

Adaptive challenge for algebraic and realistic dynamic optimisation benchmarks.

ALZA SANTOS, J.

2025


The author of this thesis retains the right to be identified as such on any occasion in which content from this thesis is referenced or re-used. The licence under which this thesis is distributed applies to the text and any original images only – re-use of any third-party content must still be cleared with the original copyright holder.

ADAPTIVE CHALLENGE FOR ALGEBRAIC AND REALISTIC DYNAMIC OPTIMISATION BENCHMARKS

JOAN ALZA SANTOS

ADAPTIVE CHALLENGE FOR ALGEBRAIC AND REALISTIC DYNAMIC OPTIMISATION BENCHMARKS

JOAN ALZA SANTOS

 0000-0002-7644-9993



A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF
ROBERT GORDON UNIVERSITY
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

This research programme was carried out in collaboration with the Intelligent Systems Group
(ISG), Department of Computing Science, University of the Basque Country

January 2025

Abstract

Dynamic optimisation focuses on finding the best possible solutions while continuously adapting to changing environments. One of the main challenges in academic dynamic optimisation is accurately representing the adaptive challenge of online algorithms to different dynamic features in benchmark generators, ensuring that only problems that obtain effective adaptation of online algorithms are classified as “dynamic”. However, academic research often simulates changes in the problem by simply adjusting certain features of the dynamic process, such as the magnitude and frequency of problem changes.

This thesis reviews academic research and practical applications to identify inconsistencies in defining dynamic optimisation problems, and replicates established methods to quantitatively evaluate how different dynamic features affect the adaptive advantage of algorithms. In particular, the study makes the following contribution:

First, the thesis establishes a theoretical framework for the fitness landscape rotation as a dynamic benchmark generator in order to demonstrate its preserving nature. Contributions include using the fitness landscape rotation as a tilting strategy to redirect the search for algorithms that are stuck in local optima.

Second, the study introduces the concept of *elusivity* to differentiate dynamic optimisation problems from sequences of unrelated instances by quantifying the adaptive advantage of online algorithms over restart. The conducted empirical research shows that certain combinations of problems, algorithms, and performance metrics indicate different degrees of elusivity, highlighting the importance of considering algorithmic performance in relation to problem dynamism.

Third, in order to extend the analysis to a practical scenario, the thesis presents a thorough analysis and a robust methodology for generating synthetic instances from real-world data, provided by ARR Craib, introducing a novel approach to developing dynamic benchmark instances, especially for dynamic scheduling problems. The specific benchmark generator characterises realistic features, allowing for the evaluation of optimisation algorithms in practical dynamic contexts.

Keywords: Dynamic Optimization; Combinatorial Optimization; XOR DOP; Fitness Landscape Rotation; Group Theory; Elusivity; Adaptive Challenge; Evolutionary Algorithms; Real-World Optimization; Truck and Trailer Scheduling.

Acknowledgements

I would like to express my deepest gratitude to my supervisors. John, thank you for giving me the opportunity to join your team as a researcher, and for inculcating in me your professional and personal values. This opportunity has definitely contributed to my development, both professionally and personally. Josu, thank you for letting me meet John, for being a constant source of inspiration and dedication, and for being an example of passion and commitment. You have always been a constant support, despite the distance. And thank you, Mark, for all the support and advices you have given me throughout my PhD journey. I deeply admire you all.

I would also like to thank Olivier and Mayowa for mentoring me during my early days as a researcher. Your guidance had a significant impact on me, and I will always be grateful for your attention and dedication to my development. I also want to extend my gratitude to my colleagues (Ellie, Martin, Benjamin, Carlos, Reggie, etc.), who not only inspired and supported me, but also created a collaborative and productive working environment.

I would like to express my gratitude to my Viva Committee members, Dr. Ciprian Zavoianu (internal), Prof. Trung Thanh Nguyen (external), and the convenor, Prof. Simon Burnett, for their time, valuable feedback, and the straight pass award granted, with only minor presentational amendments.

Thanks to ARR Craib Ltd for allowing us to use their data as part of the contributions to this thesis.

A huge thank you to NSC and RGU for the financial support provided during my PhD studies, as well as for enabling my participation in various conferences, workshops, and seminars. Together with UPV-EHU, the excellent facilities, atmosphere, and care provided have been invaluable to my research career. Lynne, Rianne, Kelly, Shona, Virginia, Kate, Sheena, Alan, Josune, Amparo, and all the administration staff on the mentioned institutions, thank you.

I would like to thank my friends, who have been with me during the happy and sad moments I have faced during these years. Thank you for the fun and the enjoyable moments we shared together, as well as your support during the challenging ones.

Especially, I would like to dedicate a paragraph to my partner, Ane. Words cannot fully express my admiration to you as the perfect example of love, perseverance, and resilience. Your strong determination to face challenges and your character not only inspire me, but everyone around you. I sincerely appreciate your sensitivity, guidance, and support throughout this journey, as well as the infinite supply of love and support I receive from you. Together, we have shown to be a strong and complementary team for enjoyment, and capable of overcoming adverse circumstances. *(esp) Te quiero mucho.*

Finally, I would like to express my gratitude to my entire family for their education and for giving me with change to develop strong values. Without their help, none of my achievements would have been possible. This thesis is dedicated to my parents, my two sisters, and my grandparents, including my grandfather who passed away during my studies. *(eus) Jasotako maitasunagatik eta emandako balioengatik, eskertuta nago eta betiko egongo naiz. Hau ere zuena da. (esp) Estoy y estaré eternamente agradecido por el cariño recibido y los valores inculcados. Esto va por vosotros.*

Declaration

I confirm that the work contained in this PhD project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

SignedJoan Alza Santos..... DateJanuary 14, 2025.....

Joan Alza Santos

Contents

I	Preliminaries	1
1	Introduction	2
1.1	Research Questions	4
1.2	Aim and Objectives	5
1.3	Research Publications	6
1.4	Thesis Overview	6
2	Literature Review	8
2.1	Combinatorial Optimisation	9
2.1.1	Combinatorial Fitness Landscape	9
2.1.2	Permutation Space	11
2.1.3	Combinatorial Optimisation Problems	14
2.2	Dynamic Optimisation	16
2.2.1	Dynamic Optimisation Problems	19
2.2.2	Benchmark Generators	21
2.3	Summary	29
II	Contributions	31
3	Fitness Landscape Rotation	32
3.1	Introduction to Landscape Rotation	34
3.1.1	Group Properties in Landscape Rotation	35
3.2	Analysis of the Landscape Rotation	35

3.2.1	Neighbourhood Preservation	36
3.2.2	Preservation of the Structure	39
3.2.3	Repercussion of the Landscape Rotation	43
3.3	Landscape Rotation as Perturbation Strategy	45
3.4	Experimentation	49
3.4.1	Case Study 1	50
3.4.2	Case Study 2	54
3.5	Summary	60
4	Elusivity of Dynamic Optimisation Problems	62
4.1	Definition	63
4.2	Case Studies	67
4.2.1	Benchmark Generators	67
4.2.2	Algorithms	68
4.2.3	Performance Metrics	68
4.2.4	Parameter Settings	71
4.2.5	Case Study I	72
4.2.6	Case Study II	81
4.3	Summary	88
5	Data Analysis on Dynamic Scheduling	90
5.1	Problem Description	92
5.1.1	Problem Formulation	94
5.2	Data Description and Preprocessing	97
5.2.1	Historical Actions and Constraints	99
5.2.2	Trucks, Trailers and Drivers	106
5.3	Overview of the Data	109
5.4	Temporal Analysis of the Data	110
5.4.1	Time-Series Decomposition Analysis	110
5.4.2	Data Characteristics and Patterns	114

5.5	Summary	119
6	Realistic Benchmark Generator	120
6.1	Synthetic Data Generation	121
6.1.1	Gaussian Copula	123
6.1.2	Evaluation Strategies	125
6.2	Experimentation	130
6.2.1	Parameter Settings	130
6.2.2	Results and Discussion	133
6.2.3	Further Analysis	142
6.3	Summary	146
III	Conclusions	149
7	Conclusions and Future Work	150
7.1	Research Questions and Major Contributions	150
7.2	Future Work	152
IV	Appendix	156
A	Extended Elusivity Calculation for the Benchmark Generators in Case Study I	157
B	Ethical and Legal Concerns	161
C	Extended Evaluation of Gaussian Copula	165
D	Time-series Analysis of the ARRC data	175
E	STL operations and Loess Regression	180
	Bibliography	182

List of Figures

2.1	Representation of permutations σ and π , and their composition operation $\sigma \circ \pi$	12
2.2	Illustration of different operations in the permutation $\sigma = 3142$	13
2.3	Classes of problems based on the frequency and magnitude of change . .	24
2.4	Visual representation of the fitness landscape rotation.	27
3.1	Fitness landscape rotation example.	43
3.2	Search Trajectory Networks of rotation-based algorithms on LOP.	53
3.3	Illustration of the search of rotation-based algorithms, under the Cayley distance metric, in an independent run on tai40a	55
3.4	Comparison between the performance of rotation-based algorithms under the Cayley distance	57
3.5	Performance evaluation of rotation-based algorithms against other local search algorithms	59
3.6	Illustration of the search process of local search algorithms with and without rotation-based strategies	60
4.1	Elusivity analysis over different algorithms	74
4.2	Elusivity analysis over different performance metrics	74
4.3	Elusivity analysis over different optimisation problems	74
4.4	Bayesian statistical analysis of elusivity	78
4.5	Optimal parameter setting based on the elusivity.	79
4.6	Influence of parameters on the elusivity	80

4.7	Elusivity heatmaps of DTSPs with city replacement to four algorithms under three different performance metrics.	83
4.8	Elusivity heatmaps of DTSPs with traffic factor to four algorithms under three different performance metrics.	83
4.9	Elusivity as a measure for adaptive advantage algorithms on DTSPs with traffic factor under robustness.	83
5.1	Entity relationship diagram showing the relationships of the data after preprocessing.	99
5.2	Frequency of tasks on the preprocessed job dataset.	101
5.3	Geographical map showing the performed tasks per location.	102
5.4	Constraint frequency of the job dataset, considering only jobs with constraints.	104
5.5	Frequency of requested trailer types in the job dataset.	105
5.6	Relationship graph showing the compatibility for each trailer type. . . .	107
5.7	Number of trailers in the ARRC fleet in 2019, organised by trailer type.	108
5.8	MSTL decomposition of the collection time of jobs	113
5.9	Annual demand of jobs for ARRC in 2019 in a daily basis.	115
5.10	Number of collected jobs in a daily basis aggregated by week days in 2019	116
5.11	Weekly aggregated input, collection, and delivery times of jobs over the year	117
5.12	Daily seasonality of the collection time for each respective month of the year	117
5.13	Daily seasonality of static and dynamic jobs	118
6.1	Data structure (metadata) of the original job dataset, and the data after <i>weekly</i> and <i>daily</i> decomposition steps.	131
6.2	Daily demand for jobs produced by Gaussian Copula with different temporal approaches	134

6.3	Weekly seasonality of the collection time of jobs over the year produced by Gaussian Copula with different temporal approaches	134
6.4	Time-distribution visualisation of Gaussian Copula with different temporal approaches	134
6.5	Time-distributions of static and dynamic jobs for the Gaussian Copula based with the time-difference approach on the data with daily decomposition.	137
6.6	Daily demand for jobs produced by Gaussian Copula with different temporal approaches.	139
6.7	Time-distribution visualisation of ensemble Gaussian Copula with different temporal approaches.	139
6.8	Influence of numerical and categorical time-related variable representation in the data on generation of synthetic data	143
B.1	Skills (constraints) covered by the drivers in 2019.	163
C.1	Correlation comparison of original and synthetic data using Gaussian Copula.	169
C.2	Correlation comparison of original and synthetic data with weekly decomposition using Gaussian Copula.	170
C.3	Correlation comparison of original and synthetic data with daily decomposition using Gaussian Copula.	171
C.4	Correlation comparison of original and synthetic data using ensemble Gaussian Copula.	172
C.5	Correlation comparison of original and synthetic data with weekly decomposition using ensemble Gaussian Copula.	173
C.6	Correlation comparison of original and synthetic data with weekly decomposition using ensemble Gaussian Copula.	174
D.1	Daily seasonality of the data in each respective month	176
D.2	MSTL decomposition of the input time of jobs	177

D.3	MSTL decomposition of the collection time of jobs	178
D.4	MSTL decomposition of the delivery time of jobs	179

List of Tables

3.1	Summary of solution exchanges following an example of fitness landscape rotation.	44
3.2	Parameter settings for the algorithms with the fitness landscape rotation as perturbation strategy for the LOP.	50
3.3	Information of the instances, and results of rotation-based algorithms on LOP instances.	51
3.4	Number of rotations and reached local optima by each rotation-based algorithm	52
3.5	Parameter settings for the algorithms with the fitness landscape rotation as perturbation strategy for the QAP.	55
3.6	Parameter settings for rotation-based algorithms in the advanced experimental study of the QAP.	58
4.1	Parameter values of the benchmark generators and algorithms	72
4.2	Overall performances of the algorithms and the elusivity value for each problem, algorithm, and performance metric combination	73
5.1	Description of the attributes of the preprocessed job dataset.	100
5.2	Frequency of each constraint in the preprocessed job dataset.	103
5.3	Summary of the data after preprocessing.	109
6.1	Analysis of the Gaussian Copula with different temporal approaches. . .	133
6.2	Performance of Gaussian Copula with different temporal approaches on the data with weekly and daily decomposition.	136

6.3	Performance of ensemble Gaussian Copulas, using different temporal approaches, on the original and (weekly and daily) decomposed data. . . .	138
6.4	Data characteristics for the considered typical day instances.	140
6.5	Performance of the conditional sampling of the ensemble Gaussian Copula with the time-difference transformation approach on the weekly decomposed data.	141
6.6	Influence of numerical and categorical time-related variable representation in the data on generation of synthetic data	144
6.7	Influence of the marginal distribution representation on the synthetic data generation process	145
6.8	Marginal distribution similarity score for each marginal for the Gaussian Copulas with time-difference approach on the data with weekly decomposition.	145
6.9	Marginal distribution similarity score for each time-related marginal for the Gaussian Copulas with time-difference approach on the data with weekly decomposition.	145
A.1	Summary of the highest posterior probabilities regarding the overall elusivity of DTSPs with fitness landscape rotation, constructed from kroA100, to the considered algorithms and performance metrics.	159
A.2	Summary of the highest posterior probabilities regarding the overall elusivity of DKPs with fitness landscape rotation to the considered algorithms and performance metrics.	160
C.1	Marginal distribution comparison of Gaussian Copulas under different time-related constraints (continued).	166
C.2	Marginal distribution comparison of time variables using Gaussian Copulas on data with weekly decomposition.	168
C.3	Marginal distribution comparison of time variables using ensemble Gaussian Copulas on data with weekly decomposition.	168

C.4	Marginal distribution comparison of time variables using Gaussian Copulas on data with daily decomposition.	168
C.5	Marginal distribution comparison of time variables under daily-feature extraction using ensemble Gaussian Copulas.	168

List of Algorithms

3.1	saHC-R1: depth-first rotation strategy	46
3.2	saHC-R2: breadth-first rotation strategy	48
5.1	MSTL: Multiple Seasonal-Trend decomposition using Loess	112
6.1	Gaussian Copula	124
E.1	LOESS: Locally Estimated Scatterplot Smoothing	180
E.2	STL: Seasonal-Trend decomposition using LOESS	181

Abbreviations

- **ACO**: Ant Colony Optimisation.
- **ADR**: Accord Dangereux Routier.
- **AI**: Artificial Intelligence.
- **ARRC**: ARR Craib Ltd.
- **CC**: Correlation Comparison.
- **DOP**: Dynamic Optimisation Problem.
- **DCOP**: Dynamic Combinatorial Optimisation Problem.
- **DKP**: Dynamic Knapsack Problem.
- **DQAP**: Dynamic Quadratic Assignment Problem.
- **DTSP**: Dynamic Travelling Salesperson Problem.
- **EA**: Evolutionary Algorithm.
- **EI{ACO, PBIL, GA}**: Elitism Immigrants-based ACO, PBIL and GA.
- **GA**: Genetic Algorithm.
- **GAN**: Generative Adversarial Network.
- **GDPR**: General Data Protection Regulation.
- **LOESS**: LOcally Estimated Scatterplot Smoothing.
- **MDC**: Marginal Distribution Comparison.
- **MMAS**: MIN-MAX Ant System.
- **MC-MMAS**: Multi-Colony MIN-MAX Ant System.
- **MPB**: Moving Peaks Benchmark.
- **MSTL**: multiple seasonal-trend decomposition using LOESS.
- **PACO**: Population-based Ant Colony Optimisation.

- **PBIL**: Population-based Incremental Learning.
- **RI**{ACO, PBIL, GA}: Random Immigrants-based ACO, PBIL and GA.
- **RDT**: Reversible Data Transforms.
- **sHC**: stochastic Hill-Climbing.
- **saHC**: steepest-ascent Hill-Climbing.
- **SDV**: Synthetic Data Vault.
- **STL**: Seasonal-Trend decomposition using LOESS.

Mathematical Abbreviations

- A : online algorithm.
- A^r : restarting algorithm.
- B : matrix for LOP.
- $\mathcal{B}(x^*)$: attraction basin of the local optimum x^* .
- C : capacity of the knapsack for KP.
- $\text{Corr}_{X,Y}$: correlation function.
- $C_{X,Y}$: contingency tables of variables X and Y .
- \mathcal{C} : copula function.
- D : distance matrix for TSP and QAP.
- $\mathcal{E}(P, A, \phi)$: elusivity of P to A under ϕ .
- F : flow matrix for QAP.
- F_X : cumulative distribution function of variable X .
- F_{BOG} : best-of-generation performance metric.
- f : objective function.
- f_X : probability density function of variable X .
- $\mathcal{G}_f(x^*)$: attraction graph of a local optimum, represented as a directed acyclic graph of an attraction basin.
- H : multivariate cumulative distribution function.
- H_{Δ_m} : accuracy performance metric.
- KS : Kolmogorov-Smirnov test.
- \mathcal{N} : neighbourhood function.
- n : size of solutions.

- O_f : collection of attraction graphs that represent the fitness landscape.
- P : dynamic optimisation problem.
- P_s : static optimisation problem.
- $p_i \in p$: profit value of the item i in the knapsack for KP.
- Q : search space.
- ρ : correlation similarity.
- \mathfrak{R} : correlation matrix.
- \mathcal{R} : robustness performance metric.
- \hat{R} : residual component of a time-series.
- \mathbf{R} : original data.
- R_X : column X in the original data.
- S : set of neighbourhood operators.
- \mathbf{S} : synthetic data.
- S_X : column X in the synthetic data.
- \mathcal{S}^c : contingency similarity.
- \hat{S} : seasonality component of a time-series.
- σ : a permutation.
- \hat{T} : trend component of a time-series.
- \mathbf{T} : time columns in the job dataset.
- \mathcal{T} : time variables' similarity.
- TV : total variance distance.
- θ : parameters of a distribution.
- $w_i \in w$: weight of the item i in the knapsack for KP.
- x : a solution from the search space.
- x^* : a local optimum of an attraction basin.
- ϕ : performance metric.
- Ω : search space.

Part I

Preliminaries

Chapter 1

Introduction

Dynamic optimisation is a field of optimisation under uncertainty, where the aim is not only to find the best solution, but also to track it over time [1]. Therefore, standard optimisation algorithms, which are designed to find the best possible solution, are generally coupled with adaptive mechanisms to balance the exploitation and exploration of its search process to deal with problem changes [2]. Without loss of generality, algorithms that deal with changes in the problem during algorithm execution are commonly referred to as *online* algorithms. Hence, dynamic optimisation involves adjusting immediate decisions to meet long-term objectives, promoting adaptive decision-making in changing conditions.

In real-world, dynamic optimisation is found in various domains, such as supply chain management, financial forecasting, network routing, and telecommunications [3]. For example, in supply chain logistics, companies continuously adjust their decisions to address demand variability and other disruptions, such as weather conditions or the influence of traffic.

The literature presents different definitions for dynamic optimisation problems (DOPs), with some researchers defining them as a *sequence of static optimisation problem instances linked up by a dynamic rule* [4, 5], whereas others define them as *optimisation problems composed by time-dependent parameters* [3, 6]. Nevertheless, these definitions

fail to distinguish between DOPs and sequences of unrelated static optimisation problems, as they encompass all types of change. For example, in cases of major changes to the problem, it may be more appropriate to restart the algorithm search randomly and consider the new environment as a separate problem instance rather than a variation of the previous one [7]. In those situations, restarting after a change can be effective, although the community usually avoids it. In [8], DOPs are further defined as *a special class of dynamic problems that are solved online by optimisation algorithms as time goes by*, emphasising the adaptive advantage of algorithms.

Related to the definition of DOPs, simulating realistic dynamic environments remains a significant challenge in academic research. Simplified dynamic benchmark problem generators have been introduced in the literature to instantiate sequences of static optimisation problem instances by incrementally changing objective functions, constraints, or variables. Despite these generators enable controlled changes of dynamic features, empirical studies often ignore their actual repercussion of changes and the real-world feature representation, as they primarily focus on the frequency and magnitude of changes [9, 10]. For instance, a widely used academic benchmark generator for dynamic combinatorial optimisation problems, the fitness landscape rotation [11, 12], does not represent real-world dynamics, and the effects of the changes have not been thoroughly reviewed yet. Hence, the literature on dynamic optimisation emphasises the importance of quantifying and comparing the adaptive challenge¹ of algorithms to different features of DOPs. In particular, quantitative evaluation of the adaptive advantage of online algorithms against a random restart after a problem change can help differentiate DOPs from sequences of unrelated problems.

In summary, there is a scarcity of academic research focused on the theoretical and empirical analysis of DOPs, which is essential to understand the impact of changes on the fitness landscape and the performance of online algorithms. Theoretical investigations can clarify differences in the definitions of DOPs and refine them by providing a framework to measure the relationship between problem dynamism and the adaptive

¹Informally, the adaptive challenge of a dynamic optimisation problem involves the “difficulty” that online algorithms face to adapt to changes in the problem.

advantage of algorithms. Practically, a systematic evaluation of algorithmic performance to different problem configurations could help analyse specific adaptations to different types and degrees of change. Additionally, incorporating data-driven benchmark generators that capture real-world complexities and patterns, such as seasonality or spatial variability, may be useful for developing algorithms capable of handling the dynamism and complexity of realistic dynamic environments [13].

1.1 Research Questions

This research project aims to answer the following question: *how can we analyse the adaptive challenge of algorithms to the features of academic and real-world dynamic optimisation problems in order to evaluate, control, and measure the adaptive advantage of online algorithms and their effectiveness based on selected evaluation criteria?*

In order to address the research question, it is essential to examine existing definitions of DOPs, identify their defining features, and evaluate how these features influence the adaptive advantages of online algorithms across different dynamisms. To that end, it is important to examine case studies with different benchmark generators, adaptive mechanisms and performance metrics on dynamic environments. Thus, the main question can be separated into the following questions:

- RQ 1.** Can we extend existing definitions for DOPs to quantitatively include the performance of online algorithms?
- RQ 2.** What essential features should benchmark generators include to construct realistic DOP instances?
- RQ 3.** To what extent can we quantify the adaptive advantage of online algorithms for solving a DOP compared to randomly restarting the algorithm after a problem change?
- RQ 4.** How can we apply the gained insights to develop advanced approaches to improve the performance of standard algorithms?

1.2 Aim and Objectives

By answering the previous research questions, this thesis aims to develop a systematic theoretical-methodological investigation on the influence of features of academic and real-world DOPs on the adaptive advantage of algorithms, using a specific performance metric, under the assumption that different forms of dynamism will pose different adaptive challenges to different algorithms.

This research is supported by the following objectives that will determine the process to reach the aim:

- OB 1** To thoroughly and systematically review the literature on definitions, dynamic features, research gaps, and methods in dynamic optimisation in order to distinguish between dynamic optimisation problems (where algorithms adapt to changes over time) and unrelated static optimisation problems that change without similarity.
- OB 2** To theoretically analyse the preservation of neighbourhood relationships between solutions and the fitness landscape topology under rotation operations, as well as the repercussion of rotation operations on the rearrangement of solutions within attraction basins. Moreover, drawn insights are used to design and evaluate novel rotation-based perturbation strategies for local search algorithms.
- OB 3** To quantitatively measure the probability of generating *elusive* problems (significant changes where restart is preferred over adaptation) by evaluating the adaptive advantage of algorithms on existing dynamic benchmark generators. The idea is to quantify the adaptive challenge of a problem by measuring the impact of dynamic features (e.g. change frequency and magnitude) on the performance of algorithms.
- OB 4** To explore the construction of dynamic benchmarks from real-world applications through synthetic data generation, in addition to demonstrate the applicability of the developed elusivity analysis (Objective **OB 3**) on a realistic framework.

1.3 Research Publications

This section summarises the publications resulting from this thesis, which are directly related to the research questions, and the aim and objectives presented in previous sections.

During the course of this doctoral research, a total of four publications (three conference and one journal publications) have been produced, which are listed below for consultation in this thesis:

1. **J. Alza**, M. Bartlett, J. Ceberio, and J. McCall. “*On the Definition of Dynamic Permutation Problems under Landscape Rotation*”. In Proceedings of GECCO, Prague, 2019.
2. **J. Alza**, M. Bartlett, J. Ceberio, and J. McCall. “*Towards the Landscape Rotation as a Perturbation Strategy on the Quadratic Assignment Problem*”. In Proceedings of GECCO, Lille, 2021.
3. **J. Alza**, M. Bartlett, J. Ceberio, and J. McCall. “*Analysing the Fitness Landscape Rotation for Combinatorial Optimisation*”. In Proceedings of PPSN, Dortmund, 2022.
4. **J. Alza**, M. Bartlett, J. Ceberio, and J. McCall. “*On the Elusivity of Dynamic Optimisation Problems*”. Swarm Evol. Comput., Volume 78, 1–13, 2023.

1.4 Thesis Overview

This thesis is divided into three parts: preliminaries (Part I), contributions (Part II), and conclusions (Part III). The rest of this thesis is organised as follows.

Chapter 2 reviews and categorises the definitions and methods from the literature. The goal of the chapter is to identify the gap and assumptions of the dynamic optimisation community and contextualise the contributions of this research work.

Chapter 3 thoroughly investigates the *fitness landscape rotation*, a widely used benchmark generator for DOPs. More precisely, an algebraic foundation is given to

analyse the preserving nature of the method, and capture the repercussion of changes. In addition, from the insights gained from the theoretical analysis, two rotation-based perturbation strategies for local search algorithms are developed and analysed.

In Chapter 4, the concept of *elusivity* for dynamic optimisation is presented, including generalised mathematical notations, to precisely describe dynamic optimisation problems and the performance of algorithms. Furthermore, two different case studies are considered to empirically illustrate and analyse the applicability of the elusivity.

Chapters 5 and 6 collectively address a real-world dynamic truck and trailer scheduling problem. Specifically, Chapter 5 provides a problem formalisation and a detailed description and analysis a thorough analysis of the provided data. Chapter 6 presents a methodology for generating synthetic problem instances from real-world data, and evaluates how closely the synthetic data replicates the distributions and patterns in the original data.

Finally, Chapter 7 concludes the thesis by emphasising the significance of the insights and findings presented in the thesis, and suggests future research directions and potential methodological approaches for further investigation.

Chapter 2

Literature Review

Optimisation algorithms have been widely applied in a range of real-world domains with the aim of reducing times, costs, or improving operational efficiency. Many of these practical applications correspond to the combinatorial domain, where the goal is to find the best combination from a finite set of solutions. Specifically, problems that require reordering or rearranging elements belong to the permutation space, commonly observed in supply chain management [14], task scheduling [15], and routing operations [16].

Generally, real-world combinatorial optimisation and permutation problems are intrinsically dynamic, where objectives, constraints, and variables change over time. For instance, vehicle routing optimisation typically depends on traffic conditions and the arrival of new deliveries. As a result, online algorithms address both the combinatorial complexity of the problem and the adaptive challenge posed by dynamic optimisation.

This section provides a systematic review of existing definitions, methods, and experiments in the dynamic optimisation literature. The chapter is structured as follows. Section 2.1 describes the fitness landscape in the combinatorial domain, the permutation space, and common combinatorial optimisation problems. Section 2.2 reviews literature on dynamic optimisation, emphasising dynamic features and benchmark generators. Finally, Section 2.3 concludes the chapter by summarising the review and outlining the motivation for this research.

2.1 Combinatorial Optimisation

Generally, combinatorial optimisation focuses on finding the best solution from a finite set of discrete possible solutions. It is commonly applied to problems where the objective is to maximise or minimise a function over a large and complex search space, such as scheduling, routing, or resource allocation [17].

The following subsections discuss the combinatorial fitness landscape, the structure of permutation spaces, and combinatorial optimisation problems studied in this thesis.

2.1.1 Combinatorial Fitness Landscape

Formally, a combinatorial optimisation problem is a tuple $P = (\Omega, f)$, where Ω is a countable finite set of structures, called search space, and $f : \Omega \rightarrow \mathbb{R}$ is an objective function that needs to be maximised or minimised. As these problems are generally NP-hard¹ [18], heuristic and metaheuristic algorithms, and especially local search algorithms, have been widely used to solve combinatorial problems [19].

A key assumption about local search algorithms is the *neighbourhood* function, which links solutions to each other through their similarity. Formally, a neighbourhood \mathcal{N} is a mapping between a solution $x \in \Omega$ and a set of solutions $\mathcal{N}(x)$ after a certain operation in the encoding of x , such that

$$\mathcal{N} : \Omega \rightarrow \mathcal{P}(\Omega), \quad (2.1)$$

where $\mathcal{P}(\Omega)$ is the power set of Ω . In other words, two solutions x and y are considered neighbours iff modifying the encoding of x results in y , so $x \in \mathcal{N}(y)$. In combinatorial optimisation, the neighbourhood function usually represents *symmetric* relations, meaning operations are invertible, i.e. $x \in \mathcal{N}(y) \Leftrightarrow y \in \mathcal{N}(x)$. This symmetry leads to *regular* neighbourhoods, where every solution in Ω has the same number of neighbours.

¹An NP-Hard problem is one for which there is no algorithm that can solve all instances in polynomial time. These problems often require exponential time in the worst case, making exact methods impractical for large instances.

Furthermore, the fitness landscape in the combinatorial domain can be defined as the collection of combinatorial optimisation problems together with the neighbourhood function [20]. Formally, the fitness landscape is a triple (Ω, f, \mathcal{N}) , where Ω is the search space, f is the objective function and \mathcal{N} is the neighbourhood function.

The fitness landscape definition helps to understand the behaviour of local search algorithms in combinatorial problems with specific neighbourhood functions. In other words, the behaviour of local search algorithms, along with the suitability of different neighbourhood functions, can be studied based on properties of the fitness landscape, such as the number of local and global optima, basins of attraction, and *plateaus* [19, 21]. These components are described in detail in the following paragraphs.

A local optimum is a solution $x^* \in \Omega$ whose objective value is better or equal than its neighbours' $\mathcal{N}(x^*) \in \Omega$, i.e. for any maximisation problem, $\forall y \in \mathcal{N}(x^*)$, $f(x^*) \geq f(y)$. The number of local optima of a combinatorial problem can be certainly associated to the difficulty of a local search algorithm to reach the global optimum (the local optimum with the best objective value) [21]. Nevertheless, there are other problem features, such as the size of the search space and symmetries, that are also valid for understanding the behaviour of local search algorithms [22].

Some studies in the combinatorial domain study the basins of attraction for local optima to analyse the fitness landscape, and evaluate the likelihood of reaching the global optimum [19, 21, 23]. Formally, an attraction basin of a local optimum, $\mathcal{B}(x^*)$, consists of solutions that lead to the local optimum x^* when a local search algorithm is applied; so $\mathcal{B}(x^*) = \{x \in \Omega \mid a_x = x^*\}$, where a_x is the final solution produced by the algorithm starting from x . The attraction basin $\mathcal{B}(x^*)$ can be represented as a tree-like directed acyclic graph, where the vertices² represent solutions, and the edges indicate the transitions between solutions. This assumption leads to the following definition.

Definition 2.1 (Attraction graph). *Let us define an attraction graph to be a directed graph $\mathcal{G}_f(x^*) = (V_f, E_f)$, where f is the objective function, $V_f \subseteq \Omega$ is a set of solutions, and E_f is a set of directed edges representing the movement from a solution to a*

²Without loss of generality, we use *vertex* for graphs and *nodes* for networks.

neighbour with a better, or equal, objective value. For every solution in the graph, there is an increasing path (sequence of solutions connected by directed edges) until reaching the local optima, such that $\forall x \in V, (x = a_1, a_2, \dots, a_h = x^*)$, where $a_{i+1} \in \mathcal{N}(a_i)$, $(a_i, a_{i+1}) \in E$, and $f(a_i) \leq f(a_{i+1})$ for any maximisation problem.

The fitness landscape can be represented as the collection of all the attraction graphs, such that $O_f = \cup_{x^* \in \mathcal{B}(x^*)} \mathcal{G}_f(x^*)$, where x^* is a local optimum of the attraction basin $\mathcal{B}(x^*) \subset \Omega$, given a triple (Ω, f, \mathcal{N}) . For clarity purposes, we define $|O_f|$ to represent the number of attraction graphs that compose the fitness landscape O_f . Note that a solution (vertex) may belong to multiple attraction graphs if some neighbours, that belong to different attraction graphs, share the same objective value.

In the case that neighbouring solutions have equal objective values, the landscape is said to have flat structures, known as *plateaus*. Formally, a plateau $\Gamma \subseteq \Omega$ is a set of solutions with the same objective value, where for any pair of solutions $x, y \in \Gamma$, there exists a path $(x = a_1, a_2, \dots, a_k = y)$ such that $a_i \in \Gamma$, $a_{i+1} \in \mathcal{N}(a_i)$, $f(a_i) = f(a_{i+1})$, $i \in \{1, 2, \dots, k\}$. The authors in [19] illustrate that combinatorial problems frequently contain plateaus and remark the importance of recognising them in the combinatorial domain. They also identify three types of plateaus and state that a plateau with multiple local optima can be considered as a single local optimum in local search algorithms, as their basins of attraction converge to the same plateau.

2.1.2 Permutation Space

One of the most studied fields in combinatorial optimisation is the permutation space, where problem solutions are represented by permutations. Formally, a permutation is a bijection from a finite set, usually composed of natural numbers $\{1, 2, \dots, n\}$, onto itself. The search space Ω represents all permutations of size n , referred to as the *symmetric group* \mathbb{S}_n , and has a cardinality of $n!$. Permutations are usually represented by $\sigma, \pi, \gamma \in \mathbb{S}_n$, except for the identity permutation $e = 12 \dots n$.

Given two permutations π and σ , their composition is defined as $(\sigma \circ \pi)(i) = \sigma(\pi(i))$, for $i \in \{1, 2, \dots, n\}$. Generally, the composition of two permutations is *non-commutative*, meaning $\sigma \circ \pi \neq \pi \circ \sigma$.

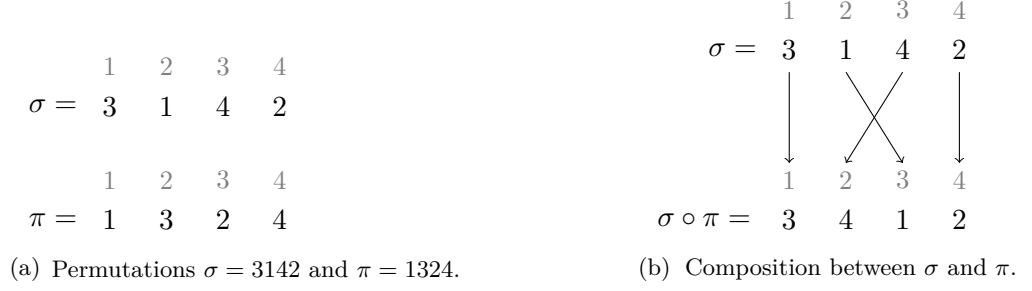


Figure 2.1: Representation of permutations σ and π , and their composition operation $\sigma \circ \pi$. The gray numbers illustrate the position of each element in the permutation.

Figure 2.1 shows two permutations, $\sigma = 3142$ and $\pi = 1324$, and illustrates their composition, $\sigma \circ \pi$. By definition, for $i \in \{1, 2, 3, 4\}$, the composition operation between π and σ can be described as follows:

$$(\sigma \circ \pi)(1) = \sigma(\pi(1)) = \sigma(1) = 3,$$

$$(\sigma \circ \pi)(2) = \sigma(\pi(2)) = \sigma(3) = 4,$$

$$(\sigma \circ \pi)(3) = \sigma(\pi(3)) = \sigma(2) = 1,$$

$$(\sigma \circ \pi)(4) = \sigma(\pi(4)) = \sigma(4) = 2.$$

Thus, the result of the composition operation is $\sigma \circ \pi = 3412$.

We direct the interested reader to [17] for more information about formal definitions on permutations and their representations.

2.1.2.1 Distance metrics on the Permutation Space

The distance between permutations can be defined as the minimum number of steps to transform one permutation into another. In the context of permutations, pairwise swaps, adjacency swaps, and insertions are fundamental operations that transform one permutation into another by modifying the order of elements. Specifically, each operator can be formally described as follows:

- A **pairwise swap** exchanges two elements in different positions. That is, given a permutation $\sigma = \sigma_1, \sigma_2, \dots, \sigma_n$ and two indices i, j such that $1 \leq i < j \leq n$, a pairwise swap transforms σ into $\sigma_1, \dots, \sigma_{i-1}, \sigma_j, \sigma_{i+1}, \dots, \sigma_{j-1}, \sigma_i, \sigma_{j+1}, \dots, \sigma_n$.

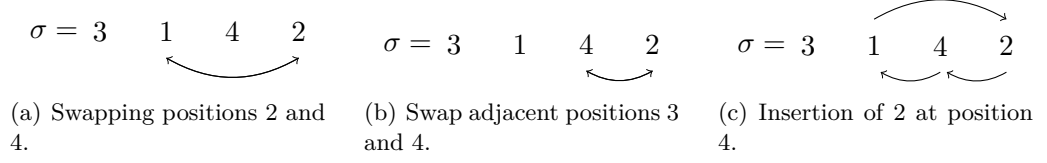


Figure 2.2: Illustration of different operations in the permutation $\sigma = 3142$. Specifically, the figure shows (a) a pairwise swap, (b) an adjacency swap, and (c) an insertion operation.

- An **adjacency swap** exchanges two consecutive elements in the permutation. That is, given a permutation $\sigma = \sigma_1, \sigma_2, \dots, \sigma_n$ and an index i such that $1 \leq i < n$, an adjacency swap transforms σ into $\sigma_1, \dots, \sigma_{i-1}, \sigma_{i+1}, \sigma_i, \sigma_{i+2}, \dots, \sigma_n$.
- An **insertion** relocates an element from one position to another within the permutation. That is, given a permutation $\sigma = \sigma_1, \sigma_2, \dots, \sigma_n$ and two indices i, j such that $1 \leq i \neq j \leq n$, the insertion operation moves the element at position i to position j , shifting the intermediate elements accordingly:

$$\begin{cases} \sigma_1, \dots, \sigma_{i-1}, \sigma_{i+1}, \dots, \sigma_j, \sigma_i, \sigma_{j+1}, \dots, \sigma_n & \text{if } i < j, \\ \sigma_1, \dots, \sigma_{j-1}, \sigma_i, \sigma_j, \dots, \sigma_{i-1}, \sigma_{i+1}, \dots, \sigma_n & \text{if } i > j. \end{cases}$$

In order to better comprehend the distance metrics between permutations, Figure 2.2 illustrates these operations on the permutation $\sigma = 3142$. In the given example, a pairwise swap exchanges elements at positions 2 and 4, an adjacent swap occurs between the elements at positions 3 and 4, and an insertion moves the element from position 2 to position 4, shifting elements at positions 3 and 4 forward.

Although there are many metrics, Kendall's- τ , Cayley, Hamming, and Ulam distances have been widely used to measure the distance between permutations in the combinatorial space [24].

Given two permutations σ and π , Kendall's- τ metric (d^K) counts the minimum number of pairwise disagreements between two permutations. Equivalently, it corresponds to the number of adjacent swaps to turn σ^{-1} into π^{-1} . The maximum distance between two permutations under Kendall's- τ metric is $d_{max}^K = \binom{n}{2}$, where n represents the size of the permutations.

The Cayley metric (d^C) counts the minimum number of (possibly non-adjacent) pairwise swaps that are needed to turn σ into π . In the case of Cayley metric, the maximum distance between two permutations is $d_{max}^C = n - 1$.

The Hamming metric (d^H) quantifies the number of positions where two permutations differ. Therefore, the maximum Hamming distance between two permutations is $d_{max}^H = n$, which means that all positions between two permutations are unequal.

Finally, the Ulam (d^U) metric represents the minimum number of insertions needed to transform a permutation into another. The maximum Ulam distance between two permutations is $d_{max}^U = n - 1$.

Additionally, Irurozki [24] presents methods for generating new permutations uniformly at random for each distance metric. For more details on permutation distance metrics and uniform permutation generation, readers can refer to [24].

2.1.3 Combinatorial Optimisation Problems

The following paragraphs describe the combinatorial problems studied in this thesis, including the knapsack problem as a binary problem, and three permutation problems. For a more detailed analysis of permutation-based problems, the interested reader is referred to [17].

Knapsack Problem

The Knapsack Problem (KP) is a well-studied maximisation combinatorial optimisation problem. Given a set of n items with a weight w and a profit value p , the goal is to collect the items that sum the largest possible profit without exceeding the capacity of the knapsack C . Mathematically, it may be modelled as follows using the binary representation:

$$f(x) = \sum_{i=1}^n p_i x_i, \quad \text{subject to } \sum_{i=1}^n w_i x_i \leq C, \quad (2.2)$$

where $x \in \{0, 1\}^n$ is a binary array of n items that represents by $x_i = 1$ the items that are selected, and w_i and p_i are the weight and the profit of the item i , for $1 \leq i \leq n$, respectively.

One way of modelling it for optimisation algorithms is applying a penalty to the fitness of the solutions when the sum of the selected weights exceeds the capacity of the knapsack [12, 25]. Hence, solutions that exceed the capacity of the knapsack are less competitive than those that satisfy the constraint, i.e. the smaller the weight over the capacity, the higher the quality of the penalised solution.

Travelling Salesperson Problem

The Travelling Salesperson Problem (TSP) is a minimisation combinatorial optimisation problem that aims to find the shortest path (or the path with the minimum cost) that crosses n cities, visiting each only once before returning to the origin. The distance between cities is represented by a distance matrix $\mathbf{D} = [d_{i,j}]_{n \times n}$, where $d_{i,i} = 0$, $i, j \in (1, n)$. Formally, the TSP may be defined as follows:

$$f(\sigma) = \sum_{i=1}^{n-1} (d_{\sigma(i), \sigma(i+1)}) + d_{\sigma(n), \sigma(1)}, \quad (2.3)$$

where $\sigma \in \mathbb{S}_n$ is the permutation that describes the ordering in which the cities are visited, n is the total number of cities and $d_{i,j}$ is the distance between the cities i and j , $i, j \in (1, n)$.

Quadratic Assignment Problem

Koopmans and Beckman introduced the Quadratic Assignment Problem (QAP) [26] as an unconstrained permutation problem that consists of assigning a set of facilities to a set of locations such that the total assignment cost is minimised. Formally, the problem consists of a distance matrix \mathbf{D} and a flow matrix \mathbf{F} , both of size $n \times n$, where $d_{x,y} \in \mathbf{D}$ is the distance between locations x and y , and $f_{i,j} \in \mathbf{F}$ is the flow between facilities i and j . The total assignment cost, represented by a permutation σ , is calculated as:

$$f(\sigma) = \sum_{i=1}^n \sum_{j=1}^n f_{i,j} d_{\sigma(i), \sigma(j)}. \quad (2.4)$$

Many QAP instances used in academia contain symmetries in the fitness landscape due to the symmetrical patterns of flow and distance matrices. As studied in [27], such symmetries exist when the locations of the facilities are grouped in a rectangular way.

Moreover, the authors in [28] state that the symmetry of QAP instances should be carefully considered when designing metaheuristics to improve their performance.

Linear Ordering Problem

The Linear Ordering Problem (LOP) [29, 30] aims to find a permutation σ that orders the rows and columns of a given matrix $\mathbf{B} = [b_{i,j}]_{n \times n}$, such that the sum of the entries above the main diagonal is maximised (or equivalently, the sum of the entries below the main diagonal is minimised). The objective function for the LOP can be formulated as follows:

$$f(\sigma) = \sum_{i=1}^{n-1} \sum_{j=i+1}^n b_{\sigma_i, \sigma_j}. \quad (2.5)$$

This representation of the LOP is also referred to as the *triangulation problem of input-output matrices* [30].

The LOP has a particular symmetrical property, i.e. the reverse of the k^{th} best solution is the exact opposite of the k^{th} worst solution [31]. Additionally, it has been demonstrated that the global optima cannot have certain items in specific positions within the permutation when the matrix \mathbf{B} matches certain structures [30].

2.2 Dynamic Optimisation

Recent research on evolutionary computation has expanded towards dynamic optimisation, a branch of *optimisation under uncertainty* [1], because of its importance in real-world applications with changing environments. Generally, dynamic optimisation problems (DOPs) involve changes to the objective function, problem instance, variables, or constraints at least once during the search for an optimal solution [32].

Population-based algorithms, such as Evolutionary Algorithms (EAs) and swarm optimisation algorithms, have been widely used to adapt to changes in DOPs due to their evolving nature [32, 33, 34, 35, 36, 37]. That is, even if the best found solution becomes infeasible or suboptimal after a change, other solutions within the population can help the algorithm adjust and adapt to the new environment. For instance, Ant

Colony Optimisation (ACO) has been widely used to address the Dynamic Travelling Salesperson Problem (DTSP) because of its graph-search capabilities [16, 35, 38].

However, traditional EAs often underperform in dynamic optimisation because they focus on converging to optimal solutions, thereby sacrificing exploration of the solution space. Instead, algorithms (commonly referred to as *online* algorithms) are coupled with adaptive mechanisms to balance exploration and exploitation, and, thus, quickly and accurately respond to changing environments [25, 39, 40].

Online algorithms often need to detect changes to handle them effectively. Problem detection is typically obtained by reevaluating the objective value of the best solution(s) at each generation, commonly known as *detectors* [9]. That is, a change is identified when detectors obtain different objective values at generations i and $i + 1$. Once a change is detected, the adaptive mechanism recalibrates the algorithm to balance the exploitation and exploration of its search process. Note that, in cases of undetectable changes, online algorithms perform inefficiently due to their challenge to adapt to problem changes [41].

Existing adaptive mechanisms can be grouped into different categories [3, 9, 42]. In the following, the adaptive mechanisms used in this research are explained.

- **Diversity approaches** balance the convergence and divergence of the algorithm to support adaptation to the new environment [2]. These approaches can be categorised into two groups: (i) the approaches that maintain the diversity through the optimisation process, and (ii) the approaches that introduce diversity after each change in the problem.

The most common diversity maintenance approaches are the immigrants-based approaches, specifically elitism- and random-immigrants [38, 43, 44]. These strategies iteratively replace a part of the population with new individuals, either randomly generated or through the mutation of the best solution in the population, based on a specified replacement rate. Note that a replacement rate of 1.0 signifies complete population replacement with new individuals, while 0.0 signifies

no immigrant generation. Generally, elitism immigrant-based algorithms are effective for slight and occasional changes [45], whereas random immigrant-based algorithms are better suited for frequent and significant changes [46].

Moreover, diversity insertion approaches require detecting changes in the problem to introduce dynamism into the population. Probably, the most common way to introduce diversity is through the *hypermutation* approach, which increases the mutation rate of Genetic Algorithms (GAs) for a number of iterations after detecting problem changes [47, 48, 49].

- **Memory approaches** are commonly employed to address *cyclic* problem changes, as previous information (e.g. best found solutions or the probabilistic model) can help to adapt to new environments [15, 44, 50, 51]. However, these approaches have limited utility and may result in redundant stored information [2].
- The **multiple populations approach** has proven to effectively deal with changes by exploring various regions of the search space through independent subpopulations [52, 53, 54]. The main challenge is determining the optimal number and size of the populations, which depends on the structure of the problem. That is, using large and many populations can enhance performance, but it also slows down the optimisation process.

Furthermore, restarting the optimisation process immediately after detecting a change is a straightforward way to address problem changes, particularly for drastically changing DOPs, as reusing prior information can be misleading [1, 9]. In short, restarting algorithms begin a new search with a random population whenever a change is detected, highlighting the need for effective change detection. This approach has been used both in academia and real-world situations. Tinos *et al.* [50] illustrate cases where restarting algorithms prove successful for the DTSP, and suggest considering this type of problems as sequences of unrelated static instances. Similarly, Allmendinger *et al.* [55] demonstrate that restarting algorithms is often optimal in a dynamic drug mixture problem with a high drug replacement rate. However, the restarting approach is

usually considered undesirable in academic contexts, as well-designed online algorithms are generally assumed to adapt effectively to a wide range of DOPs [15, 56].

The interested readers are directed to check out existing adaptive mechanisms and their applications in academic and realistic contexts [3, 9].

2.2.1 Definitions of Dynamic Optimisation Problems

The extensive research on dynamic optimisation has inspired the evolutionary computation community to present different definitions for DOPs. However, to the best of our knowledge, no unified definition has been formally established to date [8, 57]. That is, while some researchers have worked with DOPs without specific definitions [15, 58], others define them as *a sequence of static instances* [4, 34, 56, 59], and some define them as *optimisation problems with time-dependent parameters* [6, 47].

Some authors have aimed to extend those definitions. Nguyen [8] represents DOPs as *a special class of dynamic problems solved online by an optimisation algorithm over time*. Solving the problem online means continuously finding and tracking optimal solutions as time goes by, such that at time t^{now} , the objective function cannot be evaluated at time $t > t^{now}$ [13]. Similarly, Fu *et al.* [57] review existing definitions for DOPs, and propose a new framework that differentiates DOPs from static problems by emphasising the necessity for decision-makers to make sequential decisions over time.

Moreover, the presence of numerous definitions for DOPs has led researchers to develop different formulations that include different components for representing temporal changes in constraints, variables, or the objective function. In the following, some mathematical formulations derived from the literature related to DOPs are presented:

- Bosman [13] mathematically defines DOPs as maximisation problems denoted as $P = \int_0^{t^{end}} f_{\gamma(t)}(x(t))dx$, where x is the solution, f is the objective function, and γ denotes dynamic parameters, including variants and constraints. The author also notes that, for dynamic combinatorial optimisation problems, a discrete sum replaces the integral.

- Cruz *et al.* [3] use the following notation to define DOPs:

$$P = \left\{ \begin{array}{l} \text{optimise } f(x, t) \\ \text{so that } x \in F(t) \subseteq \Omega, t \in T \end{array} \right\},$$

where Ω is the search space, $t \in T$ represents the time, $f : \Omega \times T \rightarrow \mathbb{R}$ is the objective function that assigns a numerical value $f(x, t) \in \mathbb{R}$ to each possible solution $x \in \Omega$ at time t , and $F(t)$ is the set of feasible solutions $x \in F(t) \subseteq \Omega$ at time t .

- Li and Yang [60] define DOPs as $P = f(x, \phi, t)$, where f is the objective function, $x \in \Omega$ is a feasible solution within the search space Ω , t represents real-world time, and ϕ is the system control parameter. The control parameter inserts dynamism by varying the solution distribution from the actual environment by $\phi(t+1) = \phi(t) \oplus \Delta\phi$, where $\Delta\phi$ indicates the deviation from the current system control parameters.
- Rohlfshagen and Yao [5] introduce a definition for DOPs by extending the definition on static combinatorial problems by Garey and Johnson [18], where a stationary combinatorial optimisation problem P_s consists of a set of instances S_I , a finite set of candidate solutions for each instance Ω_I , and a function f that assigns a value to each solution-instance pair, $f(I, x), x \in \Omega_I$. Thus, the authors introduce a time component into the definition, and define DOPs in the context of a trajectory through a sequence of static combinatorial problem instances as the tuple $(P_s, Tr_{P_s}) = f(I(t), \mathbf{x}(t))$, where P_s represents a static combinatorial problem instance, $t \in \mathbb{N}$ indicates discrete time, $I(t) \in \Omega_I$ is the instance at time t , $Tr : I \times \mathbb{N} \rightarrow I$ defines a time-dependent trajectory through the set of instances S_I defined by $I(t+1) = Tr(I(t), t)$, and $x(t) \in \Omega_t$ is a candidate solution for the instance $I(t)$.

Furthermore, the literature presents a number of works that study the impact of problem changes on algorithmic performance. Branke [2] proposes that optimisation problems should be considered *dynamic* if and only if EAs adapt accurately to changes

over time. Conversely, problems that change independently of previous environments should be regarded as a sequence of independent problems. Younes *et al.* [61] emphasise that simply considering a time parameter in the problem definition does not imply that the problem is *dynamic*, indicating that DOPs that can be solved in advance should be considered *static*. Rohlfshagen *et al.* [7] theoretically analyse the runtime of a (1+1) EA on two simple frameworks according to the magnitude and frequency of changes, and illustrate two counter-intuitive assumptions where restarting is preferable to adaptation for slightly changing problems. Additional theoretical studies analyse the runtime analysis of different algorithms for different well-studied DOPs [62, 63, 64, 65]. Branke *et al.* [66] present metrics to analyse and characterise the nature of changes based on the shifting distance of the best solutions after a change in continuous domains. Yu *et al.* [67] demonstrate the challenges of relocating moving optima in severely and quickly changing continuous DOPs. Yazdani *et al.* [68] review the field of dynamic continuous optimisation, and identify the evaluation of the performance of online algorithms for different DOPs as a key future research direction.

In addition to formulating and defining DOPs, representing and classifying types of problem changes poses a challenge in dynamic optimisation. The next section provides an overview of the dynamic features that characterise DOPs, the classification strategies for categorising them, and the dynamic benchmark generators typically used in academic for modelling DOPs.

2.2.2 Benchmark Generators

Many studies have developed dynamic test instances or *benchmarks* to simulate varying problems for comparing algorithmic performance in dynamic environments. Due to the difficulty of obtaining real-world data, researchers often use benchmark generators with controllable features to generate specific problem variations on demand [10].

Based on the assumption that DOPs are often treated as static problem instance sequences, the literature offers various benchmark generators for transforming static optimisation problems into dynamic ones [4, 11, 15, 69]. Starting from a static problem

instance, these methods produce a sequence of static problem instances by incrementally making adjusted changes to the latest problem instance in the sequence.

In the literature on dynamic optimisation, several types of benchmark generators have been developed to simulate changes on the problem. Probably, the most popular benchmark generator involves modifying the problem variables, where the fitness landscape dynamically changes, as exemplified by the Moving Peaks Benchmark (MPB) [15] for the continuous domain or the fitness landscape rotation [11, 12] for the combinatorial domain. Another approach focuses on the modification of objective functions, where the relationships between conflicting objectives for multi-objective problems change over time [70]. Additionally, some benchmarks modify the constraints of the problem to insert dynamism, such as by varying inequality or equality constraints, thereby altering the feasible region of the fitness landscape over time [71]. Other generators introduce changes by adding or removing decision variables, leading to a variable dimensionality during optimisation [72]. Finally, note that realistic benchmarks can combine several types of changes, increasing the adaptive challenge of problems to online algorithms.

Additionally, the literature presents different dynamic features for benchmark generators, and generated problems are classified to different groups based on different criteria. However, existing benchmark generators often lack to include and combine certain important features of real-world DOPs, such as the time-linkage feature [9, 13]. The next section reviews the dynamic features and classifications for benchmark generators.

2.2.2.1 Dynamic Features and Classification Systems for Benchmark Generators

Dynamic benchmark generators are composed by certain features that characterise the dynamism of DOPs. Therefore, understanding these features is essential for the development and evaluation of online algorithms that are robust against DOPs. In short, the general features of dynamic benchmark generators are the following:

- The **time-linkage** refers to the influence of decisions made at present on future decisions. This is a crucial feature in many real-world scheduling and routing applications, for instance [13].
- The **predictability** specifies if future changes can be forecasted based on data patterns or algorithmic insights.
- The **detectability** determines whether changes are detectable by the optimisation process (e.g. detectors).
- The **cyclicity** is associated with the recurrence of changing environments, meaning that the algorithm may have already explored future environments.
- Different **components of the problem** change during the optimisation process, such as the objective function, problem instance, and number of decision variables. This dynamic feature also includes changes in dimensionality, fitness landscapes, and the presence of multi-objective and constrained problems.
- The **homogeneity** is related to the consistency of changes, such as changes with uniform frequency (periodic) and magnitude over time. That is, in homogeneous environments, all parts of the landscape change equally, whereas in heterogeneous environments, different regions of the fitness landscape that change independently, leading to more complex changes.

Literature typically identifies DOPs as having the following dynamic features [8, 42]: non-time-linkage, unpredictability, detectability, non-constraint, and variations in the objective function or problem instance. Cyclicity is also examined, although it is less common for general benchmark generators.

Additionally, based on the dynamic features of changes, several classification methods have been developed to capture the dynamic nature of DOPs, and to taxonomise the adaptive challenge of online algorithms to solve DOPs [3, 34]. The classification methods that can be summarised as follows:

- Eberhart and Shi [74] proposed a framework based on the **change direction**, which indicates whether the encoding of solutions, their objective value, or both, are altered by problem changes.

<div style="writing-mode: vertical-rl; transform: rotate(180deg);"> FREQUENCY + ↑ - </div>	Progressive (frequent-slight)	Chaotic (frequent-severe)
	Quasi-static (occasional-slight)	Abrupt (occasional-severe)
	<div style="text-align: center;"> - → + MAGNITUDE </div>	

Figure 2.3: Classes of problems based on the frequency and magnitude of change, according to the authors in [73].

- Angeline [75] focused on the **trajectory of moving optima**, distinguishing between linear, cyclic, or random changes in the continuous domain. Specifically, linear changes refer to the constant displacement of the optima, cyclic changes follow a circular movement of the optima, and random changes introduce noise, making the movement of the optima unpredictable.
- Weicker [76, 77] extended the classifications from Eberhart and Shi [74] and Angeline [75] by introducing the **homogeneity** feature.
- De Jong [78] introduces a real-world oriented classification system for DOPs primarily focused on the severity of changes. Duhain and Engelbrecht [73] extend this idea, and suggest a classification based on both the frequency and magnitude of changes. The **frequency** and **magnitude** of changes have been used to define how often changes occur and how significant those changes are, respectively. Problems with frequent but minor changes, termed quasi-static, are easier to manage than those with infrequent but drastic alterations, referred to as abrupt or chaotic environments. Figure 2.3 displays the distinction between the classes presented by Duhain and Engelbrecht, organised by the frequency and magnitude of the changes. Note that the divisions between classes are not shown, as the adaptive challenge of a problem depends on the algorithm used to solve it.
- Younes *et al.* [4] added another layer of complexity by distinguishing between dimensional and non-dimensional changes. Specifically, the **dimensionality** of

problem changes reflects whether the representation of solutions is altered, such as adding or removing of decision variables. This feature significantly increases the adaptive challenge of DOPs to online algorithms, since dimensional changes may affect the representation of the solution space, generally pose a greater challenge than non-dimensional changes, which only modify problem parameters or constraints.

Furthermore, Li and Yang [60] combine many of the previous classification concepts, and present a generalised benchmark generator that distinguishes between six types of DOPs, including small, large, random, chaotic, recurrent, and noisy recurrent changes.

2.2.2.2 Developed Benchmark Generators

The literature presents many different benchmark generators to simulate changing problems by adding sequential modifications to an initial static instance. In the following section, the benchmark generators used in this thesis are described in detail. Specifically, the fitness landscape rotation, two different dynamic TSP-based benchmarks, and the synthetic data generation as benchmark generators are covered.

2.2.2.2.1 Fitness Landscape Rotation

The *fitness landscape rotation* has been probably the most popular benchmark generator in the combinatorial domain for academic purposes because of its simplicity and ability to preserve important properties of the problem instance [4, 11, 12, 69, 79].

Introduced as the XOR DOP generator [11, 12], this method periodically applies the rotation operation to alter the mapping between solutions and objective values using the exclusive OR (rotation) operator. Formally, given a static binary problem, a change magnitude ρ and a change frequency τ , the mapping between a solution $x \in \Omega$ and its objective value $f(x)$ is *rotated* as follows:

$$f_t(x) = f(x \oplus M_t), \quad (2.6)$$

where f_t is the objective function at instance $t = \lceil \frac{i}{\tau} \rceil$, i is the iteration of the search process of the algorithm, f is the original (static) objective function, “ \oplus ” is the exclusive-OR operator and $M_t \in \Omega$ is a binary mask. The mask M_t is incrementally generated by $M_t = M_{t-1} \oplus T$. Here, T is a binary string randomly generated containing $\lfloor \rho \times n \rfloor$ number of ones, where n represents the size of the problem. The initial mask is a zero vector, $M_1 = \{0\}^n$.

According to Tinós and Yang [10, 80], the XOR DOP generator permutes the problem in a special way, so important problem properties remain the same over time, i.e. the number and quality of optima or the neighbourhood relations between solutions are preserved.

Some works in the literature extended the XOR DOP generator to other spaces [4, 60, 69, 81]. Li and Yang [60] developed a generalised dynamic benchmark generator to build DOP instances in binary, real, and combinatorial spaces. Similarly, Younes *et al.* [4] presented a generalised benchmark generator (GBG) that modifies the encoding of the problem instance rather than rotating the mapping of the objective value of each solution. The authors encode the sequence of static optimisation problem instances as follows:

$$S = \{(I_t, x_t), t \in (1, k)\}, \quad (2.7)$$

where $k > 1$ denotes the sequence size (indicating $k - 1$ changes), I_t is the problem instance at the change period t , and x_t represents the optimal solutions at that period. Instances are generated incrementally:

$$I_t = I_{t-1} \oplus \Delta_t, \quad (2.8)$$

where Δ_t is the environmental shift variable applied to the problem instance. Note that I_1 is the initial (static) problem instance. The change magnitude $\rho \in (0.0, 1.0]$ indicates the total number of elementary operations for creating Δ_t , where $\lceil n \times \rho \rceil$ is the number of exchanges applied to the mapping function.

Mavrovouniotis *et al.* [69] comment that rotating the permutation space at $\rho = 0.5$ using GBG might reorder all the elements (swap half of the variables with the other half), resulting in a more severe change than intended. Therefore, the authors extended

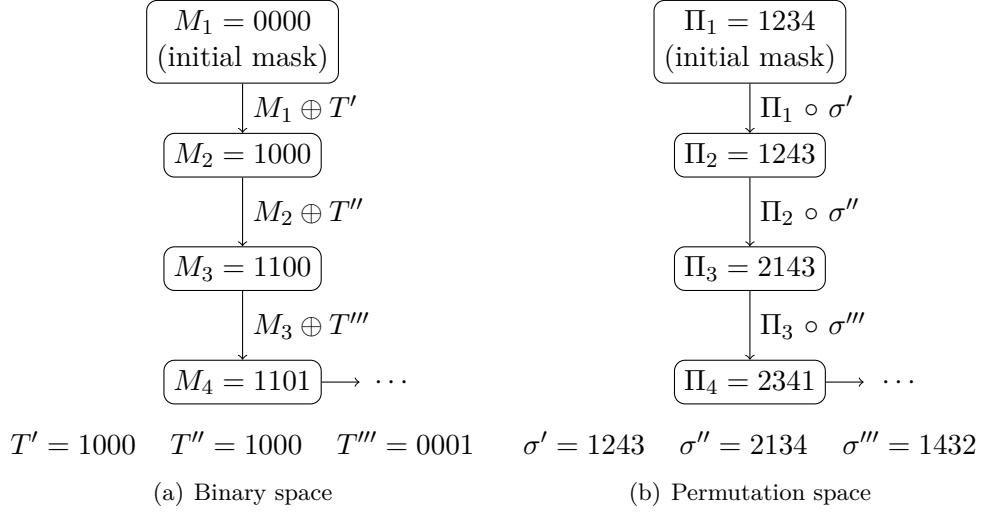


Figure 2.4: Visual representation of the fitness landscape rotation with a change severity of $\rho = 0.25$ in binary and permutation spaces for problems of size $n = 4$. Each template (T and σ) is generated at the minimum distance for the given metrics: Hamming distance $d_{min}^H = 1$ for the binary space, and Cayley distance $d_{min}^C = 1$ for the permutation space. These operations correspond to a bit-flip in the binary space, and a pairwise swap in the permutation space.

the GBG to the permutation space, specifically focusing on the DTSP and its variants, by rotating the fitness landscape through variable swaps.

In order to illustrate and clarify the fitness landscape rotation, Figure 2.4 illustrates the generation of incremental masks after rotation operations in both binary and permutation spaces for the initial three changes. In the given example, random templates are uniformly generated based on ρ using Hamming distance for the binary space, and Cayley distance for the permutation space.

2.2.2.2.2 Dynamic Travelling Salesperson Problem with Traffic

A benchmark generator that constructs a Dynamic TSP instances with traffic factor can be formulated as follows [35]:

$$f_t(\sigma) = d_t(\sigma_1, \sigma_2) + \sum_{i=1}^{n-1} d_t(\sigma_i, \sigma_{i+1}), \quad (2.9)$$

where $d_t(x, y)$ is the distance between cities x and y for the instance t . Note that the search space Ω remains constant, whereas the varying objective function f_t is incrementally modified as follows:

$$d_t(x, y) = \begin{cases} d(x, y) + r, & \text{if } (x, y) \in W_t, \\ d_{t-1}(x, y), & \text{otherwise,} \end{cases} \quad (2.10)$$

where $d(x, y)$ is the original distance between cities x and y , $d_t(x, y)$ is the distance between cities x and y in problem instance t , $r \sim N(0, d(x, y))$ is a normally distributed random variable, and W_t is a set of randomly selected arcs for change period t . For asymmetric DTSPs, the number of modified arcs in W_t is given by $\lceil n(n-1)\rho \rceil$, where ρ is the change magnitude, and $n(n-1)$ is the total number of connections between cities. In symmetric instances, this quantity is halved to $\lceil \frac{n(n-1)}{2}\rho \rceil$, since $d_t(x, y) = d_t(y, x)$.

2.2.2.2.3 Dynamic Travelling Salesperson Problem with City Replacement

The authors in [82] introduce a benchmark generator for creating DTSP instances with city replacement, where specific cities are swapped over time. Formally, given a set of cities $V = N_{in} \cup N_{out}$, with N_{in} and N_{out} being two subsets of $\frac{n}{2}$ vertices each, the generator replaces $\lceil \rho \times \frac{n}{2} \rceil$ vertices, randomly selected, in N_{in} with an equal number from N_{out} every τ iterations, where n is the problem size, and τ and ρ indicate the frequency and magnitude of changes, respectively. Similarly, as DTSPs can be modelled as a weighted graph $G = (N_{in}, E)$, the weights (distances) in E are also changed.

2.2.2.2.4 Synthetic Data Generation

Synthetic data generators are crucial to balance data protection regulations and the needs of researchers to work on real-world applications [83]. The application of synthetic data is growing rapidly in many fields, such as medical research or finance, or simply for data privacy [84]. According to James *et al.* [85], synthetic data is expected to surpass real-world data usage across multiple areas, such as Artificial Intelligence (AI) and machine learning applications. To give some relevant examples, Amazon generates large amounts of synthetic data to train the voice recognition algorithm used for Alexa, while researchers employ models for text, images, and voice recognition, among other uses [86]. Gartner [87] predicts that, by 2030, most of the data used in AI will be artificially generated by rules, statistical models, simulations or other techniques.

2.3 Summary

Chapter 2 achieves the research objective **OB 1** by systematically reviewing the literature on definitions, dynamic features, and methods in dynamic optimisation. This allows for a clear distinction between dynamic optimisation problems, where algorithms adapt over time, and unrelated static optimisation problems that change without similarity, where restarting may be efficient.

The literature on DOPs is primarily organised into four research streams: (i) definition and classification of DOPs based on their dynamic features, (ii) development for dynamic benchmarks that replicate dynamism, (iii) application of adaptive mechanisms to existing algorithms, and (iv) development of performance metrics to assess and quantify algorithmic performance.

This section provides a detailed overview of the literature in combinatorial and dynamic optimisation to contextualise the contributions of this thesis. Specifically, the section has highlighted the following research gaps:

- Despite the wide use of benchmark generators to simulate problem changes and compare algorithmic performance, academic research is often limited to two features of DOPs: the frequency and magnitude of changes [7]. Specifically, benchmark generators based on fitness landscape rotation have been widely used in the combinatorial space to construct DOPs. However, theoretical analysis of the preservation of the landscape structure and the neighbourhood relations, beyond the binary space, are still lacking [10]. Furthermore, to the best of our knowledge, the repercussion resulting from fitness landscape rotation in the permutation space has not been thoroughly investigated, beyond the intended impact with respect to the magnitude and frequency of changes. That is, it is generally assumed that a small rotation results in a minor perturbation in terms of rearrangement of solutions in the fitness landscape [4, 69]. However, this thesis demonstrates that this assumption is not necessarily valid.

- The literature presents different definitions and classification strategies for DOPs, and many of them consider the performance of online algorithms. Nevertheless, there is still the need to accurately represent the adaptive challenge of online algorithms for solving DOPs with different dynamic features.
- To the best of our knowledge, using a synthetic data generator to replicate dynamic benchmark problem instances is extremely limited or non-existent. In fact, specific dynamic benchmarks may repel the general reader due to their problem-specific nature, which complicates the generalisation of benchmark problems [4]. Nevertheless, the access and consideration of realistic applications is still a limitation in the field of dynamic optimisation [9, 68].

Part II

Contributions

Chapter 3

Fitness Landscape Rotation for Dynamic Optimisation Problem Generation and Perturbation Strategy

In the field of dynamic optimisation, researchers often design generalised benchmark problems for algorithm development in controlled changing environments. These generators create a sequence of problem instances from an existing static optimisation problem by incrementally introducing regulated changes with adjustable parameters, such as frequency and magnitude of changes.

As presented in Section 2.2.2.2.1, the fitness landscape rotation has been widely used in academia as a benchmark generator for producing a sequence of problem instances due to its simplicity and ability to retain the properties of the initial static problem instance [4, 10, 11, 12, 69, 79]. In short, this method artificially constructs a concatenated homogeneously changing problem by periodically altering the mapping between solutions and objective values over time, thus relabelling solutions encoding in the fitness landscape while maintaining its structure. According to [10], the fitness landscape rotation in the binary space progressively permutes the initial problem,

preserving important properties of the problem, such as the structure of the fitness landscape.

The popularity of this method comes from its simplicity, but especially from its ability to preserve important properties of the fitness landscape (or problem instance), such as the number and quality of the optima or the neighbourhood relations among the solutions. However, despite its popularity, the utility of this technique is questioned, since the fitness landscape rotation is not a dynamism that can be observed in real-world situations [69]. In fact, a theoretical analysis of the preservation of the structure of the fitness landscape and a further study of the applications of this operation in combinatorial optimisation problems beyond binary space are still lacking [10].

This chapter is a combination of some research outputs produced in this thesis [79, 81, 88]. Specifically, the contributions of this chapter are the following. First, we extend the formulation of the fitness landscape rotation to the permutation space by introducing permutation distance metrics to precisely calibrate the intended and resulting magnitude of rotations in the fitness landscape. Second, we provide algebraic foundations to examine the preservation of the neighbourhood of solutions, even when the fitness landscape is rotated, and to capture the repercussion of rotations in terms of the exchange of solutions between different attraction graphs. Third, from the theoretical insights gained, we experimentally investigated different ways to employ the fitness landscape rotation to perturb the algorithm search of local search algorithms. Specifically, two rotation-based strategies are presented and applied to different local search algorithms to study the applicability and exploratory profit of these strategies on different permutation problems.

The remainder of the chapter is structured as follows. Section 3.1 introduces a fitness landscape rotation method that employs permutation distance metrics to measure rotation magnitude. A thorough description of some of its properties using group theory and graph theory notions is provided in Section 3.2. Section 3.3 examines the fitness landscape rotation as a perturbation strategy for local search algorithms, and

presents two rotation-based algorithms that are studied in the experimentation. Section 3.4 describes the experimental study, and discusses the applicability of the fitness landscape rotation from the observed results. Finally, Section 3.5 concludes the chapter with a summary of insights from the theoretical analysis and experimental study.

3.1 Introduction to the Fitness Landscape Rotation

In [81], we formally define the fitness landscape rotation in the permutation space using the composition operation as follows:

$$f_t(\sigma) = f(\Pi_t \circ \sigma), \quad (3.1)$$

where $\sigma \in \mathbb{S}_n$ is a solution, “ \circ ” is the composition operation between permutations and Π_t is a permutation mask. The permutation mask is incrementally generated by $\Pi_t = \Pi_{t-1} \circ \pi$, where π is a permutation generated using the methods¹ in [24], containing $d = \lceil d_{max} \times \rho \rceil$ operations from the identity permutation given a permutation distance. The permutation mask is initialised as the identity permutation, $\Pi_1 = e$.

Furthermore, we have highlighted that swaps between variables in the permutation domain can be accurately measured using the Cayley distance (as specified in Section 2.1.2.1). This helps to reduce the chances of creating unintended rotations, as noted by Mavrovouniotis *et al.* [69]. Additionally, other permutation distance metrics can be used to create different types of operations besides swaps (see Section 2.1.2.1 for more details). Specifically, the number of operations needed to rotate an instance (which is referred to as *rotation degree*) depends on the maximum and minimum distances of the chosen metric and the magnitude of changes ρ . Formally, for the permutation distance metrics we considered, the rotation degree can be expressed as follows:

¹In short, the methods in [24] for uniformly generating random permutations at a specified distance d using Cayley, Hamming, and Kendall’s- τ distance metrics involve the following steps: (i) calculating the total number of permutations at the specified distance, (ii) applying a distance-based decomposition vector specific to each metric to generate valid permutations, and (iii) uniformly selecting from these permutations to ensure equal sampling probability.

$$d = \begin{cases} \lceil d_{max}^K \times \rho \rceil \in \{1, \dots, \binom{n}{2}\}, & \text{Kendall's-}\tau \ (d^K), \\ \lceil d_{max}^H \times \rho \rceil \in \{2, \dots, n\}, & \text{Hamming} \ (d^H), \\ \lceil d_{max}^{\{C,U\}} \times \rho \rceil \in \{1, \dots, n-1\}, & \text{Cayley} \ (d^C) \text{ or Ulam} \ (d^U), \end{cases}$$

where d_{max} is the maximum distance of a given metric and n is the size of permutations.

3.1.1 Group Properties in Fitness Landscape Rotation

The fitness landscape rotation can be represented by group actions, where both the search space of solutions and the rotation operation satisfy specific properties. Formally, given a finite set of solutions Ω and a group operation “ \cdot ”, $G = (\Omega, \cdot)$ is a group if the closure, associativity, identity, and invertibility properties are satisfied. Mathematically, these fundamental group properties (axioms) are defined as:

- **Closure:** $x, y \in G, x \cdot y \in G$.
- **Associativity:** $x, y, z \in G, (x \cdot y) \cdot z = x \cdot (y \cdot z)$.
- **Identity:** $\exists! e \in G, \forall x \in G, x \cdot e = e \cdot x = x$.
- **Invertibility:** $x, x^{-1} \in G, x \cdot x^{-1} = x^{-1} \cdot x = e$.

There is another property, the **commutativity**, that is fundamental for the definition of *Abelian* groups. Formally, the commutation of two elements x and y of a group G exists when $x \cdot y = y \cdot x$. It is worth mentioning that the commutation property holds in the binary space, but it generally does not in the permutation space. Based on the example shown in Figure 2.1, for the permutations $\sigma = 3142$ and $\pi = 1324$, and the composition operation², we have $\sigma \circ \pi = 3412$ and $\pi \circ \sigma = 2143$, demonstrating that the composition between permutations is not commutative.

3.2 Analysis of the Fitness Landscape Rotation

Many authors have found it practical to study features of the fitness landscape, since they seem to condition the behaviour of the algorithms [22, 89, 90, 91]. Abstractly, these characteristics affect the geometric properties of the fitness landscape, such as

²The symmetric group, (\mathbb{S}_n, \circ) , uses permutations and the composition operator.

the arrangement of solutions among attraction graphs. Attraction graphs can be represented as tree-like directed acyclic graphs that lead to a local optimum, where vertices represent solutions and edges indicate steepest-ascent movements to neighbouring solutions (see Definition 2.1 for a formal definition of attraction graphs).

To the best of our knowledge, there are no studies that thoroughly studied the fitness landscape rotation in permutation space, similar to the study done by Tinós and Yang did in the binary space [10, 80]. This section analyses the fitness landscape rotation in the combinatorial domain, specifically within the permutation space, by analysing the preservation of the neighbourhood, the isomorphism of the attraction graphs that compose the fitness landscape, and the effects of rotation operations on the fitness landscape are examined.

Without loss of generality, the study uses notations from Sections 2.1.1 and 3.1.1 to demonstrate its validity with proofs and examples. Additionally, the permutation space (S_n), the swap operation (i.e. the 2-exchange operator) and the steepest-ascent hill climbing algorithm (saHC) are considered to represent the fitness landscape, and more precisely, the set of attraction graphs.

3.2.1 Neighbourhood Preservation

It is worth noting that the neighbourhood of a solution may not belong to the same attraction graph. In the following, we aim to algebraically demonstrate that the neighbourhood of solutions is preserved under any rotation degree. Specifically, we represent the fitness landscape rotation as group actions and the search space of solutions as a subset of a group G .

Definition 3.1 (Neighbourhood operator). *Given a group G , a neighbourhood operator s is an element of the set $S \subseteq G$ that defines transformations for solutions within the search space. The identity element e is included in S , and for every operator $s \in S$, there exists an inverse $s^{-1} \in S$ ensuring invertibility. Therefore, the neighbourhood of a solution $x \in G$ is defined as $\mathcal{N}(x) = \{s \cdot x, x \cdot s : s \in S\}$, where $s \in S$ acts as a neighbourhood operator on x , and the operation “ \cdot ” denotes the group operation.*

Without loss of generality, we apply the neighbourhood operator with the left group operation, i.e. $s \cdot x \in \mathcal{N}(x)$. For Abelian groups, this action is trivial due to the commutativity property, meaning $s \cdot x = x \cdot s$.

Definition 3.2 (Preservative set of neighbourhood operators). *Let S be a set of neighbourhood operators. We say that S is **preservative** iff for any solution $g \in G$ and any neighbourhood operator $s \in S$, there exist operators $s', s'' \in S$ such that*

$$s' \cdot g \cdot s = g = s \cdot g \cdot s''. \quad (3.2)$$

It is worth noting that the property in Equation 3.2 is unique because it applies to Abelian groups and pairwise swaps in the permutation space, but may not hold for adjacent swaps and insertions (refer to Section 2.1.2.1 for a detailed description of these fundamental operations in the permutation space). Let us illustrate it with the following example.

In order to demonstrate that S is preservative, we must show that there exist $s, s', s'' \in S$, such that $s' \cdot g \cdot s = g = s \cdot g \cdot s''$. Consider the permutation $\sigma = 3241$ and the neighbourhood operator $s = 1324$; then, $s' = 1432$ and $s'' = 2134$. Let us examine the property in Equation 3.2 under pairwise swaps, adjacent swaps, and the insertion operations.

First, let us examine pairwise swaps as the operation of the neighbourhood function. The cardinality (size) of the set of neighbourhood operators under pairwise swaps is $|S| = \binom{n}{2} + 1$, where n is the size of the solutions. For permutation problems of size $n = 4$, the set of neighbour operators is $S = \{1234, 2134, 1324, 1243, 3214, 1432, 4231\}$. Therefore, since $s, s', s'' \in S$, we prove that S is preservative under pairwise swaps.

Next, consider adjacent swaps as the operation of the neighbourhood function. The cardinality of the set of neighbour operators under adjacent swaps is $|S| = n$, such that $S = \{1234, 2134, 1324, 1243\}$ for permutation problems of size $n = 4$. Since $s' \notin S$, we can state that the set of neighbourhood operators S is not preservative under adjacent swaps.

Finally, we analyse the insertion operation as the neighbourhood function. The cardinality of the set of neighbour operators under insertions is $|S| = (n - 1)^2 + 1$.

The set of neighbourhood operators for permutation problems of size $n = 4$ is $S = \{1234, 2134, 2314, 2341, 1324, 1342, 3124, 1243, 4123, 1423\}$. Since $s' \notin S$, we can conclude that the neighbourhood set S is not preservative under insertions.

Theorem 3.1. *Let G be a group, $\mathcal{N}(x) \subset G$ the neighbourhood of $x \in G$ generated by a preservative set of neighbourhood operators S , and $t \in G$ a mask used to rotate the fitness landscape. Then, the neighbourhood of solutions is preserved if $\mathcal{N}(t \cdot x) = t \cdot \mathcal{N}(x)$.*

Proof. Let $x, y \in G$ be neighbouring solutions that satisfy the symmetric property of the neighbourhood relation, i.e. $y \in \mathcal{N}(x)$ and $x \in \mathcal{N}(y)$. Let $S \subset G$ be a preservative set of neighbourhood operators.

In order to prove that the rotation of the fitness landscape preserves the set of neighbour solutions, we need to prove that $\mathcal{N}(t \cdot x) \subset t \cdot \mathcal{N}(x)$ and $t \cdot \mathcal{N}(x) \subset \mathcal{N}(t \cdot x)$.

First, let $y \in \mathcal{N}(t \cdot x)$ be a solution in the neighbourhood of the rotated $t \cdot x$. By definition, there exists a neighbourhood operator $s' \in S$ such that $y = s' \cdot t \cdot x$. Since S is preservative (Definition 3.2), there is a $s \in S$ satisfying $s' \cdot t \cdot s = t$, and a $s'' \in S$ such that $s \cdot x \cdot s'' = x$. Thus, we have $y = s' \cdot t \cdot s \cdot x \cdot s'' = t \cdot x \cdot s''$. Since $s'' \in S$ and according to Definition 3.1, we can prove that $y \in t \cdot \mathcal{N}(x)$.

Now, let $y \in t \cdot \mathcal{N}(x)$ be a solution in the rotated neighbourhood of x . Based on the neighbourhood operator definition (Definition 3.1), we have $y = t \cdot s \cdot x$. Applying $s' \in S$ to both sides results in $s' \cdot y = s' \cdot t \cdot s \cdot x$. Since S is preservative (Definition 3.2), there exists an $s \in S$ such that $s' \cdot t \cdot s = t$, allowing us to simplify the equation to $s' \cdot y = t \cdot x$. Since $s' \in S$, it follows that $t \cdot x \in \mathcal{N}(y)$, according to Definition 3.1. Thus, using the symmetry definition of the neighbourhood, we prove that $y \in \mathcal{N}(t \cdot x)$.

Thus, we have proven that the neighbourhood of solutions is preserved after the fitness landscape rotation. Note that the preservation of the neighbourhood is independent of the algorithm and objective function. \square

Example 3.2. *Let us consider a permutation $\sigma = 3241$, the rotation mask $\Pi_t = 1243$, the composition operation “ \circ ” as the group operation, and S a preservative set of neighbourhood operators under pairwise swaps.*

To demonstrate that $\mathcal{N}(\Pi_t \circ \sigma) \subset \Pi_t \circ \mathcal{N}(\sigma)$, we proceed with the following steps:

- The permutation $\gamma \in \mathcal{N}(\Pi_t \circ \sigma)$ is in the neighbourhood of the rotated σ . Since $\Pi_t \circ \sigma = 1243 \circ 3241 = 4231$, we conclude that $\gamma \in \mathcal{N}(4231)$.
- Using the pairwise swap neighbourhood operator $s' = 2134$ and the Definition 3.1, we define $\gamma = s' \circ \Pi_t \circ \sigma = 2134 \circ 1243 \circ 3241 = 4132$.
- In order to satisfy the preservative property in Equation 3.2, we identify two neighbourhood operators, $s = 2134$ and $s'' = 1432$. Specifically, we demonstrate that preservation holds for $s, s'' \in S$ as follows:
 - Given $s, s' \in S$ where $s' \circ \Pi_t \circ s = \Pi_t$, we have $2134 \circ 1243 \circ 2134 = 1243$.
 - Given $s, s'' \in S$ where $s \circ \sigma \circ s'' = \sigma$, we find $2134 \circ 3241 \circ 1432 = 3241$.
- Therefore, since $4132 = 1243 \circ 3241 \circ 1432$, we confirm that $\gamma \in \Pi_t \circ \mathcal{N}(\sigma)$ holds under the specified neighbourhood operators $s, s', s'' \in S$.

Next, in order to prove $\Pi_t \circ \mathcal{N}(\sigma) \subset \mathcal{N}(\Pi_t \circ \sigma)$, we follow these steps:

- The permutation $\gamma \in \Pi_t \circ \mathcal{N}(\sigma)$ is in the neighbourhood of $\sigma = 3241$.
- Using the operator $s = 2134$, we define $\gamma = \Pi_t \circ s \circ \sigma = 1243 \circ 2134 \circ 3241 = 4132$.
- By applying $s' = 2134$ to both sides of the equation, we prove that $s' \circ \gamma = \Pi_t \circ \sigma$, leading to $2134 \circ 4132 = 1243 \circ 3241 = 4231$.
- Therefore, we confirm that $\gamma \in \mathcal{N}(\Pi_t \circ \sigma)$ holds under $s, s' \in S$.

3.2.2 Preservation of the Structure

The fact that the neighbourhood of solutions remains the same after being rotated leads us to investigate the preservation of the structure of the fitness landscape. It is worth noting that the fitness landscape rotation alters the mapping of solutions to objective values (see Equation 3.1). This leads to the relabelling of solutions (vertices) in the fitness landscape. Hence, since the relabelling of solutions does not change the topology of the fitness landscape, we can say that the structure of the fitness landscape remains consistent after a rotation [10, 69, 92].

Using the notations in Section 2.1.1, we represent the fitness landscape as a set of attraction graphs $O_f = \bigcup_{x^* \in \mathcal{B}(x^*)} \mathcal{G}_f(x^*)$, where each attraction graph $\mathcal{G}_f(x^*) = (V_f, E_f)$ consists of the following:

- $V_f \subseteq \Omega$ is the set of solutions within the attraction basin of a local optimum x^* .
- E_f is the set of directed edges representing transitions between solutions and neighbouring solution based on their objective values.

When the fitness landscape is rotated by $t \in G$, the rotated fitness landscape results in a corresponding set of attraction graphs $O_{f_t} = \bigcup_{x^* \in \mathcal{B}(x^*)} \mathcal{G}_{f_t}(x^*)$.

Informally, we can point out that attraction graphs are *isomorphic* (structurally equivalent) to themselves in the rotated environment, such that $O_f \cong O_{f_t}$, where O_{f_t} is the set of attraction graphs that composes the rotated fitness landscape f_t . The following theorem employs algebraic terminology to precisely describe graph isomorphism.

Theorem 3.3. *Let (G, f, S) be a group G , an objective function $f : G \rightarrow \mathbb{R}$, and a preservative set of neighbourhood operators S . Let \mathcal{N} be the neighbourhood function defined by S . Then, for any element $t \in G$ representing a landscape rotation, the fitness landscapes O_f and O_{f_t} are isomorphic in the following sense:*

- (i) *there is a bijection between the local optima of f and f_t ,*
- (ii) *there is a graph isomorphism between the attraction graphs $\mathcal{G}_f(x^*)$ and $\mathcal{G}_{f_t}(t^{-1} \cdot x^*)$ for each local optimum x^* of f .*

Proof. (i) First, we will prove that there is a bijection between the local optima of f and f_t . Note that the rotated fitness function is defined as $f_t(x) = f(t \cdot x)$, where t is a rotation applied uniformly to all solutions in the group G .

Let x^* be a local optimum of f , meaning that for all its neighbours $y \in \mathcal{N}(x^*)$, it holds that $f(y) \leq f(x^*)$. We must prove that $z^* = t^{-1} \cdot x^*$ is a local optimum of f_t , such that for all neighbours $z \in \mathcal{N}(z^*)$, $f_t(z) \leq f_t(z^*)$.

In order to prove this, we use the neighbourhood preservation (Theorem 3.1), which states that $\mathcal{N}(z^*) = \mathcal{N}(t^{-1} \cdot x^*) = t^{-1} \cdot \mathcal{N}(x^*)$. Therefore, every neighbour $z \in \mathcal{N}(z^*)$ can be written as $z = t^{-1} \cdot y$, for some $y \in \mathcal{N}(x^*)$.

By the definition of fitness landscape rotation and the invertibility property of groups, we have $f_t(z) = f_t(t^{-1} \cdot y) = f(t \cdot t^{-1} \cdot y) = f(y)$. Since x^* is a local optimum of f , it follows that $f(y) \leq f(x^*)$, where $f(x^*) = f(t \cdot t^{-1} \cdot x^*) = f_t(t^{-1} \cdot x^*) = f_t(z^*)$. Thus, we establish $f_t(z) \leq f_t(z^*)$, proving that z^* is a local optimum of f_t .

Similarly, suppose z^* is a local optimum of f_t . We claim that $x^* = t \cdot z^*$ is a local optimum of f , meaning that for all $y \in \mathcal{N}(x^*)$, $f(y) \leq f(x^*)$.

Following the same steps as above, by using Theorem 3.1, we find that $\mathcal{N}(x^*) = \mathcal{N}(t \cdot z^*) = t \cdot \mathcal{N}(z^*)$. This implies that, for each neighbour $y \in \mathcal{N}(x^*)$, there exists $y = t \cdot z$ such that $z \in \mathcal{N}(z^*)$. Therefore, by computing $f(y) = f(t \cdot z) = f_t(z) \leq f_t(z^*) = f(t \cdot z^*) = f(x^*)$, we demonstrate that x^* is a local optimum of f .

Finally, let $L_f = \{x^* \in G : \forall y \in \mathcal{N}(x^*), f(y) \leq f(x^*)\}$ and $L_{f_t} = \{z^* \in G : \forall y \in \mathcal{N}(z^*), f_t(y) \leq f_t(z^*)\}$ denote the sets of local optima for f and f_t , respectively. We define the maps $t_f^* : L_f \rightarrow L_{f_t}$ and $t_{f_t}^* : L_{f_t} \rightarrow L_f$ by $t_f^*(x^*) = t^{-1} \cdot x^*$ and $t_{f_t}^*(z^*) = t \cdot z^*$. These maps form inverse bijections between the local optima sets L_{f_t} and L_f .

(ii) Next, we will demonstrate that the attraction graphs $\mathcal{G}_f(x^*) = (V_f, E_f)$ and $\mathcal{G}_{f_t}(t^{-1} \cdot x^*) = (V_{f_t}, E_{f_t})$ are isomorphic. Specifically, we show that the inverse actions of t and t^{-1} on G induce graph isomorphisms between these attraction graphs.

From (i), we know that $x^* \in L_f$ and $t^{-1} \cdot x^* \in L_{f_t}$ are local optima related by the bijection $t_f^*(x^*) = t^{-1} \cdot x^*$. We extend this bijection to define a map $\phi_f^t : V_f \rightarrow G$, where $\phi_f^t(a) = t^{-1} \cdot a$ for $a \in V_f$.

In particular, we will demonstrate that $\phi_f^t : V_f \rightarrow V_{f_t}$ and the product $\phi_f^t \times \phi_f^t : E_f \rightarrow E_{f_t}$ form a morphism of graphs $\phi_f^t : \mathcal{G}_f(x^*) \rightarrow \mathcal{G}_{f_t}(t^{-1} \cdot x^*)$. Then, considering the inverse morphism $\phi_{f_t}^{t^{-1}} : \mathcal{G}_{f_t}(t^{-1} \cdot x^*) \rightarrow \mathcal{G}_f(x^*)$, we will establish the graph isomorphism.

First, in order to ensure that ϕ_f^t defines a graph morphism, we show that the objective value assigned to each vertex is preserved under rotation. That is, for any $a \in V_f$, we have $f_t(\phi_f^t(a)) = f_t(t^{-1} \cdot a) = f(t \cdot t^{-1} \cdot a) = f(a)$.

Now, consider a path in $\mathcal{G}_f(x^*)$, represented as a sequence of vertices $x = a_0, a_1, \dots, a_h = x^*$, where for $0 \leq i < h$, $a_i \in V_f$, $a_{i+1} \in \mathcal{N}(a_i)$, $f(a_i) < f(a_{i+1})$, and (a_i, a_{i+1}) forms an edge in E_f . Note that $\forall a \in \mathcal{N}(a_i), f(a) \leq f(a_{i+1})$.

By applying ϕ_f^t and Theorem 3.1, we get $\phi_f^t(a_{i+1}) = t^{-1} \cdot a_{i+1} \in t^{-1} \cdot \mathcal{N}(a_i) = \mathcal{N}(t^{-1} \cdot a_i) = \mathcal{N}(\phi_f^t(a_i))$. Additionally, if $a' \in \mathcal{N}(\phi_f^t(a_i))$, then $a' = t^{-1} \cdot a$ for some $a \in \mathcal{N}(a_i)$, leading to $f_t(a') = f_t(t^{-1} \cdot a) = f_t(\phi_f^t(a)) = f(a) \leq f(a_{i+1}) = f_t(\phi_f^t(a_{i+1}))$. Thus, it establishes that $(\phi_f^t(a_i), \phi_f^t(a_{i+1}))$ is an edge in E_{f_t} .

These arguments establish the existence of a path in $\mathcal{G}_{f_t}(x^*)$ such that $t^{-1} \cdot x = \phi_f^t(x) = \phi_f^t(a_0), \phi_f^t(a_1), \dots, \phi_f^t(a_h) = t^{-1} \cdot x^*$, with each pair $(\phi_f^t(a_i), \phi_f^t(a_{i+1}))$ representing edges for $0 \leq i < h$, where $f_t(\phi_f^t(a_i)) < f_t(\phi_f^t(a_{i+1}))$. Therefore, we prove that there exists a graph morphism from $\mathcal{G}_f(x^*)$ to $\mathcal{G}_f(t^{-1} \cdot x^*)$, defined by $\phi_f^t : V_f \rightarrow V_{f_t}$ and $\phi_f^t \times \phi_f^t : E_f \rightarrow E_{f_t}$.

Similarly, we define the map $\phi_{f_t}^{t^{-1}} : V_{f_t} \rightarrow G$, where $\phi_{f_t}^{t^{-1}}(a) = t \cdot a$. Following the same steps as above, we show that $\phi_{f_t}^{t^{-1}}$ preserves edges and objective values, inducing a graph morphism from $\mathcal{G}_{f_t}(t^{-1} \cdot x^*)$ to $\mathcal{G}_f(x^*)$.

Using the definitions, we demonstrate that these are inverse morphisms. For example, for any $a_i \in V_f$, $\phi_{f_t}^{t^{-1}} \circ \phi_f^t(a_i) = \phi_{f_t}^{t^{-1}}(t^{-1} \cdot a_i) = t \cdot t^{-1} \cdot a_i = a_i$. Thus, it is shown that

$$\begin{aligned}\phi_{f_t}^{t^{-1}} \circ \phi_f^t &= Id : V_f \rightarrow V_f, \\ \phi_f^t \circ \phi_{f_t}^{t^{-1}} &= Id : V_{f_t} \rightarrow V_{f_t}.\end{aligned}$$

Similarly, we establish that

$$\begin{aligned}(\phi_{f_t}^{t^{-1}} \circ \phi_f^t) \times (\phi_{f_t}^{t^{-1}} \circ \phi_f^t) &= Id : E_f \rightarrow E_f, \\ (\phi_f^t \circ \phi_{f_t}^{t^{-1}}) \times (\phi_f^t \circ \phi_{f_t}^{t^{-1}}) &= Id : E_{f_t} \rightarrow E_{f_t}.\end{aligned}$$

Therefore, since we have established a bijective mapping between vertices and edges, we conclude that the attraction graphs are isomorphic, $\mathcal{G}_f(x^*) \cong \mathcal{G}_{f_t}(t^{-1} \cdot x^*)$, completing the proof. \square

Example 3.4. Consider a permutation problem of size $n = 4$, illustrated in Figure 3.1(a). The figure shows the fitness landscape as a set of attraction graphs, where the vertices represent solutions (permutations) and directed edges indicate the transitions to the best neighbour (by pairwise swaps) when the saHC is applied. Figures 3.1(b) and 3.1(c) show the attraction graphs after rotating the original landscape by a single swap operation, resulting in $\Pi' = 1243$ and $\Pi'' = 2134$, respectively.

As can be observed from the figures, the vertices are relabelled across the fitness landscape after rotation, although the structure of the fitness landscape, the number and arrangement of the attraction graphs, and the number and objective values of local and global optima are preserved.

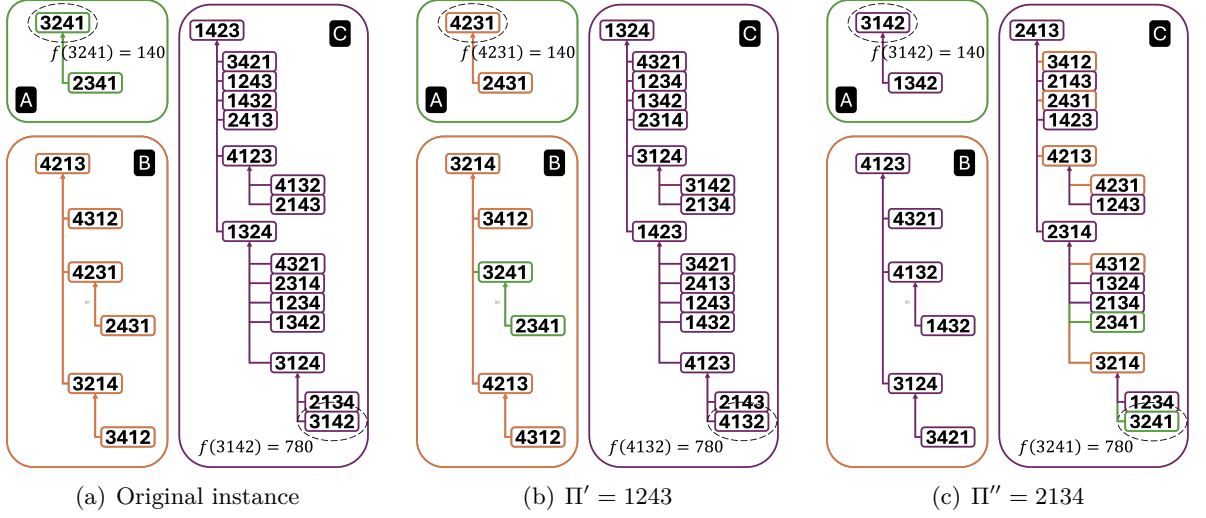


Figure 3.1: Fitness landscape (represented as a collection of attraction graphs) rotation example. (a) Fitness landscape of a permutation problem instance of size $n = 4$ under the swap neighbourhood operation. (b) Rotated fitness landscape of the original instance after swapping the variables 3 and 4. (c) Rotated fitness landscape of the original instance after swapping the variables 1 and 2. The coloured cells indicate the solutions exchanged between the attraction graphs with respect to the original instance.

Despite solutions are relabelled in different positions, the structure of the fitness landscape is preserved. However, the mapping of vertices to objective values remains unchanged. Therefore, the fitness landscapes before and after rotation, O_f and O_{f_t} , are equivalent in terms of objective values. Figure 3.1 shows that the local optimum in the attraction graph A has an objective value of $f(\sigma^*) = f_t(\Pi' \circ \sigma^*) = f_t(\Pi'' \circ \sigma^*) = 140$.

3.2.3 Repercussion of the Fitness Landscape Rotation

Although several works used the fitness landscape rotation in the permutation domain [4, 10, 69, 79], none studied the consequence of the generated dynamics, and considered that a large rotation degree is considered as a high perturbation in the fitness landscape. In particular, Tinós and Yang [10] mention that the topological structure of the fitness landscape and the neighbourhood relations must be analysed to comprehend the behaviour of algorithms.

Recent works in combinatorial optimisation have demonstrated that neighbouring solutions of a local optimum may belong to more favourable attraction graph [19, 21].

Table 3.1: Summary of solution exchanges following an example of fitness landscape rotation. This example highlights the number of solution exchanges between attraction graphs for all possible rotation masks generated by the Cayley distance metric, d_C .

$t (d_C)$	Exchanges	$t (d_C)$	Exc.	$t (d_C)$	Exc.
1234 (0)	0 (original)	1423 (2)	10	4213 (2)	13
1243 (1)	4	2143 (2)	16	4321 (2)	14
1324 (1)	8	2341 (2)	14	2314 (3)	12
1432 (1)	8	2431 (2)	14	2413 (3)	12
2134 (1)	16	3124 (2)	12	3142 (3)	12
3214 (1)	12	3241 (2)	13	3421 (3)	13
4231 (1)	14	3412 (2)	12	4123 (3)	14
1342 (2)	10	4132 (2)	14	4312 (3)	13

That is, even one movement from a local optimum is enough to explore other areas (in terms of attraction graph) of the fitness landscape. This hypothesis suggests that even the smallest rotation degree (i.e. swap between two variables) can have a significant impact on the restructuring of solutions within attraction graphs.

In order to measure the impact of the relabelling solutions in the fitness landscape after a rotation, we use the total number of solution exchanges between attraction graphs. Specifically, we refer to solution exchanges between attraction graphs as the process where solutions moves from one attraction graph to another. This idea is based on the assumption that, for local search-based algorithms, it is more likely to reach a different attraction graph when a rotation implies a significant number of solution exchanges between attraction graphs.

Let us illustrate the repercussion of the fitness landscape rotation using the example in Figure 3.1. For ease of understanding, we have highlighted in light blue the migrated solutions corresponding to Figure 3.1(a) in Figures 3.1(b) and 3.1(c). Furthermore, Table 3.1 summarises the total number of solution exchanges between attraction graphs for all possible rotations generated at a given Cayley distance (d_C).

Figures 3.1(b) and 3.1(c) show that all solutions are relabelled to different positions compared to Figure 3.1(a), although the number of solution exchanges between attraction graphs varies. That is, the swap between the items 3 and 4 of all solutions in the original instance (Figures 3.1(b)) only results in four solutions are relabelled to

different attraction graphs, i.e. the solutions 3241 and 2341 move to graph B , while 4231 and 2431 move to graph A . However, Figure 3.1(c) shows that swapping the variables 1 and 2 produces the maximum number of solution exchanges³, even though the fitness landscape is also rotated at minimum degree. Therefore, despite both rotations are considered as minor degree changes of the original instance, their impact on the relabelling of solutions in the fitness landscape is totally different.

The entries in Table 3.1 show that the rotation degree, measured by the Cayley distance, is not necessarily proportional to the number of solution exchanges between attraction graphs in the permutation space. By calculating the average number of exchanges for each Cayley distance in the table, we can deduce that, on average, rotating to $d_C = 1$ makes about 10 solution exchanges, $d_C = 2$ creates nearly 13 exchanges, and $d_C = 3$ produces nearly 12 exchanges, respectively. Therefore, for this particular instance, it is more likely that rotating to $d_C = 2$ will create more solution exchanges than rotating to $d_C = 3$.

It is worth noting that exhaustively calculating the number of local optima on medium to large size benchmark instances may be unfeasible due to the size of their search spaces. Therefore, drawn insights suggest caution in using the rotation degree in permutation space, since even low or medium rotations can result in a significant increase in the total number of exchanges of solutions between attraction graphs. Finally, it should be noted that the repercussion of changes can vary from instance to instance of the same problem.

3.3 Fitness Landscape Rotation as a Perturbation Strategy

The literature presents a wide variety of perturbation strategies to avoid getting trapped in a given attraction graph, as a local optimum in the original fitness landscape is not generally mapped to a local optimum in the rotated fitness landscape [19, 93]. We say that the algorithm is stuck when the search reaches an optimal solution and no improvement is obtained after comparing the entire neighbourhood.

³The maximum number of solution exchanges can be approximated to $|\Omega| \times (|O_f| - 1) / |O_f|$, where $|\Omega|$ is the size of the search space and $|O_f|$ represents the number of attraction graphs that compose the fitness landscape O_f .

Algorithm 3.1 saHC-R1: depth-first rotation strategy for perturbation

- 1: Let σ be a random permutation and e the identity permutation.
 - 2: Let d be a number within the boundaries of a given permutation distance metric.
 - 3: $best \leftarrow$ Best solution found by $\text{saHC}(e, \sigma)$.
 - 4: **repeat**
 - 5: $\Pi \leftarrow$ Uniformly at random permutation at distance d .
 - 6: $\sigma^* \leftarrow$ Best solution found by $\text{saHC}(\Pi, best)$.
 - 7: Update $best$ if σ^* improves it.
 - 8: Update d based on the chosen cooling scheme (e.g. linear, exponential, etc.).
 - 9: **until** Stopping criterion is met.
-

The theoretical insights gained from the previous section suggest that the fitness landscape rotation can be used as a perturbation strategy for local search-based algorithms to perturb the search of stuck algorithms (ideally) into a different attraction graph.

In short, the proposed rotation-based local search algorithms can be summarised as follows:

- 1: Run the local search algorithm until reaching a local optimum, x^* .
- 2: The rotation operation is applied to relocate the algorithm at the solution $t \cdot x^*$.
Ideally, the algorithm will reach a new local optimum, such that $t \cdot x^* \subseteq \mathcal{G}_{f_t}(y^*)$.
- 3: This process is repeated until the stopping criterion is met.

In particular, two rotation-based perturbation strategies are suggested: (i) a depth-first rotation strategy, where the already found optimum is compared with the obtained optimum on rotated environments; and (ii) a breadth-first rotation strategy that spends some time on the rotated space before returning to the original environment is designed. These methods differ in the way they use the rotation operation. Without loss of generality, the following algorithms are applied into permutation problems using saHC, although they may be extended to any other combinatorial problem and local search algorithm.

On the one hand, we design a depth-first rotation method (saHC-R1), analogous to the iterative local search (ILS) [93] or the variable neighbourhood search (VNS) [94]. Algorithm 3.1 shows the pseudocode of the depth-first rotation strategy applied into

saHC for any optimisation problem. The algorithmic details of this strategy are described in the following paragraph.

saHC-R1 starts the search from a random solution σ , and continues to improve its quality by the local search algorithm, $\text{saHC}(e, \sigma)$, until a local optimum is reached, $best$. The function $\text{saHC}(e, \sigma)$ returns the local optimum, $best$, obtained after applying the algorithm saHC on the original environment⁴, starting from σ . Once a local optimum is found, the permutation mask Π is generated uniformly at random to rotate the fitness landscape (see Equation 3.1), and the saHC is reinitialised from the previously found local optimum by $\text{saHC}(\Pi, best)$. The function $\text{saHC}(\Pi, best)$ starts from $best$ and applies the saHC on the rotated environment $f(\Pi \circ \pi)$, returning a local optimum σ^* . Note that solutions on the rotated environment are mapped to different objective values, such that $\forall \pi \in \mathbb{S}_n, \pi \rightarrow f(\Pi \circ \pi)$. The obtained local optimum, σ^* , replaces the previously found $best$ solution if it improves its quality.

On the other hand, a breadth-first rotation procedure applied to a saHC (saHC-R2) is presented. This method is comparable with the Simulated Annealing (SA) [95] or the Late Acceptance Hill-Climbing (LAHC) [96]. The implementation of this technique is summarised in Algorithm 3.2.

This strategy begins similarly to saHC-R1, but uses the rotated environment differently. Specifically, saHC-R2 allocates iterations in the rotated environment before returning to the original instance. Then, it continues the optimisation process from the new location in the original instance until a local optimum is found. The function $\text{GreedySearch}(\Pi, best, B_r)$ is initialised with $\Pi \circ best$, and carries on the search in the rotated environments for a budget of iterations, B_r . The number of iterations spent in the rotated instance, B_r , is halved as the search progresses to enhance the exploitation capacity of the algorithm. Note that the latest local optimum obtained serves as the starting point for the search in the rotated environment, even if it is not the overall best solution found (as in saHC-R1).

⁴Note that $\forall \pi \in \mathbb{S}_n, \pi \rightarrow f(e \circ \pi) = f(\pi)$ (Section 3.1.1).

Algorithm 3.2 saHC-R2: breadth-first rotation strategy for perturbation

- 1: Let σ be a random permutation and e the identity permutation.
 - 2: Let d be a number within the boundaries of a given permutation distance metric.
 - 3: $best \leftarrow$ Best solution found by **saHC**(e, σ).
 - 4: **repeat**
 - 5: $\Pi \leftarrow$ Uniformly at random permutation at distance d .
 - 6: $\sigma^* \leftarrow$ Solution obtained by **GreedySearch**($\Pi, best, B_r$).
 - 7: $B_r \leftarrow B_r/2$
 - 8: $best \leftarrow$ Best solution found by **saHC**(Π, σ^*).
 - 9: Update d based on the chosen cooling scheme (e.g. linear, exponential, etc.).
 - 10: **until** Stopping criterion is met.
-

Our perception is that both methods are suitable to escape from plateaux or local optima, changing the search direction of the algorithm [19, 21, 23] (see Section 2.1 for a detailed analysis of plateaux). Moreover, we can deduce the behaviour of both strategies. saHC-R1 jumps between attraction graphs going deeply until reaching a local optimum, so it should stand out on instances with a small number of attraction graphs. In contrast, saHC-R2 is more focused on the exploration of the solution space since it spends some iterations on the fitness landscape rotation moving the search away from the initial point. That said, saHC-R2 may be a good option when the space is composed by a large number of attraction graphs.

Apart from the algorithmic design of the methods, the degree in which the fitness landscape is rotated also influences the search process of the algorithms. Both methods use the distance between permutations to control the degree of the perturbation (rotation). Nevertheless, the rotation degree must be adjusted in accordance with the purpose of the algorithm at each moment (see Section 3.4.2.1).

Some evolutionary algorithms use a cooling schedule to guide the algorithm from exploratory behaviour to exploitative behaviour [95, 97]. This balancing idea might be used in the fitness landscape rotation by moving far away from a local optimum at the beginning (high rotation degree), and decreasing the distance until ending up moving towards a nearby location. Hence, the rotation is scaled and gradually decreased within the maximum distance and the minimum distance for a given metric, ensuring a controlled balance between the exploration and exploitation behaviour of algorithms.

It is worth noting that, although large rotation distances typically produce large perturbations, small rotations can be sufficient to significantly tilt the search process.

Mathematically, the linear cooling in terms of the rotation distance may be seen as follows:

$$d = \left\lfloor \frac{i}{B}(d_{max} - d_{min}) + d_{min} \right\rfloor, \quad (3.3)$$

where i is the iteration of the algorithm search process, B is the total budget of iterations, and d_{min} and d_{max} are the minimum and maximum distances of a permutation distance metric respectively.

Moreover, motivated by the assumption that even a small perturbation in the local optimum may lead to a different attraction graph [19, 21], this study introduces an exponential cooling scheme that starts with larger rotation distances early in the search process, promoting exploration, and smaller rotations in later iterations, allowing for more focused exploitation. Note that, although small rotations can still lead to a significant number of solution exchanges between attraction graphs, a well-calibrated exponential cooling strategy is essential to avoid premature convergence and maintain a gradual transition from exploration to exploitation throughout the search process.

Formally, the exponential cooling may be formulated as follows:

$$d = \left\lfloor d_{max} e^{-\lambda i} \right\rfloor, \quad \lambda = \frac{\ln \left(\frac{d_{min}}{d_{max}} \right)}{B}, \quad (3.4)$$

where λ is the cooling rate, i is the search iteration, B is the total budget of iterations, and d_{min} and d_{max} are the minimum and maximum distances of a permutation distance metric, respectively.

3.4 Experimentation

This section aims to analyse and prove the applicability of the fitness landscape rotation as a perturbation strategy on two permutation problems, i.e. the LOP and QAP, respectively. We direct the reader to Section 2.1.3 for the formal definition of the considered optimisation problems.

Table 3.2: Parameter settings for the algorithms with the fitness landscape rotation as perturbation strategy for the LOP.

Parameter	Description
Local search algorithm	Steepest-ascent hill-climbing algorithm (saHC)
Rotation degree	Exponential cooling
Distance metric	Cayley distance
Number of repetitions	30
Stopping criterion	10^3n iterations

This investigation is divided into two distinct studies. The first case study focuses on explaining and comparing the behaviour of the developed rotation-based algorithms to understand the underlying mechanism of the fitness landscape rotation and its repercussion in the performance of the algorithm for each problem instance.

On the second case study, we extend the previous analysis to compare the performance of the proposed approaches against other local search algorithms. To that end, we first adjust the rotation parameters for rotation-based algorithms. Then, a comparison of the performance of tuned rotation-based algorithms against other versions of the same algorithm is performed.

3.4.1 Case Study 1

This section implements the developed fitness landscape rotation strategies into saHC and evaluates their behaviour in solving the LOP. The specific instances used are obtained from the supplementary material in [19], which provides 12 LOP instances: eight of size 10 with a large number of plateaux, and four of size 50 that contain several large plateaux. The algorithm parameters for this study are summarised in Table 3.2.

Obtained results and instance properties are summarised in Table 3.3, which displays the best objective values, the number of rotations, and the count (and percentage, if applicable) of visited local optima for each algorithm. The total number of local optima for each instance is obtained from [19]. However, for instances of size $n = 50$ with incomplete data, we only report the number of explored local optima, excluding percentages.

Table 3.3: Information of the instances, and results of rotation-based algorithms on LOP instances. The percentages for instances of size $n = 50$ are not available, since their total number of local optima is unknown.

Instance	#LO	saHC-R1			saHC-R2		
		Best Obj. Value	Rotations	LO (%)	Best Obj. Value	Rotations	LO (%)
Instance 1	13	1605	2015	13 (100%)	1605	2636	13 (100%)
Instance 2	24	1670	2011	24 (100%)	1670	2629	24 (100%)
Instance 3	112	4032	1956	104 (92.8%)	4032	2545	106 (94.6%)
Instance 4	129	3477	1988	121 (93.8%)	3477	2565	125 (96.9%)
Instance 5	171	32952	2093	169 (98.8%)	32952	2712	171 (100%)
Instance 6	226	40235	1954	215 (95.1%)	40235	2571	220 (97.3%)
Instance 7	735	22637	2138	683 (92.9%)	22637	2742	716 (97.4%)
Instance 8	8652	513	2528	6063 (70%)	513	3351	6887 (79.6%)
N-be75eec	> 500	236464	8527	62143 (−%)	236464	10737	75404 (−%)
N-be75np	> 500	716994	8306	36046 (−%)	716994	10364	58423 (−%)
N-be75oi	> 500	111171	8507	80001 (−%)	111170	10698	96371 (−%)
N-be75tot	> 500	980516	8437	31550 (−%)	980516	10606	53450 (−%)

The results show the good performance of rotation-based algorithms, as well as their exploratory ability. Specifically, both algorithms are able to find the same optimal solutions (except for N-be75oi), in terms of objective values, but they differ in the number of rotations and local optima explored. The percentages represent the ability of the algorithms to explore the attraction graphs that compose the fitness landscape. saHC-R2 always performs more rotations than saHC-R1, so we can say that saHC-R2 tends to be more exploratory than saHC-R1.

Table 3.4 illustrates the influence of the rotation degree on the algorithmic performance, showing the number of rotations and local optima achieved by each algorithm, over 30 runs, for specific Cayley distances on **Instance 8**. The table shows that the highest exploratory behaviour of the algorithms holds on medium-small distances, i.e. both algorithms find more local optima when the rotation operates at $d_C = \{3, 4, 5\}$. This performance matches with the example in Table 3.1, where rotating to low or medium distances is sufficient to perturb the search of algorithms to different attraction graphs. Nevertheless, both rotation-based algorithms are coupled with the exponential cooling (see Equation 3.4), meaning that they spend more iterations rotating at low and medium distances.

Table 3.4: Number of rotations and reached local optima by each rotation-based algorithm on **Instance 8**.

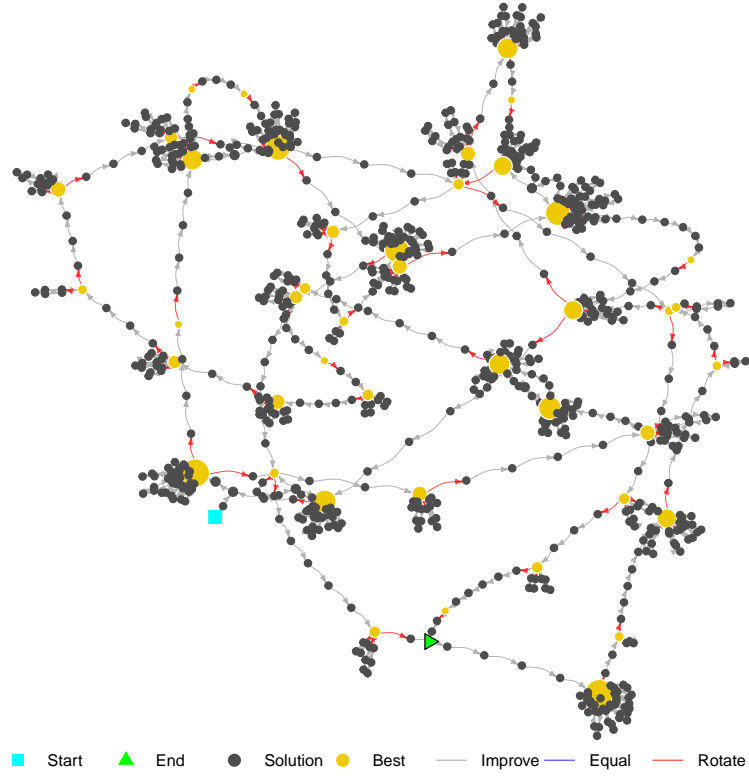
		Cayley distance								
		1	2	3	4	5	6	7	8	9
saHC-R1	Rotations	19594	19304	10851	7416	5623	4525	3778	3279	1452
	LO	678	1968	2500	2679	2725	2633	2438	2234	1227
saHC-R2	Rotations	27680	26090	14191	9432	7039	5633	4676	4027	1746
	LO	736	3809	4014	3719	3337	3037	2761	2528	1346

In order to visually represent and analyse the evolution of the algorithms, we use the Search Trajectory Networks (STNs) tool [98], a directed-graph-based model that displays the search trajectories of algorithms in two or three dimensions. Figure 3.2 displays a single run of each algorithm on **Instance 8** using STNs. The colours in the figures highlight the starting and ending points of the search (blue and green nodes), the best found solutions (yellow nodes) and the rotation operations (red edges), respectively. The entire experimentation is available online⁵.

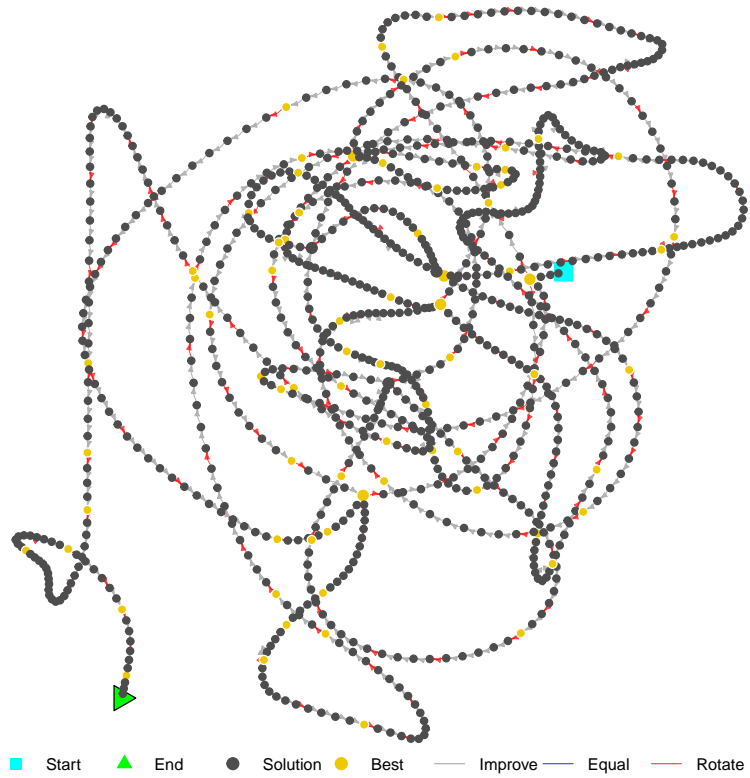
The plots illustrate the behaviour of each algorithm in a two-dimensional space. Specifically, Figure 3.2(a) shows saHC-R1, which systematically applies the rotation operation from the best solution found. This visualisation illustrates the structure of the fitness landscape, as the algorithm navigates through paths of the attraction graphs. In contrast, Figure 3.2(b) shows saHC-R2, where the algorithm passes through attraction graphs, rarely becoming stuck in the same local optimum. This behaviour emerges because saHC-R2 rotates from the most recent local optimum found, rather than the best solution identified.

Finally, it is worth noting the presence of a plateau composed of local optima in **Instance 8**, i.e. multiple local optima have the same objective value, which turns out to be the optimal value. The figures confirm that both algorithms can detect and deal with plateaux. Interestingly, Figure 3.2(a) shows that some local optima that form the plateau are visibly larger, which means that saHC-R1 visits them several times. Contrarily, saHC-R2 reaches more local optima than saHC-R1.

⁵Available at <https://zenodo.org/record/6406825#.Ykcxaw7MI-Q>



(a) saHC-R1



(b) saHC-R2

Figure 3.2: Search Trajectory Networks of rotation-based algorithms on LOP Instance 8.

3.4.2 Case Study 2

The previous section has presented a preliminary study to illustrate the applicability of the fitness landscape rotation for the LOP. However, for the sake of demonstrating the nature of rotation-based algorithms, the study has omitted the adjustment of parameters and the comparison of suggested strategies with other algorithms.

This section expands on the previous experiments to thoroughly investigate the benefits of both rotation-based algorithms. Specifically, we examine a different optimisation problem and algorithm to demonstrate the fidelity of both rotation-based strategies. Therefore, we begin by adjusting the rotation parameters, and then, we compare the performance of the tuned rotation-based algorithms against other algorithm versions.

3.4.2.1 Rotation-based Algorithm Calibration

This study considers the QAP instances `tai40a`, `tai40b` and `bur26a`⁶ [99], of size $n = 40$ and $n = 26$, respectively. We direct the reader to Section 2.1.3 for a formal definition of the QAP.

Moreover, this study uses the stochastic hill-climbing heuristic (sHC) as a local search algorithm to apply both fitness landscape rotation strategies. Using the swap neighbourhood operation, sHC moves towards the first randomly chosen neighbour that improves the observed objective value. Note that, in order to reduce the computational cost of sHC, the number of neighbour comparisons before considering the algorithm to be stuck at a local optimum (referred to as *improvement trials* or simply *trials*). The parameters of the algorithm used in this study are summarised in Table 3.5.

⁶The QAP instances and their best-known values have been obtained from <https://coral.ise.lehigh.edu/data-sets/qaplib/qaplib-problem-instances-and-solutions/>.

Table 3.5: Parameter settings for the algorithms with the fitness landscape rotation as perturbation strategy for the QAP.

Parameter	Description
Permutation Distance Metrics	Cayley, Hamming, Kendall's- τ
Improvement Trials	$\{n, n + 74, n + 148, \dots, n + 740\}$ (11 equally distributed points between $n = 40$ and $780 = \binom{n}{2}$)
Cooling Strategies	Linear and Exponential (see Equations 3.3 and 3.4)
Stopping criterion	$B = 10^3 n$ iterations
Nnumber of repetitions	30 independent runs for each algorithm

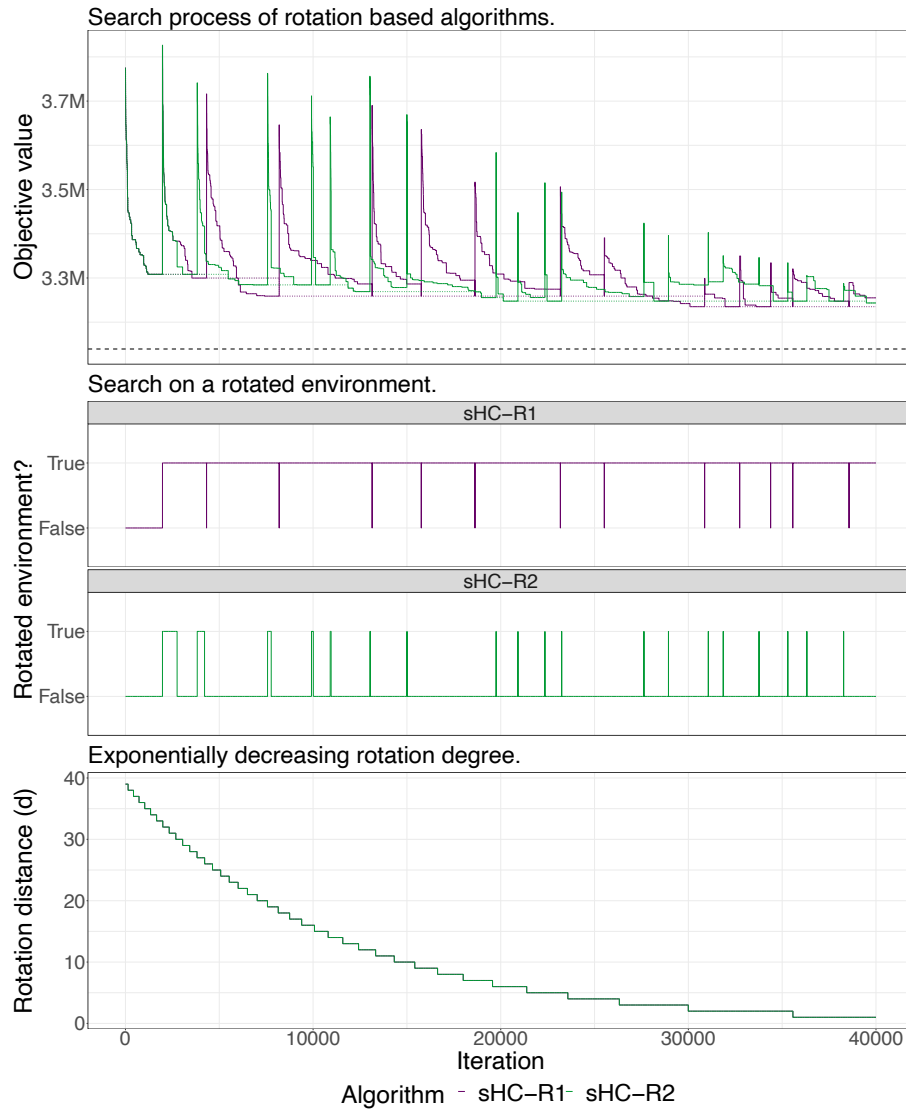
**Figure 3.3:** Illustration of the search of rotation-based algorithms, under the Cayley distance metric, in an independent run on tai40a.

Figure 3.3 shows the behaviour of a single run of rotation-based algorithms with exponential cooling under the Cayley distance metric in `tai40a`. The horizontal axis represents the iteration of the search process across all plots. The plots are organised as follows: the top plot displays the evolution of algorithms, where the solid-coloured line indicates the objective value of each candidate solution, the dotted line shows the best-found solution, and the thick dashed line denotes the best-known objective value. The middle plot indicates whether the algorithm is exploring the original or the rotated fitness landscape at each iteration. Finally, the bottom plot shows the distance at which the fitness landscape may be rotated, measured by the Cayley distance.

The top and bottom charts reflect the relationship between the rotation distance and the steepness of the modification. In general terms, the statement “a higher rotation distance means a larger perturbation” holds, since the environment is rotated with a higher entropy than it does at the end of the search. However, the third peak for `sHC-R2` in the top graph goes further than the previous one, even when the rotation distance is smaller (similar to the example shown in Figure 3.1).

Due to the infeasibility to show the performance of each algorithm for different parameter and metric configurations, the results have been summarised as heatmaps in Figure 3.4. The tables are organised by metrics, trials, and rotation distances for each of the algorithms. The colours in the tables are used for guidance only, where the lighter background colour, the better performance of the algorithm, i.e. the algorithm ends closer to the best-known solution in terms of objective value.

The algorithms demonstrate consistent performance across different metrics, indicated by the similar colour gradients in the tables for all permutation metrics. However, their performance depends on the number of improvement trials considered. Heatmaps for the `sHC-R1` algorithm reveal a darker tone on the left (fewer trials) compared to the right (full neighbourhood exploration), highlighting the importance of thoroughly exploring the neighbourhood before rotating the environment to effectively reach and compare local optima. Additionally, rotating at the minimum distance for any metric is generally more advantageous to cooling schemes, with exponential cooling proving

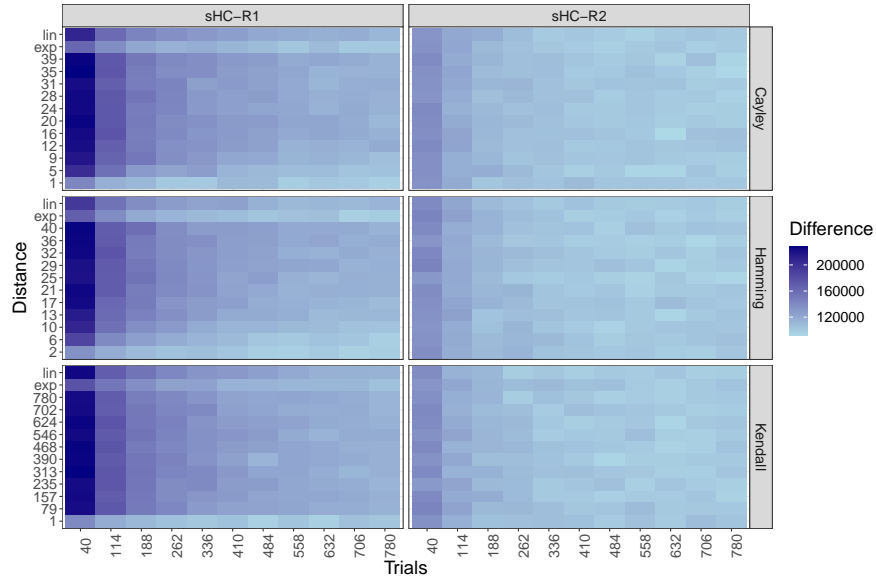


Figure 3.4: Comparison between the performance of rotation-based algorithms under the Cayley distance metric. The difference between the mean performance of the algorithms to the best-known objective value for `tai40a` for each configuration are indicated.

more efficient than linear cooling for the fitness landscape rotation. Overall, sHC-R1 performs well when it thoroughly explores the neighbourhood and applies slight rotations. In contrast, sHC-R2 shows a consistent colour gradient across configurations, although it performs better after a few iterations of neighbourhood exploration before the fitness landscape rotation. Thus, sHC-R2 generally benefits from a rotated environment, regardless of the nature and extent of the rotation.

The calibration reveals no significant differences between the modifications produced by the permutation metrics. In the following, we will demonstrate how the symmetry of the instances greatly influences the algorithmic performance.

3.4.2.2 Advanced Experimental Study

Following the comparison of previously tuned algorithms, two additional sHC variants are examined: the traditional sHC, and the multi-starting sHC (sHC-r). In sHC-r, the number of improvement trials matches that of sHC-R1 and sHC-R2, but the rotation-based algorithm parameters are adjusted due to random reinitialisation of the search.

Table 3.6: Parameter settings for rotation-based algorithms in the advanced experimental study of the QAP.

Parameter	Value
Local search algorithm	Stochastic hill-climbing algorithm (sHC)
Rotation degrees	$\{d_{min}, \dots, \lceil d_{max} * 0.25 \rceil\}$, linear and exponential cooling
Number of improvement trials	$n, 2n, \dots, d_{max}^K$
Number of independent runs	30
Stopping criterion	$10^3 n$ iterations

The parameter setting used for the following experiments is described in Table 3.6.

This study uses 11 instances from QAPLIB with different properties and sizes:

- **Totally symmetric** [99]: tai40a, tai60a.
- **Symmetric distance and asymmetric flow matrices** [100]: lipa40a, lipa40b, lipa60a, lipa60b.
- **Asymmetric distance and symmetric flow matrices** [101]: tai40b, tai60b.
- **Totally asymmetric** [102]: bur26a, bur26b, bur26c.

Figure 3.5 shows the average difference between the best solutions found by each algorithm and the best-known values for instances with different symmetry structures⁷. The sub-captions detail the characterisation (symmetries) of the distance and flow matrices for the selected QAP instances. Note that sHC provides only provides the average performance across all runs, as it does not adapt when the search gets stuck.

The plots indicate that the benefits of rotation-based algorithms vary by instance, although they generally help avoid getting trapped in attraction graphs. As can be observed, the problem characterisation significantly impacts algorithmic performance; in particular, the distance matrix representation. Figures 3.5(a) and 3.5(c) illustrate the algorithmic performance on instances with a symmetric distance matrix. The plots show that rotation-based algorithms outperform other sHC variants when rotating at close distances and exploring the entire neighbourhood (as detailed in Section 3.4.2.1). In particular, sHC-R2 performs better than sHC-R1 with optimised rotation parameters. However, for asymmetric distance matrix, the results become more chaotic and differ significantly from previous findings, as can be seen in Figures 3.5(b) and 3.5(d).

⁷Note that some results are selected to highlight interesting aspects. Complete results are available at https://github.com/joanalza/GECCO_2021.git

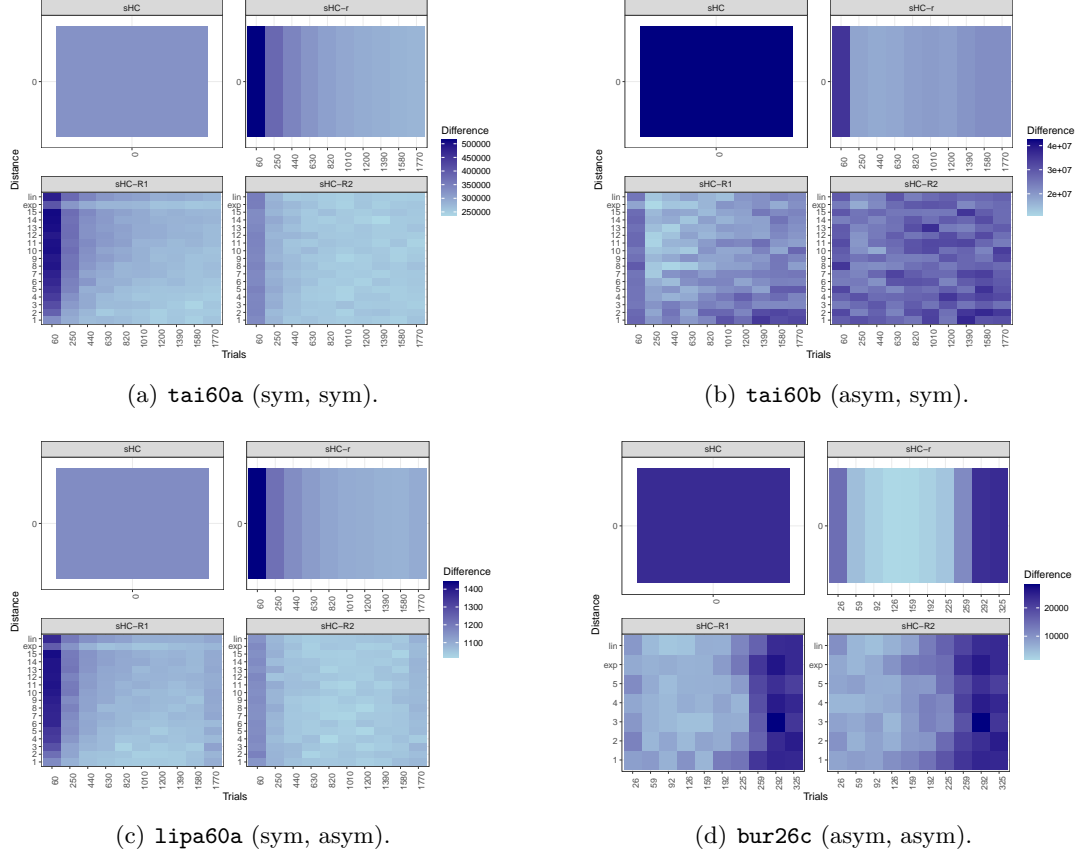


Figure 3.5: Performance evaluation of rotation-based algorithms against other local search algorithms. The performance of the algorithms sHC, sHC-r, sHC-R1 and sHC-R2, are described on four QAP instances with different symmetry structures on the distance and flow matrices, respectively.

In the case of the instance **tai60b**, which is composed of an asymmetric distance matrix and a symmetric flow matrix, sHC-R1 and sHC-R2 work worse than sHC-r in general, although better than sHC. In any event, sHC-R1 shows good performance when the rotated parameters are correctly tuned. In contrast, both sHC-R1 and sHC-R2 perform similarly in the **bur26c** instance, which is completely asymmetric, i.e. the flow and distance matrices of the instance are both asymmetrical. Nevertheless, a simple restart is often more valuable than rotating the fitness landscape.

As a point of curiosity, the linear and exponential cooling schemes presented in Equations 3.3 and 3.4 excessively focus on the exploration of the space during initial

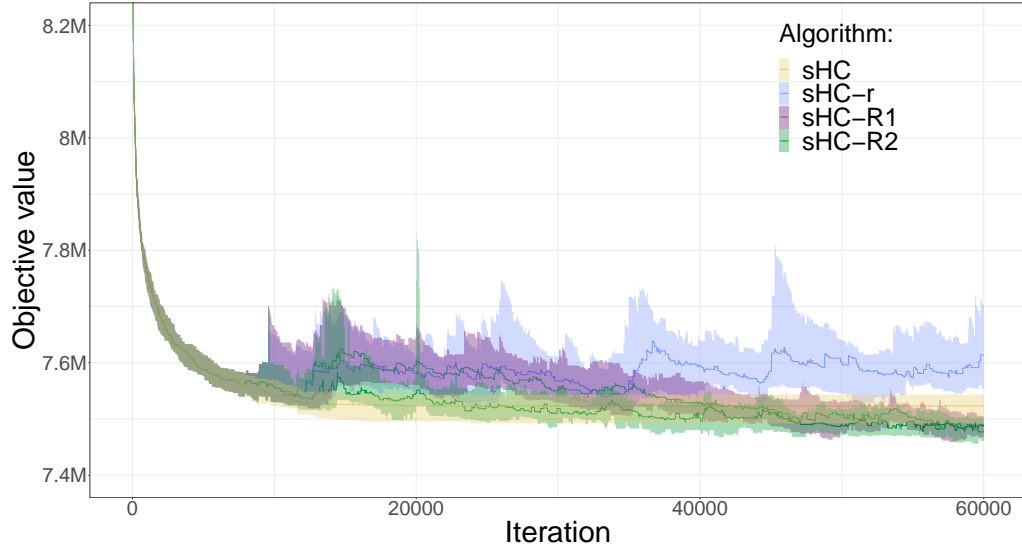


Figure 3.6: Illustration of the search process of local search algorithms with and without rotation-based strategies. The interquartile range of each algorithm on `tai60a` under the exponential cooling and the exploration of the entire neighbourhood are shown.

iterations. Figure 3.6 illustrates the interquartile range of the search process of algorithms on the `tai60a` instance, considering the entire exploration of the neighbourhood and the exponential cooling. The figure shows how `sHC` gets stuck in the first quarter of the entire budget. Although other algorithms react to getting stuck, it is not until the middle (or even third quarter) of the search when rotation-based methods surpass `sHC`; in particular, when the environment rotates at lower distances. Therefore, results suggest that low distance rotations are generally more beneficial, as larger rotations may result in a similar behaviour to a random restart in terms of the number of relabelling solutions.

3.5 Summary

Chapter 3 achieves the research objective **OB 2** by theoretically analysing the preserving nature and repercussions of the fitness landscape rotation, in addition to introduce and evaluate two rotation-based perturbation strategies designed to improve the performance of local search algorithms.

The fitness landscape rotation has been widely used to generate DOPs due to its preserving nature, where important properties of the problem are maintained. This chapter has thoroughly revised the fitness landscape rotation in the permutation space with proofs and examples, and has suggested further applications of this method to perturb the search of local search algorithms when they get stuck.

Specifically, we have algebraically demonstrated how the structure of the fitness landscape and the relationships between solutions are preserved. Furthermore, we found that the degree of rotation can be misleading, as it does not always rotate the fitness landscape as intended. Therefore, we suggest caution when employing the fitness landscape rotation in combinatorial space, since even a slight rotation can have a significant impact in terms of solution exchanges between attraction graphs.

Based on the insights gained, we have presented two perturbation strategies based on the fitness landscape rotation to change the search direction of local search algorithms in a controlled way. On the one hand, a depth-first rotation approach has been suggested. This strategy moves through local optima values by comparing them against new local optima found in rotated environments. On the other hand, a breadth-first rotation strategy was designed to kick a stuck search at some degree, and resume the search until reaching a new local optimum.

The experiments carried out have illustrated the exploratory behaviour of both fitness landscape rotation strategies to perturb the search and reach out further areas of the solution space. However, obtained results have also shown that restarting the optimisation may be preferable in some cases. Therefore, conducted experiments suggest using the restart as the baseline for the comparison of perturbation strategies.

Chapter 4

Elusivity Concept to Measure the Adaptive Challenge of Dynamic Optimisation Problems to Online Algorithms

As pointed out in Section 2.2.1, many authors point out that non-static optimisation problems should be considered *dynamic* only if the *online* algorithm adapts accurately to changes over time; otherwise, the problem should be regarded as sequences of independent problems. This definition emphasises the dynamic nature of the problem and the adaptive advantage of online algorithms. Most empirical studies in dynamic optimisation operate under the implicit assumption that adapting to problem changes with different frequent and severe configuration is more efficient and stable than restarting the search from scratch [4, 11, 35, 69, 81]. Nevertheless, restarting algorithms have proven to be effective for certain optimisation problems [50, 55].

To the best of our knowledge, a systematic quantitative analysis that evaluates and relates the variability of problems, the adaptive challenge that DOPs present to algorithms, and performance metrics has not been established to date. Hence, this chapter presents the *elusivity* concept to quantitatively measure the extent to which

an online algorithm can accurately adapt to a DOP. Thus, the performance of online algorithms are compared against the restarting version of the same algorithm, specified by a performance metric. In addition, we distinguish elusive and non-elusive problems based on the adaptive challenge to algorithms with regard to restart.

The remainder of the chapter is structured as follows. Section 4.1 presents mathematical notations to introduce the elusivity concept using the language of stochastic processes. Then, the elusivity concept is applied and thoroughly analysed by reproducing two already published case studies in Section 4.2. The goal is to empirically identify DOP instances in the studies where adapting is less effective than restarting, under a performance metric. In addition, we use the elusivity concept to distinguish between elusive and non-elusive problems based on the adaptive advantage of online algorithms. Finally, Section 4.3 provides a summary of the conclusions and insights from obtained results.

4.1 A Definition Framework for the Elusivity

The following definitions introduce some mathematical terminology to precisely describe the performance of online and restarting algorithms and define a formulation of the elusivity (or the adaptive challenge) of a problem to an algorithm under a performance metric. That is, DOPs and algorithms are described in sufficient generality to apply to most situations of interest and avoid unnecessary assumptions about problems or algorithms. To that end, we use the language of stochastic processes to abstract any detail of the internal state of algorithms into random variables that represent successive states of the search. In applying these ideas to specific algorithms, internal state parameters are known, and the extent to which they condition the search state transitions can be estimated experimentally.

Definition 4.1 (Static optimisation problem). *A static optimisation problem $P_s = (\Omega, f)$ is a tuple consisting of Ω , which represents the search space, and $f : \Omega \rightarrow \mathbb{R}$, which is an objective function mapping solutions in Ω to real values that reflect their objective value.*

This notion is flexible, and can be extended to multi-objective problems without difficulty, accommodating different representations of solutions.

Definition 4.2 (Search algorithm). *A search algorithm A solves an optimisation problem P_s by exploring solutions in Ω with the goal of optimising f . The algorithm search is bounded by a budget B , which defines the maximum number of objective value evaluations (or iterations) it can perform on f . Thus, a run of A on P_s with budget B can be represented as a sequence of random variables $S_1(A, P_s), \dots, S_i(A, P_s), \dots, S_B(A, P_s)$, where $S_i(A, P_s)$ is a sample of one or more solutions in Ω , commonly referred to as a population.*

For non-population based algorithms (e.g. local search algorithms), each S_i may represent a candidate solution as a population.

To initiate a run, algorithms typically begin from an initial distribution of populations, often assumed to be uniform $U_{(A, P_s)}$ ¹. This indicates that the A starts from a randomly chosen population distributed uniformly over Ω . Formally, this is represented as:

$$S_1(A, P_s) \sim U_{(A, P_s)}, \quad (4.1)$$

where $S_1(A, P_s)$ is the initial population sampled according to this distribution.

Definition 4.3 (Trajectory of search algorithms). *The trajectory of an algorithm A on a problem P_s refers to the sequence of populations s_1, \dots, s_B produced throughout its execution, where $\Pr(S_i = s)$ indicates the probability that s is the i^{th} population generated by A .*

Note that, since the trajectory is generated by sampling a sequence of random variables, it is itself a random variable. That is, $Tr(A, P_s)$ denotes the random variable of trajectories of the algorithm A on the problem P_s , and $tr(A, P_s)$ refers to a specific value sampled from $Tr(A, P_s)$. Therefore, an algorithm $A : P_s \rightarrow Tr(A, P_s)$ can be defined as a map between a problem P_s and a trajectory distribution $Tr(A, P_s)$.

¹This statement however could equally apply to algorithms starting with a predefined initialisation strategy by replacing $U_{(A, P_s)}$ with the approach used to generate the initial random variable.

Definition 4.4 (Performance metric). *A performance metric $\phi(A, P_s)$ is a function that assigns a real number to a run $tr(A, P_s)$ of the algorithm A in the problem P_s , i.e. $\phi : tr(A, P_s) \rightarrow \mathbb{R}$. Since $Tr(A, P_s)$ is stochastic, the performance of algorithms during R runs can be used to estimate the expected performance $\mathbb{E}[\phi(A, P_s)]$.*

In order to extend previous (static optimisation) definitions to the domain of dynamic optimisation, it is essential to precisely formulate DOPs and online algorithms. Without loss of generality, based on the definitions in Section 2.2.1, we define DOPs from the perspective of benchmark generators as follows:

Definition 4.5 (Dynamic optimisation problem). *A dynamic optimisation problem P can be defined as a sequence of static optimisation problem instances P_1, \dots, P_m , where $P_t = (\Omega_t, f_t)$ for each $j = 1, \dots, m$, where $m > 1$, and P_t is defined on the space Ω_t and the objective function $f_t : \Omega_t \rightarrow \mathbb{R}$.*

Empirical benchmark studies in dynamic optimisation assume that adaptation is usually beneficial in efficiently tracking moving optima and reducing the computational cost [2]. This assumption leads us to the following definition.

Definition 4.6 (Online algorithm). *A search algorithm A is run online on the dynamic optimisation problem P with budget $B = \sum_{j=1}^m B_t$ means that A outputs a related trajectory $tr(A, P)$ while successively processing a sequence of problem instances P_t .*

Using the above notation, the online algorithm A is conditioned on the trajectory $tr(A, P_{j-1})$ due to changes in the objective function f_t . Specifically, the i^{th} population s_i of the online algorithm A in the problem instance P_t is sampled from $S_i(A, P_t)$, which depends on the trajectory (often simplified to the previous population s_{i-1}) of A .

This notation describe how an online algorithm runs on a DOP, and clarifies the concept of restarting the algorithm for DOPs.

Definition 4.7 (Restarting algorithm). *Assuming changes are detectable, a restarting algorithm A^r is defined as the independent execution of an algorithm A with random restarts on a problem P_t , with a budget $B = \sum_{j=1}^m B_t$.*

This means that the trajectory $tr(A^r, P)$ is a concatenation of the m independent trajectories $tr(A, P_1), \dots, tr(A, P_m)$, where the initial population of each independent trajectory is set by the starting distribution (Equation 4.1). In other words, A runs from initialisation on P_1 to produce $tr(A, P_1)$, then, on P_2 for $tr(A, P_2)$, and so on.

It is worth noting that, in general, $S_i(A, P_t)$ and $S_i(A^r, P_t)$ will follow different probability distributions, determined by the different trajectories of A and A^r , respectively. In this way, the trajectories and performance metrics for A and A^r over general DOPs can be computed and compared.

We define the *elusivity* of a DOP to an algorithm based on a performance metric based on previous definitions in dynamic optimisation.

Definition 4.8 (Elusivity). *Let P be a non-noisy dynamic optimisation problem, A an online algorithm for P , A^r the restarting version of A , and ϕ a performance metric to minimise. The elusivity of P to A under ϕ is defined as:*

$$\mathcal{E}(P, A, \phi) = \mathbb{E}[\phi(A, P) - \phi(A^r, P)].$$

P is elusive to A under ϕ if and only if $\mathcal{E}(P, A, \phi) \geq 0$.

For maximising performance metrics, $\mathcal{E}(P, A, \phi) = \mathbb{E}[\phi(A^r, P) - \phi(A, P)]$.

Informally, the elusivity concept allows to quantitatively compare algorithmic performance over dynamic benchmark sets. For example, we can say that a problem P is less elusive to an algorithm A_1 than A_2 if $\mathcal{E}(P, A_1, \phi) < \mathcal{E}(P, A_2, \phi)$. This prompts the following definition.

Definition 4.9 (Adaptive advantage). *Let $\mathcal{E}(P, A, \phi)$ denote the elusivity of problem P to online algorithm A with respect to performance metric ϕ . The adaptive advantage refers to the performance improvement gained by A in response to changes in the problem P .*

Therefore, the adaptive advantage provides a measure of how well or badly a particular algorithm adapts to a particular DOP. A strongly negative elusivity will indicate that an algorithm is well-suited to a particular benchmark set under a performance metric.

Note that, if P is elusive to A under ϕ , then, there is no adaptive advantage to be gained from A as the problem changes with reference to ϕ . In other words, equally good or even better performance can be expected from simply restarting the algorithm when detecting a problem change. This establishes restart as the baseline against which all online algorithms should be compared with, i.e. online algorithms must, at least, beat their restarting version in order to work with *non-elusive* problems. Trivially from previous definitions, all DOPs will be elusive to algorithms that adapt by reinitialising the entire population (not necessarily uniformly at random), because in this case $Tr(A^r) = Tr(A)$. Hence, $\mathcal{E}(P, A, \phi) = \mathcal{E}(P, A^r, \phi)$.

4.2 Case Studies for the Elusivity Analysis

In this section, the elusivity concept is evaluated by using and extending the analysis on already published experimental works [4, 11, 35, 69, 81]. That is, a number of well-studied non-noisy DOPs, algorithms, and performance metrics are re-implemented from two existing frameworks, and the resulting trajectories are subjected to elusivity analysis. In that context, there are a number of critical elements that are addressed in this section: (i) the dynamic generators used to insert dynamism into static optimisation problems, (ii) a brief review of the algorithms, the adaptation mechanisms incorporated to enhance their adaptive advantage and their restarting version, and (iii) the functions used to measure the performance of the algorithms.

4.2.1 Dynamic Benchmark Generators

In dynamic optimisation, the community commonly formulates DOPs as sequences of static optimisation problems that change during the algorithm execution. This study replicates popular benchmark generators to create DOPs. Specifically, the fitness landscape rotation (explored in Section 3.1) has been employed for both binary and permutation spaces [4, 11, 69, 81]. Additionally, we replicate the benchmark generators from [35] to develop DTSPs with traffic factor and city replacement, respectively. See Section 2.2.2.2 for more details on the benchmark generators.

4.2.2 Algorithms and adaptation mechanisms

This experimentation examines a representative set of algorithms in dynamic optimisation and utilises benchmark generators from published works [11, 35, 38, 44, 45, 46, 103].

The first case study focuses on three well-studied algorithms supplemented by elitism and random immigration approaches (described in Section 2.2) [38, 43, 44]. Specifically, this includes elitism and random immigrants-based genetic algorithms (EIGA and RIGA) [44], population-based incremental learning algorithms (EIPBIL and RIPBIL) [11, 45], and ant colony optimisation algorithms (EIACO and RIACO) [103]. Note that EIACO and RIACO replace the worst solutions in a memory of past solutions instead of replacing the overall population [38].

For the second case study, we consider four ant colony optimisation (ACO) variants from [35]: two diversity-based algorithms (MMAS and MC-MMAS) and two memory-based algorithms (PACO and EIACO). These types vary in their way of updating pheromone trails, affecting how they adapt to problem changes. That is, diversity-based algorithms reduce the evaporation of previous promising solutions, whereas memory-based approaches store good solutions to accelerate the decrease of outdated pheromone trails.

The restarting version of the algorithms, denoted by the superscript “ A^r ”, restarts the search process of the algorithm, discarding previously gathered information when a problem change occurs. This restart strategy is one of the simplest and most straightforward ways to cope with problem changes, although it requires accurate change detection. In this study, a single detector (defined in Section 2.2.1) is employed to identify problem changes by iteratively evaluating the objective value of the best solution in the population.

4.2.3 Performance Metrics

Algorithms are usually evaluated using performance metrics, where different factors (specified by the practitioner) are measured depending on the goals of the problem. For example, the financial field is more focused on obtaining good solutions quickly,

rather than finding the best solution possible for each problem instance [9]. As a result, performance metrics can be classified into two classes: optimality-based and behaviour-based performance metrics. The former evaluates the ability of algorithms to find high-quality solutions, whereas the latter studies the internal nature of algorithms, such as their recovery speed or stability. This work uses three optimality-based metrics from replicated studies, in addition to a behaviour-based metric.

First, the *best-of-generation* metric F_{BOG} [11] averages the value of the best objective value at each iteration over several runs. Formally, it may be described as:

$$\mathbb{E}[F_{BOG}(A, P)] = \frac{1}{R} \sum_{k=1}^R (F_{BOG}(A, P)) = \frac{1}{R} \sum_{k=1}^R \left(\frac{1}{B} \sum_{i=1}^B x_{i,k}^*(A, P) \right), \quad (4.2)$$

where R is the total number of independent runs and $x_{i,k}^*(A, P)$ is the best objective value of the population $s_i(A, P_t)$ at run k . Therefore, the elusivity of a problem to an algorithm under best-of-generation F_{BOG} is measured as:

$$\mathcal{E}(P, A, F_{BOG}) = \frac{1}{R} \sum_{k=1}^R \left(\frac{1}{B} \sum_{i=1}^B [x_{i,k}^*(A, P) - x_{i,k}^*(A^r, P)] \right). \quad (4.3)$$

Second, the *accuracy* metric $H_{\Delta m}$ [104] averages the difference between the optima and the value of the best individual at the end of each change period. Li et al. [105] adapted it to average the accuracy over the runs. It may be represented as:

$$\mathbb{E}[H_{\Delta m}(A, P)] = \frac{1}{R} \sum_{k=1}^R (H_{\Delta m}(A, P)) = \frac{1}{R} \sum_{k=1}^R \left(\frac{1}{m} \sum_{j=1}^m h_j(A, P) \right), \quad (4.4)$$

where $h_j(A, P)$ is the best-error² when the problem instance P_t reaches the budget B_t of the algorithm A on run k , and m is the total number of instances. By replacing the best-error h_j with the best found objective value just before changing P_t to P_{j+1} , x_j^* , the *best-before-change* performance metric, $P_{\Delta m}$, is obtained that is used as the third metric in this experimentation (as done by [35]). The elusivity of a problem to an algorithm under the accuracy metric $H_{\Delta m}$ is calculated as:

²The difference between the best-known value of the instance and the value of the best individual in the population.

$$\mathcal{E}(P, A, H_{\Delta m}) = \frac{1}{R} \sum_{k=1}^R \left(\frac{1}{m} \sum_{j=1}^m [h_j(A, P) - h_j(A^r, P)] \right). \quad (4.5)$$

Similarly, the elusivity of a problem to an algorithm under the best-before-change performance metric is measured as:

$$\mathcal{E}(P, A, P_{\Delta m}) = \frac{1}{R} \sum_{k=1}^R \left(\frac{1}{m} \sum_{j=1}^m [x_j^*(A, P) - x_j^*(A^r, P)] \right). \quad (4.6)$$

It is worth noting that, since the accuracy measures the difference between the objective value of the best found solution and the optimal objective value, the elusivity analysis of the accuracy or the best-before-change metrics is the same. Hence, $\mathcal{E}(P, A, H_{\Delta m}) = \mathcal{E}(P, A, P_{\Delta m})$.

Fourth, the *robustness* metric \mathcal{R} [106] measures the stability, persistence, and degradation of an algorithm by comparing the best already found solution with the best solution found on the last change (instance) in the following way:

$$\mathbb{E}[\mathcal{R}(A, P)] = \frac{1}{R} \sum_{k=1}^R (\mathcal{R}(A, P)) = \frac{1}{R} \sum_{k=1}^R \left(\frac{1}{m} \sum_{j=1}^m \min\left(1, \frac{\Delta x_{j-1}^*(A, P)}{x_t^*(A, P)}\right) \right), \quad (4.7)$$

where $\Delta x_{j-1}^*(A, P)$ is the best objective value found by the algorithm A for the problem P , and $x_j^*(A, P)$ is the best objective value found by A for the problem instance j . $\mathcal{R} \in [0, 1]$ is a maximisation performance metric, i.e. the closer to 1, the better average robustness of the algorithm A . The \min operation evaluates whether the algorithm can reach the same or a better objective value than previous changes. Note that this equation is designed for minimisation problems, but it can be adapted for maximisation by replacing \min with \max . Thus, the elusivity of a minimisation problem to an algorithm, as measured by its robustness \mathcal{R} , is obtained by:

$$\mathcal{E}(P, A, \mathcal{R}) = \frac{1}{R} \sum_{k=1}^R \left(\frac{1}{m} \sum_{j=1}^m \left[\min\left(1, \frac{\Delta x_{j-1}^*(A, P)}{x_t^*(A, P)}\right) - \min\left(1, \frac{\Delta x_{j-1}^*(A^r, P)}{x_t^*(A^r, P)}\right) \right] \right). \quad (4.8)$$

4.2.4 Parameter Settings

In this section, an elusivity analysis on the extended experimental frameworks in [11, 12, 35, 44, 103] is conducted to (i) introduce the elusivity concept and accurately identify the different elusivity degrees of problems to algorithms, (ii) characterise DOPs as *elusive* or *non-elusive* to well-studied algorithms, (iii) analyse the adaptive advantage and the behaviour of state-of-the-art online algorithms regarding the elusivity of a realistic problem, and (iv) perform the overall elusivity analysis of the algorithms for each DOP considering all dynamic configurations. The experimentation is divided into two parts.

First, a comprehensive and straightforward case study is replicated to evaluate the elusivity concept, where the landscape is rotated for the dynamic travelling salesperson problem (DTSP) and the dynamic knapsack problem (DKP) [11, 44, 103]. In the case of DTSP, the instances³ `kroA100`, `kroA150` and `kroA200` (containing 100, 150 and 200 cities) are used to insert dynamism, and their best-known values (for static instances) are obtained from [107], used for the accuracy metric. For DKPs, three static optimisation problem instances have been constructed following the patterns presented in [12], and they are available online⁴ repository as supplementary material. The parameter setting employed for this experimentation (extracted from previous works [11, 12, 44, 103]) is described in the second column (Case Study I) of Table 4.1.

In the second case, the aim is to apply the concept of elusivity to assess the benefits of adaptation versus restarting, providing a more complete and realistic comparison of the adaptive advantage of various algorithms. Specifically, the experiments in [35] are reproduced and extended to evaluate the complete elusivity degree of four ACO variants, using identical algorithmic parameters to obtain consistent results. The authors introduce dynamism (traffic factor) into the instances `kroA100` and `kroA200` (with 100 and 200 cities, respectively) by considering 110 change configurations (10 change magnitudes and 11 change frequencies). The parameters used in this experimentation are described in the third column (Case Study II) of Table 4.1.

³The static TSP instances and their best-known values have been obtained from <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/>.

⁴Available at https://zenodo.org/record/7346818#.Y3yfw0x_pqs

Table 4.1: Parameter values of the benchmark generators and algorithms extracted from reference works [11, 12, 35, 44, 103].

Parameter	Case Study I Value	Case Study II Value
Change periods (τ)	10, 25, 50, 100, 200	$\lceil n \cdot s \rceil, s = \{1, 2.5, 5, \dots, 22.5, 25.0\}$
Change magnitudes (ρ)	0.1, 0.25, 0.5, 0.75, 1.0	0.1, 0.2, ..., 1.0
Number of repetitions (R)	50	30
Elitism criteria	TRUE	FALSE
Stopping criterion (B)	1000 iterations	$\tau * 100$ (99 changes)
Population size ¹	n ($\lceil 0.25n \rceil$ ants for ACO)	25
Immigrant replacement rate	0.2	0.5 (EIACO)
RIGA mutation probability	0.01	-
RIPBIL learning rate	0.25	-
RIPBIL mutation probability	0.02	-
RIPBIL shift operator	0.05	-
RIACO initial pheromone trail	$1/n$	$1/n$
RIACO relative influence rate	$\alpha = 1, \beta = 5$	$\alpha = 1, \beta = 5$
Pheromone evaporation rate	-	$\theta = 0.8$ (MMAS & MC-MMAS)
Memory size	$\lceil 0.25n \rceil$	3 (PACO & EIACO)
Performance metrics	F_{BOG} & $H_{\Delta m}$	F_{BOG} & $P_{\Delta m}$ & \mathcal{R}

¹ Number of ants for ACO variants.

Note that, due to the space limit, only certain results that illustrate the elusivity concept are selected and presented in the following sections. Complete implementation and results are available online⁵.

4.2.5 Case Study I

For illustrative purposes, a set of four problems with different dynamism, two algorithms and two performance metrics are selected as example. We denote as P' and P'' two DTSPs that change every $\tau = 100$ iterations with magnitudes $\rho = 0.1$ and $\rho = 0.5$, respectively. Similarly, P''' and P'''' represent two DKPs changing at period $\tau = 100$ and magnitudes $\rho = 0.1$ and $\rho = 0.5$. Note that P' and P'' are minimisation problems, and P''' and P'''' are maximisation problems. As for the algorithms and performance metrics, RIPBIL and RIGA are employed under the best-of-generation and the accuracy metrics. Table 4.2 summarises the overall performances of the algorithms,

⁵ Available at https://zenodo.org/record/7346818#.Y3yfw0x_pqs

Table 4.2: Overall performances of the algorithms and, in bold, the elusivity value for each problem, algorithm, and performance metric combination (elusive problems highlighted in orange).

	$\mathbb{E}[F_{BOG}]$	$\mathcal{E}(P, A, \phi)$	$\mathbb{E}[H_{\Delta m}]$	$\mathcal{E}(P, A, \phi)$
$(P', RIPBIL)$	186049.80	-31204.70	170126.30	-34361.50
$(P', RIPBIL^r)$	217254.50		204487.80	
$(P'', RIPBIL)$	208870.90	-8847.90	191811.10	-13238.90
$(P'', RIPBIL^r)$	217718.80		205050.00	
$(P''', RIGA)$	1788.26	-9.03	20.40	-9.00
$(P''', RIGA^r)$	1779.23		29.40	
$(P''', RIGA)$	1773.59	5.95	31.00	2.00
$(P''', RIGA^r)$	1779.54		29.00	

and also captures the elusivity of problems to the algorithm and performance metric combinations (following Definition 4.8). The results show that online algorithms are useful on $P'-P'''$, but restarting the RIGA is beneficial on P''' . In other words, P''' proves *elusive* to RIGA, regardless of the performance metric.

Once the elusivity concept has been introduced and described, we proceed to replicate the first case study. Several change period and magnitude combinations are considered to change the same initial problem, and the elusivity values are shown in a two-dimensional representation (heatmaps) based on the period and magnitude combinations, as shown in Figure 2.3. Each cell of the heatmap represents the elusivity of the problem, with a specific period-magnitude setting, to an algorithm. Figures 4.1, 4.2 and 4.3 display in heatmaps the elusivity of DTSPs and DKPs with the fitness landscape rotation, solved by RIACO, EIGA, RIGA and RIPBIL and measured by the best-of-generation metric, $F_{BOG}(P, A)$. The colours in the tables are used for guidance only, where the red colour indicates the problem is elusive to the algorithm; also, the higher its intensity, the larger elusivity of the problem to the algorithm. On the contrary, the more green the colour, the less elusive the problem is to the algorithm.

The following observations can be extracted from the heatmaps. First, they demonstrate that adapting to slightly changing problems (left side) is generally beneficial rather than restarting, although the preference is gradually reversed to the point that restarting after a change is more favourable to severely changing problems (right side).

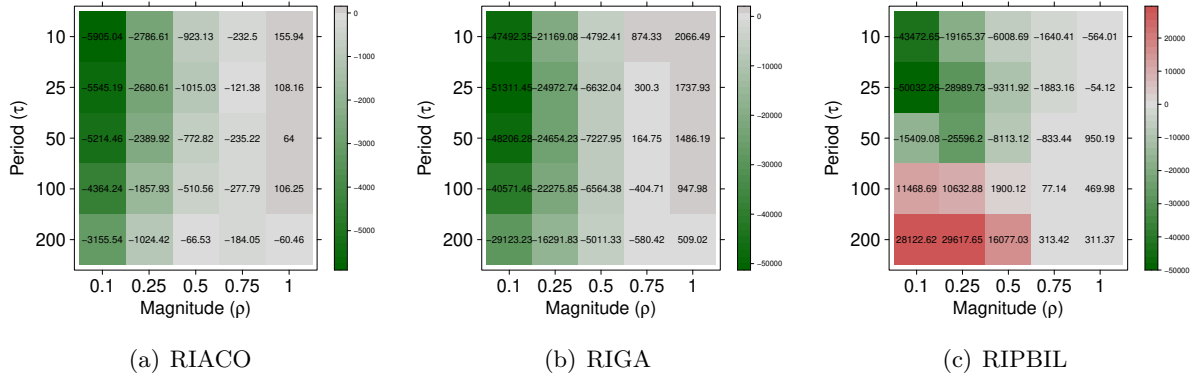


Figure 4.1: Elusivity analysis over different algorithms. The elusivity heatmaps of DTSPs with fitness landscape rotation constructed from kroA150 to RIACO, RIGA and RIPBIL under the best-of-generation are shown.

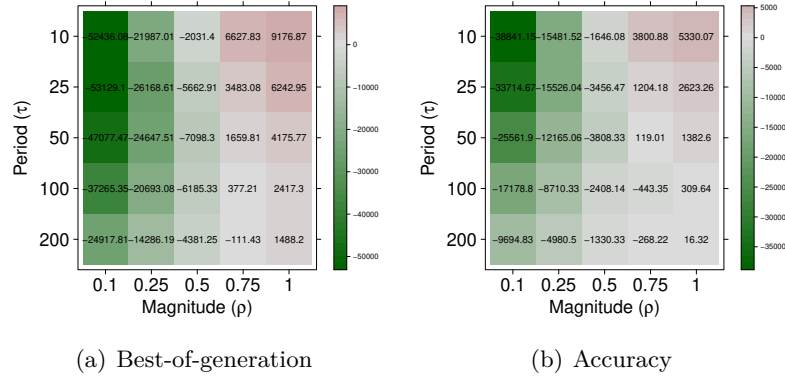


Figure 4.2: Elusivity analysis over different performance metrics. The effect of the performance metric on DTSPs with fitness landscape rotation constructed from kroA150 to EIGA under best-of-generation and accuracy are examined.

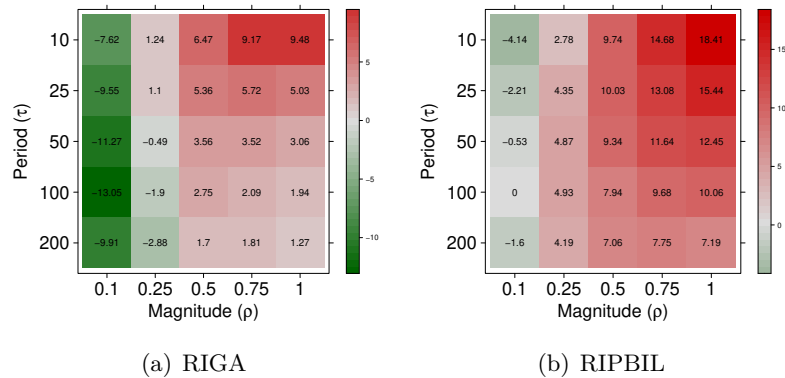


Figure 4.3: Elusivity analysis over different optimisation problems. The elusivity heatmaps of DKPs to RIGA and RIPBIL under the best-of-generation are shown.

This result may sound obvious, since slight changes retain certain similarities of the previous problem instance, and the increase on the magnitude and frequency of changes (decrease of the period) complicates the adaptive advantage of algorithms. However, obtained results show that this statement may not hold for certain cases. For example, DTSPs with fitness landscape rotation changing at $\rho = 0.1$ and $\tau = 200$ (see Figure 4.1(c)) are more elusive to RIPBIL under F_{BOG} than the same problem changing at $\rho = 0.75$ and $\tau = 200$. This situation may be due to two factors. First, even small rotations can significantly impact the fitness landscape, as discussed in Section 3.2.3, making online algorithm adaptation less effective than a simple restart. Moreover, the random immigrants approach may not be effective for slight changes, as new immigrants can introduce an excessive variability to the algorithm search. Hence, since occasional changes may allow algorithms to reconverge, restarting proves more effective.

Second, the images show different elusivity values for every problem and algorithm combination. For example, DTSPs with fitness landscape rotation changing at $\rho = 1$ and $\tau = 200$ are less elusive to RIACO under F_{BOG} than the same problem changing at $\rho = 1$ and $\tau = 10$ (see Figure 4.1(a)); in other words, the adaptive advantage of RIACO is higher in DTSPs changing severely and occasionally than the same problem changing severely and frequently. Therefore, it can be said that the same algorithm proves more or less *elusive* depending on the problem configuration.

Third, from Figure 4.2, it can be seen different elusivity values for DTSPs with fitness landscape rotation to EIGA under both performance metrics, although the general pattern of elusivity holds for both performance metrics. That is, for example, DTSPs that change (rotate) at $\rho \leq 0.5$ are non-elusive to EIGA under F_{BOG} and $H_{\Delta m}$ to different degrees, depending on the frequency and magnitude of the change, and the performance metric considered. Therefore, these observations highlight the effect of the performance metric on the elusivity formulation.

Fourth, by contrasting Figures 4.1(b) and 4.2(a), the influence of the adaptation mechanism on the elusivity of DTSPs with fitness landscape rotation to EIGA and RIGA under best-of-generation can be analysed. Based on this comparison, slightly

different elusivity values of the DTSPs for EIGA and RIGA under F_{BOG} can be observed, respectively, although the general elusivity pattern is certainly similar for both immigrant-based algorithms. That is, although to different degrees, slightly and frequently changing DTSPs are non-elusive to EIGA and RIGA under F_{BOG} , and severely and frequently changing DTSPs are elusive under the same conditions.

Finally, the results show that fitness landscape rotation generates mostly *non-elusive* DTSPs no matter the algorithm used, i.e. adapting to changes is usually beneficial (only 40 problems out of 125, approximately $\frac{1}{3}$ of all problems, are *elusive* according to Figures 4.1 and 4.2). In the case of the DKPs, however, online algorithms are rarely beneficial, since 38 problems out of 50 prove *elusive* to random immigrants-based algorithms (see Figure 4.3). Hence, the inclusion of elusivity analysis adds value by revealing that, under the set conditions, restart is more effective than random immigrant adaptation on DKPs generated by the XOR DOP benchmark generator. Moreover, it is worth noting that DKPs changing at $\rho = 0.1$ and $\tau = 100$ are on the threshold to prove elusive to RIPBIL under F_{BOG} . Hence, in order to validate the experimental results, a statistical analysis is performed in the next section.

4.2.5.1 Statistical Analysis

In order to ensure that such results are still valid when assessing the uncertainty related to the experimentation, a Bayesian statistical analysis, equivalent of the pairwise Wilcoxon signed-rank test⁶, is carried out. This technique estimates the expected probability of two algorithms obtaining the best results (winning probability), given some observations (experimental results), and some prior belief (usually uniformly generated from a Dirichlet distribution [109, 110]). Generally speaking, two algorithms perform similarly (tying probability) if the performance difference between them is within the Range Of Practical Equivalence (ROPE):

$$ROPE = (0, |\mathbb{E}[\phi(A^r, P)]| \cdot \gamma), \quad (4.9)$$

⁶Available in the R package `scmamp` [108].

where $\mathbb{E}[\phi(A^r, P)]$ is approximated with the mean performance of the restarting version of the algorithm A on the problem P under the performance metric ϕ and γ is a variance parameter. Certainly, the γ parameter is used to define the ROPE of the contrasted algorithms in the Bayesian analysis, where a value of $\gamma = 0$ denotes that two algorithms have equivalent performance when the difference in scores is equal to 0. The parameter $\gamma = 0.001$ is set to a relatively low value to consider both algorithms performing similarly. Note that this strategy could be also added to the elusivity formulation by replacing the zero in Definition 4.8 with $|\mathbb{E}[\phi(A^r, P)]| \cdot \gamma$.

Based on this analysis, the previously shown elusivity heatmaps are extended to more general *statistical heatmaps*. Specifically, for each experimental setting, three complementary heatmaps are created: (i) the bottom-left shows the probability of the online algorithm being the winner (green cells), (ii) the bottom-right shows the probability of the restart algorithm being the winner (red cells), and (iii) the heatmap in the top shows the probability of the equivalent performance (blue cells) of both algorithm.

To illustrate the statistical analysis of the elusivity of the results obtained in the previous section, the combination of DKPs with fitness landscape rotation, RIPBIL and best-of-generation are considered (see Figure 4.4). Visually, guided principally by the colours, the online winning heatmap (green) in Figure 4.4 looks similar to the elusivity heatmaps presented in Figure 4.3(b). Nevertheless, the heatmaps in Figure 4.4 demonstrate that most DKPs with fitness landscape rotation prove elusive to RIPBIL in this experimental setting, although for some problems, RIPBIL and RIPBIL^r achieve similar performance. Only two DKPs changing at $\rho = 0.1$ and $\tau = 10, 20$ are clearly *non-elusive* to RIPBIL under F_{BOG} . In contrast, DKPs changing at $\tau = 50, 100, 200$ and $\rho = 0.1$ are on the threshold of being *elusive* to RIPBIL, with winning and tying probabilities for these changes nearly equivalent between RIPBIL and RIPBIL^r . All other DKP configurations are identified as *elusive* to RIPBIL under F_{BOG} . Therefore, it can be stated that the XOR DOP generator primarily produces *elusive* DKPs for RIPBIL under best-of-generation.

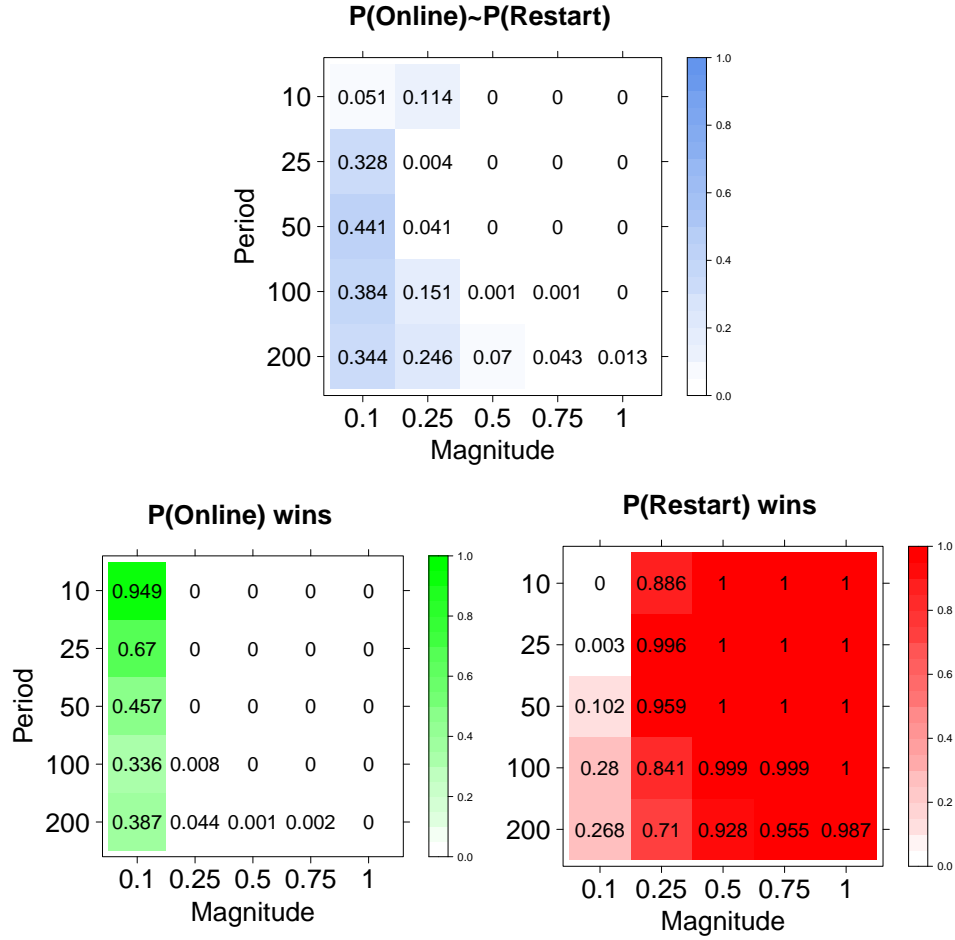


Figure 4.4: Bayesian statistical analysis of elusivity. The heatmaps show the probabilities of the online algorithm winning (green cells, bottom-left), the restart algorithm winning (red cells, bottom-right), and both algorithms performing equivalently (blue cells, top) for solving DKPs with fitness landscape rotation, measured by best-of-generation.

4.2.5.2 Effect of the Parameter Setting on the Elusivity

This study analyses to what extent tuning the immigrant replacement rate affects the performance of immigrant-based algorithms, specifically in relation to the elusivity and adaptive advantage of DOPs and online algorithms, respectively. Previous studies set the replacement rate at 0.2, a typical value used in the literature [36, 44, 45]. Hence, in order to analyse the effect of this parameter on the performance of immigrant-based algorithms, we systematically evaluate a wider range of replacement rates, specifically 0.1, 0.2, ..., 0.9.

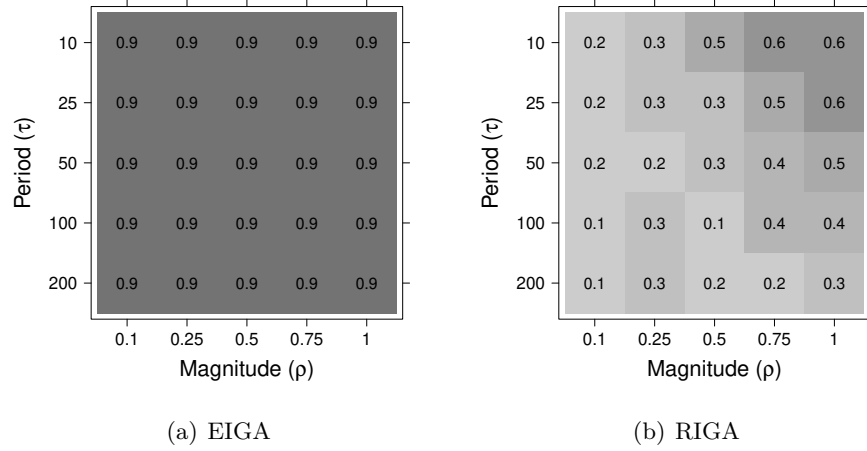


Figure 4.5: Optimal parameter setting based on the elusivity. The immigrant replacement rate of EIGA and RIGA under best-of-generation with the best elusivity (or adaptive advantage) are shown for DTSPs with fitness landscape rotation constructed from `kroA100`.

First, an evaluation of the performance achieved by the immigrants-based online algorithms (for each problem under the considered performance metrics) has been performed to get the optimal immigrant replacement rates. Figure 4.5 shows in heatmaps the optimal immigrant replacement rate setting for EIGA and RIGA to solve DTSPs with different frequency and magnitude of change under best-of-generation, respectively.

Figure 4.5(a) indicates that the highest immigrant replacement rate for EIGA obtains the best performance under F_{BOG} when solving DTSPs with fitness landscape rotation, regardless of the frequency or magnitude of changes. In contrast, smaller replacement rates for RIGA typically perform better under F_{BOG} (see Figure 4.5(b)), although this varies with the frequency and magnitude of changes. Specifically, lower replacement rates are preferable for occasional and slightly changing DTSPs with fitness landscape rotation, while larger rates become more suitable for frequent and significant changes. These findings align with those in [45].

Furthermore, after determining optimal parameter settings for each problem and algorithm, we analyse how the immigrant replacement rate affects problem elusivity

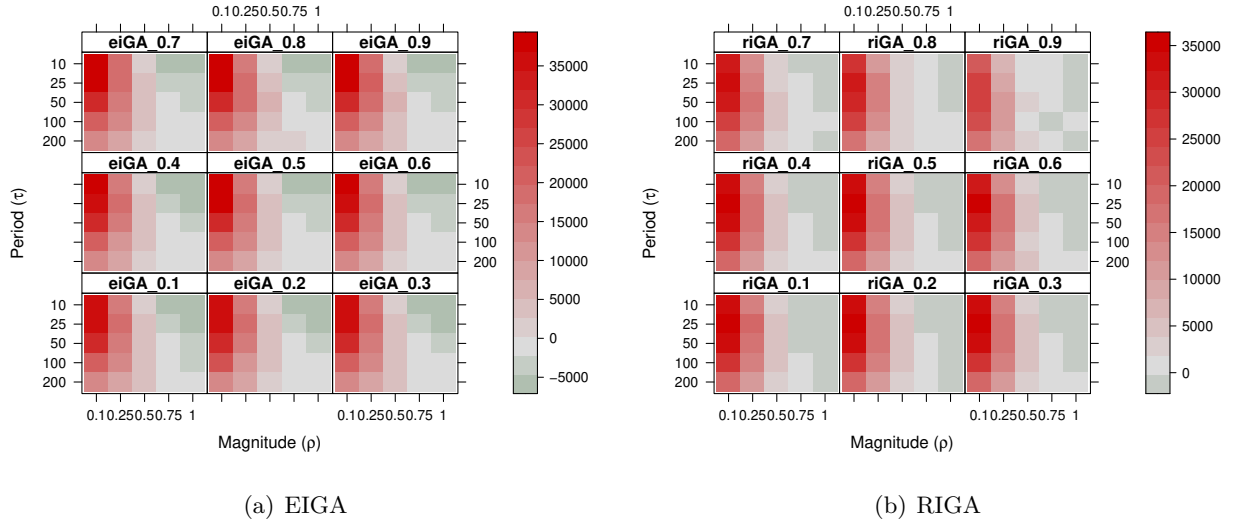


Figure 4.6: Influence of parameters on the elusivity. The elusivity heatmaps illustrate how immigrant replacement rates for EIGA and RIGA, under best-of-generation, affect the elusivity of DTSPs with fitness landscape rotation constructed from `kroA100`.

to algorithms under specified performance metrics. Figure 4.6 presents heatmaps displaying the elusivity values for each EIGA and RIGA under the best-of-generation for each DTSP. The figure illustrates that the elusivity of DTSPs with fitness landscape rotation is sensitive to algorithmic parameters, although the overall elusivity pattern remains consistent. That is, despite slight variations in elusivity values due to changes in the immigrant replacement rate, some problem configurations remain elusive to the algorithms, showing no adaptive advantage even with optimal parameter tuning. Specifically, even with optimal replacement rate tuning (as shown in Figure 4.5), certain DTSPs may still prove elusive to EIGA under F_{BOG} , as shown in the top-right heatmap in Figure 4.6(a).

4.2.5.3 Overall outcome

Despite previous observations confirm the utility of elusivity in quantifying the adaptive challenge of DOPs to online algorithms under a performance metric, they do not specify how to report the elusivity concept in published work. Therefore, the expected probability of the fitness landscape rotation producing *elusive* problems for the algorithms and performance metric has been considered. Informally, we say that an online

algorithm A provides no adaptive advantage over restart, A^r , on problem P under a performance metric ϕ if and only if elusivity values $\mathcal{E}(P, A, \phi) \geq 0$, or if the performance of A and A^r are practically equivalent (see Equation 4.9).

For clarification purposes, let us consider Figure 4.4 as an example. It can be observed that online RIPBIL is superior 4 times out of 25 (i.e. DKPs changing at $\rho = 0.1$ and $\tau = 10, 20, 50, 200$), and ties restart once out of 25 (i.e. DKPs changing at $\rho = 0.1$ and $\tau = 100$). Hence, these results reflect an expected probability of 0.12 to produce non-elusive DTSPs with fitness landscape rotation to RIPBIL under the best-of-generation, and a probability of 0.08 to produce DTSPs with fitness landscape rotation where online and restart RIPBIL are practically equivalent.

That said, the expected probability that permutation-based landscape rotation creates *elusive* DTSPs is 0.3. In the case of the XOR DOP, the expected probability is 0.66 that the generated problems present adaptive challenge to the used algorithms and performance metrics, whereas 0.14 represents practical equivalence.

Appendix A provides a detailed analysis of the winning and tying counts and probabilities for each algorithm version in the experiments conducted.

Now that we can ensure the generation of dynamic, but non-elusive, problems to algorithms, we focus on extending the elusivity concept to quantify the effectiveness of adaptation mechanisms or comparing algorithm performances regarding the elusivity.

4.2.6 Case Study II

In the previous section, it has been demonstrated that the elusivity of problems generated by the fitness landscape rotation, which are mainly used for academic purposes, varies with the change period and magnitude tuning, as well as with the selected algorithm. In this section, the goal is to make the most of the elusivity formulation to capture and evaluate the adaptive advantage and behaviour of online algorithms on a more realistic framework. The idea is to use the elusivity concept to measure the extent to which adaptation improves or degrades algorithm performance in comparison to restart.

The experiments carried out are reproduced from a state-of-the-art research work [35], where DTSPs with different dynamisms (traffic factor and city replacement), four ant colony variants (MMAS, PACO, EIACO and MC-MMAS) and three performance metrics are used. The authors categorise changes as *fast* when the change period is $\tau \leq 2.5n$, and *slow* for $\tau \geq 25n$. Similarly, they refer a change to be *small*, *medium* or *large* based on the magnitudes $\rho \in \{0.1, 0.25, 0.5, 0.75\}$. However, it can be observed that change magnitudes are unevenly distributed across categories, as four values ($\rho \in \{0.1, 0.25, 0.5, 0.75\}$) are assigned to three categories (*small*, *medium* or *large*). The precise parameters for this experimentation are detailed in Table 4.1.

In summary, the paper states that adaptation mechanisms enhance the adaptive advantage of algorithms, although their performance depends on the settings (dynamism) of the problem. Besides, they exhibit the following observations. First, PACO and EIACO outperform MMAS and MC-MMAS for most quickly changing DTSPs under F_{BOG} . Second, MMAS and MC-MMAS perform better than PACO and EIACO for most slowly changing DTSPs under F_{BOG} and $P_{\Delta m}$, although it is gradually inverted with the increase of problem size. Third, all algorithms obtain very good results under \mathcal{R} . Fourth, the restarting version of the algorithms are not effective for DTSPs with traffic factor when changes do not affect the best solution found, since it would result in an undetected change for the restarting algorithm.

Finally, authors assure that the following statements are consistent with the observations found in their previous studies [34]: (i) MC-MMAS is competitive with MMAS, i.e. both maintain a competitive performance; (ii) EIACO generally outperforms PACO; and (iii) PACO gradually outperforms MMAS as problem size increases.

4.2.6.1 Elusivity Analysis

First of all, note that the aim of this experimentation is to illustrate and study the application of the elusivity concept to quantify the adaptive advantage of online algorithms over restarting the search after detecting a change. To that end, it has been decided to extend the change period and magnitude settings used in [35], and show the

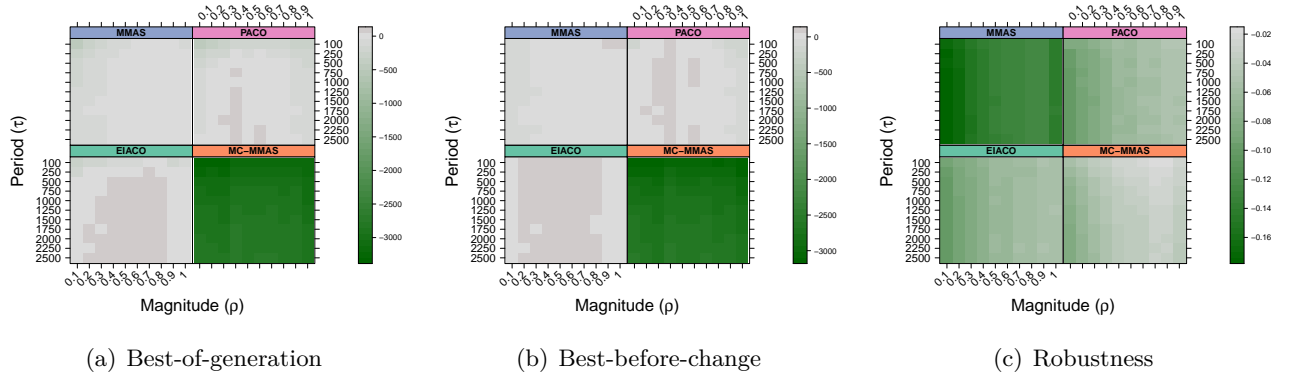


Figure 4.7: Elusivity heatmaps of DTSPs with city replacement to four algorithms under three different performance metrics.

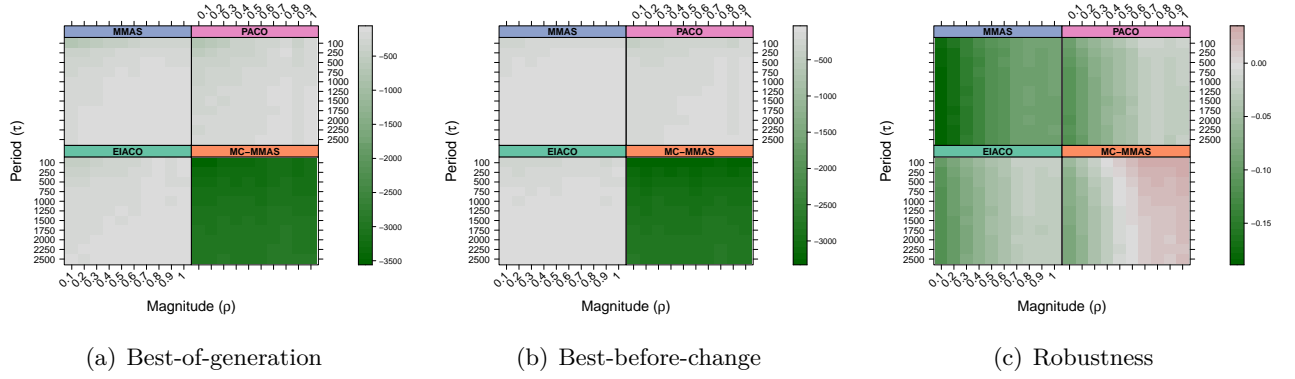


Figure 4.8: Elusivity heatmaps of DTSPs with traffic factor to four algorithms under three different performance metrics.

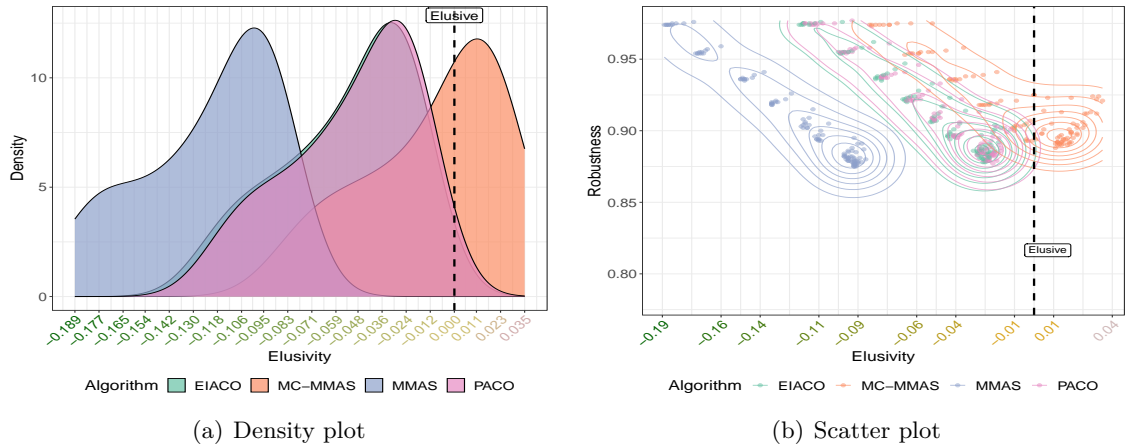


Figure 4.9: Elusivity as a measure for adaptive advantage algorithms on DTSPs with traffic factor under robustness. The elusivity threshold (zero) is represented by a vertical dashed line, and the algorithms by the colour of the header of the heatmaps.

elusivity values in heatmaps to accurately highlight the adaptive advantage of online algorithms to solve DTSPs with different settings. Figures 4.7 and 4.8 capture the elusivity of the DTSPs with city replacement and traffic factor, respectively, to the algorithms under three performance metrics.

Figures 4.7(a), 4.7(b), 4.8(a) and 4.8(b) demonstrate that most DTSPs (either for city replacement or traffic factor) prove *non-elusive* to the algorithms under F_{BOG} and $P_{\Delta m}$. Generally, MC-MMAS exhibits lower elusivity than PACO, MMAS and EIACO under F_{BOG} and $P_{\Delta m}$, meaning that the adaptive advantage over its restarting version is large under these performance metrics. The heatmaps for PACO, MMAS and EIACO reveal that adaptation is slightly better than restarting the search for slight and frequent changing DTSPs, although both online and restarting versions obtain similar results in general. Hence, DTSPs with city replacement or traffic factor are in the threshold to become elusive to PACO, MMAS and EIACO under F_{BOG} and $P_{\Delta m}$, respectively, since the adaptation mechanisms for these algorithms confer little or no advantage over their restart for occasional and severe changes.

In the same way, Figure 4.7(c) shows that DTSPs with city replacement prove non-elusive to the algorithms under \mathcal{R} . However, Figure 4.8(c) shows that 49 DTSPs with different traffic factor prove elusive to MC-MMAS under \mathcal{R} . That is, the robustness of MC-MMAS deteriorates with the increase of the traffic factor, to the point that DTSPs with medium to severe traffic factor become elusive to MC-MMAS under \mathcal{R} . This particular case allows demonstrating the role of performance metrics in the elusivity, in addition to the problem and the algorithm. Nevertheless, it is worth noting that most empirical studies in dynamic optimisation design and compare algorithms that aim to optimise the best-of-generation. In fact, as stated in [111], robustness and best-of-generation metrics conflict with each other, where algorithms perform better in terms of F_{BOG} on less robust problems.

4.2.6.2 Elusivity as a Measure for the Advantage of Adaptive Mechanisms

So far, obtained results have been presented and analysed using elusivity heatmaps to quantify the advantage that adaptation brings over a restart of the algorithm. Nevertheless, heatmaps fall short when analysing the effectiveness and sensitivity of the algorithms, and comparing the best performance of each version of algorithms against other algorithms'. Therefore, the elusivity analysis is extended using density plots and scatter plots.

For illustration purposes, the paradigmatic case study (Case Study II) is considered to analyse the relation between the robustness and elusivity of the four ACO algorithms for DTSPs with traffic factor in more depth. It is worth noting that the observations drawn in the following analysis may not hold for other problem, algorithm and performance metric combinations. In fact, as mentioned in [111], the observations drawn from robustness may conflict with the ones from best-of-generation, thus affecting the elusivity analysis. In Figure 4.9, the elusivity heatmap in Figure 4.8(c) is supported with a density plot and a scatter plot. The density plot in Figure 4.9(a) uses a kernel probability density to estimate the elusivity distribution for DTSPs with traffic factor to each algorithm under \mathcal{R} , described in Equation 4.8, on the same set of DTSPs (see Table 4.1). The scatter plot in Figure 4.9(b) shows the relation between the robustness obtained by the best algorithm version (online or restart) for each problem setting, and the adaptive advantage obtained by each algorithm.

Figure 4.9(a) reveals that PACO, MMAS and EIACO generally prove *non-elusive* under \mathcal{R} , whereas the performance of MC-MMAS varies with the frequency and magnitude of change. That is, from the heatmaps, it can be observed that DTSPs with traffic factor changing at $\rho = 1$ and $\tau = 100$ prove *elusive* to MC-MMAS under \mathcal{R} , whereas problems changing at $\rho = 0.1$ and $\tau = 100$ prove non-elusive. Therefore, it can be said that, for these change configurations, the adaptation mechanisms for MC-MMAS is giving little or no advantage in terms of robustness.

Similarly, densities also demonstrate that PACO, MMAS and EIACO have a more concentrated elusivity distribution, whereas the elusivity of DTSPs with traffic factor to

MC-MMAS under \mathcal{R} is more variable. This might be because of (i) the high variability of adaptation mechanisms, that stand out depending on the period and frequency setting of changes; or (ii) a bad tuning of adaptation mechanism parameters, such as immigrant replacement rate or the memory size in EIACO or PACO, for example.

Figure 4.9(b) displays the relation between the robustness of the best version for each algorithm and their respective elusivity in a scatter plot. Points on the left-side of the drawn vertical line (elusivity lower than zero) represent a better performance of online algorithms over their restarting version (PACO, MMAS, EIACO and MC-MMAS), and the opposite for right-sided points (PACO^r, MMAS^r and EIACO^r). From the plot, it can be said that DTSPs with traffic factor prove *non-elusive* to PACO, MMAS and EIACO under \mathcal{R} , and also for problems changing at $\rho \leq 0.4$ to MC-MMAS. The figure also shows the line of best fit for each algorithm, aiming to highlight the elusivity and performance trend of algorithms with respect to the configuration of the changes. In a certain way, this chart shows the elusivity distributions in Figure 4.9(a) from a top view, and the depth is determined by the robustness of the algorithms. Recall that robustness is a maximisation measure, so larger robustness means a better performance of the algorithm.

From Figure 4.9(b), a similar pattern for all algorithms can be observed when measured under robustness, where slightly changing DTSPs are concentrated in the upper part of the plot and severe changes at the bottom, although all algorithms differ in the elusivity distribution. Aforementioned, PACO and EIACO usually prove *non-elusive* for DTSPs with traffic factor under \mathcal{R} , although points are concentrated close to the threshold to become *elusive* (elusivity close to zero) as the frequency and magnitude of change increase. MC-MMAS^r proves more or less *elusive* depending on the setting of changes under \mathcal{R} , although its robustness is certainly maintained, no matter the frequency and magnitude of change. That said, it can be concluded that, in this case study, the adaptation mechanisms for MC-MMAS may be disadvantageous for some frequencies and magnitudes of change when measured under robustness, although the algorithm is quite robust by nature. Nevertheless, note that these observations are

drawn from a particular illustration of the obtained results, and they are limited to the exposed experimental setup. Finally, it is interesting to note the trend on the distribution of the algorithms: the less elusive, the more robust become the algorithms. Therefore, it can be inferred that the adaptive advantage is influenced by the change frequency and magnitude setting.

From these observations, PACO, MMAS and EIACO are less robust than MC-MMAS^r for some change settings, i.e. except from the DTSPs changing at $\rho \geq 0.5$, even a restarting behaviour of MC-MMAS can be more effective than PACO, MMAS and EIACO under \mathcal{R} . These insights demonstrate that adaptation mechanisms do not always improve the robustness of algorithms, since PACO, EIACO and MMAS are less robust than MC-MMAS^r for DTSPs with traffic factor changing at $\rho \geq 0.5$. Obtained results suggest the inclusion of the restart, in future research, to avoid erroneous evaluation of algorithms in elusive problems where there is no adaptive advantage.

Finally, an interesting observation that cannot be ignored is that, for each algorithm, points are grouped based primarily on the magnitude of changes, but also on the change period. This statement allows for the classification of problems according to the performance and adaptive advantage of algorithms under a performance metric. For example, from Figure 4.9(b), 7 distinct groups can be perceived for all algorithms under \mathcal{R} . However, the characterisation of groups varies by the algorithm. That is, for MMAS, EIACO and PACO, the increase in the change magnitude is related to the variability in the robustness and the concentration of elusivity of groups. In the case of MC-MMAS^r, robustness is maintained throughout each change period, regardless of the magnitude of changes, while the elusivity of groups varies with the magnitude of change. That is, the more severe the change, the more effective is the restart over the adaptation for MC-MMAS.

4.2.6.3 Overall outcome

This case study has demonstrated that the elusivity formulation can be extended to quantify the adaptive advantage (and degradation in performance) of online algorithms against restart. Specifically, the case study presents various visual representations that reveal that (i) heatmaps effectively illustrate the elusivity degree of DOPs, varying their change frequency and magnitudes, to an algorithm and a performance metric, (ii) density plots highlight the adaptive challenge and advantage of online algorithms across a set of DOPs, and (iii) scatter plots allow for a precise comparison of the performance and adaptive advantage of online algorithms over restart.

4.3 Summary

Chapter 4 meets the research objective **OB 3** by quantitatively measuring the probability of generating elusive problems, where restarting is preferred over adaptation, using existing dynamic benchmark generators with different change frequencies and magnitudes.

The field of dynamic optimisation presents a wide variety of online algorithms with adaptation mechanisms to solve problems that change over time. Empirical works often compare the performance of algorithms under different configurations of the frequency and magnitude of changes, but often ignore whether the problem can be effectively solved by an online algorithm.

The mathematical formulations and the systematic elusivity analysis of this chapter have proved the validity of the elusivity concept to (i) distinguish *elusive* and *non-elusive* problems based on the adaptive advantage of algorithms to changes, and (ii) evaluate the advantage of the adaptation mechanisms over the restart. In particular, this study has emphasised the limitation to compare algorithm performance under change classifications based only on the frequency and magnitude of changes, since the dynamism also depends on the adaptation challenge of algorithms to deal with changes and the performance metric used. In fact, performance metrics are usually neglected in previous research on this topic, but they are crucial to a full definition of the elusivity.

The conducted experiments have demonstrated the existence of elusive problems in already published studies [11, 12, 35, 44, 103] to different extents according to the problem, algorithm, and performance metric combinations. Therefore, this work suggests systematically including the restarting version of the algorithms in the experimentation to eliminate erroneous study of algorithms with disadvantageous adaptation mechanisms in future research.

Note that presented definitions do not make any assumptions about the problem, algorithm, performance metric, representation, type of dynamism, and adaptation mechanisms. Hence, the presented elusivity concept can be applied into any DOP with “detectable” changes.

Chapter 5

Data-Driven Analysis for a Real-World Dynamic Scheduling Problem

The studies in the previous chapter, and the literature on dynamic optimisation, have primarily focused on benchmark generators for permutation problems, which have been used to construct dynamic optimisation problems as sequences of related static optimisation problem instances. Such generators are generally designed for empirical testing in academic research, where the generated dynamisms often fail to capture the complexities of real-world applications. In fact, the development of dynamic benchmark generators that recreate the dynamism observed in real-world situations is still a fundamental challenge for the field of dynamic optimisation [3, 4, 9, 42].

This thesis aims to address this research gap by introducing a preliminary benchmark generator that generates synthetic dynamic truck and trailer scheduling problem instances based on real-world historical data. Similar to other real-world scheduling problems that show different types of dynamism [15, 112, 113], the considered scheduling problem is formulated as a permutation problem [14]. Bui *et al.* [112] highlight that, generally, parameter variations (e.g. fluctuating task durations due to equipment issues or weather) cause minor repercussion, whereas changes in resource availability

(e.g., equipment breakdowns or maintenance) and constraints (e.g. shifts in task priorities) tend to be more severe, making previous solutions infeasible. The authors also note that, for such severe changes, a complete restarting approach may be beneficial, although more advanced adaptive approaches are generally preferred.

For the sake of clarity and readability, we have divided our contribution into two distinct chapters. In this first chapter, we examine a dataset, formulate a real-world optimisation problem, and perform a descriptive analysis to identify and characterise the dynamic features of the problem [8, 68]. In Chapter 6, we propose a synthetic data generation method and a an initial methodology applicable to other real-world scenarios.

The data examined in this study has been provided by a transportation and logistics company, ARR Craib Ltd (ARRC). This local transportation company specialises in the subsea, oil & gas, pallet network and construction sectors in the North East Scotland, although it also covers operations throughout the UK. A wide and heterogeneous fleet of vehicles and a network of locations covers road transport and storage.

In 2016, Regnier-Coudert *et al.* [14] collaborated with ARRC and developed an automated framework that allowed operators to track the vehicle fleet and interact with drivers in real time to improve the decision-making process. As a result, ARRC has been able to fully store their historical logistical operations, preserving comprehensive records of requested resources, the driver skills required to perform the operations, and detailed description of the assignments, including their respective timings. The company has granted us access to their historical records to support our collaboration in creating a parametrisable synthetic benchmark generator that reproduces the characteristics and patterns of the original data without revealing commercially sensitive information.

The chapter is structured as follows. Section 5.1 provides a detailed exposition and methodical formulation of the dynamic and realistic optimisation problem at hand. The aim is to establish the theoretical foundation of the optimisation problem and to determine the necessary variables in the data for the generation of synthetic instances. Then, Section 5.2 describes the data provided by the company, and Section 5.4 extends

the analysis of the data to reveal some of its properties and hidden patterns. This analysis will help us identify the expected insights from instances that are not only statistically representative, but also temporally consistent. Finally, Section 5.5 provides a summary of the insights gained from the description and analysis of the data.

5.1 Dynamic Truck and Trailer Scheduling Problem

The optimisation problem considered in this study can be seen as a variation of a real-world, uncertain and heterogeneous dynamic vehicle routing problem [16], where the aim is to find an optimal schedule to meet all customer deadlines, reducing the transportation costs and increasing the efficiency of vehicle fleet management.

Operators (decision makers) were originally responsible for meeting customer demands and organising the priority and assignment of jobs to trucks and trailers, which were constantly changing due to the arrival of new ones, making it a major scheduling challenge. Regnier-Coudert *et al.* [14] developed an *offline* constructive algorithm, which treats each change on the DOP as a different static optimisation problem, to balance efficiency and flexibility in the decision-making process to improve the adaptive advantage of algorithms to problem changes. Therefore, proper assignment resulted in promoting sustainability in the operations to meet customer requirements and deadlines, as well as reducing the environmental impact.

The considered optimisation problem presents some characteristics that make benchmark generators worth investigating and designing [8, 68]. Some features of the problem are listed below.

- **Dynamic.** The main dynamism of the problem comes from the arrival of new jobs over time, which affects the dimensionality of the search space. Specifically, these jobs (eventually, we refer to them as *dynamic* jobs) are received during working hours and require immediate action, particularly when the requested completion time is within the same day. Alternatively, dynamic jobs that requested a posterior completion date can be classified as *static* jobs, which means that they can be postponed until the requested completion date, and planned in advance, i.e. before the start of the working day.

Furthermore, the dynamism varies significantly depending on the time of day, week, or month (seasonalities). For example, weekdays (Monday to Friday) generally have more jobs (mostly dynamic jobs) than on weekends, except on certain days, such as Christmas or New Year.

- **Heterogeneous.** The heterogeneity of the considered problem can be found in two ways. On the one hand, dealing with geographic heterogeneity is a typical challenge encountered in real-world applications [114]. As it will be explored in Section 5.2.1, the locations where jobs are executed are distributed across North East Scotland, with a notable concentration in Aberdeen and Peterhead.

In addition, heterogeneity can be attributed to the available number of resources (truck, trailer, and drivers), which varies daily due to maintenance or unavailability of drivers.

- **Constrained.** The problem presents a variety of static hard and soft constraints related to meeting the deadlines specified by the customers and international regulations, as well as constraints on resource management. For example, some jobs present a *driver-skill* constraint, which means that they can only be performed by certain types of trailers or by drivers with specialised handling skills (i.e. European Agreement concerning the International Carriage of Dangerous Goods by Road¹).
- **Realistic.** This feature is probably the biggest contribution to the field, as it bridges the research gap between academia and real-world. Indeed, the field of dynamic optimisation has primarily used synthetic or simulated instance generators that, while useful for initial analysis, often fail to capture the complexity of real-world scenarios. Therefore, obtained outcomes are not only theoretically important, but also practically applicable.

¹<https://www.gov.uk/guidance/driving-dangerous-goods-and-special-loads>

It is worth noting that the problem considered is primarily characterised by its dynamic nature. Hence, in order to construct a valid benchmark generator that accurately reflects these dynamics, it is essential to thoroughly analyse and understand the types of changes. The changes in the considered problem can be broadly classified by the following attributes:

- **Cyclicity:** certain seasonality components in the data are repeated daily and weekly, such as the input, collect and deliver times of the jobs.
- **Frequency:** although there is not a predefined fixed frequency of changes, daily variations might be estimated from existing seasonality components.
- **Predictability:** future changes can be derived from the seasonality of the data, e.g. the input time of jobs can be predicted from the daily distribution of the input time of jobs.
- **Detectability:** the arrival of new jobs is easily detected by the system, so algorithms would not need to deal with the detection of problem changes over time.
- **Dimensionality:** it is influenced by the completion of jobs and the arrival of new ones, the latter decreasing every time each job is performed and increasing with each new job. Therefore, the dimension of the search space varies over time, which intrinsically influences the objective function.
- **Unknown optimum:** since the historical decision given by the operators is not considered, optimal solutions are not available.
- **Time-linkage:** there is a temporal dependency between the different stages of the problem, since previously made decisions directly affect future assignments.

5.1.1 Mathematical Formulation

This section formulates the problem considered using a mixed-integer linear programming model, adapted from [14], although it varies slightly due to the structure of the data. First, the time window of each job only considers the earliest collection and the latest delivery times, rather than having separate windows for collection and delivery tasks, respectively. Second, the considered problem does not differentiate between job types, such as inbound or outbound.

The model applies variables, invariants, and constraints to capture all requirements, resulting in a sophisticated model of 11 constraints. Note that this problem is a variation of the vehicle routing problem with pickup and delivery [16], in which jobs and trailers are considered separate “objects” to be picked up and delivered between different locations by trucks [113].

Let us assume that the working hours of the company are determined by the variables W_s and W_e , which refer to the opening and closing times of the company, respectively.

Let $J = \{J_1, \dots, J_n\}$ be a set of jobs, where each job J_i contains a collect time J_i^c and a delivery time J_i^d , J_i^s denotes the expected execution time of the job J_i and J_i^t is the time when job J_i starts. The distance between two successive jobs, J_i and $J_{i'}$, is denoted by $\delta_{i,i'}$.

Let $T = \{T_1, \dots, T_m\}$ be a set of trailers, and a trailer T_j is meant to be subcontracted by the notation $T_j^s = 1$. Similarly, let $L = \{L_1, \dots, L_l\}$ be a set of trucks, where a truck is subcontracted when $L_k^s = 1$. Let define the function $\mathcal{T}(L_k)$ to return the trailer associated with a truck L_k . Note that certain trucks are rigid, i.e. they have a trailer constantly attached to them. These types of truck are denoted as $L_k^r = 1$, which means that a trailer is associated with the truck L_k . Similarly, $\mathcal{C}_{J_i, T_j} = 1$ shows that J_i fits in the trailer T_j according to its capacity.

The distance travelled between a trailer T_j and a job J_i is denoted as δ_{J_i, T_j} , and the distance travelled between a trailer T_j and a truck L_k is indicated as δ_{T_j, L_k} . Similarly, the travel time between a job and a trailer, as well as the time between a trailer and a truck, is based on the travel distance and the maximum speed of the trailer type. The function $\tau(J_i, T_j)$ represents the travel time between job J_i and trailer T_j , and $\tau(T_j, L_k)$ the travel time between a trailer T_j and a truck L_k , respectively.

Let $S = \{S_1, \dots, S_w\}$ be the set of skills required to transport certain jobs. In order to deal with driver-skills constraints, the function \mathcal{S} is used. That is, $\mathcal{S}^{job}(J_i, S_s) = 1$ determines whether the job J_i requires the skill S_s , and $\mathcal{S}^{truck}(L_k, S_s) = 1$ if the driver assigned to the truck L_k has the skill S_s .

Finally, the decision variables $X_{J_i, T_j} = 1$ and $Y_{J_i, L_k} = 1$ represent if the trailer T_j and the truck L_k are assigned to job J_i , respectively.

Therefore, the objective function can be formulated as follows:

Minimise:

$$\sum_{i=1}^n \max \left\{ \sum_{j=1}^m X_{J_i, T_j} \cdot T_j^s, \sum_{k=1}^l Y_{J_i, L_k} \cdot L_k^s \right\}, \quad (5.1)$$

subject to

$$\sum_{j=1}^m X_{J_i, T_j} = 1, \quad \forall J_i \in J, \quad (5.2)$$

$$\sum_{k=1}^l Y_{J_i, L_k} = 1, \quad \forall J_i \in J, \quad (5.3)$$

$$J_i^c \leq J_i^t + J_i^s \leq J_i^d, \quad \forall J_i \in J, \quad (5.4)$$

$$W_s \leq J_i^t + J_i^s \leq W_e, \quad \forall J_i \in J, \quad (5.5)$$

$$J_i^t + J_i^s + \delta_{J_i, J_{i'}} - J_{i'}^t \leq (2 - X_{J_i, T_j} - X_{J_{i'}, T_j}) \cdot M, \quad \forall J_i \neq J_{i'} \in J, J_i^t < J_{i'}^t, \forall T_j \in T, \quad (5.6)$$

$$J_i^t + J_i^s + \delta_{J_i, J_{i'}} - J_{i'}^t \leq (2 - Y_{J_i, L_k} - Y_{J_{i'}, L_k}) \cdot M, \quad \forall J_i \neq J_{i'} \in J, J_i^t < J_{i'}^t, \forall L_k \in L, \quad (5.7)$$

$$\tau(T_j, L_k) + \tau(J_i, L_k) - J_i^t \leq (2 - X_{J_i, T_j} - Y_{J_i, L_k}) \cdot M, \quad \forall J_i \in J, \forall T_j \in T, \forall L_k \in L, \quad (5.8)$$

$$Y_{J_i, L_k} \cdot \mathcal{S}^{job}(J_i, S_s) \leq \mathcal{S}^{truck}(L_k, S_s), \quad \forall J_i \in J, \forall L_k \in L, \forall S_s \in S, \quad (5.9)$$

$$X_{J_i, T_j} - \mathcal{C}_{J_i, T_j} \leq 0, \quad \forall J_i \in J, \forall T_j \in T, \quad (5.10)$$

$$Y_{J_i, L_k} \cdot L_k^r \leq X_{J_i, \mathcal{T}(L_k)} \cdot L_k^r, \quad \forall J_i \in J, \forall L_k \in L, \quad (5.11)$$

where M is a large enough constant that is calculated as follows:

$$M > \max \left(\max_{J_i, J_{i'} \in J} (J_i^t + J_i^s + \delta_{J_i, J_{i'}}), \max_{J_i \in J, T_j \in T, L_k \in L} (\tau(T_j, L_k) + \tau(J_i, L_k)) \right).$$

The objective function 5.1 aims to minimise the number of subcontracted jobs that are either performed by subcontracted trucks or subcontracted trailers, subject to the following constraints.

Constraints 5.2 and 5.3 ensure that a job is performed by one and only one trailer and truck, respectively.

Constraints 5.4–5.8 are all related to the timing of jobs. Specifically, constraint 5.4 specifies that all jobs are collected and delivered on time. The fact that a job is performed during the working hours of the company is determined by constraint 5.5. Constraints 5.6 and 5.7 ensure that a trailer and a truck perform a job at a given time, respectively. Constraint 5.8 guarantees that a truck has enough time to collect a trailer and a job on time.

Finally, constraints 5.9–5.11 ensure that the assigned resources are compatible with the job in terms of driver-skill capability and compatibility. In particular, constraint 5.9, referred to as driver-skill constraint, links the skills of the driver assigned to the truck to those required to perform the job. Constraint 5.10 states that the assigned trailer can perform the specified job. Finally, constraint 5.11 requires that a job assigned to a rigid truck also be performed by the trailer associated with the rigid truck.

5.2 Data Description and Preprocessing

The use of real-world data highlights the importance of collecting, managing and analysing data in the optimisation process. In order to manage the data, it is essential to use cleaning, filtering and processing techniques to convert the raw data into a formalisation of the optimisation problem. This section provides a detailed description of the data, which is crucial to ensure that synthetic data generation models preserve the relevant variables of the optimisation problem instances.

ARRC has given us access to their historical database, which includes full information of jobs carried out in 2019, as well as their historical record of resources (drivers, trucks, and trailers) and locations and their definition of driver-skill constraints. Overall, the data serves as a realistic and dynamic representation of historical operations,

which are inherently influenced by the temporal nature of the logistics workflow. The data provided can be organised into the following datasets:

- The *job dataset* serves as the primary source of data to recreate historical operations. Each job contains a set of tasks, which specifies the nature of the actions, their location, and their precise execution time (e.g. “*collect* a cargo from *A* at 10 a.m.”). Moreover, each job encompasses the driver-skill constraints, the requested collection and delivery times, and the preferred type of trailer, specified by the customer. In addition, there is a history of the resource (drivers, trucks and trailers) assigned to these jobs, although this information is beyond the scope of this study.
- The *location dataset* indicates the name and precise geographic coordinates where the tasks are performed.
- The *constraints dataset* captures all job-specific requirements (skills) that the drivers must meet for the execution of the jobs.
- The *driver dataset* documents the personal information and skills of the drivers employed by ARRC, which is crucial for the execution of jobs with specific driver-skill constraints.
- The *trailer dataset* represents the trailer fleet, encompassing each identifier (registration number), the geographic location and type of each trailer.
- The *trailer type dataset* classifies trailers according to their capacity and specifications, such as flatbeds and curtainside trailers.
- The *truck dataset* contains information on the truck fleet, including the identifier (registration number) of the truck, its geographic location, and the details of the assigned driver and trailer, if applicable.

Figure 5.1 presents an entity relationship diagram to show the structure and relationship between datasets after filtering, cleaning, and processing the data in the following sections. Note that this figure should only be used as a visual representation of the relationship between the datasets, as the number of variables in each dataset is significantly reduced after the preprocessing step (see Section 5.3).

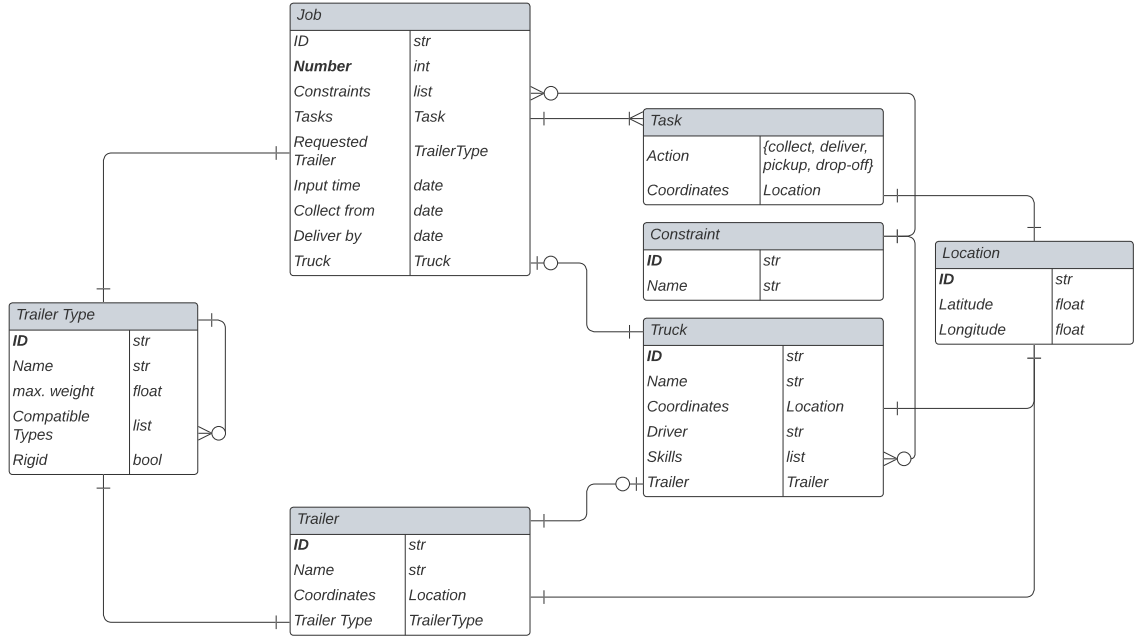


Figure 5.1: Entity relationship diagram showing the relationships of the data after preprocessing.

The following sections provide a complete description of each dataset and describe the data cleaning, filtering and processing steps separately, considering only the variables that are essential for the defined case study.

5.2.1 Historical Actions and Driver-Skill Constraints

Provided data consists of a total of 41,939 jobs completed in 2019, varying in complexity, i.e. from jobs with simple collection and delivery tasks to more complex multitasking jobs. Note that tasks are part of jobs, and define the action and the location where they must be performed. The tasks considered in this study are the following:

- *Collection* tasks involve collecting of cargo from a predetermined location.
- *Delivery* tasks encompass the delivery of a cargo to an assigned destination.
- *Pickup* tasks refer to the specific action of securing a loaded trailer from an established location. These tasks are generally associated with actions in Aberdeen Harbour.

Table 5.1: Description of the attributes of the preprocessed job dataset.

Attribute	Description
Job ID	A unique identifier for each job used to identify the records in the original database.
Job number	A unique number for each job that may be used as an alternative identifier.
Constraints	A list of driver-skill requirements to perform each job, which can be null if none exist.
Input time	The specific time in which the job was received and inserted in the system.
Requested times	The time-window requested by the customer to perform the jobs, denoted as <i>collect from</i> and <i>deliver by</i> . Moreover, the customer can add flags to indicate that a job must be collected or delivered on the specific time requested, denoted as <i>on time collect</i> and <i>on time delivery</i> .
Requested trailer	The trailer type requested by the customer to perform the job.
Tasks	A list of tasks that represent the job, where each task owns the following attributes: <ul style="list-style-type: none"> - Task ID: the action that must be performed, e.g. <i>collect</i>, <i>deliver</i>, <i>pickup</i> or <i>drop-off</i>; - Location ID: the identification of the location where the task is to be executed.

- *Drop-off* tasks entail the placement of a loaded trailer at a designated location, which is often linked to Aberdeen Harbour.

The total number of tasks performed by ARRC during 2019 is 88,340.

In addition to task-related information, each job has a detailed description about the customer, the resources that were used for the tasks, and the timings of the job process. Nevertheless, the job dataset has been refined to include only the essential attributes for the generation of artificial information or the simulation of a typical working day. Table 5.1 describes the job dataset after filtering the information that is trivial to the generation of synthetic instances. It is essential not only to list and describe the attributes, but also to analyse and understand their characteristics, which is presented in the following section (Section 5.4).

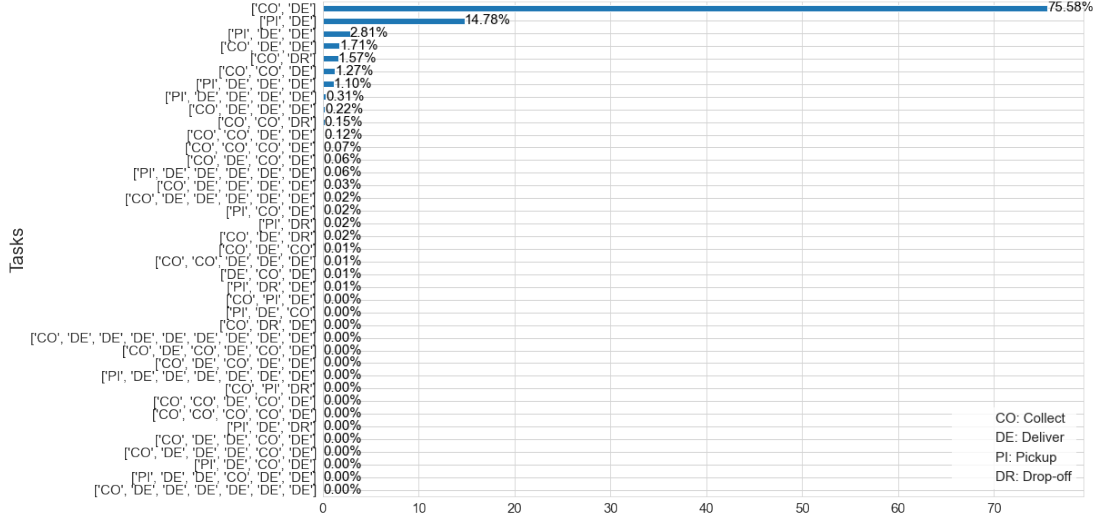


Figure 5.2: Frequency of tasks on the preprocessed job dataset.

The dataset has been filtered to only consider the jobs entered and performed in 2019. That is, the jobs entered in 2018 and the jobs scheduled for 2020 have been excluded for the synthetic data generation process. As a result, the total number of tasks has decreased from 88,340 to 87,859. In order to maintain consistency in task scheduling, another time-related property of the dataset is to ensure that the input time of a job can be no later than the collection time. Furthermore, the collection time must occur strictly before the delivery time. Therefore, in order to clean invalid jobs from the dataset, jobs that do not meet this property have been eliminated, decreasing the total number of tasks to 87,185. Therefore, this resulted in the job dataset containing 87,185 tasks (distributed over 41,391 jobs), representing approximately 98.5% of the jobs in the original data.

Figure 5.2 shows the frequency of tasks per job in 2019. As can be observed from the figure, around 95% of the filtered jobs are two-tasks (i.e. *collect* and *deliver*, *pickup* and *deliver*, *collect* and *drop-off* and *pickup* and *drop-off*), whereas the remaining 5% consists of three or more tasks per job.

Another important aspect of the data is the geographical location of the job tasks, which can be extracted from the job dataset, as can be observed in Table 5.1. Locations operate as nodes within the company’s logistical network. The provided data

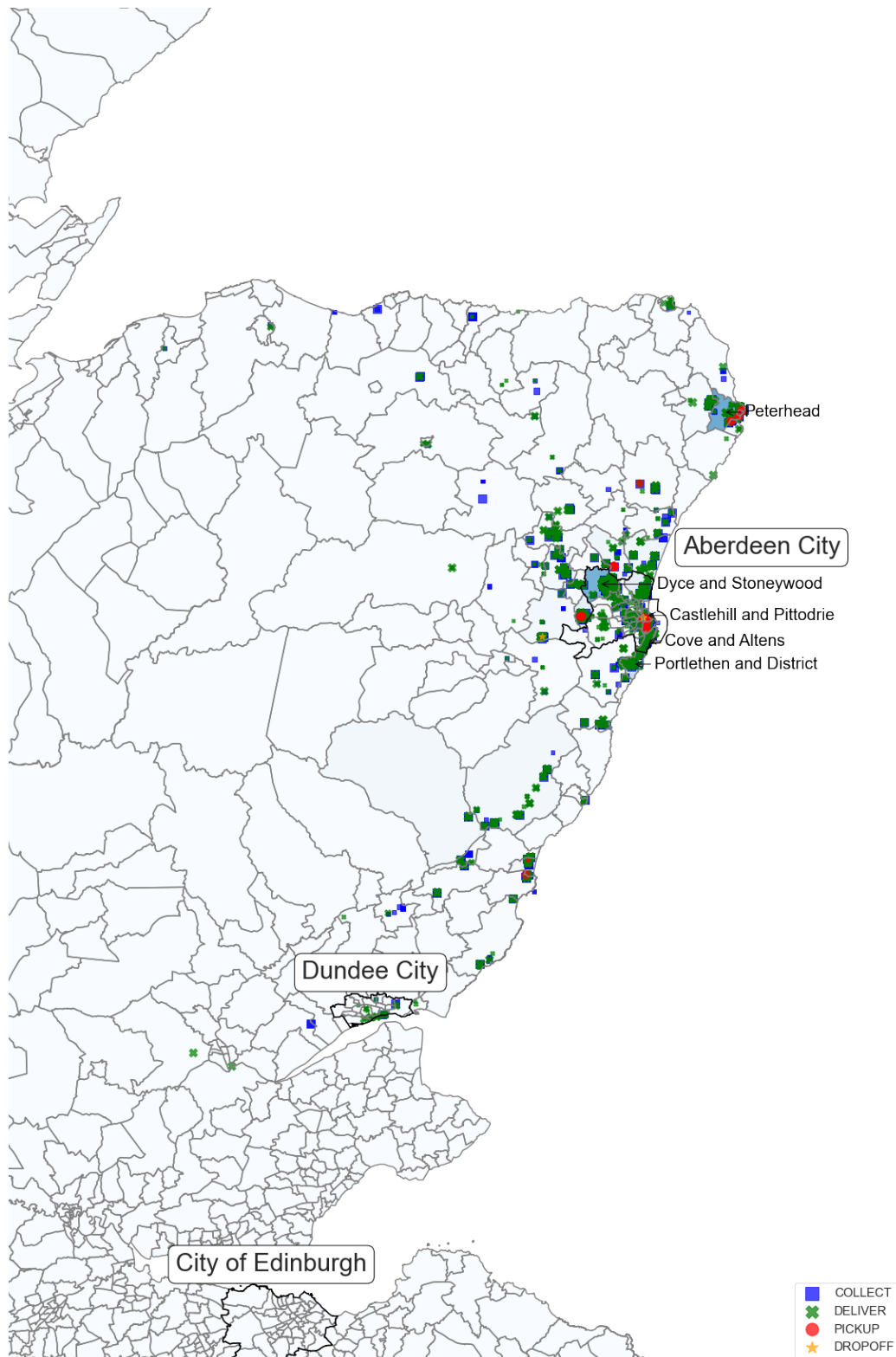


Figure 5.3: Geographical map showing the performed tasks per location.

Table 5.2: Frequency of each constraint in the preprocessed job dataset.

Number of Constraints	Frequency (%)
None	70,104 (80.561%)
One constraint	13,708 (15.753%)
Two constraints	2,622 (3.013%)
Three constraints	480 (0.552%)
Four constraints	95 (0.109%)
Five constraints	9 (0.010%)
Over five constraints	2 (0.002%)

contains of dataset containing 3,328 locations, from which their identification, name and geographical locations have been extracted. Nevertheless, the 2019 data includes jobs that are performed throughout 1,085 locations, meaning that only around 1/3 of the stored locations are actually used in 2019. Figure 5.3 shows a map organised by the boundaries of the community council² with geographical locations and the number of tasks completed in 2019. As depicted in the figure, most of the jobs are located near Aberdeen city, especially in Aberdeen harbour (Castlehill and Pittodrie), Dyce, Altens or Portlethen; the main economic and industrial areas in the region. However, it is worth highlighting the significant number of jobs performed in Peterhead, a town in the North East Scotland, which can be attributed to its fishing port (the largest in Europe) or the oil and gas industry. Indeed, Peterhead is the second most visited location, just after Aberdeen harbour.

Alternatively, it is worth remembering that the aim of this study is to simulate and optimise the dynamic decision-making process on a typical working day at the company. As a result, long-distance jobs have been filtered out. These jobs are typically assigned to a resource that will perform the job throughout the day, thus it is unlikely to reschedule the same resource for other jobs on the same day. Based on the distance and distribution of the locations, we have delimited the case study to jobs performed in North East Scotland; more specifically in Moray, Aberdeenshire, Aberdeen city, Angus, and Dundee city council areas. Some locations in the Scottish Highlands and Perth and Kinross councils have also been considered if they are within the defined geographical

²https://data.spatialhub.scot/dataset/community_council_boundaries-is

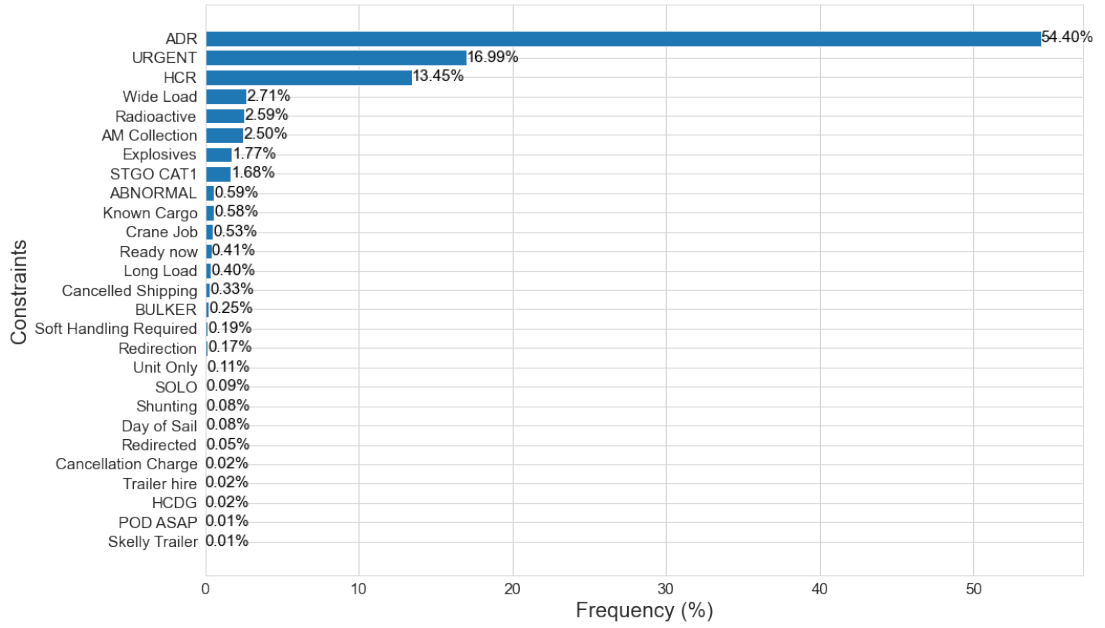


Figure 5.4: Constraint frequency of the job dataset, considering only jobs with constraints.

boundaries³. This spacial filtering reduced the total number of job-tasks and locations to 87,020 and 1,055, respectively.

Additionally, certain jobs require driver-skill constraints, which refer to the specialised restrictions and requirements that drivers must meet to perform the job. Note that driver-skill constraints are a special case of constraints (as presented in Section 5.1.1), and should not be confused with other constraints, such as temporal constraints that are inherent attributes of jobs. Therefore, without loss of generality, in the remainder of this chapter, we use “constraint” to refer to the driver-skill requirements that drivers must satisfy in order to perform the job. Table 5.2 summarises the presence of constraints in the preprocessed job dataset, and Figure 5.4 shows the frequency of constraints in constrained jobs as a bar plot.

The analyses performed have revealed that more than 80% of the jobs are constraint-free, meaning that they can be assigned to any available driver. The remaining 20% of the jobs are subject to constraints that restrict the assignment of qualified drivers to these jobs; approximately 15% of the jobs have a single constraint, while the remaining

³Latitude within 56.35 and 57.70 degrees, and longitude within -4.0 and 0.0 degrees.

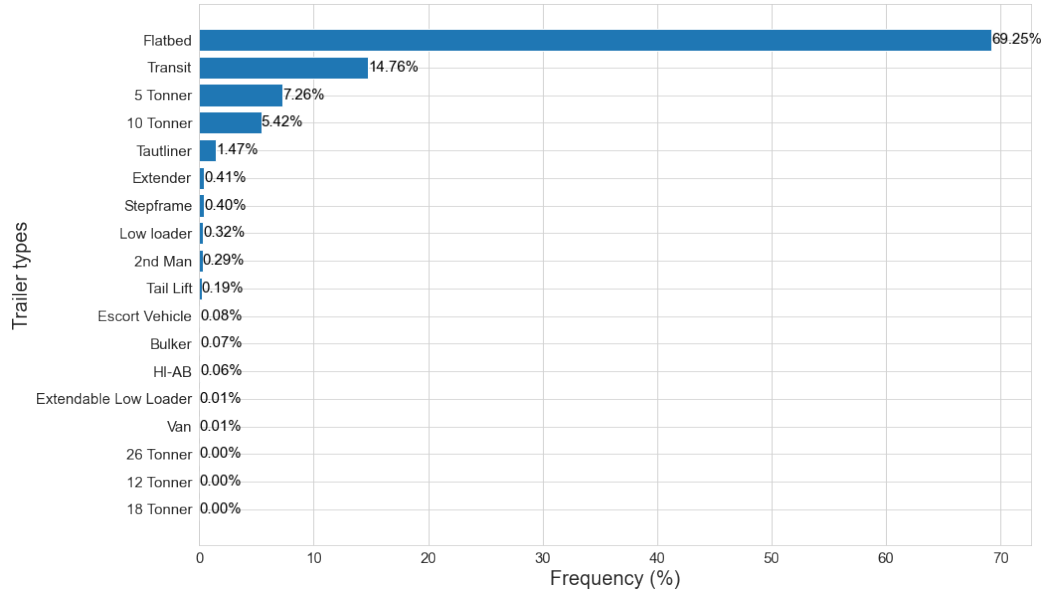


Figure 5.5: Frequency of requested trailer types in the job dataset.

5% have two or more constraints. As can be observed in Figure 5.4, the most common constraint is the need for an ADR licence. This special skill comes from the European Agreement concerning the International Carriage of Dangerous Goods by Road⁴, which regulates the transportation of hazardous goods, such as flammable liquids, gases or radioactive materials. This is present in more than 50% of the constrained jobs (around 10% of all jobs). Other constraints include urgency (around 3.4% of all jobs), high-cost rental (HCR) (around 2.7% of all jobs), etc.

The last aspect that has been analysed from the job dataset is the requested trailer types for each job. The trailer type determines the capacity and suitability of the trailer to transport different kinds of goods. For example, some trailers are better suited for handling heavy loads, oversized equipment or bulk cargo, while others are more suitable for carrying standard-sized pallets or containers. Therefore, it is crucial to match the type of trailer to the job requirements.

Figure 5.5 shows the frequency of each trailer type in the job dataset as a bar plot. The figure shows that flatbed trailers account for nearly 70% of all trailer types requested. These trailers are large, semi-articulated and widely used for transporting

⁴<https://www.gov.uk/guidance/driving-dangerous-goods-and-special-loads>

various types of goods, such as oversized equipment or bulk cargo. The second most requested trailer type is the transit trailer, which represents around 15% of the requests. Transit trailers are vans often used to transport small construction materials, agricultural products, or industrial goods. The third and fourth most requested trailer types are the 5- or 10-tonner trailers, respectively, which represent around 13% of customer requests. These small trailers can carry up to 5 or 10 tonnes of goods, such as parcels or containers. Other trailer types were also requested, such as curtain sided trailers, extender trailers, step frames and others.

In order to evaluate whether the company can handle the demands of its customers, the next section examines the resource capabilities that the company possesses.

5.2.2 Resources: Trucks, Trailers and Drivers

ARRC operates a large and diverse fleet of vehicles to meet the needs of their customers throughout Scotland. The fleet consists of 1,306 trailers and 541 trucks, classified into 21 and 26 different types, respectively. In addition, the company provided the records of 738 drivers, each of whom have various skills and qualifications to drive different types of trucks. In the following paragraphs, a complete description for each dataset and the processing steps are described.

From the trailer dataset, the identification, name, geographical location and types for each trailer have been extracted. Specifically, trailer types include their own names, pseudonyms, maximum weight and length specifications (if available) and compatibility list. The compatibility indicates the feasibility for replacing a particular trailer type for another to meet customer demands. Figure 5.6 shows a directed graph representing the compatibility between the different types of trailer. The colours in the figure are used for distinction purposes only, and the arrows represent the compatibility of one trailer to another.

The figure shows two main clusters, one on the left, representing the compatibility of light-weighted trailers (e.g. van, transit trailers, etc.), and another on the right that represents the heavy-weight trailer compatibility (e.g. flatbeds, stepframes, etc.). Note

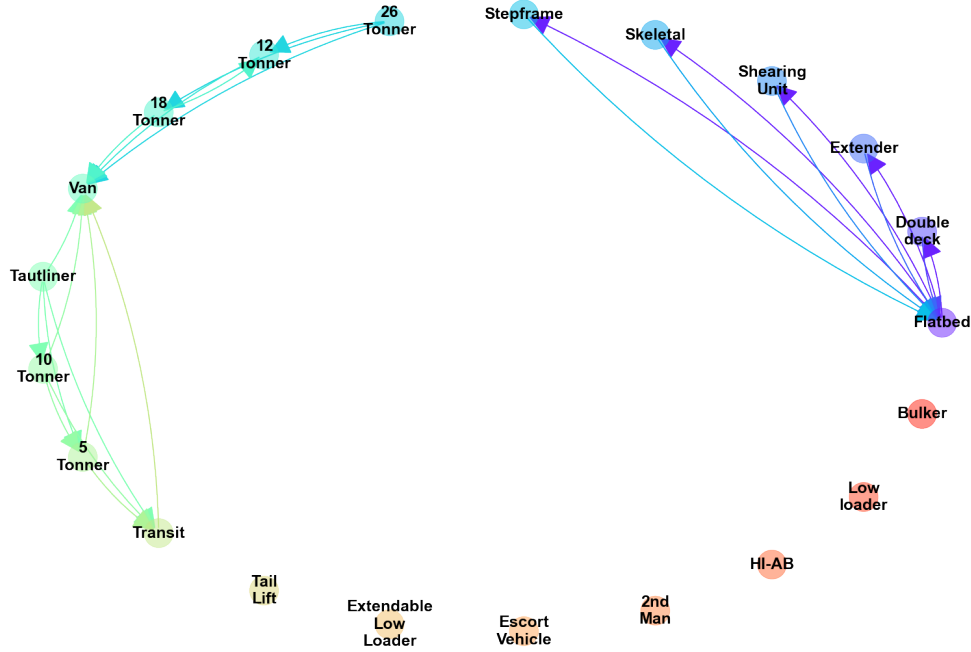


Figure 5.6: Relationship graph showing the compatibility for each trailer type.

that heavy-weight trailers are all compatible with each other, which is not the case with light-weight trailers. For example, it can be seen that flatbeds and stepframes complement each other, while tautliners can be used as an alternative to vans, but not inversely. Furthermore, the plot includes some trailer types that do not have compatible trailer types (e.g. bulker), which means that they cannot be replaced by any other trailer type.

The data provided contains a total of 1,306 trailers with various dimensions and specifications. Note that the performed data description states that there are up to 21 trailer types. However, as can be seen in Figure 5.7, the ARRC trailer fleet covers 9 of the 21 trailer types in 2019, where the largest trailer type in the fleet is the flatbed, followed by different variations of articulated trailers. Alternatively, note that the fleet of trailers is composed by heavy-weight vehicles that are compatible with light-weight vehicles (see Figure 5.5). Therefore, the company may decide to subcontract a small vehicle (van) rather than using a compatible but large trailer. In any case, operators typically prefer to avoid using large trailers for small cargo [14].

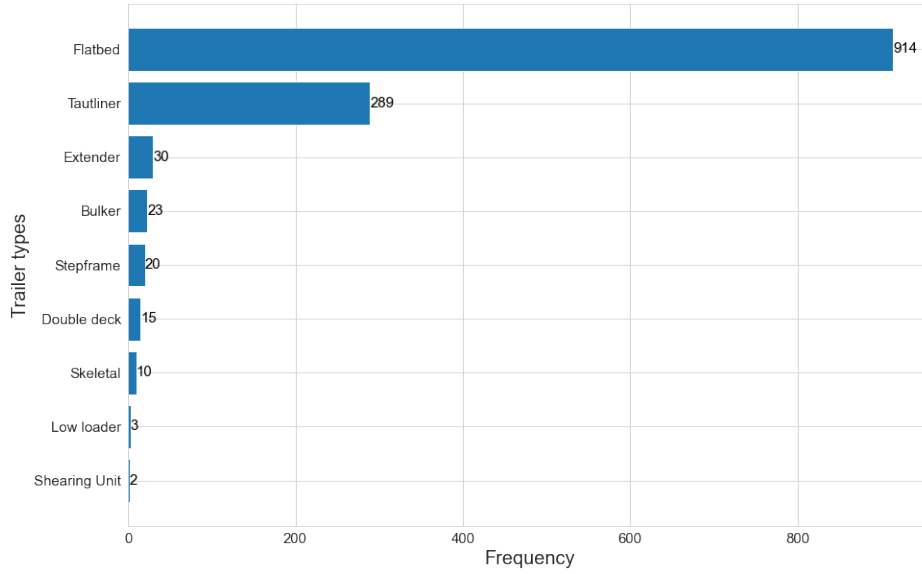


Figure 5.7: Number of trailers in the ARRC fleet in 2019, organised by trailer type.

ARRC has records of 541 trucks in 2019 in their database, from which the truck identification, name, geographical location, and the associated trailer (if the truck is rigid) and driver information have been considered. The data provided also distinguishes 26 truck types, which includes an identification, the name of the truck type, the maximum weight and length, and the *rigidity*. Rigidity determines whether the trailer is directly attached to the truck, i.e. rigid trucks cannot be separated from their trailers. Therefore, the rigidity variable, originally part of the truck type dataset, has been integrated as an additional variable within the truck dataset. Consequently, the truck type dataset has been discarded for the synthetic data generation process.

The company also provided a total of 738 drivers with different skills and qualifications, from which their identification, skills, and assigned truck have been pruned. It is worth noting that both trucks and drivers are interconnected by variables that link to each other. This link resulted in 296 truck-driver combinations that include the identification, name, location, and trailer attached to each truck, in addition to the assigned driver and its skills. For clarity purposes, these combinations will be treated as trucks hereafter.

5.3 Overview of the Data

This section summarises the essential properties of the data provided after the data cleaning, filtering and processing steps carried out in the previous sections. Note that this data is used as original (input) data for the generation of synthetic instances and their subsequent validation in the upcoming chapter. Table 5.3 summarises the number of entries for each dataset after the preprocessing step. This process has been done in two phases in the previous sections.

On the one hand, the job dataset has been analysed from three different perspectives. Firstly, approximately 80% of jobs have no constraints on driver skills, while the remaining 20% tend to have one or two constraints. Second, jobs are heterogeneously distributed around Aberdeen, with a notable exception in Peterhead, which has a large fishing and oil industry. Third, we have found that nearly 70% of total jobs require flatbed trailers, while 15% need transit trailers, and 12% require 5- or 10-ton trailers.

On the other hand, the resources of the company have been studied. The analysis carried out on the fleet of vehicles and drivers has revealed the diversity and heterogeneity of the fleet, as well as the compatibility between trailers or the skills of drivers to perform constrained jobs. We have found out that the company consists of 9 out of the 21 trailer types in the data, so they address the demand by optimising the compatibility between trailer types.

These findings provide valuable insights into the characteristics of the data provided. Figure 5.1 shows the attributes and relationship between the different datasets on an entity relationship diagram.

Table 5.3: Summary of the data after preprocessing.

Dataset	Number of entries	Number of columns
Jobs	87,020	9
Locations	1,055	3
Constraints	50	2
Trailers	1,306	4
Trailer types	22	5
Trucks	296	6

5.4 Temporal Analysis of the Data

Synthesising data with temporal attributes is challenging because it involves capturing the distributions and relationships among variables in addition to their temporal patterns. Therefore, it is essential to thoroughly analyse temporal variables to capture seasonality and noise in the data, ensuring that synthetic instances maintain the quality and consistency of real-world records.

In this section, in order to examine long-term variations and seasonal fluctuations of the data, we use time-series decomposition to capture and describe different patterns in the data, such as the trend, seasonality and residual components. As described in Section 5.2.1, the temporal variables are found in the job dataset, which includes the input time and the requested collection and delivery times of the jobs. The following sections provide a detailed temporal analysis of these variables to understand their underlying patterns.

5.4.1 Time-Series Decomposition Analysis

Time-series data often show a range of temporal patterns, so it is useful to divide it into different components, each of which represents a different type of pattern in the data. Time-series decomposition is a well-known approach that captures statistical properties and time dependencies by splitting data into trend, seasonality, and residual components [115, 116]. Therefore, the extraction process can be used to provide insights about individual components and their interactions, such as the long-term pattern of the data (trend), periodic fluctuations (seasonality) and random noise (residual). Indeed, these features are related to some features of DOPs, such as frequency, predictability, and cyclicity of changes.

The time-series decomposition can be formulated in the following way:

$$X_t = \hat{T}_t + \hat{S}_t + \hat{R}_t, \quad (5.12)$$

where X_t is the observed value at time t , \hat{T}_t is the estimated trend component, \hat{S}_t is the estimated seasonal component, and \hat{R}_t is the estimated residual component. The

addition operation has been used in this study because the seasonal variation in the data is relatively stable over time. However, for time-series data where seasonal variation increases or decreases over time by exponential or quadratic trend, a multiplicative decomposition must be used. An alternative is to transform the data until the variation is stable over time (e.g. log transformation), and employ an additive decomposition.

The literature presents a wide number of time-series decomposition methods, such as the regression or exponential smoothing [117, 118] or the Seasonal Trend decomposition using Loess (STL) [115]. Bandara *et al.* [116] presented an extended approach of the STL decomposition that includes multiple seasonal components, called Multiple Seasonal-Trend decomposition using Loess (MSTL).

MSTL considers a set of seasonal components that reflect different seasonality (e.g. daily, weekly, monthly, yearly, etc.) as $\hat{S}_t = (\hat{S}_t^1 + \hat{S}_t^2 \dots + \hat{S}_t^K)$, where K is the total number of seasonalities found in X_t . Thus, MSTL applies STL to each seasonal component to identify different seasonal variations in the time-series. The seasonal components must be sorted from the shortest to the longest period (e.g. hourly, daily, weekly, etc.) to prevent the erroneous seasonality decomposition, as otherwise the shorter seasonality would be considered as part of the longer seasonal component. The scheme of the MSTL method is given in Algorithm 5.1. We direct the interested reader to Appendix E for more details about STL and Loess regression (which are part of MSTL).

In short, the MSTL approach consists of three steps. First, each seasonal component $\hat{S}_t^k, k \in \{1, \dots, K\}$ is iteratively extracted using Loess, and the time-series data is updated by subtracting the seasonal components until a seasonally adjusted (non-seasonal) time-series is obtained. Thus, all K seasonal components in \hat{S}_t are independently extracted from the time-series data (lines 2–5). Then, STL is iteratively performed on each estimated seasonal component individually to refine the estimated seasonal components (lines 6–12). This process (called *robustness iterations* [115]) allows the model to tolerate larger errors, as the time-series X' provided to STL contains

Algorithm 5.1 MSTL: Multiple Seasonal-Trend decomposition using Loess

Input: Data X_t , periods of the seasonal components \mathbf{k} , Loess function \mathcal{L} , number of *robustness* iterations.

Output: Trend \hat{T}_t , Seasonality \hat{S}_t , Residuals \hat{R}_t .

```

1:  $X' \leftarrow X_t$ 
2: repeat
3:   Estimate  $\hat{S}_t^k$  from  $\mathcal{L}(X')$ .
4:    $X' \leftarrow X' - \hat{S}_t^k$ 
5: until  $k \in \mathbf{k}$ 
6: repeat
7:   repeat
8:      $X' \leftarrow X' + \hat{S}_t^k$ 
9:     Estimate  $\hat{S}_t^k$  from  $\mathcal{L}(X')$ .
10:     $X' \leftarrow X' - \hat{S}_t^k$ 
11:   until  $k \in \mathbf{k}$ 
12: until robustness iterations is met
13: Estimate  $\hat{T}_t$  from  $\mathcal{L}(X')$ .
14: Estimate  $\hat{R}_t$  from  $\mathcal{L}(X' - \hat{T}_t)$ .
```

only the seasonal component of interest \hat{S}_t^k , along with the trend and residual components. Finally, the trend and the residual components are extracted from the latest seasonally adjusted time-series (lines 13–14). Note that the original implementation of MSTL [116] presents a preprocessing step to impute missing data and transform the time-series into an additive decomposition if necessary. For simplicity, these steps have been omitted in the pseudocode.

The main parameter of MSTL is the period of each seasonal component in the time-series. Therefore, MSTL can take the following periods as input: 24 hours (day seasonality), $24 \times 7 = 168$ hours (weekly seasonality), $24 \times 30 = 720$ hours (monthly seasonality) and $24 \times 365 = 8,760$ hours (annual seasonality). In addition, other parameters related to STL and Loess can be adjusted, such as window size s and polynomial degree δ . In the subsequent analysis, the values used in [116] are replicated, that is, we set the window size $s \in \{11, 15\}$ (different for each respective seasonal component), the polynomial degree $\delta = 0$ (moving average) and the *robustness* iterations to 2.



Figure 5.8: MSTL decomposition of the collection time of jobs. Specifically, the trend, daily (seasonal_24) and weekly (seasonal_168) seasonalities, and residual components are individually displayed.

Figure 5.8 visualises the collection time⁵ of jobs in 2019 and time-series components after performing the MSTL decomposition, i.e. the trend, daily and weekly seasonalities and residual components. A slight fluctuation of the trend can be perceived from the figure, although there is no clear pattern that can discern a further seasonal component. That is, the trend reveals that months between May and September are generally busier than winter months (October to April), although no further pattern that repeats monthly, quarterly or half-yearly can be identified. The lower number of jobs in winter time may be due to the weather conditions in the Scottish North Sea area, where severe weather conditions are common, and may result in a decrease in job arrivals.

From the weekly seasonal component (Seasonal_168), we can see that the collection of jobs gradually increases as the week progresses. Moreover, consecutive weeks have similar patterns, although future weeks may have different seasonal patterns. Specifically, we can observe a higher weekly collection activity from April to July. The daily seasonality (Seasonal_24), however, is more difficult to evaluate because of its high frequency. To that end, the following section performs a more accurate analysis to examine the daily and weekly seasonality components in the preprocessed data.

5.4.2 Data Characteristics and Patterns

After identifying and describing different temporal components of jobs, this section presents and studies the raw temporal characteristics and patterns of the dataset. The total numbers of inputted, collected and delivered jobs per day in 2019 are shown in Figure 5.9. The plot confirms a slight increase in the number of jobs from May to September, as pointed out in the previous section, when the number of jobs typically exceeds 150 per day. As an exception, March presents a short busy period of 3 days (13–15 March 2019), when they handled approximately 200 jobs by the end of the working week. The figure also highlights another outlier on 24 June 2019, where the number of jobs inputted to the system increased to a total of 277. This is because a customer requested resources for multiple future jobs months in advance.

⁵The MSTL results for the input, collect, and delivery times of jobs are presented in Appendix D.

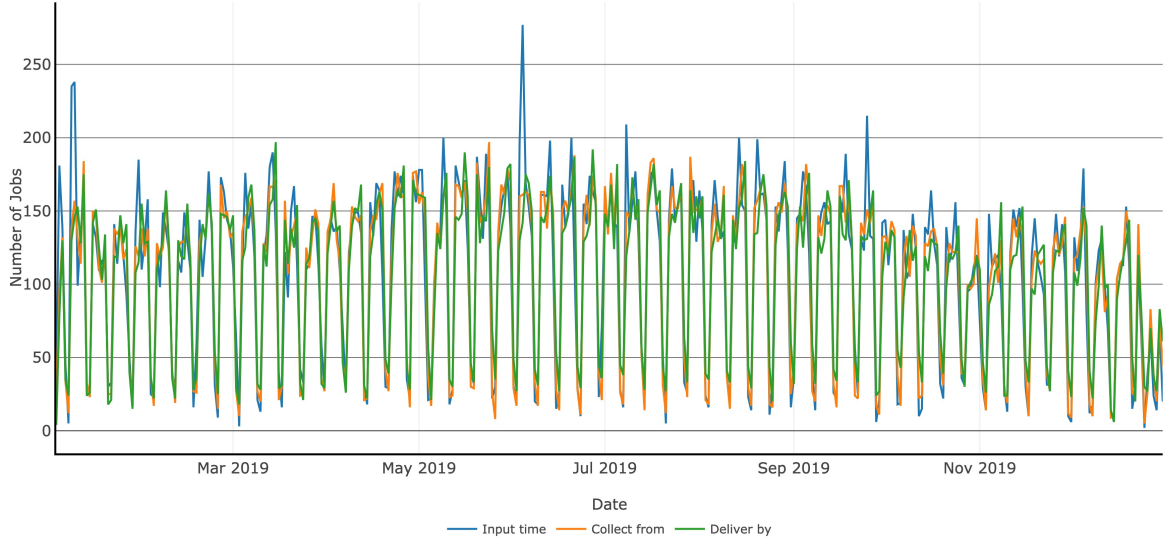


Figure 5.9: Annual demand of jobs for ARRC in 2019 in a daily basis.

A weekly seasonality over the year can also be perceived from Figure 5.9, where the input, collection, and delivery of jobs decrease on weekends. This weekly seasonality can be perceived in detail in Figure 5.10, where the collection time of jobs is aggregated by week days. The plot shows that the number of job collections are higher on Monday to Friday than on weekends. In addition, it can be seen that the number of collected jobs increases slightly over the week, as highlighted in the time-series decomposition analysis in the previous section. Both figures show certain anomalies, although these are related to the typical holiday period. As an example, the last week of the year contains two consequent drops, one of which is related to the day after Christmas (Thursday, 26 December 2019) and the other is due to the weekend (28–29 December 2019). Consequently, the reduced workload at the end of the year makes the beginning of the following year more demanding, in terms of the number of inputted jobs, compared to the usual weeks of winter months (October to April).

Figures 5.11 and 5.12 provide additional information on the daily and weekly seasonalities of the data, respectively, providing a temporal breakdown of the day. In Figure 5.11, it can be seen that the input, collection, and delivery times of jobs have

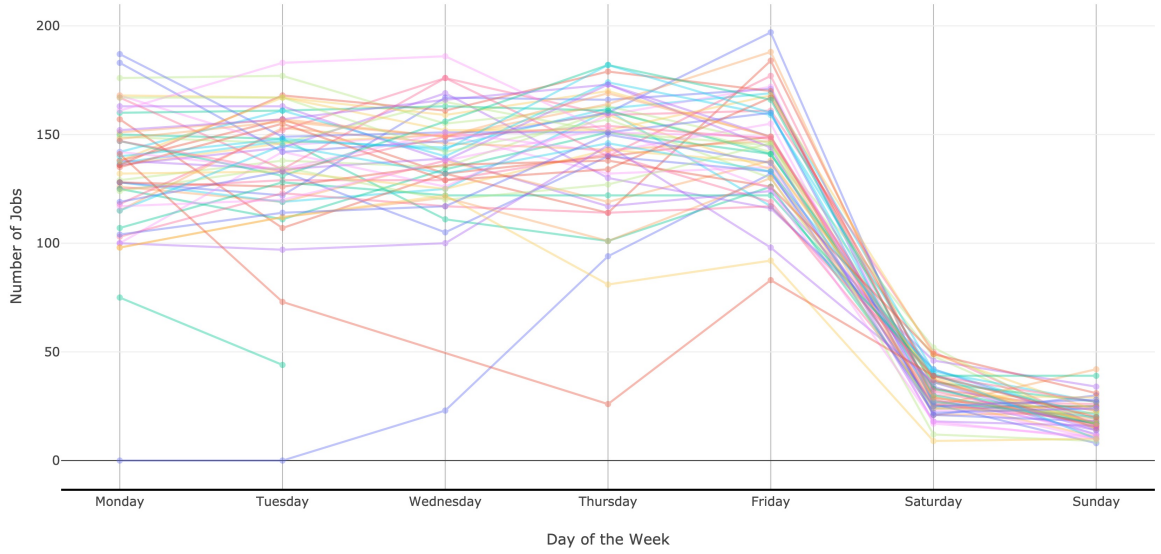


Figure 5.10: Number of collected jobs in a daily basis aggregated by week days in 2019. Each line corresponds to a certain week in 2019.

different distributions, although they show a cyclical pattern on a daily basis. Generally speaking, the input and delivery times of jobs follow a bimodal distribution, whereas the collection time follows a unimodal distribution. Further analysis of these distributions is presented in the subsequent paragraphs.

The seasonality analysis⁶ presented in Figure 5.12 provides strong evidence for the daily seasonality of the collection time for each month of the year, demonstrating a consistent pattern that persists throughout the months. Furthermore, the figure reveals that the daily seasonality is not related to the month of the year, since the mean daily seasonality slightly varies over the months.

It is worth remarking that the observations carried out in previous analyses have not considered the dynamism of jobs, where the distribution of static and dynamic jobs may be different. Note that we refer to a *static job* as a job that arrives days in advance and can be planned, while *dynamic jobs* are those that arrive during the working hours and must be performed before the end of the day (see Section 5.1). Moreover, certain dynamic jobs are flexible, which means that they may be performed on the day of their

⁶The full seasonality analysis for the input and delivery times of jobs are shown in Appendix D.

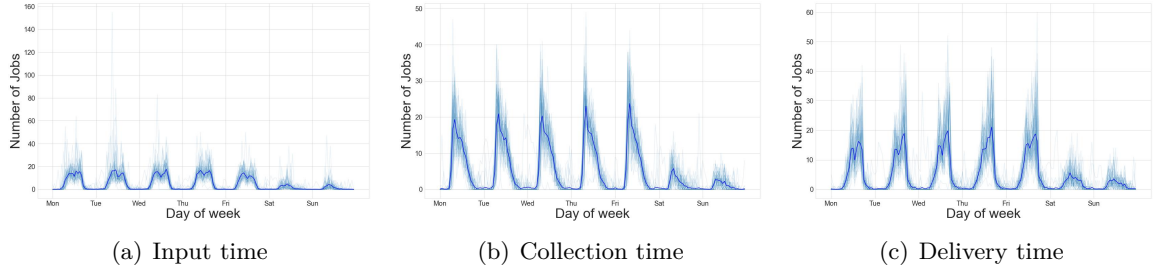


Figure 5.11: Weekly aggregated input, collection, and delivery times of jobs over the year. The light blue lines indicate the time for which jobs are requested to be collected for each day, and the dark blue line is the average collection time.

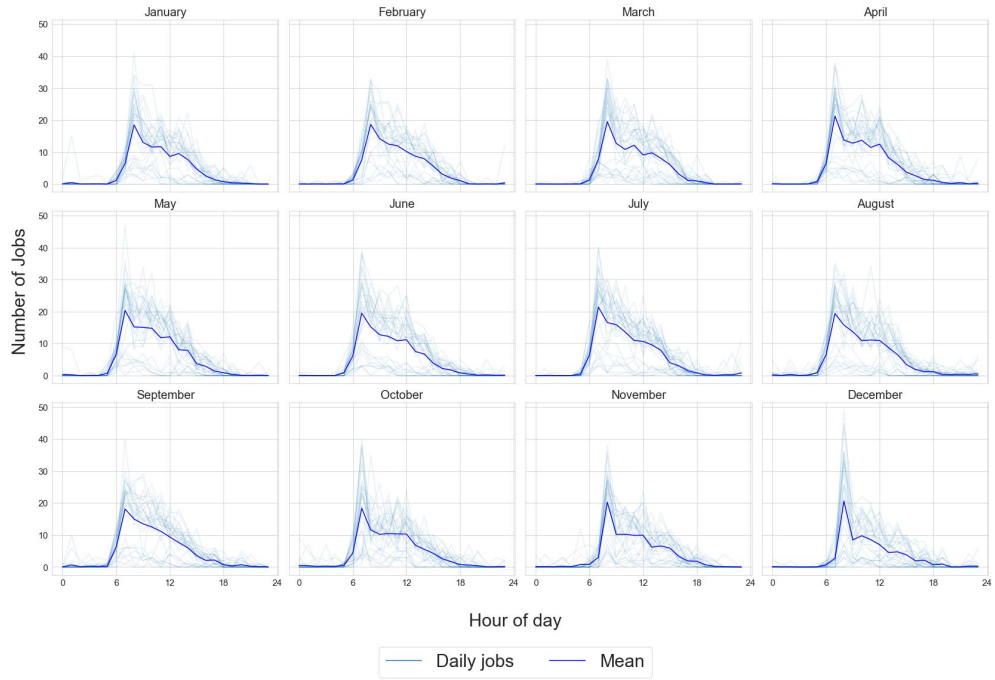


Figure 5.12: Daily seasonality of the collection time for each respective month of the year. Light blue lines indicate the time at which jobs are inputted or requested to be collected or delivered for each day, and the dark blue line is the average collection time.

arrival (and treat them as dynamic jobs) or may be postponed to become a static job on a subsequent day.

In order to show the possible bias of the dynamism on the temporal patterns of the data, Figure 5.13 shows the distributions of input, collection, and delivery times for static and dynamic jobs, separately, in the dataset on a daily basis. The plot reveals that there is a clear difference in the way these two types of jobs are distributed over time.

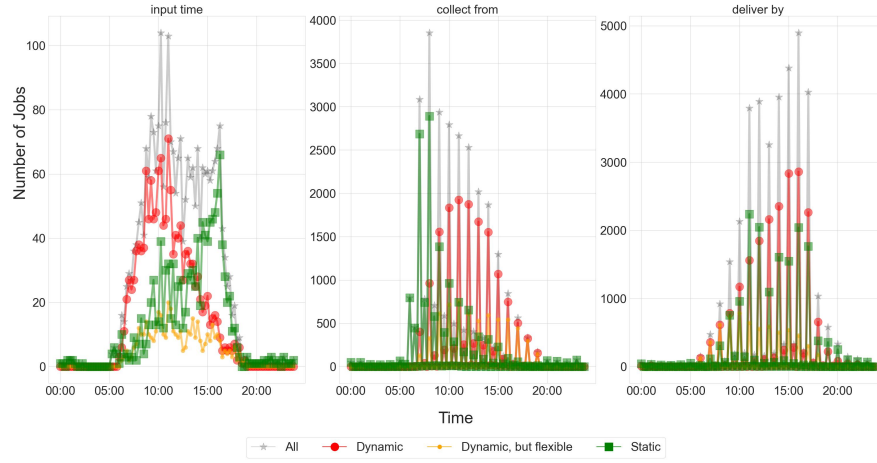


Figure 5.13: Daily seasonality of static and dynamic jobs. Specifically, half-hourly time-distributions of static (green) and dynamic jobs (red) over 2019 are visualised. The cumulative amount of jobs is represented in grey.

Dynamic jobs are generally inputted during the early hours of the day, with a peak around 8–11 a.m., and need to be posteriorly collected during midday and delivered by the afternoon. Static jobs however show a different pattern, where their input time distribution presents a linear growth from 7 a.m. to the late hours of the day (around 5 p.m.). This scenario indicates that static jobs are planned days in advance, and are usually scheduled to be collected at the beginning of a posterior working day (from 7–8 a.m.) and delivered in the late morning (between 11 a.m. and noon) or afternoon (from 2 p.m. to 5 p.m.).

Finally, an interesting observation is that the distributions for the requested collection and delivery times for both static and dynamic jobs resemble discrete random variable distributions. That is, these distributions do not follow a smooth and continuous pattern, but instead show discrete frequencies at certain hours of the day. This may be because customers round their requests to the nearest hour. For example, a customer generally specifies “collect from 10:00 a.m.” rather than “collect from 10:17 a.m.”, giving operators more time for planning their requests. This rounding procedure results in higher frequencies of job collection and delivery requests at these times, giving the distribution a discrete appearance.

5.5 Summary

Chapter 5 partially achieves the research objective **OB 4** and contributes to **OB 1** by analysing the dynamic aspects of a realistic truck and trailer scheduling problem. Specifically, it offers insights into how real-world constraints, dynamic features, and variable dependencies impact the synthetic data generation process.

This chapter has provided a comprehensive description and analysis of data obtained from a logistics company with the aim of understanding and reproducing the distributions, patterns, and dependencies between different variables in real-world data. The aim is to identify and understand the structure and dependencies of the data for the reproduction of synthetic, but still realistic, instances.

In order to contextualise and demonstrate the practical utility of the data, we have mathematically formalised the optimisation problem using a mixed-integer linear programming approach, where the main objective of allocating incoming jobs to a minimum number of resources (trucks and trailers) while dealing with a set of constraints.

Then, we have thoroughly described and analysed the different datasets and their respective variables. This process involved meticulous cleaning, filtering, and processing steps to ensure a reliable version of the data for further analysis. The conducted analysis has revealed that the recorded workload is predominantly Monday to Friday, with a notable peak period during the summer months, when the workload experiences a slight increase. In addition, we have identified several dependencies among time-related variables that highlight their essential role in replicating the income and dynamism of jobs.

Finally, note that the insights obtained from the analysis carried out will be used in the next chapter to identify and understand the expected results of the synthetic data generation process, ensuring that they conform to the observed patterns and dependencies.

Chapter 6

Developing a Benchmark Generator for a Real-World Dynamic Scheduling Problem

In the previous chapter, we have meticulously described the structure, attributes, and dependencies of the data provided by ARRC. However, despite the detailed analysis of the data in the previous sections, it is worth remarking that the company has not given us permission to share their historical records due to the privacy and sensitive exposure of the made decisions and the data itself. Instead, the company let us use their data to develop a benchmark generator that creates synthetic instances from their historical records.

This chapter presents a methodology for developing a benchmark generator for a realistic dynamic truck and trailer scheduling problem, using synthetic data generation based on historical records. However, this approach is not limited solely to this particular application. In fact, the main contribution of this chapter is to provide practical and generic guidelines for generating synthetic instances that capture the complexity and properties of real-world applications and approximate the distributions and patterns found in the original data, such as the temporal nature, chronology of actions, and seasonality.

The study proposes using Gaussian copulas as a benchmark generator to generate synthetic instances. Specifically, Gaussian copulas learn the marginal distributions of the original data and their correlations by means of copulas to sample new data. This approach has been adopted due to its simple configuration and functionality to ensure the generation of synthetic instances that are statistically robust and practically relevant [119, 120]. Nevertheless, although validation and testing are still being conducted before its release, the proposed methodology is general enough to generate synthetic data from any real-world application.

This chapter is structured in the following way. Section 6.1 presents the used synthetic data generator and evaluation metrics to understand their functionality and limitations. Section 6.2, the synthetic data generation approach and its parameters, are thoroughly empirically analysed. The goal is to thoroughly explore different strategies and approaches to improve the performance of the generative model to capture the distributions and patterns in the data. Finally, Section 6.3 concludes the chapter by giving a detailed summary of the findings, the methodology for the generation of new synthetic problem instances and future lines of research.

6.1 Synthetic Data Generation

Synthetic data generation refers to the use of real-world data (in text, images, or video formats) to generate similar (but not identical) data using algorithms or mathematical models [121]. That is, a model is first constructed by fitting the original data, and then, the model is used to generate new samples to approximate the probability distributions and patterns in the original data (i.e. marginal and joint probability distributions), although more complex patterns may require more sophisticated modelling techniques and a deeper understanding of the underlying data characteristics.

Synthetic data can be used to replace real-world data when it is limited or unavailable (data scarcity), lacks high-quality information (data quality), or requires privacy preservation (data privacy) [84, 86, 122].

The main advantage of synthetic data generation models is their ability to generate synthetic data by approximating to the distributions and patterns of the original data,

but without the confidentiality and privacy restrictions. This allows for a valuable and realistic analysis without confidentiality and privacy restrictions, facilitating the production and exchange of data.

Although synthetic data presents many advantages, it also has some limitations. First, synthetic data generation models may fall short in capturing the complexity and variability of real-world data, leading to erroneous conclusions. Nevertheless, the impact of similarity between original and synthetic data varies from problem and domain. For example, machine learning generally requires more data than medical research, which prioritises data quality to gain credibility in the medical community [86]. Second, models may not capture certain dependencies that require deep understanding of the data, such as rare events or logical rules. Hence, such patterns need to be identified and defined in advance. Third, since synthetic data generation models use real-world data as input, certain patterns or biases in the original data, or unreliable models, may lead to ethical and legal violations. Therefore, a thorough analysis of ethical and legal implications is essential to ensure compliance with the law, protect against possible vulnerability and address ethical concerns [123] (see Appendix B for a detailed description of ethical and legal requirements).

The literature presents different methods for generating synthetic data, ranging from standard statistical methods to advanced deep learning techniques [123]. Each method has its own advantages and disadvantages, and the outcome varies depending on the quality of the data or the domain. Probably, the most widely used methods for synthetic data generation in various domains, including computer vision and medical imaging, are advanced deep learning models, specifically Generative Adversarial Networks (GANs) [124] and, more recently, Diffusion Models [125].

This work aims to use synthetic data generation models to create benchmark instances of the problem described in Section 5.1. Specifically, we have considered the Gaussian Copula as the statistical model to generate synthetic problem instances, which has been typically used to analyse and model dependencies in financial time-series data [119, 120]. The following paragraphs provide further insights into this model.

6.1.1 Gaussian Copula

A *copula* is a probability distribution function that describes the non-linear dependence between variables. Specifically, a copula describes the correlation of the marginal probability distribution of continuous random variables into a multivariate distribution by using a correlation matrix. The foundation of copulas is based on Sklar's theorem [126], which states that any multivariate distribution can be written in terms of univariate marginal distributions and a copula. Formally, Sklar's theorem is stated as follows:

Definition 6.1 (Sklar's theorem). *Let $\mathbf{X} = (X_1, \dots, X_n)$ be a n -dimensional random vector, where each marginal $X_i, i \in \{1, \dots, n\}$ has its cumulative distribution function $F_{X_i}(x_i)$ and probability density function $f_{X_i}(x_i)$. Then, a multivariate cumulative distribution function H can be written as $H(x_1, \dots, x_n) = \mathcal{C}(F_{X_1}(x_1), \dots, F_{X_n}(x_n))$, and its probability density function as $h(x_1, \dots, x_n) = c(F_{X_1}(x_1), \dots, F_{X_n}(x_n)) \cdot \prod_{i=1}^n f_{X_i}(x_i)$, where $\mathcal{C} : [0, 1]^n \rightarrow [0, 1]$ is the copula function of n dimensions and c is the density function of the copula \mathcal{C} . Similarly, a copula \mathcal{C} can be defined as a cumulative distribution function of n dimensions with marginal distributions $i \in \{1, \dots, n\}, F_{X_i}(x_i)$, such that $\mathcal{C}(F_{X_1}(x_1), \dots, F_{X_n}(x_n)) = H(F_{X_1}^{-1}(F_{X_1}(x_1)), \dots, F_{X_n}^{-1}(F_{X_n}(x_n)))$, where $F_{X_i}^{-1}$ is the inverse function of F_{X_i} .*

Each random variable X_i is defined with a particular cumulative distribution F_{X_i} and density function f_{X_i} . Then, the copula \mathcal{C} defines the dependence structure between random variables, independent of their marginal distributions. This means that copulas are invariant to the transformations of the variables used, even if the variables belong to different probability distributions [127].

Gaussian Copulas, a special family of copulas, have been widely used in the literature to generate synthetic data due to their simplicity [119, 127, 128]. They convert different type of marginal distributions to standard normal before computing the correlation matrix. Formally, given n to be the dimension of the copula, a Gaussian Copula $\mathcal{C}^{\mathfrak{N}}$ can be formulated as:

$$\mathcal{C}^{\mathfrak{N}}(F_{X_1}(x_1), \dots, F_{X_n}(x_n)) = \Phi^{\mathfrak{N}}(\Phi_{X_1}^{-1}(F_{X_1}(x_1)), \dots, \Phi_{X_n}^{-1}(F_{X_n}(x_n))), \quad (6.1)$$

Algorithm 6.1 Gaussian Copula

-
- 1: **function** LEARNING(\mathbf{X})
 - 2: Transform the random vector \mathbf{X} into continuous random variables with non-missing values.
 - 3: Learn univariate marginal distributions, $(\Phi_{X_1}(x_1), \dots, \Phi_{X_n}(x_n))$.
 - 4: Compute correlation matrix \mathfrak{R} from $\text{Corr}[X_i, X_j], i \neq j$.
 - 5: **end function**

 - 6: **function** SAMPLING($(\Phi_{X_1}(x_1), \dots, \Phi_{X_n}(x_n)), \mathfrak{R}$)
 - 7: Generate a n -dimensional vector of correlated normal random variables \mathbf{X}' using Cholesky decomposition of $\mathfrak{R} = LL^T$, where L is the lower triangular matrix.
 - 8: Estimate the marginal distributions $\Phi_{X_i}(x_i), x_i \in \mathbf{X}'$, where the joint distribution is the Gaussian Copula $\mathcal{C}^{\mathfrak{R}}$.
 - 9: Apply the inverse of the copula function to each marginal $\Phi_{X_i} \in \mathbf{X}', \Phi_{X_i}^{-1}(\Phi_{X_i})$.
 - 10: Apply the inverse transformation to generate a random vector \mathbf{X}' back to the original space.
 - 11: **end function**
-

where $H = \Phi^{\mathfrak{R}}$ is the joint cumulative distribution function of a multivariate normal distribution with mean vector zero and correlation matrix $\mathfrak{R} \in [-1, 1]^{n \times n}$, and $F_{X_i}^{-1} = \Phi_{X_i}^{-1}(x_i)$ is the inverse Gaussian distribution of the random variable X_i . Let I be the identity matrix, then, the density of the multivariate Gaussian distribution $\Phi^{\mathfrak{R}}$ is represented as follows:

$$c^{\mathfrak{R}}(F_{X_1}(x_1), \dots, F_{X_n}(x_n)) = \frac{1}{\sqrt{|\mathfrak{R}|}} \exp \left[-\frac{1}{2} \left(\Phi_{X_1}^{-1}(x_1), \dots, \Phi_{X_n}^{-1}(x_n) \right)^T (\mathfrak{R}^{-1} - I) \left(\Phi_{X_1}^{-1}(x_1), \dots, \Phi_{X_n}^{-1}(x_n) \right) \right]. \quad (6.2)$$

Based on copula theory, the estimation of marginal distributions and the copula function can be used to generate new samples that are statistically similar to the original data. That is, since the multivariate distribution H is represented by the marginals $F_{X_1}(x_1), \dots, F_{X_n}(x_n)$ and a copula \mathcal{C} , a random vector $\mathbf{X}' = (X'_1, \dots, X'_n)$ can be generated from \mathcal{C} and the inverted function $F_{X'_1}^{-1}(F_{X'_1}(x'_1)), \dots, F_{X'_n}^{-1}(F_{X'_n}(x'_n))$ to obtain the desired random vector \mathbf{X}' . The learning and sampling processes of the Gaussian Copulas are described in detail in Algorithm 6.1. The interested reader in Sklar's theory and copulas is referred to [126, 127, 128, 129].

In this study, the *Synthetic Data Vault* (SDV) [119] Python package (version 1.8.0) has been used as the implementation of the `GaussianCopula`, which uses the `RDT: Reversible Data Transforms and Copulas` Python package to work with copulas. SDV is an open-source environment based on machine learning, developed by the Massachusetts Institute of Technology (MIT) and supported by DataCebo ©, designed to learn patterns of the original data and sample synthetic data. Additionally, it allows transforming variables to fit the model, using constraints to depict complex patterns in the data, and performing a conditional sampling to create synthetic data that matches certain characteristics while preserving the statistical patterns in the data.

Indeed, conditional sampling can be applied to generate specific instances of the optimisation problem described in Section 5.1. That is, the fitted annual model can be used to generate synthetic samples of specific days while maintaining the distributions and patterns of the data. Specifically, the Gaussian Copula performs conditional sampling by fixing the values of certain variables and sampling the others from the previously calculated conditional distribution. Formally, let $(X, Y) \sim C^{\mathfrak{R}}$ denote that random variables X and Y have a joint distribution with a Gaussian Copula $C^{\mathfrak{R}}$. Then, the conditional distribution of Y given $X = x$, denoted $C_X^{\mathfrak{R}}$, is given by:

$$c_X^{\mathfrak{R}}(Y) = P[Y \leq y | X = x] = \frac{\partial C^{\mathfrak{R}}(X, Y)}{\partial X}. \quad (6.3)$$

Note that previous notations have been reduced to two dimensions for the sake of simplicity, although it can be extended to higher dimensions [120].

6.1.2 Evaluation Strategies

Validating the quality of synthetic data is inherently complex due to the need to evaluate several critical aspects. Ideally, generated synthetic data should be sampled from the same underlying probability distributions as the original data, but these distributions are often unknown, making direct comparison impossible [130]. Consequently, researchers rely on various metrics to assess the statistical fidelity of synthetic data, ensuring it reflects the patterns and dependencies of the original data. This approach, however, complicates the selection of suitable evaluation metrics.

Additionally, the utility of synthetic data must be assessed, particularly its effectiveness in specific applications, such as training machine learning models. Despite privacy preservation being usually measured to prevent re-identification of individuals in the original dataset, this evaluation is skipped on further analysis since the original data does not include commercially sensitive information.

In this study, we employ a variety of evaluation strategies to validate the fidelity and utility of synthetic data, aiming to achieve an optimal balance among these competing requirements. Specifically, we use `SDMetrics` Python module [131], which is part of the `SDV` environment, to quantitatively measure the similarity and utility of synthetic data over the original data.

6.1.2.1 Statistical and Structural Similarity

In order to compare the similarity of the synthetic data with the original data, it is crucial to examine how effectively synthetic data approximates mathematical properties from the original data. That is, statistical and structural comparisons must be performed to verify the similarity between the original and synthetic data.

This process, also called data fidelity, can be categorised into two primary distinct analyses: a comparison of marginal distributions between the original and synthetic data columns, and a comparison of bivariate distributions (correlations) for all pairs of columns between the original and synthetic data. Further statistical comparisons may also be made, such as comparing the mean, median, and standard deviation.

6.1.2.1.1 Marginal Distribution Comparison

The marginal distribution comparison (\overline{MDC}) evaluates, for each variable (marginal), the statistical similarity between the original and synthetic data. In particular, a score is calculated for each data column, which represents the total difference in distribution between the original and synthetic marginals, and returns the average of the similarities of the marginal distribution as the overall score. The scores range from 0 to 1, where 1 indicates that all marginal distributions are likely the same, and 0 means that all distributions are as distinct as possible.

Depending on the type of each variable, different comparison tests are performed to quantify the difference between the probability distributions. Specifically, the Kolmogorov-Smirnov test is used for continuous variables (numerical and date), and the total variation distance is used for discrete variables (categorical and boolean). It is worth noting that, even though both measures are used to compare distributions, they focus on different aspects of the distributions.

Kolmogorov-Smirnov test. It determines if one or two samples come from the same non-parametric probability distribution [132]. It has been widely used to measure the absolute maximum distance between the same marginals of the original and synthetic data [119, 133]. Formally, let $F_{R_X}(x)$ and $F_{S_X}(x)$ be the cumulative probability functions for the continuous variable X in the original and synthetic data, R_X and S_X , respectively. Then, the Kolmogorov-Smirnov score for the column X is computed as:

$$KS(X) = \left(1 - \max_{x \in R_X \cup S_X} \{|F_{R_X}(x) - F_{S_X}(x)|\}\right), \quad (6.4)$$

where $x \in R_X \cup S_X$ represents all possible values in the combined range of samples R_X and S_X . $KS \in [0, 1]$ calculates the score based on the largest difference between $\hat{F}_{R_X}(x)$ and $\hat{F}_{S_X}(x)$. Note that the higher the score, the higher the quality of the generated data.

Total Variation Distance. It is a typical statistical distance metric that measures the difference between probability distributions [134]. In order to deal with discrete random variables, it computes the frequency distribution (probability mass function) for the same discrete column X in the original and synthetic data, R_X and S_X , to determine their distribution, $F_{R_X}(x)$ and $F_{S_X}(x)$, respectively. The total variation distance score for the column X is calculated as follows:

$$TV(X) = \left(1 - \frac{1}{2} \sum_{x \in R_X \cup S_X} |F_{R_X}(x) - F_{S_X}(x)|\right), \quad (6.5)$$

where the fraction $1/2$ ensures that the normalisation $TV \in [0, 1]$. Note that the score may be biased by a low number of samples, since the frequency of missing values in the synthetic data would be 0. The score $TV(X) = 1$ means that the same marginal X for both the original and synthetic data fit together.

6.1.2.1.2 Correlation Comparisons

The correlation comparison (\overline{CC}) evaluates the relationship between all pairs of variables as bivariate distributions, and measures to what extent correlations in the original dataset have been captured in the synthetic data. A score of 1 indicates that the correlations have been perfectly established.

Depending on the type of the variables, different metrics are used. That is, for two continuous variables, the correlation similarity is used, while for two discrete variables, the contingency table is used. It is worth noting that to compare the relationship between a continuous and a discrete variable, contingency tables are also employed by discretising the continuous columns into separate bins.

Correlation similarity. Given R_X and R_Y to be a pair of columns in the original data, and S_X and S_Y to be the same columns in the synthetic data, then, the bivariate correlation score for the columns X and Y can be calculated as:

$$\rho(X, Y) = \rho(Y, X) = \left(1 - \frac{1}{2} |\text{Corr}(R_X, R_Y) - \text{Corr}(S_X, S_Y)|\right), \quad (6.6)$$

where $\text{Corr}(R_X, R_Y), \text{Corr}(S_X, S_Y) \in [0, 1]$ represent the correlation function that returns the correlation coefficient between the given random variables. Both Pearson's and Spearman's rank correlation coefficients can be used as the correlation function Corr . By default, the Pearson's correlation is used as the correlation function. A score of $\rho(X, Y) = 1$ means a total pairwise match between correlation of the same columns in the original and synthetic data.

Contingency similarity. It uses normalised contingency tables to statistically compare the original and synthetic data. A contingency table shows the multivariate frequency distribution of all combinations between pairs of categorical variables. By normalising these tables, it is possible to calculate the joint probability between two variables. The absolute difference between two normalised contingency tables can be calculated using the total variation distance (see Equation 6.5).

Formally, let X and Y be two discrete variables representing two columns of the original and synthetic data, \mathbf{R} and \mathbf{S} , and let $C_{X,Y}^R$ and $C_{X,Y}^S$ denote their normalised

contingency tables, respectively. Then, the contingency similarity score for X and Y can be measured as:

$$\mathcal{S}^c(X, Y) = \left(1 - \frac{1}{2} \sum_{x \in X} \sum_{y \in Y} |C_{X,Y}^R(x, y) - C_{X,Y}^S(x, y)| \right), \quad (6.7)$$

where $C_{X,Y}^R(x, y), C_{X,Y}^S(x, y) \in [0, 1]$ represent the probability of $x \in X$ and $y \in Y$ in the original and synthetic data, respectively. Note that the higher the score, the more similar the two distributions are.

6.1.2.1.3 Evaluation of Temporal Dependencies

Relying exclusively on the comparison of marginal distributions and correlation structures may fall short for capturing the dependencies between variables, as well as the intrinsic complexities and hidden patterns of the problem addressed in this study. Specifically, as discussed in Section 5.4, it is important to analyse the seasonal components in the data, which includes examining the relationship among time-related variables, i.e. the input, collection, and delivery times of jobs.

It is worth noting that temporal variables are determinants of the dynamic nature and complexity of the problem being studied. Consequently, the replication of temporal patterns with a high degree of similarity is essential for the characterisation, reliability and practical utility of synthetic data in replicating real-world situations.

The similarity of the time-distribution variables has been quantitatively measured using the Kolmogorov-Smirnov test (see Equation 6.4), since time-related variables are considered continuous variables. Generally speaking, the overall time-distribution similarity score can be described as follows:

$$\bar{\mathcal{T}} = \frac{1}{|\mathbf{T}|} \sum_{T \in \mathbf{T}} KS(T), \quad (6.8)$$

where $T \in \mathbf{T}$ is a time-related variable (column) in the data. Similar to the Kolmogorov-Smirnov test, a higher score $\bar{\mathcal{T}} \in [0, 1]$ indicates greater similarity between the distributions of time-related variables. Note that $\bar{\mathcal{T}}$ could be applied to measure the time-distribution similarity of different seasonal components, i.e. $\bar{\mathcal{T}}_{annual}$ and $\bar{\mathcal{T}}_{weekly}$ represent the similarity score of the annual and weekly distributions, respectively.

For the daily distribution similarity, however, it may be more appropriate to also include dynamism in the formulation, as observed in Section 5.4.2. Note that we refer to dynamic jobs to those that arrive during the working hours of the company and must be completed before the end of the day. Therefore, the overall daily distribution similarity score can be extended as follows:

$$\overline{\mathcal{T}}_{daily} = \frac{1}{|\mathbf{T}|} \sum_{T \in \mathbf{T}} \frac{1}{2} (KS(T_{dyn}) + KS(T_{stat})), \quad (6.9)$$

where T_{dyn} and T_{stat} correspond to the values associated with dynamic and static jobs for the time-related variable T , respectively.

In addition, an intuitive and comprehensive understanding of time-distribution similarities can be obtained by visually examining and comparing the shape of the original and synthetic time-distributions. This visual interpretation may allow us to identify patterns that are hidden in the quantitative similarity scores.

6.2 Experimentation

This section provides a detailed empirical analysis of the parameter setting and performance of the Gaussian Copula for capturing the statistical and structural similarities of the original data and generating useful problem instances. To do this, we first process the original data and adjust the model parameters in Section 6.2.1. Then, Section 6.2.2 presents an exhaustive analysis of the behaviour of the model for different data inputs and parameters. Finally, Section 6.2.3 summarises the results obtained.

6.2.1 Parameter Settings

In this study, the Gaussian Copula has been considered as the synthetic data generation model to produce synthetic instances from the processed job dataset. The Gaussian Copula is primarily characterised by the distribution of the marginals and their correlation. Each marginal distribution can be independently adjusted to be a normal, beta, truncated normal, gamma, or uniform distribution. The Gaussian Copula implementation used considers all marginal distributions to be beta by default.

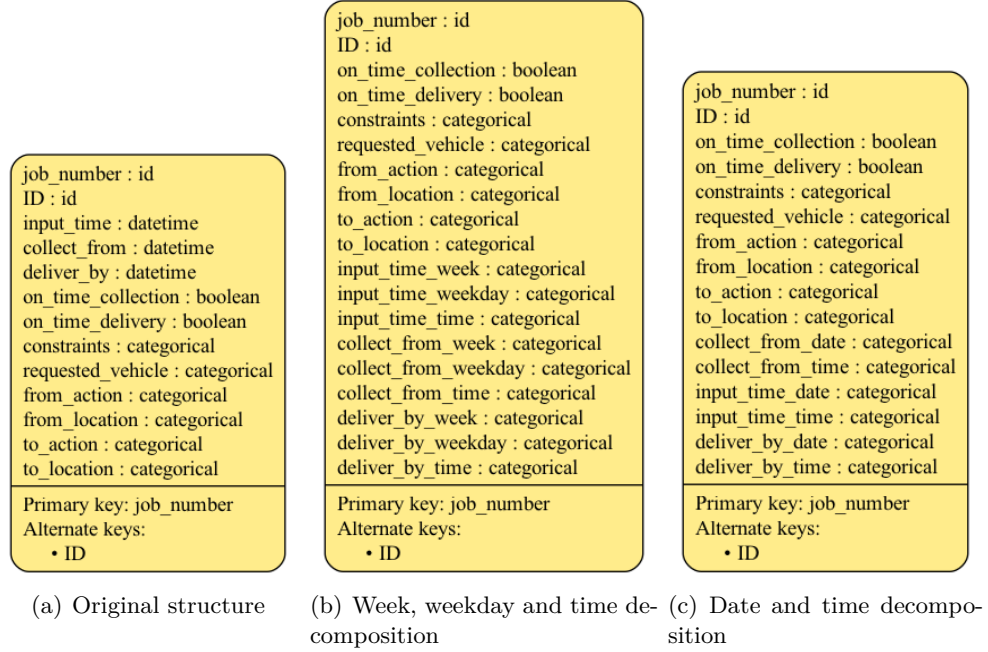


Figure 6.1: Data structure (metadata) of the original job dataset, and the data after *weekly* and *daily* decomposition steps.

In order to adjust the data to the considered synthetic data generation implementation, further data processing steps are performed to the job dataset. First, we get rid of columns that consist of lists as column entries, such as the driver-skill constraints column. Instead, a hash function is used to assign a unique identifier to each list of constraints. A total of 92 different driver-skill constraint lists have been identified.

After the preprocessing of the dataset, we can observe that the job dataset is composed of 87,020 rows representing the tasks performed in 2019 (see Section 5.2.1 for more details about the preprocessing of jobs and tasks). Nevertheless, the presence of repeated keys (job *ID* and *number*) for different rows (tasks) in the table presents a conflict with the implementation for the generation of synthetic data. Thereby, the first two tasks per job (which account for more than 90% of all jobs, as can be seen in Figure 5.2) have been represented as separate columns in the job dataset as *from_location* and *to_location*, and *from_action* and *to_action*, respectively. The structure of the processed job data as the original data and the type of each variable are shown as metadata in Figure 6.1(a).

Finally, different temporal approaches are defined to ensure a reliable generation of feasible samples. That is, as discussed in Section 6.1, the chronology of time-related variables of jobs is a particular pattern that models must consider to sample feasible data. Specifically, the input time of feasible samples must be no later than the requested collection time, and the collection time must be strictly before the delivery time.

In this study, the following temporal approaches have been defined and analysed separately to meet temporal requirements.

- The **validation-only** approach iteratively produces batches of samples until the specified number of feasible samples is reached (infeasible samples are ignored). The number of batch iterations and the size of batches are specified by the user.
- Two transformation approaches are defined to convert time-related columns into relative values to better deal with temporal dependencies, referred as **minutes transformation** approach and **time-difference transformation** approach. The minutes transformation strategy converts time-related columns into minutes since the beginning of the year. The time-difference transformation considers the difference between input and delivery times to the collection time of jobs, respectively. These methods, which are accompanied by the validation approach, are designed to capture the temporal dependencies between the highlighted variables [119].

Furthermore, in order to avoid the generation of undesirable tasks, such as a succession of pickup tasks only, the combination of the first and second tasks' action and location must be identical to the original data. That is, the collection (or pickup) of jobs must be strictly accompanied by a delivery (or drop-off) action at the locations identified in the original data.

Finally, note that the results and visualisations presented in this section are obtained considering the default configuration for Gaussian Copula with the same random seed (set to 1). Moreover, the number of samples to be generated is set to the total number of entries in the original data.

6.2.2 Results and Discussion

The experimentation and results of this study are divided into four parts. First, the defined temporal approaches are analysed to capture dependencies among time-related variables. Then, the influence of the decomposing time-related variables on the original data in different features is rigorously evaluated. Additionally, an extended analysis is carried out to improve the model to capture the hidden distribution of static and dynamic jobs. Finally, considering the findings of the previous analyses, a model is selected for conditional sampling to analyse the distributions and dependencies of the specific synthetic instances generated.

Appendix C presents details of the marginal distribution and correlation comparison for the entire experimentation.

6.2.2.1 Evaluation of Temporal Approaches to Feasible Sampling

Table 6.1 shows the overall marginal distribution (\overline{MDC}) and correlation (\overline{CC}) similarity scores, as well as the annual, weekly, and daily similarity score ($\overline{\mathcal{T}}_{annual}$, $\overline{\mathcal{T}}_{weekly}$, and $\overline{\mathcal{T}}_{daily}$, respectively), for each temporal approach presented in Section 6.2.1 when applied to the Gaussian Copula. For further information about the employed metrics, see Section 6.1.2.

The results obtained show that the validation-only approach obtains the highest correlation similarity score, the time-difference transformation method obtains the highest marginal distribution similarity and annual seasonality similarity, and the highest weekly and daily seasonality similarity scores are achieved by the minutes transformation strategy. In order to understand the temporal similarity scores, the temporal distributions for each approach are displayed in Figures 6.2– 6.4.

Table 6.1: Analysis of the Gaussian Copula with different temporal approaches.

SDV Constraint	\overline{MDC}	\overline{CC}	$\overline{\mathcal{T}}_{annual}$	$\overline{\mathcal{T}}_{weekly}$	$\overline{\mathcal{T}}_{daily}$
Validation-only	97.94%	92.00%	96.70%	79.55%	66.20%
Minutes transformation	96.81%	90.20%	91.46%	98.79%	83.62%
Time-difference transform.	98.96%	88.86%	98.85%	79.48%	67.04%

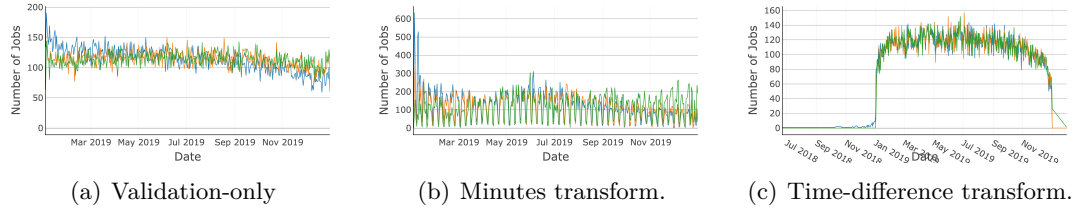


Figure 6.2: Daily demand for jobs produced by Gaussian Copula with different temporal approaches. The lines represent the input (blue), collection (orange), and delivery (green) times, respectively.

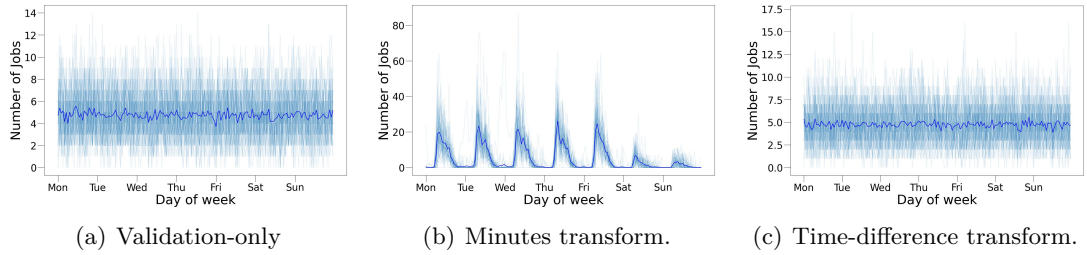


Figure 6.3: Weekly seasonality of the collection time of jobs over the year produced by Gaussian Copula with different temporal approaches. The light blue lines indicate the time for which jobs are requested to be collected for each day, and the dark blue line is the average collection time.

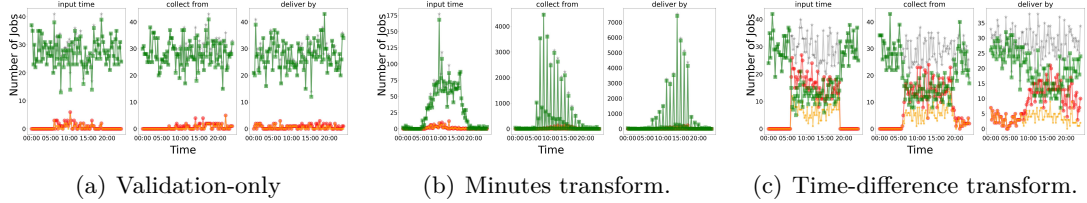


Figure 6.4: Time-distribution visualisation of Gaussian Copula with different temporal approaches. The colours represent the distribution of static (green), dynamic (red), and dynamic-flexible (orange-yellow) jobs, and the cumulative amount of jobs (black).

Figures 6.2 and 6.4 show the annual and daily distribution of input, collection, and delivery times, respectively. Figure 6.3 presents the weekly distribution of the collection time, for different constraints.

Visually, the shape of the distributions of the Gaussian Copula with the minutes transformation approach reveals a rough approximation of temporal dependencies to the original data. Figure 6.2(b) visualises the annual distribution for the minutes transformation strategy, which reflects that the model is able to capture the weekly seasonality of the original data (also seen in Figure 6.3(b)). Nevertheless, the annual similarity

score in Table 6.1 indicates the lowest similarity score for the minutes transformation approach, which may be caused by the large number of inputted jobs at the beginning of the year. Note that the scale of the vertical axis of the minutes transformation plot is larger than the plots for the validation-only and time-difference approaches.

Despite the minutes transformation-based model may be advantageous in capturing the annual and weekly patterns in the data, it fails to capture the time-distribution of the original static and dynamic jobs. That is, as can be seen in Figure 6.4(b), the daily similarity score ($\bar{\mathcal{T}}_{daily}$) of the Gaussian copula with minutes transformation may be biased by the distribution similarity of static jobs, since the model fails to capture the distributions for dynamic jobs.

Finally, Gaussian Copulas with the time-difference transformation and validation-only approaches are unable to capture any temporal patterns in the original data, and show an excessive variation on the annual, weekly, and daily distribution plots.

In summary, the Gaussian Copula with the minute transformation approach shows reasonable performance, although the results obtained still provide room for improvement in capturing the temporal patterns of the original dataset.

6.2.2.2 Temporal-Feature Decomposition of the Data

Previous experiments demonstrated that the minutes transformation temporal approach (which is based on calculating the minutes since the beginning of the year for each time-related variable) produced reasonably good performance. Motivated by the good performance of using a more specific granularity for temporal variables, this study proposes to decompose the time-related variables into date and time components to capture temporal dependencies in the data more effectively.

Specifically, two different methods are used to decompose time-related variables in the data. The first method decomposes dates into the week of the year, weekday, and time (minutes since midnight), thus, dividing each time-related variable into three columns, as illustrated in Figure 6.1(b). The second method represents dates as the day of the year and the time as the minute of the day (see Figure 6.1(c)).

Table 6.2: Performance of Gaussian Copula with different temporal approaches on the data with weekly and daily decomposition.

SDV Constraint		\overline{MDC}	\overline{CC}	$\overline{\mathcal{T}}_{annual}$	$\overline{\mathcal{T}}_{weekly}$	$\overline{\mathcal{T}}_{daily}$
Weekly	Validation-only	95.38%	81.47%	87.70%	96.25%	84.62%
	Minutes transformation	96.22%	83.02%	91.46%	98.79%	83.62%
	Time-difference transform.	94.00%	81.36%	99.32%	96.35%	82.19%
Daily	Validation-only	94.15%	74.56%	88.60%	94.34%	84.81%
	Minutes transformation	77.60%	51.66%	96.04%	79.35%	66.92%
	Time-difference transform.	92.31%	73.55%	99.47%	96.20%	81.98%

Note that both methods differ in the configuration of the date, i.e. the first method uses the week and weekday to represent dates, whereas the second uses the day of the year. The remaining experiments refer to these methods as the *weekly decomposition* and *daily decomposition*, respectively.

Table 6.2 shows the performance of different Gaussian Copula coupled with the temporal approaches defined in the previous section after performing the weekly and daily decomposition in the original data. Generally speaking, the table reveals slightly higher scores (except for the annual similarity score) for models under the weekly decomposition compared to the daily decomposition, especially for the Gaussian Copula with the minutes transformation approach. That is, the Gaussian Copula with minutes transformation approach considerably decreases its performance for daily decomposition, to the point of having difficulty in capturing both the statistical and temporal patterns in the data. Contrarily, Gaussian Copulas with the validation-only and time-difference transformation approaches show consistent and robust results for both weekly and daily decompositions. In particular, the Gaussian Copula with time-difference transformation approach shows promising yearly and weekly similarity scores for the daily decomposition, although shows a lower score for the daily seasonality component.

Furthermore, comparing the results in Tables 6.1 and 6.2, the temporal patterns of the models are captured more precisely using temporal-feature decomposition techniques. Specifically, weekly and daily seasonality similarity scores show promising results for models with decomposition methods.

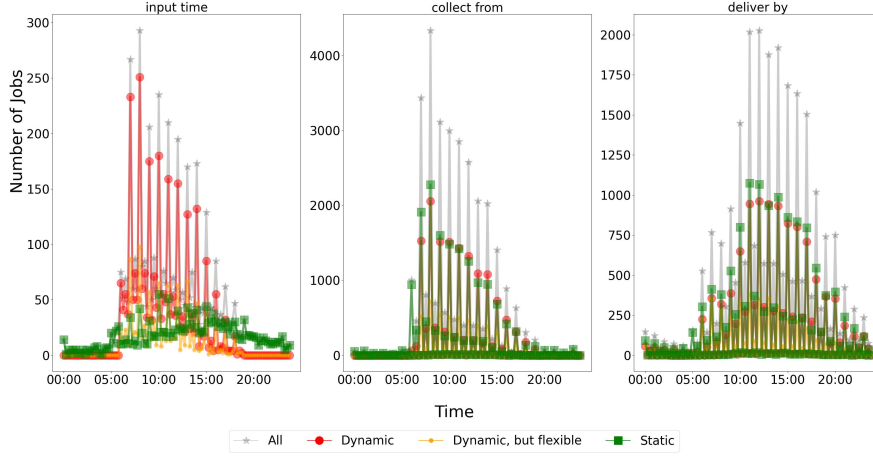


Figure 6.5: Time-distributions of static and dynamic jobs for the Gaussian Copula based with the time-difference approach on the data with daily decomposition.

As an illustrative example, the time-distributions of the Gaussian Copula with the time-difference transformation approach on data with the daily decomposition are shown in Figure 6.5. The figure demonstrates an accurate and robust representation of the collection time-distribution, resulting in a 99.50% similarity score if we apply the metric in Equation 6.8 (which eludes the dependencies between dynamic and static jobs). Nevertheless, the daily similarity score of the model (81.98%) reflects that the model fails to capture the distributions for dynamic and static jobs. Additionally, the input time- and delivery time-distributions show different shapes to the original data (see Figure 5.13). This may be caused by the relative values (i.e. time-difference to collection time of jobs) used for these variables, which perturb the mathematical properties of these variables.

In summary, obtained results demonstrates that the Gaussian Copula performs well when trained on data with time-related variables decomposed into different granularities, allowing the model to better capture temporal dependencies. In particular, the time-difference transformation approach results promising under any decomposition method, despite the distributions for the input and delivery times of jobs differ from the original distributions. Furthermore, Gaussian Copulas under the developed decomposition methods still fail to capture the dependencies for the dynamic and static job for the input, collection, and delivery times of jobs.

6.2.2.3 Dynamic and Static Model Ensembling

Motivated by the limitation of previously analyses to capture the dynamic patterns in the data, this study suggests (i) dividing the dataset into dynamic and static jobs, (ii) training separate sub-models, and (iii) combining the outputs of these sub-models to produce synthetic data. In other words, a dynamic model is trained from a dataset with only dynamic jobs, and a static model is trained from a separate dataset with only static jobs. Since there is no dependency between dynamic and static jobs, the samples produced by both models are then combined to create synthetic data.

This strategy (referred to as *ensemble modelling*) appears highly promising to capture the dynamic dependencies of the original data. Nevertheless, it is worth noting that models do not exclusively generate their respective job types, i.e. dynamic models may produce static jobs, and the other way around.

The results of ensemble Gaussian Copulas with different temporal approaches on the original and decomposed data are depicted in Table 6.3. Although similar conclusions can be drawn, there is a notable improvement compared to the outcomes from the previous sections, particularly when it comes to capturing temporal patterns. That is, ensemble modelling more accurately captures the dynamic properties in the original data, especially when considering the data under the weekly decomposition.

Table 6.3: Performance of ensemble Gaussian Copulas, using different temporal approaches, on the original and (weekly and daily) decomposed data.

SDV Constraint		\overline{MDC}	\overline{CC}	$\overline{\mathcal{T}}_{annual}$	$\overline{\mathcal{T}}_{weekly}$	$\overline{\mathcal{T}}_{daily}$
Original	Validation-only	97.88%	91.77%	96.24%	79.24%	67.47%
	Minutes transformation	97.76%	91.28%	94.90%	99.11%	85.47%
	Time-difference transform.	98.99%	89.39%	98.82%	79.60%	67.55%
Weekly	Validation-only	95.75%	82.67%	91.89%	95.99%	85.05%
	Minutes transformation	97.08%	84.96%	94.90%	99.11%	85.47%
	Time-difference transform.	94.89%	82.82%	99.31%	96.85%	92.69%
Daily	Validation-only	95.59%	76.49%	93.21%	96.85%	85.12%
	Minutes transformation	77.34%	52.03%	95.97%	79.38%	67.71%
	Time-difference transform.	93.60%	75.38%	99.44%	96.91%	92.63%

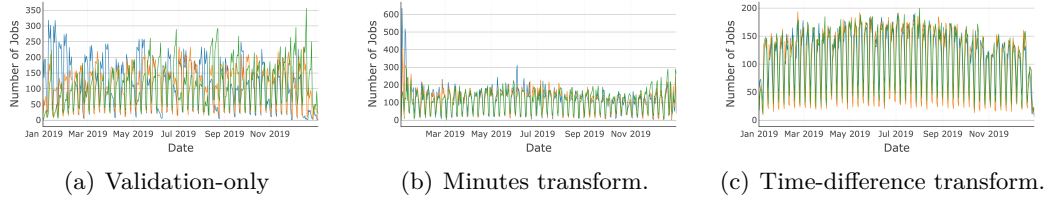


Figure 6.6: Daily demand for jobs produced by Gaussian Copula with different temporal approaches.

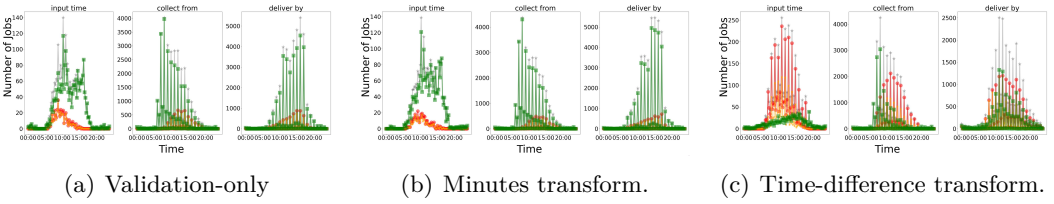


Figure 6.7: Time-distribution visualisation of ensemble Gaussian Copula with different temporal approaches.

Figures 6.6 and 6.7 display the annual and daily time-distributions for ensemble Gaussian Copulas with the different temporal approaches on the data with the weekly decomposition. The plots show that the ensemble Gaussian Copulas with validation-only and the minutes transformation approaches exhibit certain dissimilarities in the annual distributions, and hardly capture the dynamism of the original data. The ensemble Gaussian Copula based on the time-difference transformation shows consistent time-distributions, in particular for the collection time, where the distributions of static and dynamic jobs are closely matched with the original distributions. However, as highlighted in previous sections, the main drawback of the time-difference approach is that the distribution shape of the input and delivery times is different from the original data due to the relative values used to represent them.

In summary, obtained results demonstrate the feasibility and utility of the ensemble modelling for the Gaussian Copula to capture the statistical and structural similarities, in addition to the temporal patterns and the dynamic characteristics of the original data. In particular, the ensemble Gaussian Copulas with the time-difference transformation approach showed good results for both decomposition methods, although the distributions of the input and delivery times of jobs are not captured.

6.2.2.4 Conditional Sampling for the Generation of Problem Instances

Sections 6.2.2.1– 6.2.2.3 have focused on developing models that approximate the complexities, dependencies, and patterns in the original data in a yearly basis. Despite this, a critical part of this research work focuses on the generation of synthetic benchmark instances that may be used to replicate a typical working day for the company. Therefore, the process consists of defining *conditions* (fixed values of certain columns of the data) to which samples must comply.

Conceptually, the Gaussian Copula may be efficient considering that it allows using conditional sampling by calculating the conditional distributions. However, since the data is subject to temporal dependencies among variables, conditional sampling based on conditional distributions may produce infeasible samples (e.g. jobs where the collection time is later than the delivery time). To that end, a sampling procedure (referred to as *rejection sampling*) is carried out to iteratively discard infeasible samples, and rerun the model to obtain the same number of new ones.

In this study, we define a specific condition based on date of collection for jobs. That is, only those jobs that are scheduled to be collected on a given date are taken into account for conditional sampling. Specifically, four different collection dates have been randomly specified (by setting the random seed to 1 – 4) by providing a week number between 1 and 53, and a weekday number between 1 and 5. The specified date is converted into the date format used by the model (i.e. dates for original data, day of the year for the data with daily decomposition, and week and weekday for the data with weekly decomposition). Table 6.4 summarises the job information for each date.

Table 6.4: Data characteristics for the considered typical day instances.

	13 March	11 April	14 May	16 May
Weekday	Wednesday	Thursday	Tuesday	Thursday
Input jobs	180	149	170	171
Collect jobs	167	143	167	170
Deliver jobs	154	150	144	190
Dynamic jobs	112	80	95	104

Table 6.5: Performance of the conditional sampling of the ensemble Gaussian Copula with the time-difference transformation approach on the weekly decomposed data.

Date	\overline{MDC}	\overline{CC}	$\overline{\mathcal{T}}_{annual}$	$\overline{\mathcal{T}}_{weekly}$	$\overline{\mathcal{T}}_{daily}$
13 March	57.24%	29.85%	87.13%	91.39%	83.43%
11 April	57.21%	29.80%	81.48%	91.12%	81.17%
14 May	57.73%	30.29%	85.56%	91.77%	77.64%
16 May	57.96%	30.38%	85.44%	92.86%	83.61%

In order to illustrate the validity of models to perform the conditional sampling, a consistent procedure from previous analyses has been considered, i.e. the ensemble Gaussian Copula with the time-difference transformation approach has been used on the data with weekly decomposition. Table 6.5 presents the results of the considered model to perform the conditional sampling on the considered typical days.

Obtained results show a significant decline in the performance of conditional sampling for the ensemble Gaussian Copula using the time-difference approach on weekly decomposed data. That is, the model effectively captures annual and daily patterns in the annual data ($\overline{\mathcal{T}}_{annual} = 99.31\%$ and $\overline{\mathcal{T}}_{daily} = 92.69\%$, shown in the weekly decomposed row in Table 6.3), but performs poorly with conditional sampling, achieving around 80% similarity for both annual and daily temporal patterns. This is primarily caused by the rejection sampling, which needs several iterations to comply with the condition (i.e. specific collection time), and meet the temporal dependencies between variables.

It is worth noting that the model is fitted using annual data, and conditional sampling is employed to sample specific dates (and therefore, problem instances) from the model. However, the distributions for specific dates may differ from those observed annually, which can affect the similarity scores. For instance, the distribution of collection time in the annual jobs dataset tends to be concentrated at the beginning of the day (see Section 5.4.2), whereas the distribution of the collection time of jobs for a particular date might follow a binomial pattern. Nevertheless, despite this variability, the model fitted with annual data is sufficiently general to approximate annual distributions and produce realistic synthetic instances using conditional sampling.

In summary, conditional sampling shows promise for creating specific synthetic instances based on a model derived from annual data. However, it should be used cautiously as a benchmark generator, since the distributions of the generated instances may differ from those in the original data.

6.2.3 Further Analysis

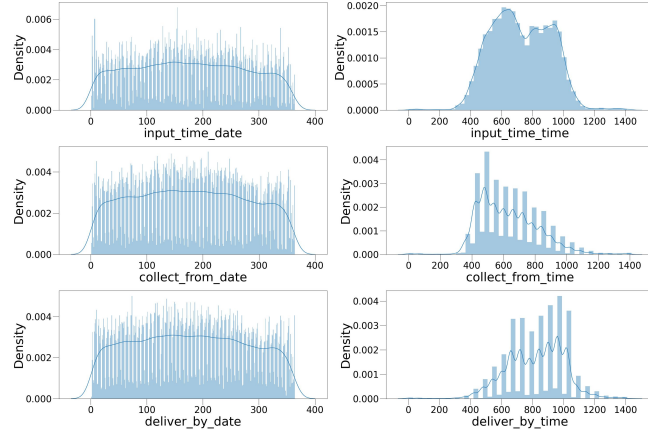
The experiments carried out in the previous section are limited by certain considerations, such as considering beta as the distribution of all marginals for the Gaussian Copula. This section extends the previous experiment to gain further insight into the obtained results. In particular, the following sections (i) illustrate the reason for considering times as categorical variables instead of numerical on the data with weekly and daily decompositions, and (ii) demonstrate the performance of Gaussian Copulas under different marginal distributions.

For interpretability purposes, non-ensemble Gaussian Copula models (presented in Section 6.2.2.3) have been considered in the following analyses.

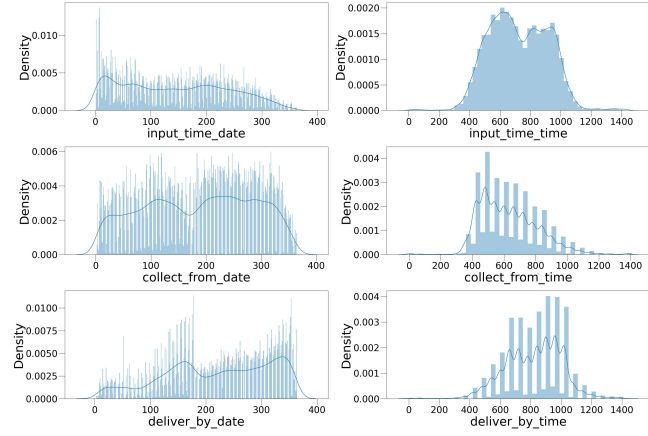
6.2.3.1 Categorical and Numerical Time Representation

As shown in Figures 6.1(b) and 6.1(c)), time-related features of the data under the weekly and daily decompositions are treated as categories rather than as numerical values. The categorisation of time-related variables into categorical (discrete) is based on the observation that the distributions for the collection and delivery times resemble a distribution of discrete random variables (see Section 5.4 for more information on the distributions in the original data).

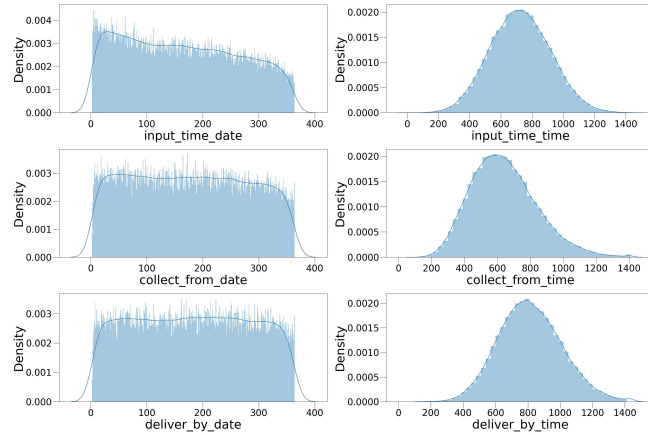
In order to construct empirical evidence, Table 6.6 contrasts the influence of the representation of time-related variables on the performance of the Gaussian Copula with the time-difference approach on the daily decomposition. In addition, the date and time densities of the input, collection, and delivery times distributions for the original data and the synthetic data (with both numerical and categorical time variables) are shown in Figure 6.8.



(a) Original data



(b) Synthetic data with categorical time-related variables



(c) Synthetic data with numerical time-related variables

Figure 6.8: Influence of numerical and categorical time-related variable representation in the data on generation of synthetic data. This includes a comparison of the densities of these variables in both the original data and the synthetic data produced by the Gaussian Copula using a validation-only approach with daily decomposition.

Table 6.6: Influence of numerical and categorical time-related variable representation in the data on generation of synthetic data. The performance of the Gaussian Copula with the validation-only approach on the data with daily decomposition (considering numerical and categorical representations) is shown.

Time formatting	\overline{MDC}	\overline{CC}	$\overline{\mathcal{T}}_{annual}$	$\overline{\mathcal{T}}_{weekly}$	$\overline{\mathcal{T}}_{daily}$
Categorical	92.31%	73.55%	99.47%	96.20%	81.98%
Numerical	94.29%	87.71%	97.74%	79.34%	76.37%

The results in Table 6.6 demonstrate that the good performance of the Gaussian Copula with the validation-only approach on daily decomposed data, using time-related variables as numerical (continuous), in terms of marginal distribution and correlation similarities. However, the table also shows that representing time-related variables as categorical provides a more precise representation of temporal patterns in the data. Figure 6.8(c) shows the densities of the date and time time-related variables in the original data, along with both numerical and categorical representations in the synthetic data. The time densities of the numerical time-related variables resemble a Gaussian distribution. In contrast, the time densities for the categorical variables closely approximate the shape of the original time-related distributions (see Figure 6.8(b)).

In summary, these findings indicate that the generation of synthetic samples (related to time) using categorical representation is generally more appropriate than numerical representation, which may introduce a large variance to the distributions.

6.2.3.2 Influence of Marginal Distributions

The influence of considering other marginal distributions in the Gaussian Copula is analysed. Specifically, this study examines the previously analysed beta distribution against the gamma, Gaussian, truncated Gaussian, and uniform distributions. Table 6.7 shows the performance of the Gaussian Copula with the time-difference transformation approach and different marginal distributions on the data with weekly decomposition.

The outcomes demonstrate an equivalent performance of the model for the beta, truncated Gaussian and uniform distributions, and a lower performance for the Gaussian and gamma distributions. The poor performance of the Gaussian and gamma

Table 6.7: Influence of the marginal distribution representation on the synthetic data generation process. The performance of Gaussian Copula with different marginal distributions using the time-difference transformation approach on the data with weekly decomposition is described.

Marginal Distribution	\overline{MDC}	\overline{CC}	$\overline{\mathcal{T}}_{annual}$	$\overline{\mathcal{T}}_{weekly}$	$\overline{\mathcal{T}}_{daily}$
Beta	94.00%	81.36%	99.32%	96.35%	82.19%
Gamma	86.16%	72.90%	92.28%	94.60%	79.71%
Gaussian	86.13%	72.89%	92.24%	94.66%	79.63%
Truncated Gaussian	93.65%	81.25%	99.91%	96.18%	81.91%
Uniform	94.01%	81.36%	99.32%	96.35%	82.22%

Table 6.8: Marginal distribution similarity score for each marginal for the Gaussian Copulas with time-difference approach on the data with weekly decomposition.

Marginal Distrib.	Job Constraints	Input Time	Collect Time	Deliver Time	Req. Vehicle	From Action	From Location	To Action	To Location	\overline{MDC}
Beta	99.51%	Extended in Table 6.9			99.74%	99.76%	97.07%	99.82%	96.51%	94.00%
Gamma	91.23%				90.11%	92.42%	88.00%	96.66%	87.18%	86.16%
Gaussian	91.24%				90.12%	92.44%	87.51%	96.66%	87.11%	86.13%
Truncated Gaussian	99.32%				99.02%	99.40%	96.78%	99.81%	96.66%	93.65%
Uniform	99.56%				99.70%	99.84%	97.02%	99.93%	96.54%	94.01%

Table 6.9: Marginal distribution similarity score for each time-related marginal for the Gaussian Copulas with time-difference approach on the data with weekly decomposition.

Marginal Distribution	Input Time			Collection Time			Delivery Time		
	Week	Weekday	Time	Week	Weekday	Time	Week	Weekday	Time
Beta	97.66%	92.60%	72.28%	98.48%	99.68%	97.70%	98.28%	95.63%	53.49%
Gamma	83.84%	91.44%	67.56%	82.44%	88.19%	84.64%	82.91%	89.08%	53.26%
Gaussian	83.77%	91.61%	67.47%	82.44%	88.22%	84.66%	82.91%	89.14%	53.20%
Truncated Gaussian	97.06%	92.91%	71.95%	97.68%	98.35%	96.93%	97.74%	95.02%	54.04%
Uniform	97.56%	92.55%	72.26%	98.42%	99.82%	97.68%	98.24%	95.73%	53.47%

distributions may be derived from their unbounded nature or their inability to approximate the original distribution of the marginals.

One interesting point in this experimentation is to identify the best marginal distribution for each variable. Tables 6.8 and 6.9 represent the marginal distribution similarity score for each marginal for the Gaussian Copulas with the time-difference approach on the data with weekly decomposition. The obtained results show that, generally, the beta distribution provides a good approximation of the marginals, followed by the uniform and truncated Gaussian distribution.

Finally, it is worth mentioning that, despite these findings are derived specifically from the Gaussian Copula with the time-difference transformation approach on the data with weekly decomposition, similar insights can be drawn from other configurations (i.e. temporal approaches and decomposition methods).

6.3 Summary

Chapter 6 partially achieves the research objective **OB 4** by presenting a methodology for constructing dynamic benchmark instances using synthetic data generation. However, the applicability of elusivity analysis (the research objective **OB 3**) to the developed realistic framework remains a work in progress, requiring further investigation to complete the objective.

The field of dynamic optimisation has identified the simulation of real-world scenarios as a challenging concept for existing benchmark generators. This chapter has presented a thorough study of the synthetic data generation process to capture the structure, dependencies, and statistical patterns of the provided real-world historical data. In order to generate synthetic data, a simple statistical model has been used in this study: the Gaussian Copula. Moreover, due to the difficulty of obtaining accurate distributions of original data, robust metrics have been considered and defined to validate and quantitatively assess the fidelity of the synthetic and original data, ensuring it reliably approximates the characteristics and patterns of the original dataset.

Conducted experiments have demonstrated the challenges encountered when incorporating temporal patterns into the synthetic data generation process. Furthermore, obtained results have shown that models generally sacrifice, to a certain extent, the marginal distribution and correlation similarities to meet temporal dependencies. These observations highlight the complexity of balancing trade-offs between capturing hidden patterns in the original data and the accuracy of the model to capture marginals and their correlations.

In addition to the empirical analysis of the synthetic data generation process, this work contributes to the field of dynamic optimisation by suggesting a practical and adaptable framework to model the complexities and variations in the real-world data.

Moreover, the synthetic data generation model can be used to produce realistic benchmark instances for a real-world dynamic scheduling problem. This framework will be available to the research community to promote its wider usage and development in the field. In summary, this illustrative framework allows bridging the gap between academic research and practical applications.

Finally, from the insights gained from the analyses presented in the previous sections, we list a set of systematic steps for generating synthetic data from real-world problems:

1. **Data collection, processing, and analysis.** First, data must be collected from the real world. Next, it needs to be processed to deal with sensitive information, process missing data or clean up unwanted values. Finally, the data must be thoroughly analysed to understand its structure, distribution and inherent patterns. This step is crucial for understanding the results of the subsequent synthetic data modelling step.
2. **Model selection and training.** Choosing an appropriate synthetic data generation model, such as statistical or deep learning models, is essential to capture the underlying distributions and dependencies in the processed data. The selected model will be then trained using the original data to generate synthetic data that approximates the original dataset.
3. **Evaluation metric definition and validation.** Establish metrics to assess the quality of the synthetic data by quantifying the distributional and correlational similarities with the original data, in addition to measuring the patterns that the model must capture. This step could also include utility and privacy analysis, especially in applications where the data is commercially sensitive.
4. **Iterative Refinement and Deployment.** Based on the evaluation results of the previous step, it may be worth refining the model and evaluating the outputs (steps 2 and 3, respectively) to improve the quality and utility of the synthetic data.

The reader can find more information on the ethical and legal considerations regarding data acquisition, risk exposure, and bias identification of this study in Appendix B.

Part III

Conclusions

Chapter 7

Conclusions and Future Work

This thesis has expanded in some existing research gaps in the field of dynamic optimisation. Specifically, this thesis has provided a comprehensive analysis of dynamic optimisation problems and benchmark generators, promoting further advancements in problem formulation, algorithmic performance evaluation, and benchmarking across different combinatorial problems.

This chapter expands on the contributions of this thesis in regard to the described research question (described in Section 1.1), discusses the limitations of the study carried out, and suggests directions for further research.

7.1 Summary of Research Questions and Major Contributions

This section provides an overview of the research questions and the main contributions of this thesis (presented in Chapter 1). Moreover, note that a brief summary of each contribution can be found at the end of each chapter in Part II, i.e. Chapters 3–6.

RQ 1. *Can we extend existing definitions for DOPs to quantitatively include the performance of online algorithms?*

A thorough and systematic review of the literature on definitions, dynamic features, and methods in dynamic optimisation has been performed to understand and identify significant research gaps in dynamic optimisation. Specifically, the conducted literature review shows the absence of a consistent definition that includes the adaptive challenge of DOPs to distinguish them from unrelated static optimisation problems that change without similarity.

RQ 2. *What essential features should benchmark generators include to construct realistic DOP instances?*

A systematic analysis of real-world data from the dynamic operations of a haulage company has been performed to gain insights into dynamic optimisation features, distributions, and patterns. Based on these insights, a preliminary benchmark generator has been proposed to simulate the operational workflow and complexity of a real-world dynamic truck and trailer scheduling application. This work aims to promote the utility and validity of synthetic data generation models as benchmark generators, supporting rigorous benchmarking in a dynamic, constrained, and heterogeneous optimisation context.

RQ 3. *To what extent can we quantify the adaptive advantage of online algorithms for solving a DOP compared to randomly restarting the algorithm after a problem change?*

A novel concept, called *elusivity*, has been introduced to quantitatively measure the adaptive challenge of DOPs to online algorithms. That is, the adaptive advantage of online algorithms has been compared to restarts across various DOPs. A comprehensive case study has empirically evaluated the extent to which elusivity provides additional insights in experimental research. Specifically, by replicating existing experimental frameworks, our study has quantified the adaptive advantage of algorithms across different DOPs and performance metrics.

RQ 4. *How can we apply the gained insights to develop advanced approaches to improve the performance of standard algorithms?*

Extend the theoretical study of the fitness landscape rotation benchmark generator to the permutation space to mathematically prove the preservation of structure and neighbourhood relations, in addition to the repercussion of *rotations* on the fitness landscape. This study emphasises the careful application of this method for evaluating and comparing online algorithms, as even minor rotations can significantly affect the reconfiguration of the fitness landscape. Moreover, based on the insights gained, two advanced perturbation strategies for local search algorithms utilising fitness landscape rotation have been developed.

7.2 Future Work

The research carried out provides a solid foundation for a better understanding of the dynamic features of DOPs, but there are still several research streams for further investigation. In the following, potential directions for future work are briefly summarised.

Further Examination of Fitness Landscape Analysis for Permutation Problems

In Chapter 3, the theoretical analysis of the fitness landscape rotation has demonstrated the preserving nature of the method in regard to the structure of the fitness landscape and the relationships between solutions. However, certain aspects of fitness landscapes related to the rotation operation remain unexplored, including the number and size of attraction basins and the distribution and centrality of local optima within these basins [19, 21, 23]. Examining these properties, in addition to problem-specific features, such as symmetries, may lead to different discoveries of the fitness landscape rotation.

Another promising research direction would be to analyse alternative fitness landscape characterisations, such as the first-improvement hill-climbing as the neighbourhood function. The study could also evaluate the impact of the rotation under various distance metrics, such as Kendall's- τ or Ulam distances for permutation problems.

Benchmark Generators and the Elusivity Concept

This thesis has implemented several benchmark generators to incorporate different levels of dynamism and complexity into the experimental frameworks. Despite elusivity being introduced as a preliminary method to evaluate the adaptive advantage of online algorithms to their restarting version, further research could explore additional properties of benchmark generators and expand on the concept of elusivity, such as considering the time-linkage property and uncertain environments.

Within the properties of DOPs, the most promising research direction would probably be to further investigate the detectability of problem changes. Conducted experiments have assumed that changes in problems can be easily detected by the alteration in the objective value of the best solution. However, in many real-world situations, this may not be feasible, negatively impacting the applicability of restarting algorithms and the adaptive advantage of online algorithms. This presents an opportunity to develop self-adaptive mechanisms that can balance adaptation and restart based on performance and the long-term robustness of solutions. To that end, predictive models (e.g. surrogate models) could be incorporated to predict future conditions and improve algorithm selection.

Another extension involves considering the elusivity to quantitatively measure the advantage of adaptive mechanisms over standard algorithms that do not respond in DOPs. That is, building on the idea of using elusivity to guide the adaptive advantage of algorithms, the elusivity can also be used to balance adaptation and continuous search in online algorithms.

Furthermore, the integration of multi-objective optimisation in dynamic contexts is a prominent research area, which requires balancing competing objectives while adapting to problem changes. Thus, we find it interesting to systematically analyse the elusivity of these problems for specific algorithms and performance metrics, particularly where objectives conflict among them.

Extend the Real-World Benchmark Generator

The benchmark generator described in Chapters 5 and 6 offers a preliminary method for creating synthetic benchmark instances from historical real-world data. Future research should explore additional strategies, such as utilising interconnected datasets and sequential data representation for temporal analysis. These considerations could improve the modelling of the complexity and realism of the data, and allow for the expansion of the current study, such as including multiple tasks per job. Moreover, the methodology could be adapted to incorporate further uncertainties, such as noise in travel times, resource heterogeneity, and the stochastic nature of time-related variables. For instance, a realistic truck and trailer scheduling problem may involve the dynamic nature of incoming jobs, the variations in fleet characteristics, and disruptions from weather and traffic conditions. Furthermore, extending the proposed methodology to address dynamic constraints, such as varying resource availability or time-dependent restrictions, would enhance the realism and applicability of synthetic benchmark generators, while also increasing the difficulty of ensuring solution feasibility [112, 135].

Future research should investigate advanced synthetic data generation models, such as Generative Adversarial Networks (GANs) [124], Diffusion Models [125], and Variational Autoencoders (VAEs) [136]. Although these models offer potential advantages, they also introduce challenges, such as interpretability, parametric complexity, and significant computational costs [121]. In any case, despite their limitations, analysing how these models could complement the proposed statistical method presents an attractive future stream for capturing complex data distributions and high-dimensional structures [86].

Recent studies have focused on integrating deep learning models with copulas for synthetic data generation, and point out the potential in combining the strengths of both approaches [119, 128]. Specifically, deep learning models can enhance the representation of complex dependencies identified by copulas, thereby improving synthetic data generation, while preserving the relationships between variables. Additionally, the robustness of deep learning models allows them to adapt to varying data distributions,

whereas copulas effectively handle static dependencies. This is particularly relevant in sectors with variable relationships influenced by external factors, such as fluctuation in finance [120].

Finally, a possible future direction is to extend the proposed methodology to other optimisation problems, such as truck and trailer allocation, with the aim to reduce operational costs and promote sustainability by minimising fuel consumption and the carbon footprint of transportation while maximising productivity.

This step will involve the integration of advanced data analytics and machine learning within dynamic combinatorial optimisation presents to improve solution methodologies. That is, by learning from previous solutions or decisions, algorithms can adaptively adjust their parameters, providing improvements to the predictive capability of algorithms. Hence, advanced problem-solving approaches can be developed by the interaction between online algorithms and new computational techniques.

Part IV

Appendix

Appendix A

Extended Elusivity Calculation for the Benchmark Generators in Case Study I

Tables A.1 and A.2 present the highest posterior probabilities regarding the overall elusivity of benchmark generators to the considered algorithms and performance metrics considered in Section 4.2.5. Specifically, the overall elusivity values for a generic case study is calculated as follows:

1. First, the online and restart version of the algorithms (A and A^r , respectively) must be implemented and executed for each DOP P , considering all dynamic configurations (changing at a certain frequency and magnitude of change), and performance metric ϕ .
2. Then, the performance of both versions of the algorithm is calculated by averaging their expected performance over several runs, $\mathbb{E}[\phi(A, P)]$.
3. Afterwards, the elusivity of each problem P (with a given frequency and magnitude of change) to algorithm A under performance metrics ϕ is calculated by the equation in Definition 4.8 in the manuscript, i.e. $\mathcal{E}(P, A, \phi) = \mathbb{E}[\phi(A, P) - \phi(A^r, P)]$, for a ϕ with minimisation purposes.

4. Next, in order to ensure that the results remain valid for the uncertainty related to the experimental process, a pairwise comparison of the performances of the online and restart algorithm version can be made from a statistical analysis.
5. Finally, the overall elusivity values for the different problems are calculated by counting the highest probability of an algorithm version being superior to the other (or a similar performance of both algorithm versions) for each dynamic optimisation problem P and performance metric ϕ .

Having said that, in Section 4.2.5.3, the overall elusivity values for the different problems, algorithms, and performance metrics considered in Case Study *I* have been calculated as follows:

1. First, the frameworks from the reference works that have considered for the experimentations have been replicated [11, 12, 35, 44, 103], and the restarting version of each algorithm has been incorporated as the baseline for the elusivity.
2. Then, after running the experiments and calculating the expected performance of the algorithms, we have calculated the elusivity values for each problem (step 3 in the list above) to an algorithm under a performance metric.
3. Finally, after performing a Bayesian analysis equivalent to the Wilcoxon pairwise signed-rank test, we have counted and averaged the highest posterior probability for each problem (with all combinations of frequency and magnitude of change), algorithm and performance metrics combination.

Table A.1: Summary of the highest posterior probabilities regarding the overall elusivity of DTSPs with fitness landscape rotation, constructed from **kroA100**, to the considered algorithms and performance metrics.

Instance	Algorithm	$\mathbb{E}[F_{BOG}]$			$\mathbb{E}[H_{\Delta m}]$		
		Online superior	Similar performance	Restart superior	Online superior	Similar performance	Restart superior
kroA100	EIACO	19	6	0	20	0	5
kroA100	RIACO	17	6	2	22	0	3
kroA100	EIGA	15	0	10	17	13	7
kroA100	RIGA	17	0	8	17	0	8
kroA100	EIPBIL	15	2	8	18	0	7
kroA100	RIPBIL	11	1	13	9	0	16
TOTAL		94 (0.63)	15 (0.10)	41 (0.27)	103 (0.69)	0 (0.00)	47 (0.31)

Table A.2: Summary of the highest posterior probabilities regarding the overall elusivity of DKPs with fitness landscape rotation to the considered algorithms and performance metrics.

Instance	Algorithm	$\mathbb{E}[F_{BOG}]$			$\mathbb{E}[H_{\Delta m}]$		
		Online superior	Similar performance	Restart superior	Online superior	Similar performance	Restart superior
joanA100	EIGA	6	1	18	8	3	14
joanB100	EIGA	7	0	18	9	2	14
joanC100	EIGA	6	1	18	9	2	14
joanA100	RIGA	7	4	14	5	13	7
joanB100	RIGA	7	4	14	5	11	9
joanC100	RIGA	5	6	14	5	11	9
joanA100	EIPBIL	5	2	18	3	4	18
joanB100	EIPBIL	5	2	18	4	3	18
joanC100	EIPBIL	5	1	19	3	3	19
joanA100	RIPBIL	4	1	20	1	2	22
joanB100	RIPBIL	4	1	20	0	4	21
joanC100	RIPBIL	4	1	20	0	3	22
TOTAL		65 (0.22)	24 (0.08)	211 (0.70)	52 (0.17)	61 (0.20)	187 (0.63)

Appendix B

Ethical and Legal Concerns

This chapter aims to systematically and transparently document the ethical and legal implications related to the data acquisition and the limitations of synthetic data generation (i.e. potential biases, errors, and inaccuracies), while ensuring confidentiality and compliance with legal guidelines regarding the inherent risks and limitations of the framework. In particular, the following sections describe (i) the performed data acquisition and processing steps, (ii) the identification of risk of synthetic data, and (iii) the identification of possible bias in the data and modelling process.

Note that a comprehensive overview of data processing and evaluation is provided in Chapter 5, whereas Chapter 6 provides information on the synthetic data model, its parameters, model validation and evaluation metrics, and the synthetic data generation pipeline, as well as its utility.

Data Acquisition, Processing, and Storage

The data is restricted by the EU General Data Protection Regulation (GDPR)¹, preventing the sharing of sensitive personal data due to privacy concerns in academic and industry research. It is worth noting that non-personal data can be classified as personal if it can identify or relate to an individual. As an alternative, synthetic data can hide sensitive information and help identify bias when appropriately manipulated [84].

¹<https://gdpr-info.eu/>

ARRC provided us their 2019 data using MongoDB, a database that uses NoSQL to store data as JSON. The database consists of 8 different datasets (collections) with different variables (fields). Data transformation and cleaning steps have been performed to correct and eliminate erroneous or incomplete data, and to transform the data into a suitable tabular format by retaining only essential information. The data has also been filtered through the geographical region to emphasise on the dynamic nature of real-world scheduling operation.

Risk Reduction

Synthetic data may contain hidden patterns or information from the original data that an attacker can use to infer certain details about the original data. The risk of synthetic data can be seen as the relationship between the likelihood of an inference and its impact, which depends on the scenario and domain. For example, synthetic data can lead to data misuse, privacy breaches, and unintended disclosures to identify personal data, medical records, or protected data. Therefore, identification and considerable care and awareness of manipulation and encryption processes are essential to mitigate potential vulnerabilities and comply with regulations, such as GDPR pseudonymisation guidelines. This ensures data integrity and confidentiality, while preventing the exploitation of any hidden patterns that could compromise the privacy of the original data.

In this work, the names of drivers, locations, and datasets for trucks and trailers (using registration numbers as identifiers) have been pseudonymised. For instance, locations are renamed as *Location1*, *Location2*, ..., *Location1070*. However, pseudonymisation should be approached cautiously, as certain attributes may inadvertently reveal identity information. Research by the authors in [137] demonstrated that original data could be approximated through reverse engineering. Thus, it is crucial to balance data accuracy with privacy to minimise the risk of disclosing sensitive information.

From the data provided, a risk related to linking data about driver skills has been identified. That is, even when drivers are anonymised, their specific skills may still lead to reidentification. Figure B.1 illustrates the number of drivers sharing the same skills,

B. Ethical and Legal Concerns

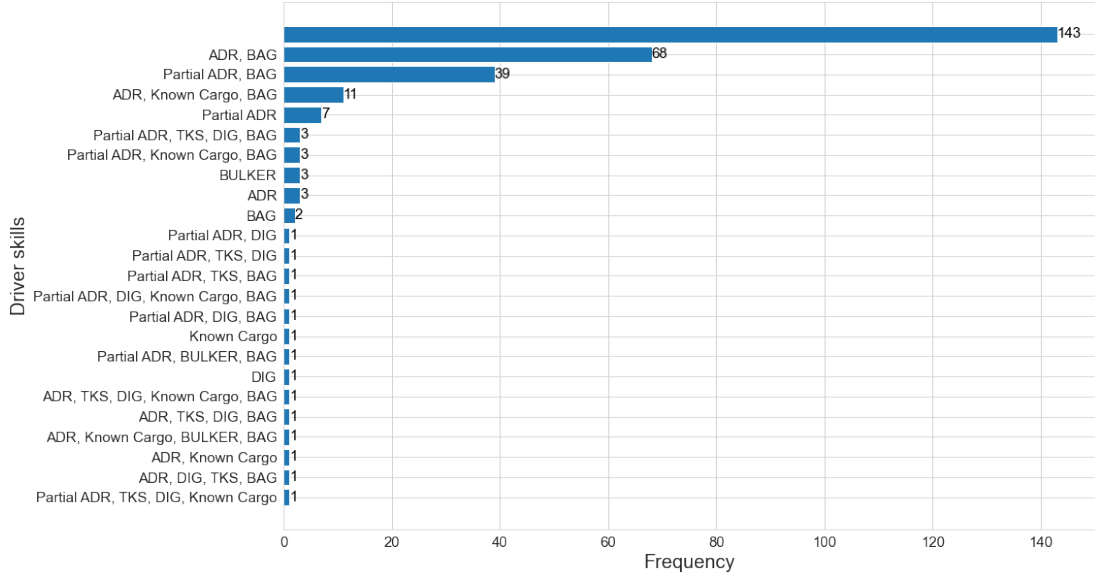


Figure B.1: Skills (constraints) covered by the drivers in 2019.

all of which are aggregated. The figure shows that most drivers have full or partial ADR skills, as well as specialised driver-skill constraints, such as *BAG* and *TKS*, which correspond to the packaging of ADR in a bag or a tank, respectively. Additionally, the skills *known cargo*, *DIG*, and *BULKER* reflect the skills in airport security, digicard, and bulker transportation, respectively. Therefore, the likelihood of using prior knowledge to disclose sensitive data for driver reidentification is very low.

Finally, it should be noted that this study deals with company-related information, which means that GDPR does not govern data about companies or any other legal entities. However, one-person companies may constitute personal data. Although it is not the case of this study, companies' geographical coordinates have been truncated to 4 digits to protect possible location-related privacy.

Bias Identification

Synthetic data generation models can learn and reproduce biases from their original data sources, affecting the quality, utility, and fairness of synthetic data. This issue is particularly evident in cases of contextual, social, and historical biases, such as gender or race biases [121]. In this study, simulated scenarios may reflect historical inequalities

or patterns that are no longer relevant or desired, such as the time distributions or the collection and deliver locations that may not hold at present or future. Another possible bias related to data may be on the preprocessing process. The privacy-preserving approach of the location must be used cautiously to generate reliable geographic locations, e.g. locations near the coastline should not be moved towards the sea, as this would affect the utility of synthetic data [138]. Users of synthetic data must be aware of these limitations and the degree to which locations have been altered to maintain privacy.

Although we analyse the modelling bias in detail in Section 6.2.1, there are two biases related to the modelling phase. First, an exclusion bias exists in the modelling process because single table synthetic data generation models in SDV consider non-repeating identification per table. Hence, for each job, the location and requested time of the first two tasks are considered. Although this approach covers approximately 92% of the jobs from the original dataset, it inadvertently omits certain events, resulting in the generation of partial synthetic data. Second, synthetic data generation models present a sampling bias when handling time constraints. Specifically, the models sacrifice certain mathematical properties of the original data to ensure that the input time does not exceed the collection time, and that the collection time remains shorter than the delivery time. Addressing this issue without negatively impacting the mathematical relationships within the time-related variables presents a considerable challenge.

Appendix C

Extended Performance

Evaluation of the Gaussian Copula

Tables C.1 – C.3 summarise the marginal distribution comparison for the analyses carried out in Section 6.2.2. Specifically, the marginal distribution comparison under the Kolmogorov-Smirnov test and total variation distance is shown for different time-related constraints after processing the input data, such as performing the time-feature extraction and the data splitting to generate separate dynamic and static models.

Figures C.1 – C.6 show the correlation comparison for the analyses carried out in Section 6.2.2. Figures show the correlation and contingency similarity scores for all combinations of columns as a matrix.

Table C.1: Marginal distribution comparison of Gaussian Copulas under different time-related constraints (continued).

SDV Constraint	Job		Input		Collect		Deliver		Req.		From		To		\overline{MDC}
	Constraints	Time	Time	Time	Time	Vehicle	Action	Location	Action	Location	Action	Location			
Original	Validation-only	99.35%	94.93%	97.97%	97.19%	99.19%	98.34%	95.53%	99.47%	95.47%	97.94%				
	Minutes transformation	99.32%	87.81%	93.92%	92.65%	99.70%	99.63%	96.86%	99.78%	95.80%	96.81%				
	Transformation – time diff.	99.34%	98.72%	98.90%	98.93%	99.70%	99.74%	96.91%	99.81%	96.72%	98.96%				
Weekly extrac.	Validation-only	99.31%	Extended in Table C.2				99.25%	99.27%	96.93%	99.73%	96.50%	95.38%			
	Minutes transformation	99.32%	Extended in Table C.2				99.70%	99.63%	96.86%	99.78%	95.80%	96.22%			
	Transformation – time diff.	99.51%	Extended in Table C.2				99.74%	99.76%	97.07%	99.82%	96.51%	94.00%			
Daily extrac.	Validation-only	99.44%	Extended in Table C.4				98.28%	98.07%	96.12%	99.40%	95.84%	93.18%			
	Minutes transformation	99.20%	Extended in Table C.4				99.44%	98.77%	95.93%	99.48%	95.66%	73.88%			
	Transformation – time diff.	99.64%	Extended in Table C.4				99.45%	99.57%	96.76%	99.89%	96.49%	91.04%			

Table B.1: Marginal distribution comparison of ensemble Gaussian Copulas under different time-related constraints.

SDV Constraint	Job		Input		Collect		Deliver		Req.		From		To		\overline{MDC}
	Constraints	Time	Time	Time	Time	Vehicle	Action	Location	Action	Location	Action	Location			
Original	Validation-only	99.51%	99.61%	96.78%	97.32%	99.13%	98.17%	96.09%	99.86%	95.63%	97.45%				
	Minutes transformation	99.56%	94.00%	96.08%	94.63%	99.52%	98.83%	96.75%	99.77%	96.48%	97.29%				
	Transformation – time diff.	99.48%	98.73%	98.88%	98.85%	99.47%	99.77%	97.16%	99.90%	96.77%	98.78%				
Weekly	Validation-only	99.67%					99.48%	99.54%	96.51%	99.83%	96.38%	95.22%			
	Minutes transformation	99.56%	Extended in Table C.3				99.52%	98.83%	96.75%	99.77%	96.48%	96.71%			
	Transformation – time diff.	99.35%					99.76%	99.58%	96.98%	99.91%	96.44%	94.22%			
Daily	Validation-only	99.61%					99.60%	99.48%	96.41%	99.83%	96.31%	94.86%			
	Minutes transformation	99.29%	Extended in Table C.5				98.87%	98.18%	96.18%	99.55%	95.66%	73.58%			
	Transformation – time diff.	99.44%					99.72%	99.92%	97.04%	99.81%	96.69%	92.55%			

Table C.2: Marginal distribution comparison of time variables using Gaussian Copulas on data with weekly decomposition.

SDV Constraint	Input Time			Collection Time			Delivery Time		
	Week	Weekday	Time	Week	Weekday	Time	Week	Weekday	Time
Validation-only	83.18%	98.68%	94.32%	87.42%	95.09%	97.60%	80.97%	94.92%	98.33%
Minutes transformation	87.47%	99.52%	93.61%	93.37%	99.48%	96.28%	90.94%	97.13%	87.40%
Time-difference transformation	97.66%	92.60%	72.28%	98.48%	99.68%	97.70%	98.28%	95.63%	53.49%

Table C.3: Marginal distribution comparison of time variables using ensemble Gaussian Copulas on data with weekly decomposition.

SDV Constraint	Input Time			Collection Time			Delivery Time		
	Week	Weekday	Time	Week	Weekday	Time	Week	Weekday	Time
Validation-only	84.32%	95.80%	93.00%	92.96%	97.22%	96.69%	84.46%	94.79%	97.75%
Minutes transformation	91.94%	98.56%	93.47%	93.96%	99.42%	97.29%	93.45%	98.33%	93.32%
Time-difference transformation	96.95%	94.31%	82.06%	98.28%	99.42%	97.84%	98.08%	96.36%	58.00%

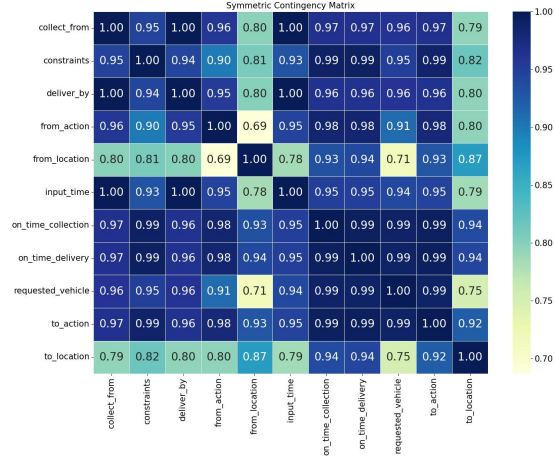
Table C.4: Marginal distribution comparison of time variables using Gaussian Copulas on data with daily decomposition.

SDV Constraint	Input Time		Collection Time		Delivery Time	
	Date	Time	Date	Time	Date	Time
Validation-only	83.31%	94.33%	86.17%	97.46%	71.66%	98.14%
Minutes transformation	76.88%	49.73%	76.97%	10.97%	79.48%	4.07%
Time-difference transformation	88.74%	71.45%	96.35%	97.65%	92.97%	53.56%

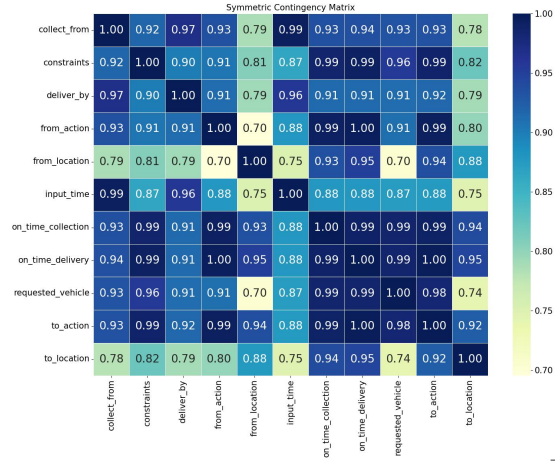
Table C.5: Marginal distribution comparison of time variables under daily-feature extraction using ensemble Gaussian Copulas.

SDV Constraint	Input Time		Collection Time		Delivery Time	
	Date	Time	Date	Time	Date	Time
Validation-only	86.08%	93.78%	88.87%	96.58%	83.15%	98.66%
Minutes transformation	75.68%	49.15%	76.73%	10.56%	79.08%	3.99%
Time-difference transformation	90.26%	82.00%	96.57%	97.92%	93.46%	57.76%

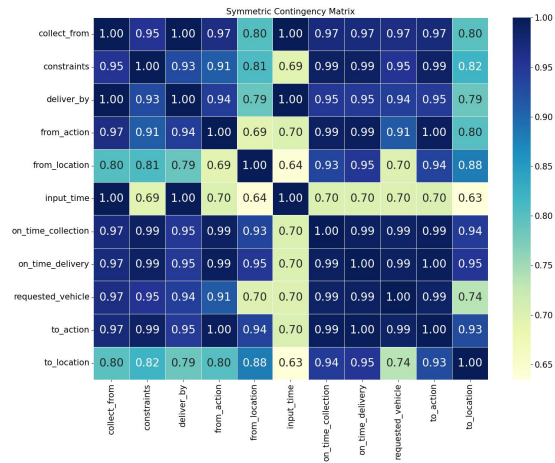
C. Extended Evaluation of Gaussian Copula



(a) Validation-only



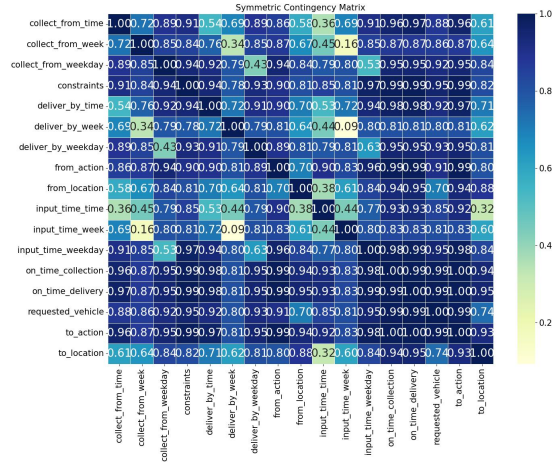
(b) Minutes transformation



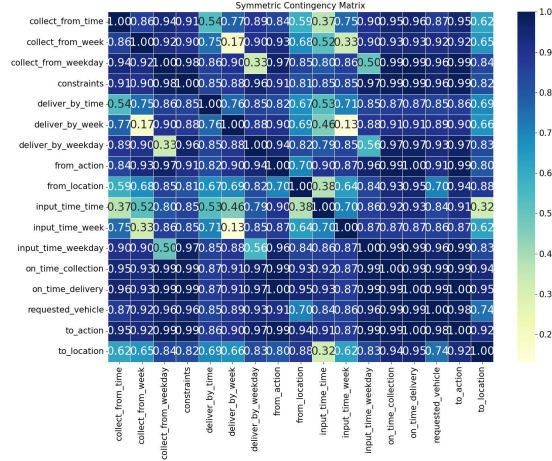
(c) Time-difference transformation

Figure C.1: Correlation comparison of original and synthetic data using Gaussian Copula.

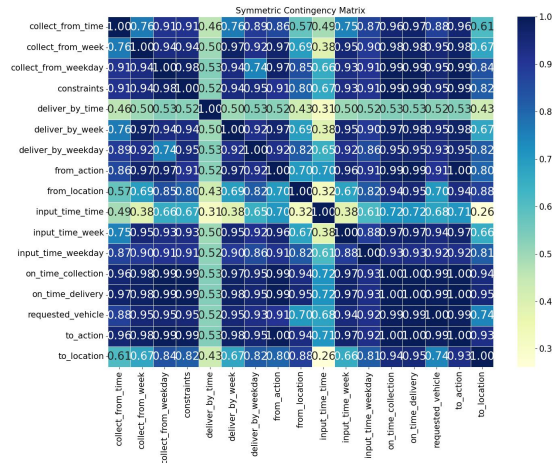
C. Extended Evaluation of Gaussian Copula



(a) Validation-only



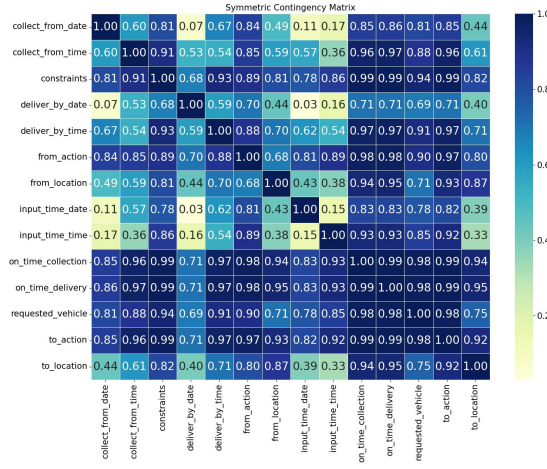
(b) Minutes transformation



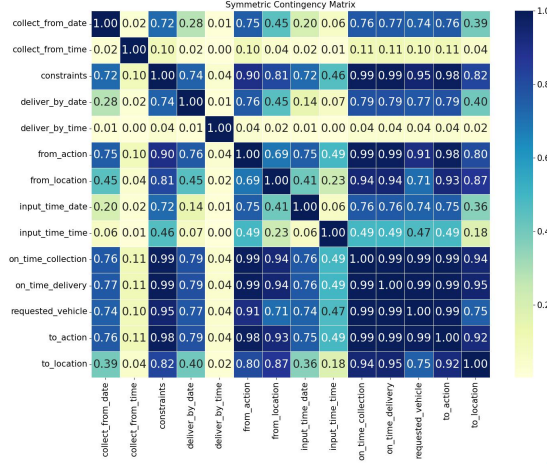
(c) Time-difference transformation

Figure C.2: Correlation comparison of original and synthetic data with weekly decomposition using Gaussian Copula.

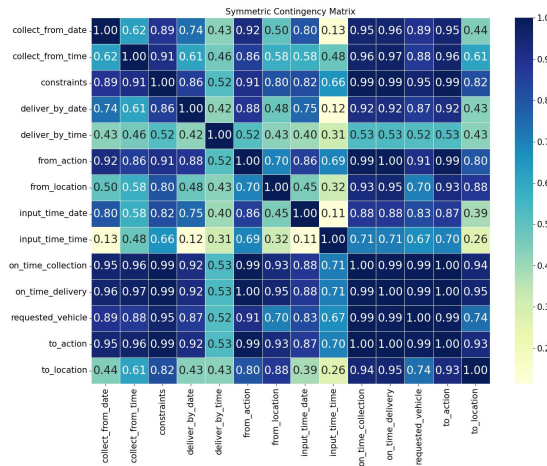
C. Extended Evaluation of Gaussian Copula



(a) Validation-only



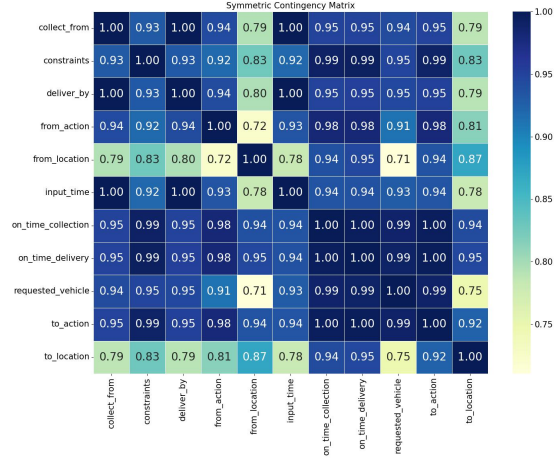
(b) Minutes transformation



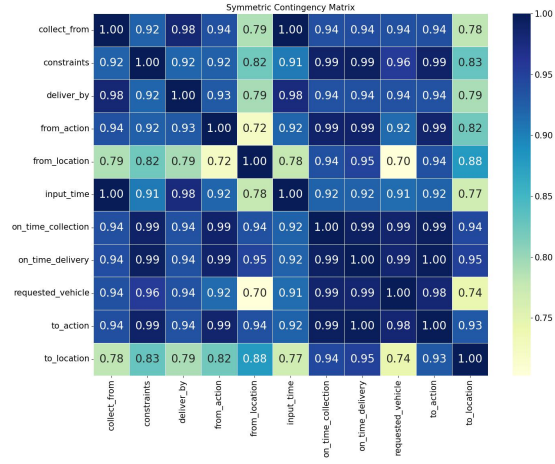
(c) Time-difference transformation

Figure C.3: Correlation comparison of original and synthetic data with daily decomposition using Gaussian Copula.

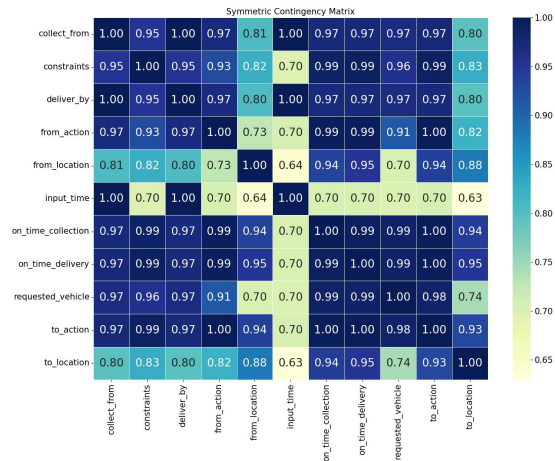
C. Extended Evaluation of Gaussian Copula



(a) Validation-only



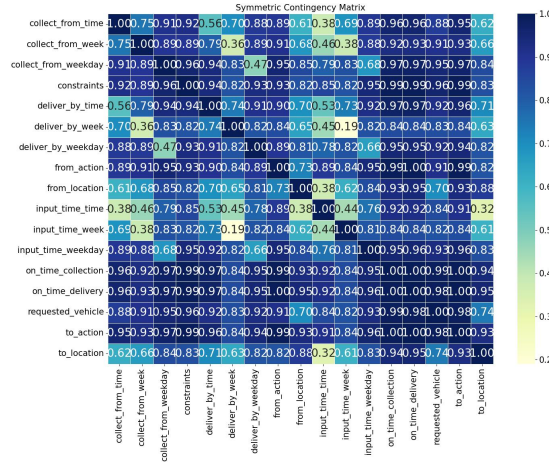
(b) Minutes transformation



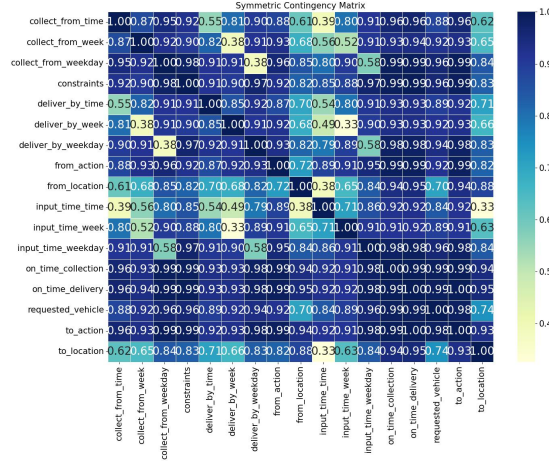
(c) Time-difference transformation

Figure C.4: Correlation comparison of original and synthetic data using ensemble Gaussian Copula.

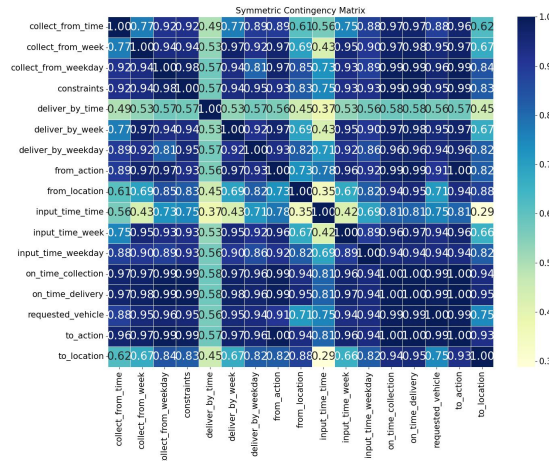
C. Extended Evaluation of Gaussian Copula



(a) Validation-only



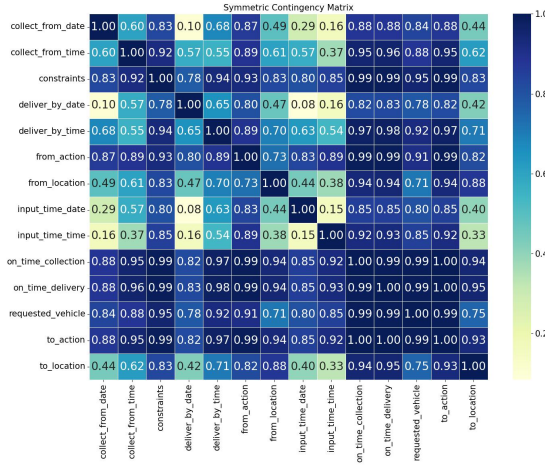
(b) Minutes transformation



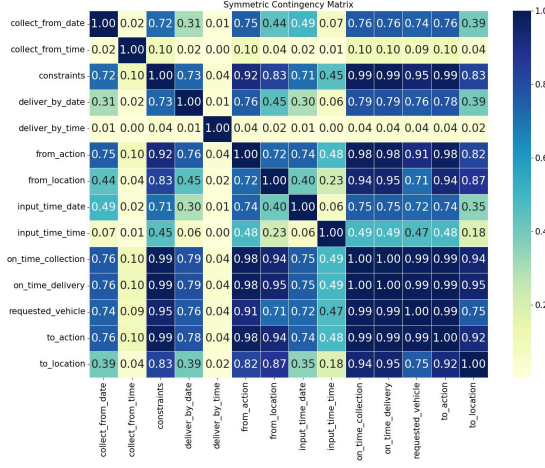
(c) Time-difference transformation

Figure C.5: Correlation comparison of original and synthetic data with weekly decomposition using ensemble Gaussian Copula.

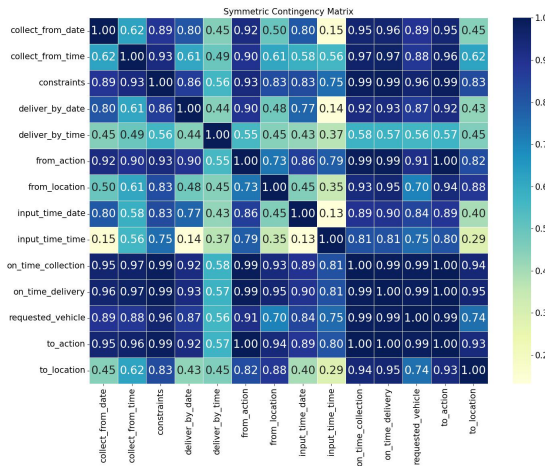
C. Extended Evaluation of Gaussian Copula



(a) Validation-only



(b) Minutes transformation



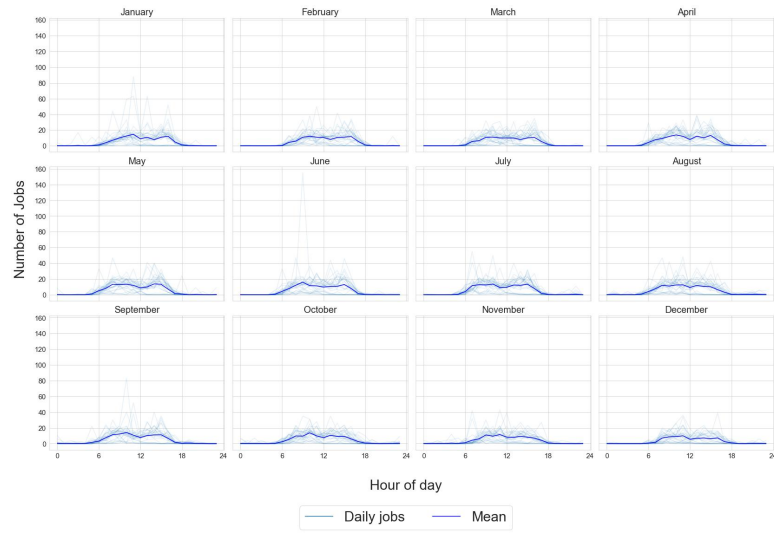
(c) Time-difference transformation

Figure C.6: Correlation comparison of original and synthetic data with weekly decomposition using ensemble Gaussian Copula.

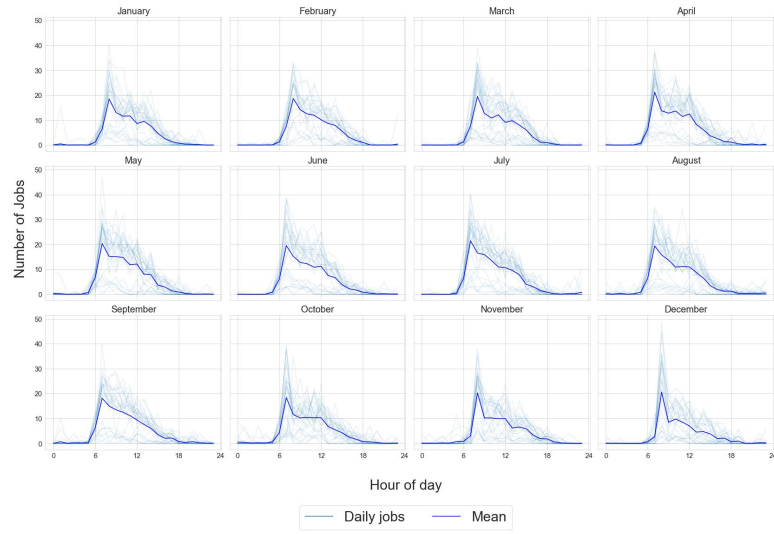
Appendix D

Time-series Analysis of the ARRC data

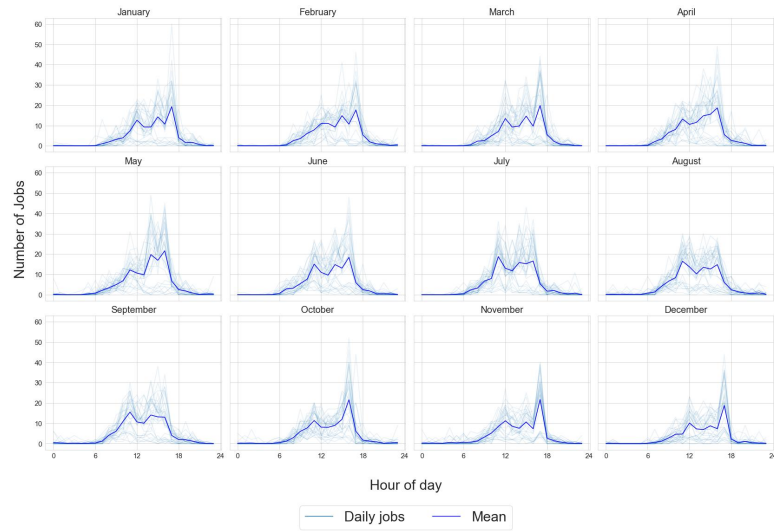
Figure D.1 shows the hourly input, collection and delivery times for each day in a monthly segregated way. The goal is to see if the daily seasonality is related to the month of the year.



(a) Input time



(b) Collection time



(c) Delivery time

Figure D.1: Daily seasonality of the data in each respective month. The light blue lines indicate the time at which jobs are inputted or requested to be collected or delivered for each day, and the dark blue line is the average collection time.

D. Time-series Analysis of the ARRC data

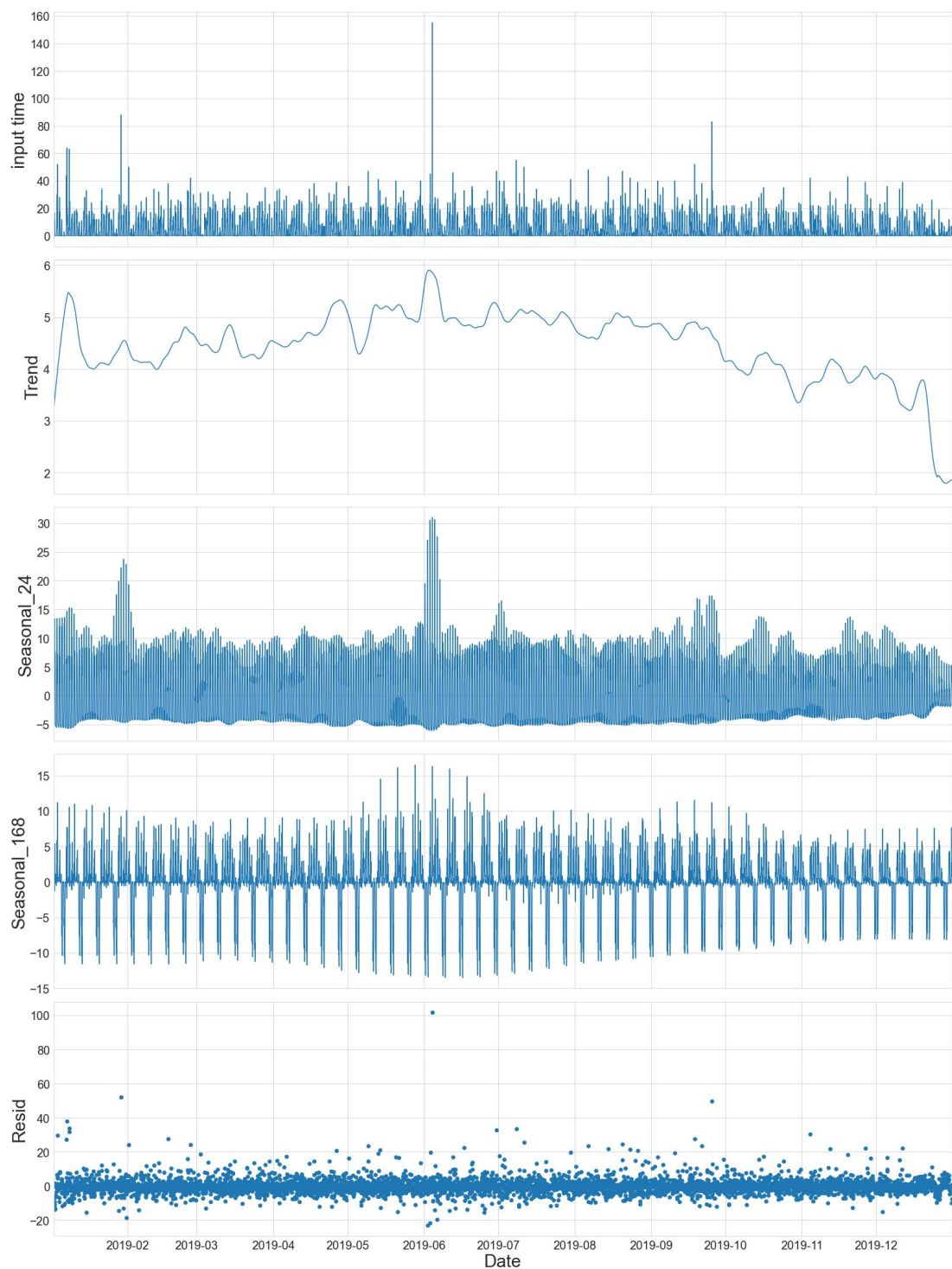


Figure D.2: MSTL decomposition of the input time of jobs. Specifically, the trend, daily (seasonal_24) and weekly (seasonal_168) seasonalities, and residual components are individually displayed.

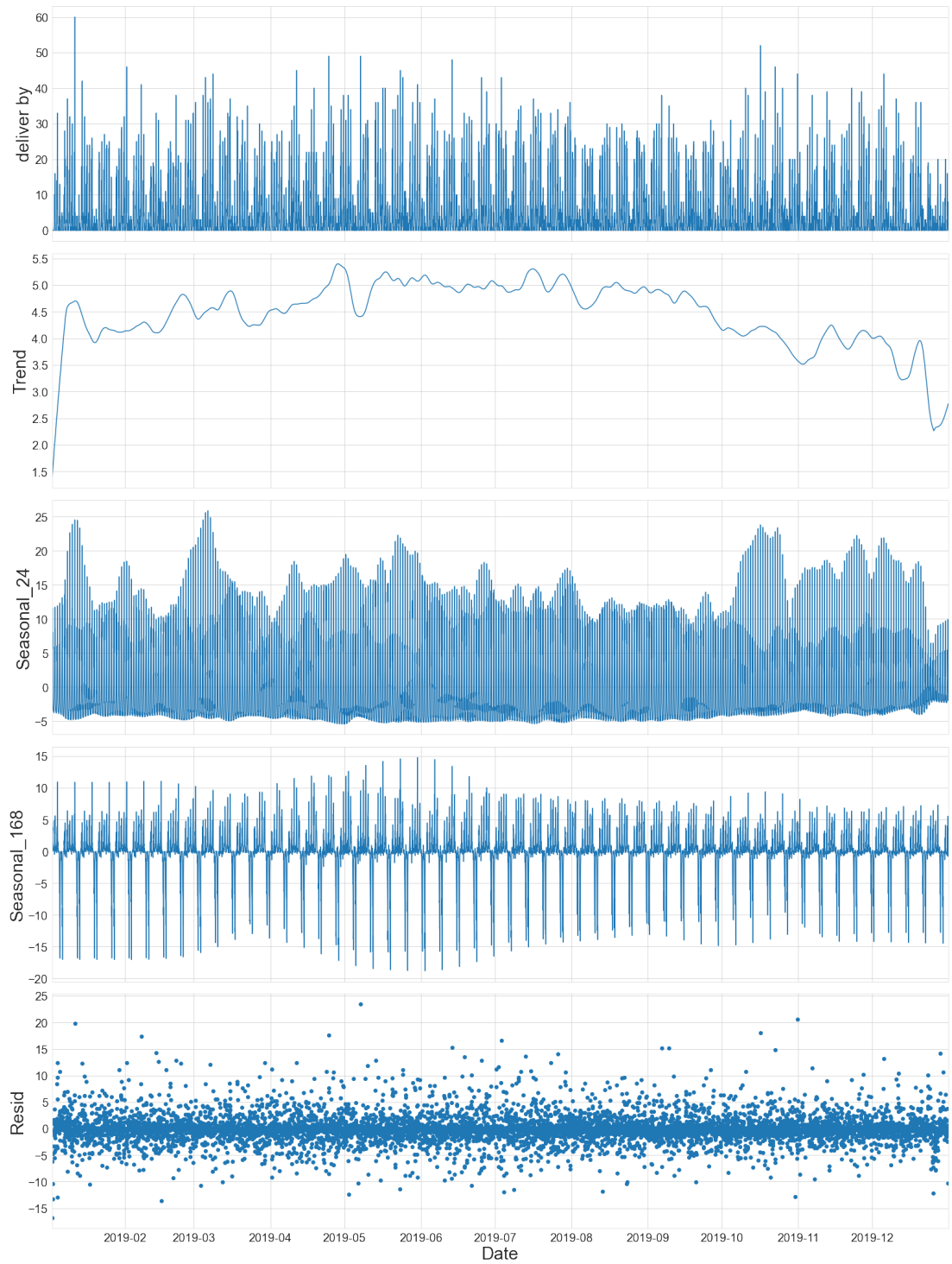


Figure D.3: MSTL decomposition of the collection time of jobs. Specifically, the trend, daily (seasonal_24) and weekly (seasonal_168) seasonalities, and residual components are individually displayed.

D. Time-series Analysis of the ARRC data

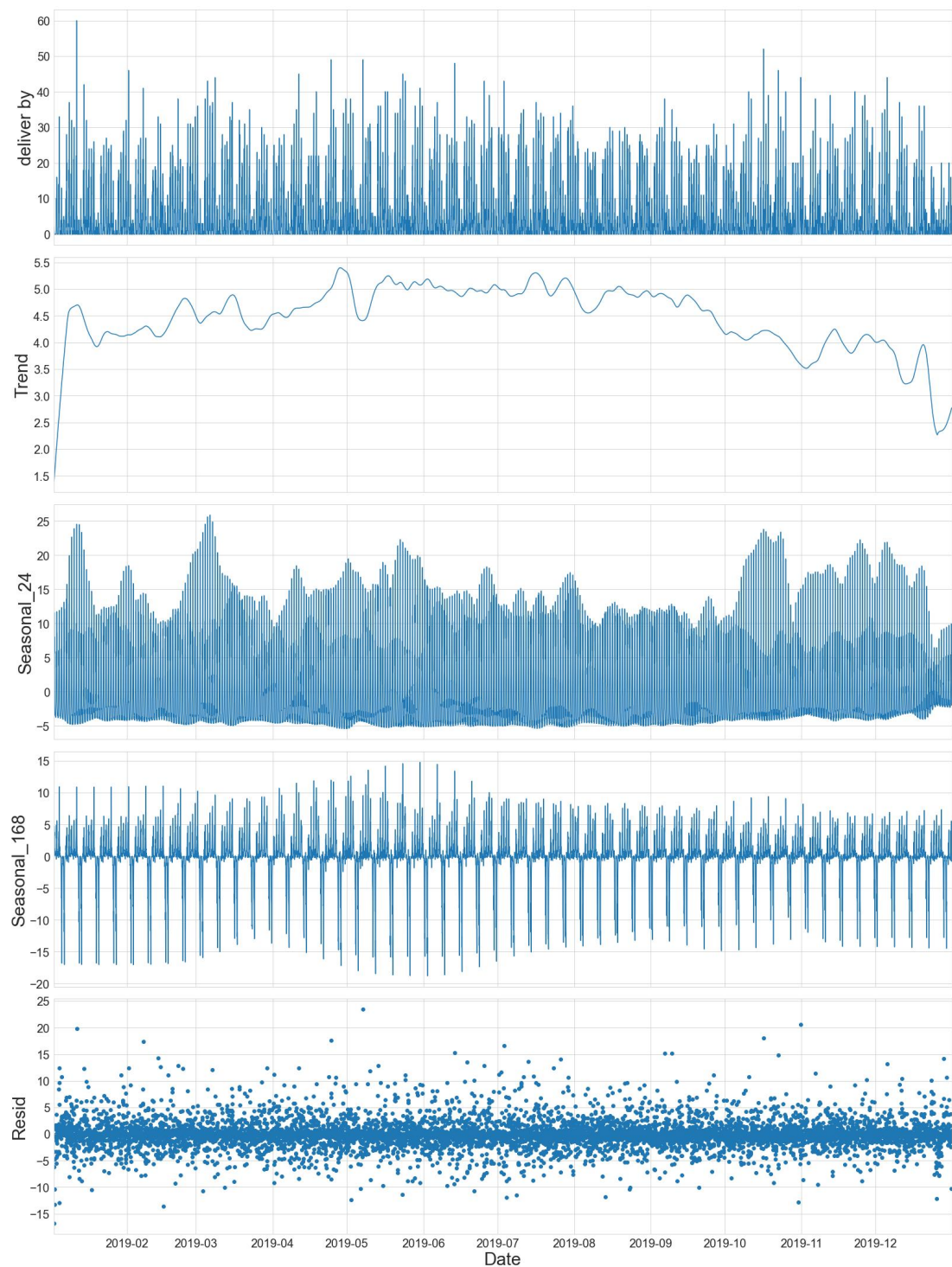


Figure D.4: MSTL decomposition of the delivery time of jobs. Specifically, the trend, daily (seasonal_24) and weekly (seasonal_168) seasonalities, and residual components are individually displayed.

Appendix E

STL operations and Loess Regression

The Loess regression, which stands for *locally estimated scatterplot smoothing*, is a non-parametric method that iteratively applies a locally weighted regression to a subset of data points within a moving window [139]. In short, for each point in the data, a subset of data points is filtered, and it is fitted with a local polynomial that has more weight on the data points closer to the center of the window. Note that the window size affects the number of data points in each subset, where smaller values reveal more local details and larger values create smoother curves. Polynomials of degrees zero, one or two (constant, linear, or quadratic models, respectively) are usually fitted to the subset of the data. The regression function is then calculated from the local polynomials to get a smooth curve, also called loess curve, that may reflect trend, seasonality or residual components. Algorithm E.1 summarises the Loess regression.

Algorithm E.1 LOESS: LOcally Estimated Scatterplot Smoothing

Input: Time-series data X data, the window size s , polynomial degree δ .

Output: Time-series component as a Loess curve.

- 1: **repeat**
 - 2: Find the s closest data points to x , $X_x \subset X, |X_x| = s$.
 - 3: Fit a weighted linear regression model of degree δ .
 - 4: **until** $\forall x \in X$.
-

Algorithm E.2 STL: Seasonal-Trend decomposition using LOESS

Input: Original data X_t , stopping criterion, Loess function \mathcal{L} .

Output: Trend \hat{T}_t , Seasonality \hat{S}_t , Residuals \hat{R}_t .

- 1: Estimate \hat{T}_t from $\mathcal{L}(X_t)$.
 - 2: Estimate \hat{S}_t from $\mathcal{L}(X_t - \hat{T}_t)$.
 - 3: **repeat**
 - 4: Update \hat{T}_t from $\mathcal{L}(X_t - \hat{S}_t)$.
 - 5: Update \hat{S}_t from $\mathcal{L}(X_t - \hat{T}_t)$.
 - 6: **until** Stopping criterion is met.
 - 7: Estimate \hat{R}_t from $\mathcal{L}(X_t - \hat{T}_t - \hat{S}_t)$.
-

The Seasonal-Trend decomposition using LOESS (STL) is a method of time-series analysis that separates a time-series into trend, seasonality, and residual components using Loess [115]. It iteratively applies Loess to multiple transformations of the original time-series data to extract the time-series components. More details in Algorithm E.2.

STL presents several advantages compared to other methods, such as its simplicity, flexibility, or robustness. However, does not handle multiplicative without processing the data first. To that end, time-series data must be transformed (logs) to use the additive decomposition for STL, and then transform it back to its original format.

Bibliography

- [1] Y. Jin and J. Branke, “Evolutionary Optimization in Uncertain Environments – A Survey,” IEEE TEVC, vol. 9, no. 3, pp. 303–317, 2005.
- [2] J. Branke, Evolutionary optimization in dynamic environments. Springer Science & Business Media, 2002.
- [3] C. Cruz, J. R. González, and D. A. Pelta, “Optimization in Dynamic Environments: A Survey on Problems, Methods and Measures,” Soft Computing, vol. 15, no. 7, pp. 1427–1448, 2011.
- [4] A. Younes, P. Calamai, and O. Basir, “Generalized Benchmark Generation for Dynamic Combinatorial Problems,” in Proceedings of GECCO, pp. 25–31, 2005.
- [5] P. Rohlfshagen and X. Yao, “Attributes of Dynamic Combinatorial Optimisation,” in Simulated Evolution and Learning, pp. 442–451, 2008.
- [6] T. Back, “On the Behavior of Evolutionary Algorithms in Dynamic Environments,” in Proceedings of CEC, pp. 446–451, 1998.
- [7] P. Rohlfshagen, P. K. Lehre, and X. Yao, “Dynamic evolutionary optimisation: An analysis of frequency and magnitude of change,” in Proceedings of GECCO, pp. 1713–1720, 2009.
- [8] T. T. Nguyen, Continuous Dynamic Optimisation using Evolutionary Algorithms. PhD thesis, University of Birmingham, 2011.

- [9] T. T. Nguyen, S. Yang, and J. Branke, “Evolutionary Dynamic Optimization: A Survey of the State of the Art,” Swarm Evol. Comput., vol. 6, pp. 1 – 24, 2012.
- [10] R. Tinós and S. Yang, “Analysis of Fitness Landscape Modifications in Evolutionary Dynamic Optimization,” Information Sciences, vol. 282, pp. 214–236, 2014.
- [11] S. Yang and X. Yao, “Dual Population-based Incremental Learning for Problem Optimization in Dynamic Environments,” in Asia Pacific Symposium on Intelligent and Evolutionary Systems, 2003.
- [12] S. Yang and X. Yao, “Experimental study on population-based incremental learning algorithms for dynamic optimization problems,” Soft Computing, vol. 9, no. 11, pp. 815–834, 2005.
- [13] P. A. N. Bosman, “Learning, Anticipation and Time-Deception in Evolutionary Online Dynamic Optimization,” in Proceedings of GECCO, pp. 39–47, 2005.
- [14] O. Regnier-Coudert, J. McCall, M. Ayodele, and S. Anderson, “Truck and Trailer Scheduling in a Real World, Dynamic and Heterogeneous Context,” Transportation Research Part E: Logistics and Transportation Review, vol. 93, pp. 389–408, 2016.
- [15] J. Branke, “Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems,” in Proceedings of CEC, vol. 3, pp. 1875–1882, 1999.
- [16] X. Xiang, J. Qiu, J. Xiao, and X. Zhang, “Demand Coverage Diversity based Ant Colony Optimization for Dynamic Vehicle Routing Problems,” Engineering Applications of Artificial Intelligence, vol. 91, 2020.
- [17] J. Ceberio, Solving Permutation Problems with Estimation of Distribution Algorithms and Extensions Thereof. PhD thesis, UPV/EHU, 2014.
- [18] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., 1979.

- [19] L. Hernando, A. Mendiburu, and J. A. Lozano, “Anatomy of the Attraction Basins: Breaking with the Intuition,” Evolutionary Computation, vol. 27, no. 3, pp. 435–466, 2019.
- [20] C. M. Reidys and P. F. Stadler, “Combinatorial Landscapes,” Working Papers 01-03-014, Santa Fe Institute, Mar. 2001.
- [21] L. Hernando, A. Mendiburu, and J. A. Lozano, “An Evaluation of Methods for Estimating the Number of Local Optima in Combinatorial Optimization Problems,” Evolutionary Computation, vol. 21, no. 4, pp. 625–658, 2013.
- [22] M.-H. Tayarani-N. and A. Prügel-Bennett, “On the Landscape of Combinatorial Optimization Problems,” IEEE TEVC, vol. 18, no. 3, pp. 420–434, 2014.
- [23] S. Verel, F. Daolio, G. Ochoa, and M. Tomassini, “Local Optima Networks with Escape Edges,” in International Conference on Artificial Evolution, pp. 10 – 23, 2011.
- [24] E. Irurozki, Sampling and Learning Distance-based Probability Models for Permutation Spaces. PhD thesis, University of the Basque Country, 2014.
- [25] J. Branke, M. Orbayı, and Ş. Uyar, “The Role of Representations in Dynamic Knapsack Problems,” in Applications of Evolutionary Computing, pp. 764–775, Springer Berlin Heidelberg, 2006.
- [26] T. C. Koopmans and M. Beckmann, “Assignment Problems and the Location of Economic Activities,” Econometrica, vol. 25, no. 1, pp. 53–76, 1957.
- [27] Z. Drezner, “Taking Advantage of Symmetry in some Quadratic Assignment Problems,” Information Systems and Operational Research, vol. 57, no. 4, pp. 623–641, 2019.
- [28] X. Benavides, J. Ceberio, and L. Hernando, “On the Symmetry of the Quadratic Assignment Problem through Elementary Landscape Decomposition,” in Proceedings of GECCO, p. 1414–1422, 2021.

- [29] R. Martí and G. Reinelt, The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization, vol. 175. Springer Science & Business Media, 2011.
- [30] J. Ceberio, A. Mendiburu, and J. A. Lozano, “The Linear Ordering Problem Revisited,” EJOR, vol. 241, no. 3, pp. 686–696, 2015.
- [31] L. Hernando, A. Mendiburu, and J. A. Lozano, “Journey to the Center of the Linear Ordering Problem,” in Proceedings of GECCO, pp. 201—209, 2020.
- [32] S. Yang, Y. Jiang, and T. T. Nguyen, “Metaheuristics for Dynamic Combinatorial Optimization Problems,” IMA Journal of Management Mathematics, vol. 24, no. 4, pp. 451–480, 2012.
- [33] G. Zames, N. Ajlouni, N. Ajlouni, N. Ajlouni, J. Holland, W. Hills, and D. Goldberg, “Genetic Algorithms in Search, Optimization and Machine Learning,” Information Technology Journal, vol. 3, no. 1, pp. 301–302, 1981.
- [34] M. Mavrovouniotis, C. Li, and S. Yang, “A Survey of Swarm Intelligence for Dynamic Optimization: Algorithms and Applications,” Swarm Evol. Comput., vol. 33, pp. 1 – 17, 2017.
- [35] M. Mavrovouniotis, S. Yang, M. Van, C. Li, and M. Polycarpou, “Ant Colony Optimization Algorithms for Dynamic Optimization: A Case Study of the Dynamic Travelling Salesperson Problem,” IEEE Computational Intelligence Magazine, vol. 15, no. 1, pp. 52–63, 2020.
- [36] M. Mavrovouniotis and S. Yang, “Ant Colony Optimization with Immigrants Schemes in Dynamic Environments,” in Proceedings of PPSN, pp. 371–380, Springer-Verlag, 2010.
- [37] D. Green, A. Aleti, and J. Garcia, The Nature of Nature: Why Nature-Inspired Algorithms Work, pp. 1–27. Cham: Springer International Publishing, 2017.

- [38] M. Mavrovouniotis and S. Yang, “Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors,” in Applied Soft Computing, pp. 4023 – 4037, 2013.
- [39] J. Branke, “Evolutionary Approaches to Dynamic Optimization Problems – Updated Survey,” 2001.
- [40] M. Črepinšek, S.-H. Liu, and M. Mernik, “Exploration and Exploitation in Evolutionary Algorithms: A Survey,” ACM, vol. 45, no. 3, 2013.
- [41] A. Baykasoğlu and Z. D. U. Durmuşoğlu, “Dynamic Optimization in a Dynamic and Unpredictable World,” in Proceedings of PICMET, pp. 1–8, 2011.
- [42] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, “A Survey of Evolutionary Continuous Dynamic Optimization Over Two Decades—Part A,” IEEE TEVC, vol. 25, no. 4, pp. 609–629, 2021.
- [43] J. J. Grefenstette et al., “Genetic Algorithms for Changing Environments,” in Proceedings of PPSN, vol. 2, pp. 137–144, Citeseer, 1992.
- [44] S. Yang, “Genetic Algorithms with Memory-and Elitism-Based Immigrants in Dynamic Environments,” Evolutionary Computation, vol. 16, no. 3, pp. 385–416, 2008.
- [45] M. Mavrovouniotis and S. Yang, “Population-Based Incremental Learning with Immigrants Schemes in Changing Environments,” in 2015 IEEE Symposium Series on Computational Intelligence, pp. 1444–1451, 2015.
- [46] R. Tinós and S. Yang, “A Self-Organizing Random Immigrants Genetic Algorithm for Dynamic Optimization Problems,” Genetic Programming and Evolvable Machines, vol. 8, no. 3, pp. 255–286, 2007.
- [47] H. G. Cobb, “An Investigation into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms having Continuous, Time-dependent Nonstationary Environments,” tech. rep., Naval Research Lab Washington DC, 1990.

- [48] T. T. Nguyen and Xin Yao, “Benchmarking and Solving Dynamic Constrained Problems,” in IEEE CSC, pp. 690–697, 2009.
- [49] G. R. Kramer and J. C. Gallagher, “An Examination of Hypermutation and Random Immigrant Variants of mrCGA for Dynamic Environments,” in Proceedings of GECCO, pp. 454–455, Springer Berlin Heidelberg, 2003.
- [50] R. Tinós, D. Whitley, and A. Howe, “Use of Explicit Memory in the Dynamic Traveling Salesman Problem,” in Proceedings of GECCO, pp. 999–1006, 2014.
- [51] S. Yang, “Memory-Enhanced Univariate Marginal Distribution Algorithms for Dynamic Optimization Problems,” in Proceedings of CEC, vol. 3, pp. 2560–2567, 2005.
- [52] J. Branke and H. Schmeck, Designing Evolutionary Algorithms for Dynamic Optimization Problems, pp. 239–262. Springer Berlin Heidelberg, 2003.
- [53] R. Mendes and A. S. Mohais, “DynDE: a Differential Evolution for Dynamic Optimization Problems,” in Proceedings of CEC, vol. 3, pp. 2808–2815, 2005.
- [54] H. Ma, S. Shen, M. Yu, Z. Yang, M. Fei, and H. Zhou, “Multi-Population Techniques in Nature Inspired Optimization Algorithms: A Comprehensive Survey,” Swarm Evol. Comput., vol. 44, pp. 365–387, 2019.
- [55] R. Allmendinger and J. Knowles, “Evolutionary Optimization on Problems Subject to Changes of Variables,” in Proceedings of PPSN, pp. 151–160, 2010.
- [56] P. Rohlfshagen and X. Yao, “Dynamic Combinatorial Optimisation Problems: An Analysis of the Subset Sum Problem,” Soft Computing, vol. 15, no. 9, pp. 1723–1734, 2011.
- [57] H. Fu, P. R. Lewis, B. Sendhoff, K. Tang, and X. Yao, “What are Dynamic Optimization Problems?,” in Proceedings of CEC, pp. 1550–1557, 2014.
- [58] D. E. Goldberg and R. E. Smith, “Nonstationary Function Optimization Using Genetic Algorithm with Dominance and Diploidy,” in Proceedings of the

- Second International Conference on Genetic Algorithms and Their Application, pp. 59–68, L. Erlbaum Associates Inc., 1987.
- [59] K. Weicker, “An Analysis of Dynamic Severity and Population Size,” in Proceedings of PPSN, pp. 159–168, Springer Berlin Heidelberg, 2000.
- [60] C. Li and S. Yang, “A Generalized Approach to Construct Benchmark Problems for Dynamic Optimization,” in Simulated Evolution and Learning, pp. 391–400, Springer Berlin Heidelberg, 2008.
- [61] A. Younes, O. Basir, P. Calamai, and S. Areibi, Adapting Genetic Algorithms for Combinatorial Optimization Problems in Dynamic Environments. INTECH Open Access Publisher, 2008.
- [62] B. Doerr, C. Doerr, and F. Neumann, “Fast Re-Optimization via Structural Diversity,” in Proceedings of GECCO, pp. 233–241, 2019.
- [63] J. Bossek, F. Neumann, P. Peng, and D. Sudholt, “Runtime analysis of randomized search heuristics for dynamic graph coloring,” in Proceedings of GECCO, pp. 1443–1451, 2019.
- [64] A. Lissovoi and C. Witt, “Runtime Analysis of Ant Colony Optimization on Dynamic Shortest Path Problems,” in Proceedings of GECCO, pp. 1605–1612, 2013.
- [65] M. Pourhassan, W. Gao, and F. Neumann, “Maintaining 2-Approximations for the Dynamic Vertex Cover Problem Using Evolutionary Algorithms,” in Proceedings of GECCO, pp. 903–910, 2015.
- [66] J. Branke, E. Salihoğlu, and c. Uyar, “Towards an Analysis of Dynamic Environments,” in Proceedings of GECCO, pp. 1433–1440, 2005.
- [67] X. Yu, Y. Jin, K. Tang, and X. Yao, “Robust Optimization Over Time — A New Perspective on Dynamic Optimization Problems,” in Proceeding of CEC, pp. 1–6, 2010.

- [68] D. Yazdani, R. Cheng, D. Yazdani, J. Branke, Y. Jin, and X. Yao, “A Survey of Evolutionary Continuous Dynamic Optimization Over Two Decades—Part B,” IEEE TEVC, vol. 25, no. 4, pp. 630–650, 2021.
- [69] M. Mavrovouniotis, S. Yang, and X. Yao, “A Benchmark Generator for Dynamic Permutation-Encoded Problems,” in Proceedings on PPSN, pp. 508–517, 2012.
- [70] S. Jiang, S. Yang, X. Yao, K. C. Tan, M. Kaiser, and N. Krasnogor, “Benchmark Functions for the CEC’2018 Competition on Dynamic Multiobjective Optimization,” tech. rep., Newcastle University, 2018.
- [71] G. Pamparà and A. P. Engelbrecht, “A Generator for Dynamically Constrained Optimization Problems,” in Proceedings of GECCO, p. 1441–1448, Association for Computing Machinery, 2019.
- [72] H. Li and G. Zhang, “Designing Benchmark Generator for Dynamic Optimization Algorithm,” IEEE Access, vol. PP, pp. 1–1, 12 2021.
- [73] J. G. O. L. Duhain and A. P. Engelbrecht, “Towards a more Complete Classification System for Dynamically Changing Environments,” in Proceedings of CEC, pp. 1–8, 2012.
- [74] R. C. Eberhart and Y. Shi, “Tracking and Optimizing Dynamic Systems with Particle Swarms,” in Proceedings of CEC, vol. 1, pp. 94–100, 2001.
- [75] P. J. Angeline, “Tracking Extrema in Dynamic Environments,” in Evolutionary Programming VI, pp. 335–345, Springer Berlin Heidelberg, 1997.
- [76] K. Weicker, “An Analysis of Dynamic Severity and Population Size,” in Proceedings of PPSN, pp. 159–168, Springer Berlin Heidelberg, 2000.
- [77] K. Weicker, “Performance Measures for Dynamic Environments,” in Proceedings of PPSN, pp. 64–73, Springer Berlin Heidelberg, 2002.
- [78] K. De Jong, “Evolving in a Changing World,” in Foundations of Intelligent Systems, pp. 512–519, Springer Berlin Heidelberg, 1999.

- [79] J. Alza, M. Bartlett, J. Ceberio, and J. McCall, “On the Definition of Dynamic Permutation Problems under Landscape Rotation,” in Proceedings of GECCO, pp. 1518–1526, 2019.
- [80] R. Tinós and S. Yang, “An Analysis of the XOR Dynamic Problem Generator Based on the Dynamical System,” in Proceedings of PPSN, pp. 274–283, Springer Berlin Heidelberg, 2010.
- [81] J. Alza, M. Bartlett, J. Ceberio, and J. McCall, “Towards the Landscape Rotation as a Perturbation Strategy on the Quadratic Assignment Problem,” in Proceedings of GECCO, pp. 1405–1413, 2021.
- [82] M. Guntsch, M. Middendorf, and H. Schmeck, “An Ant Colony Optimization Approach to Dynamic TSP,” in Proceedings of GECCO, pp. 860–867, 2001.
- [83] A. McCrabb, H. Nigatu, A. Getachew, and V. Bertacco, “DyGraph: a Dynamic Graph Generator and Benchmark Suite,” in Proceedings of ACM SIGMOD, 2022.
- [84] Y. Lu, M. Shen, H. Wang, X. Wang, C. van Rechem, and W. Wei, “Machine Learning for Synthetic Data Generation: A Review,” arXiv preprint arXiv:2302.04062, 2023.
- [85] S. James, C. Harbron, J. Branson, and M. Sundler, “Synthetic Data Use: Exploring Use Cases to Optimise Data Utility,” Discover Artificial Intelligence, vol. 1, no. 1, p. 15, 2021.
- [86] A. Figueira and B. Vaz, “Survey on Synthetic Data Generation, Evaluation Methods and GANs,” Mathematics, vol. 10, no. 15, p. 2733, 2022.
- [87] E. L. Yu and P. N. Suganthan, “Maverick Research: Forget About Your Real Data — Synthetic Data Is the Future of AI,” 2021.
- [88] J. Alza, M. Bartlett, J. Ceberio, and J. McCall, “Analysing the Fitness Landscape Rotation for Combinatorial Optimisation,” in Proceedings of PPSN, pp. 533–547, Springer International Publishing, 2022.

- [89] E. Pitzer and M. Affenzeller, A Comprehensive Survey on Fitness Landscape Analysis, pp. 161–191. Springer Berlin Heidelberg, 2012.
- [90] K. M. Malan and A. P. Engelbrecht, “A Survey of Techniques for Characterising Fitness Landscapes and some Possible Ways Forward,” Information Sciences, vol. 241, pp. 148–163, 2013.
- [91] A. Prugel-Bennett and M.-H. Tayarani-Najaran, “Maximum Satisfiability: Anatomy of the Fitness Landscape for a Hard Combinatorial Optimization Problem,” IEEE TEVC, vol. 16, no. 3, pp. 319–338, 2012.
- [92] S. Yang, “Non-Stationary Problem Optimization Using the Primal-Dual Genetic Algorithm,” in Proceedings of CEC, vol. 3, pp. 2246–2253, 2003.
- [93] H. R. Lourenço, O. C. Martin, and T. Stützle, Iterated Local Search, pp. 320–353. Springer US, 2003.
- [94] N. Mladenović and P. Hansen, “Variable Neighborhood Search,” Computers & Operations Research, vol. 24, no. 11, pp. 1097–1100, 1997.
- [95] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing,” Science, vol. 220, no. 4598, pp. 671–680, 1983.
- [96] E. K. Burke and Y. Bykov, “The Late Acceptance Hill-Climbing Heuristic,” 2017.
- [97] M. Ayodele, J. Mccall, O. Regnier-Coudert, and L. Bowie, “A Random Key based Estimation of Distribution Algorithm for the Permutation Flowshop Scheduling Problem,” in Proceedings in GECCO, pp. 2364–2371, 06 2017.
- [98] G. Ochoa, K. M. Malan, and C. Blum, “Search Trajectory Networks: A Tool for Analysing and Visualising the Behaviour of Metaheuristics,” Applied Soft Computing, 2021.
- [99] E. Taillard, “Robust Taboo Search for the Quadratic Assignment Problem,” Parallel Computing, vol. 17, no. 4, pp. 443–455, 1991.

- [100] Y. Li, P. M. Pardalos, and M. G. Resende, “A Greedy Randomized Adaptive Search Procedure For The Quadratic Assignment Problem,” 1994.
- [101] Éric D. Taillard, “Comparison of Iterative Searches for the Quadratic Assignment Problem,” Location Science, vol. 3, no. 2, pp. 87–105, 1995.
- [102] R. Burkard and J. Offermann, “Entwurf von Schreibmaschinentastaturen Mittels Quadratischer Zuordnungsprobleme,” Zeitschrift für Operations Research, vol. 21, pp. B121–B132, 1977.
- [103] M. Mavrovouniotis and S. Yang, “Elitism-based Immigrants for Ant Colony Optimization in Dynamic Environments: Adapting the Replacement Rate,” in Proceedings of CEC, pp. 1752–1759, 2014.
- [104] K. Trojanowski and Z. Michalewicz, “Searching for Optima in Non-Stationary Environments,” in Proceedings of CEC, vol. 3, pp. 1843–1850, 1999.
- [105] C. Li, S. Yang, T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H. Beyer, and P. Suganthan, “Benchmark Generator for CEC 2009 Competition on Dynamic Optimization,” tech. rep., CEC 2009 competition on dynamic optimization, 2008.
- [106] W. Rand and R. Riolo, “Measurements for Understanding the Behavior of the Genetic Algorithm in Dynamic Environments: A Case Study Using the Shaky Ladder Hyperplane-Defined Functions,” in Proceedings of GECCO, pp. 32–38, 2005.
- [107] M. Mavrovouniotis and S. Yang, “Empirical Study on the Effect of Population Size on MAX-MIN Ant System in Dynamic Environments,” in Proceedings of CEC, pp. 853–860, 2016.
- [108] B. Calvo and G. Santafé Rodrigo, “SCMAMP: Statistical Comparison of Multiple Algorithms in Multiple Problems,” The R Journal, 2016.
- [109] B. Calvo, J. Ceberio, and J. A. Lozano, “Bayesian Inference for Algorithm Ranking Analysis,” in Proceedings of GECCO, pp. 324–325, 2018.

- [110] B. Calvo, O. M. Shir, J. Ceberio, C. Doerr, H. Wang, T. Bäck, and J. A. Lozano, “Bayesian Performance Analysis for Black-Box Optimization Benchmarking,” in Proceedings of GECCO, pp. 1789–1797, 2019.
- [111] X. Yu, T. Chen, and X. Yao, “Empirical Analysis of Evolutionary Algorithms with Immigrants Schemes for Dynamic Optimization,” Memetic Computing, vol. 1, pp. 3–24, 03 2009.
- [112] L. T. Bui, Z. Michalewicz, E. Parkinson, and M. B. Abello, “Adaptation in Dynamic Environments: A Case Study in Mission Planning,” IEEE TEVC, vol. 16, no. 2, pp. 190–209, 2012.
- [113] V. S. Nguyen, Q. D. Pham, and T. T. Huynh, “Modelling and Solving a Real-World Truck-Trailer Scheduling Problem in Container Transportation with Separate Moving Objects,” OPSEARCH, 2024.
- [114] R. Boucekkine, G. Fabbri, S. Federico, and F. Gozzi, “Managing Spatial Linkages and Geographic Heterogeneity in Dynamic Models with Transboundary Pollution,” Journal of Mathematical Economics, vol. 98, p. 102577, 2022.
- [115] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, “STL: A Seasonal-Trend Decomposition,” J. Off. Stat, vol. 6, no. 1, pp. 3–73, 1990.
- [116] K. Bandara, R. J. Hyndman, and C. Bergmeir, “MSTL: A Seasonal-Trend Decomposition Algorithm for Time Series with Multiple Seasonal Patterns,” 2021.
- [117] A. M. De Livera, R. J. Hyndman, and R. D. Snyder, “Forecasting Time Series with Complex Seasonal Patterns Using Exponential Smoothing,” Journal of the American statistical association, vol. 106, no. 496, pp. 1513–1527, 2011.
- [118] A. Dokumentov and R. J. Hyndman, “STR: Seasonal-Trend Decomposition Using Regression,” INFORMS Journal on Data Science, vol. 1, no. 1, pp. 50–62, 2022.
- [119] N. Patki, R. Wedge, and K. Veeramachaneni, “The Synthetic Data Vault,” in IEEE DSAA, pp. 399–410, 2016.

- [120] J. M. Hernández-Lobato, J. R. Lloyd, and D. Hernández-Lobato, “Gaussian Process Conditional Copulas with Applications to Financial Time Series,” in Proceedings of NIPS, p. 1736–1744, Curran Associates Inc., 2013.
- [121] J. Jordon, L. Szpruch, F. Houssiau, M. Bottarelli, G. Cherubin, C. Maple, S. N. Cohen, and A. Weller, “Synthetic Data – What, Why and How?,” 2022.
- [122] S. McLachlan, “Realism in Synthetic Data Generation,” Master’s thesis, Massey University of New Zeland, 2017.
- [123] S. Hao, W. Han, T. Jiang, Y. Li, H. Wu, C. Zhong, Z. Zhou, and H. Tang, “Synthetic Data in AI: Challenges, Applications, and Ethical Implications,” ArXiv, vol. abs/2401.01629, 2024.
- [124] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Nets,” in Advances in Neural Information Processing Systems, vol. 27, Curran Associates, 2014.
- [125] R. Po, W. Yifan, V. Golyanik, K. Aberman, J. T. Barron, A. H. Bermano, E. R. Chan, T. Dekel, A. Holynski, A. Kanazawa, C. K. Liu, L. Liu, B. Mildenhall, M. Nießner, B. Ommer, C. Theobalt, P. Wonka, and G. Wetzstein, “State of the Art on Diffusion Models for Visual Computing,” 2023.
- [126] M. Sklar, “Fonctions de Répartition an Dimensions et Leurs Marges,” University of Paris, 1959.
- [127] A. AghaKouchak, A. Bárdossy, and E. Habib, “Copula-based Uncertainty Modelling: Application to Multisensor Precipitation Estimates,” Hydrological Processes, vol. 24, no. 15, pp. 2111–2124, 2010.
- [128] J. Größer and O. Okhrin, “Copulae: An Overview and Recent Developments,” WIREs Computational Statistics, vol. 14, no. 3, p. e1557, 2022.
- [129] Salinas-Gutiérrez, Estimation of Distribution Algorithms based on Copula Functions. PhD thesis, Center for Research in Mathematics, 2011.

- [130] F. Bahrpeyma, M. Roantree, P. Cappellari, M. Scriney, and A. McCarren, “A Methodology for Validating Diversity in Synthetic Time Series Generation,” MethodsX, vol. 8, p. 101459, 2021.
- [131] DataCebo, Inc., Synthetic Data Metrics, 2024. Version 0.13.0.
- [132] N. V. Smirnov, “Estimate of Deviation between Empirical Distribution Functions in Two Independent Samples,” Bulletin Moscow University, vol. 2, no. 2, pp. 3–16, 1939.
- [133] J. Munkhammar and J. Widén, “An Autocorrelation-based Copula Model for Generating Realistic Clear-Sky Index Time-Series,” Solar Energy, vol. 158, pp. 9–19, 2017.
- [134] M. B. Wilk and R. Gnanadesikan, “Probability Plotting Methods for the Analysis of Data,” Biometrika, vol. 55, pp. 1–17, 03 1968.
- [135] D. Zhang, S. Cai, F. Ye, Y.-W. Si, and T. T. Nguyen, “A Hybrid Algorithm for a Vehicle Routing Problem with Realistic Constraints,” Information Sciences, vol. 394-395, pp. 167–182, 2017.
- [136] D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” 2022.
- [137] T. Stadler, B. Oprisanu, and C. Troncoso, “Synthetic Data – Anonymisation Groundhog Day,” 2022.
- [138] T. Cunningham, G. Cormode, and H. Ferhatosmanoglu, “Privacy-Preserving Synthetic Location Data in the Real World,” CoRR, vol. abs/2108.02089, 2021.
- [139] W. S. Cleveland and S. J. Devlin, “Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting,” Journal of the American statistical association, vol. 83, no. 403, pp. 596–610, 1988.