FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE
COIMBRA

Pedro Miguel Duque Rodrigues

Principled Modelling Of The Google Hash Code Problems For Meta-Heuristics

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems, advised by Professor Alexandre B. Jesus and Professor Carlos M. Fonseca, and presented to the Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2023

The work presented in this thesis was carried out in the Algorithms and Optimization Laboratory of the Adaptive Computation group of the Centre for Informatics and Systems of the University of Coimbra and financially supported by the Foundation for Science and Technology under a scholarship with reference UIDP/00326/2020.

Abstract

Acknowledgments

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"I've got a bad feeling about this..."

Han Solo, Star Wars

1.1 Motivation

Optimization problems are ubiquitous in real-world scenarios. Consider, for example, the task of planning a road trip, which involves various optimization challenges. These range from selecting the best route considering factors like fuel efficiency, travel time, and budget limitations, to efficiently packing luggage. Solving these problems entails finding good solutions, ideally in the most time-efficient manner possible.

Tackling an optimization problem typically begins by understanding the problem and its details. Then, a strategy is employed to solve the problem. This highlights the two main steps: first, describing the problem's details (model), and second, employ a strategy (solver) that uses those details to discover solutions.

In fact, this approach serves as the foundation for one widely employed method to address to address optimization problems, wherein a standard mathematical formulation (model) describing the problem is developed. Subsequently, one of the numerous available generic solvers, *e.g.* the simplex method, is utilized to find solutions. Notably, the simplex method, is an exact solver, which ensures the optimality of the obtained solutions.

Although this method for solving optimization problems is clearly established in the community there is recently growing interest in how to employ heuristic and meta-heuristic strategies to solve these problems, specifically, in the

Combinatorial Optimization (CO) field.

Heuristic are procedures, often problem-specific, that attempt to quickly solve a problem and provide a helpful "rule of thumb" for achieving reasonably good results, usually in a greedy fashion. Meta-heuristic methods combine several subordinate heuristics thus being more generic and can be applied to broad a range of problems. Natural processes and phenomena, such as collective behavior, natural selection, and some physical properties of materials, inspire several meta-heuristic search processes making them flexible and adaptable, although more computationally intensive than more conventional heuristics. Remarkably, both strategies are generally more time-efficient than exact approaches, though without an associated guarantee of optimality.

Given the nature of meta-heuristic methods, and the inherent diversity of problems, crafting universal meta-heuristic solvers is challenging. Unlike mathematical optimization, there's no established modelling framework for meta-heuristics that follows the same principled approach, possibly due to skepticism about its feasibility and effectiveness.

The establishment of this framework would standardize problem-solving approaches, facilitate the reuse of meta-heuristic methods, and distinctly separate the tasks of problem modeling and solver development. Moreover, it would provide researchers and practitioners with a useful tool to experimentally assess the performance of meta-heuristic methods across a range of diverse problems.

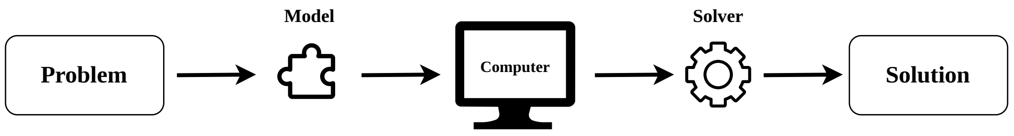

Figure 1.1: Modelling and Problem Solving

In summary, the concept of a modeling framework for problem tackling can be visually depicted as shown in Figure 1.1. For a given problem, a computational model that describes the problem is generated and provided to a machine, which then employs a meta-heuristic solver to obtain solutions. This approach is built upon prior work [1, 3] that has initiated the formalization of this objective.

Simultaneously, alongside the development and application of meta-heuristic strategies to address combinatorial optimization problems, there exists a community interest in constructing a collection of benchmark optimization problems that hold both theoretical and practical significance. The Google Hash Code Competition problems, arguably, present themselves as suitable candidates.

The Hash Code programming competition, formerly hosted annually by Google, challenged teams of up to four members to solve intricate CO problems within a four-hour time frame. Participants were allowed to use any tools, resources, and programming languages of their choice. These problems often drew inspiration from real-world issues and engineering challenges, such as vehicle routing, task scheduling, and Wi-Fi router placement. Essentially, can be classified as “open” research problems.

Given the pertinence of these problems, they serve as apt benchmarks for the evaluation of meta-heuristics. Moreover, they offer a suitable approach to assess the feasibility of the aforementioned principled modeling approach. Not only do these problems present challenges from theoretical and practical standpoints, but they also have well written and detailed statements. The competition context in which they were presented also provides a wealth of empirical data regarding the best solutions attained thus far.

1.2 Goals & Scope

This thesis focuses on creating, implementing, and evaluating meta-heuristic approaches using a principled modelling framework. The Google HashCode Problems serve as benchmarks for our study.

Our approach encompasses two primary aspects. First, from a modeling perspective, we aim to expand upon the modelling concepts that have been partially explored in previous research [1, 2, 3]. Our primary objective is to solidify existing concepts while introducing additional functionality, both in conceptual understanding and practical application.

Additionally, we aim to construct models for the Hash Code problems. These models will not only be analyzed in this thesis but will also serve as illustrative examples for explaining and teaching the modeling concepts. Furthermore, they will enable a critical evaluation of the merits and shortcomings of this structured approach in comparison to more ad-hoc and traditional methods of problem-solving.

Secondly, the implementation of state-of-the-art meta-heuristic solvers. This development enables a thorough examination of the models and the overarching strategy. This assessment will encompass an analysis of both performance metrics and the quality of solutions achieved through this approach.

As such, the main research questions we outline for this thesis are:

R1. Can we formalize the existing ideas explored by previous work on the modelling framework and produce a practical implementation, potentially contributing with new features?

R2. Can we develop general-purpose meta-heuristic solvers and utilities with respect to the principled modelling framework implementation?

R3. Can Google Hash Code problems be solved using a modelling approach?

1.3 Contributions

The main contributions of this thesis related to the aforementioned research questions, are as follows:

C1. With the existing research on principled modeling for meta-heuristics [1, 2, 3], this document aims to consolidate and formalize a comprehensive specification. Our objective is to encapsulate all the concepts and developments made thus far. Additionally, we have created a practical Python implementation of this framework. In essence, both in the formalization and implementation, we endeavor to synthesize the existing ideas concerning modeling for constructive and local search techniques. These techniques are integral components of meta-heuristic solvers and from our perspective, they should be integrated into a model, as we will elaborate upon in this thesis.

C2. We developed several meta-heuristic solvers and utilities both for gathering the solutions and for testing the developed models. Given that these are general-purpose they can work with any model that is developed under the practical implementation of the framework we devised.

C3. After a careful selection process we selected two Google Hash Code problems for which some models for each of the problems were developed that explore the different properties of the problems in an attempt to both obtain the best solutions possible. Those models not only provide a practical example on how to model a problem but also explore the framework capabilities in great detail.

1.4 Software

The following software resulted from the development of this thesis and has been distributed under an open source license.

S1. Python Framework (to be continued...)

S2. Models and Experiments Code (to be continued...)

1.5 Outline

The remainder of thesis is structured as follows. In Chapter 2, we provide an overview of optimization concepts, meta-heuristics, and modeling in the context of meta-heuristics. Moving to Chapter 3, we analyze the Google Hash Code competition, focusing on problem characteristics and their relation to existing CO literature. In Chapter 4, we discuss a modeling framework and its role in meta-heuristic development. Chapters 5 and 6 present detailed studies of the Hash Code problems "Optimize Data Center" and "Book Scanning", with experimental results. Finally, Chapter 7 summarizes findings and suggests future research directions.

Chapter 2

Background

Chapter 3

Google Hash Code Competition

"I've got a bad feeling about this..."
Han Solo, Star Wars

This chapter presents an overview of the Google Hash Code competition. In Section 3.1, we provide a concise review of the competition, encompassing both its historical background and format. Subsequently, in Section 3.2, we delve into the problems presented to participants over the years, attempting to categorize them and establish connections with well-known combinatorial optimization problems described in the literature. Moving forward, Section 3.3 sheds light on the design of competition instances. Concluding this chapter, Section 3.4 offers remarks that highlight key aspects of the competition problems, deemed pertinent to this work.

3.1 Format

3.2 Problems

3.2.1 Hash Code 2014

Street View Routing

In the context of constructing street view maps there is a need to collect imagery that is taken by specialized vehicles equipped for that purpose. This constitutes a challenging problem since given a fleet of cars which may only be available for a limited amount of time a route for each must be defined as to maximize the number of streets photographed. City streets are modelled as a graph where nodes are junctions and the edges are streets connecting said junctions.

Moreover, streets are defined by three distinct properties: direction, length and cost that will take for the car to traverse the street.

The challenge consists of scheduling the routes for street view cars in the city, adhering to a pre-determined time budget. The goal is to optimize the solution by maximizing the sum of the lengths of the traversed streets, while minimizing the overall time expended in the process. The quality of the solution for this problem is evaluated by using the sum of the lengths of the streets as the primary criterion and the time spent as a tie-breaker.

The problem at hand bears a strong resemblance to a combination of the Vehicle Routing Problem and the Maximum Covering Problem. This is because the scheduling of routes for the fleet of cars must be done in a way that ensures that the combination of all sets of streets visited by each car encompasses the entire city, in the most time-efficient manner possible.

3.2.2 Hash Code 2015

Optimize a Data Center

The optimization of server placement problem is a concern that pertains to the design of data centers, as various factors must be taken into account to ensure optimal efficiency. In this context, the ‘optimizing servers” problem portrays a scenario in which contestants are in the position of designing a data center and seeking to determine the optimal distribution of servers. The data center is physically organized in rows of slots where servers can be placed. Hence, the challenge is to efficiently fill the available slots in a Google data center with servers of varying sizes and computing capacities, while also ensuring that each server is assigned to a specific resource pool.

Objectively, the goal is to assign multiple servers to available slots and resource pools in such a way as to maximize the guaranteed capacity for all resource pools. This metric serves as the criterion for evaluating solutions to this problem. The guaranteed capacity, in this context, refers to the lowest amount of computing power that will remain for a specific resource pool in the event of a failure of an arbitrary row of the data center. It is important to note that this objective function is considered a bottleneck, as small changes in a solution may not result in significant changes in the score, making the optimization process more difficult.

Notably, the problem of optimizing the placement of servers in a data center can be thought of as a combination of a Multiple-Knapsack Problem and an assignment problem. This is because the servers must be placed within the constraints of the available space in the data center rows, and subsequently, they must be assigned to resource pools.

Loon

Project *Loon*, which was a research endeavor undertaken by Google, aimed at expanding internet coverage globally by utilizing high altitude balloons. The problem presented in this competition drew inspiration from this concept, requiring contestants to devise plans for position adjustments for a set of balloons, taking into consideration various environmental factors, particularly wind patterns, with the objective of ensuring optimal internet coverage in a designated region over a specific time frame.

The objective of this problem was to develop a sequence of actions, including ascent, descent, and maintaining altitude, for a set of balloons with the goal of maximizing a score. In this case, the score is calculated based on the aggregate coverage time of each location, represented as cells on a map of specified dimensions, at the conclusion of the available time budget.

In summary, this problem can be classified as both a simulation and a coverage and routing problem, based on the properties previously described. It is important to note that the simulation aspect of this problem has a direct impact on the calculation of the score, and is not solely limited to constraints on the available time budget for operations. Furthermore, this problem can be represented in a forest, where the vertices represent spatiotemporal coordinates (x, y, z, t), and the edges symbolize changes in altitude and lateral movement (wind) for a given balloon.

3.2.3 Hash Code 2016

Delivery

In today's world, with the widespread availability of internet, online shopping has become a prevalent activity. As a consequence, there is an ever-growing need for efficient delivery systems. This competition challenges participants to manage a fleet of drones, which are to be used as vehicles for the distribution of purchased goods. Given a map with delivery locations, a set of drones, each with a set of operations that can be performed (load, deliver, unload, wait), a number of warehouses, and a number of orders, the objective is to satisfy the orders in the shortest possible time, taking into consideration that the products to be delivered in an order may have product items stored in multiple different warehouses and therefore require separate pickups by drones.

In this problem, the simulation time $\mathcal{T}$ is given and the goal is to complete each order within that time frame. The score for each order is calculated as $\frac{(\mathcal{T}-t)}{\mathcal{T}} \times 100$, where t is the time at which the order is completed. The score ranges from 1 to 100, with higher scores indicating that the order was completed sooner. The overall score for the problem is the sum of the individual scores for all orders, and it is to be maximized.

In summary, this problem can be classified as a variant of the Vehicle Routing Problem, specifically as a Capacitated, Pickup and Delivery Time Windowed Multi-Depot Vehicle Routing Problem. This classification takes into account the pickup and delivery of items, the time window for delivery, the multiple routes and warehouses that each vehicle may need to visit in order to fulfill the orders.

Satellites

Terra Bella was a Google division responsible for managing and operating a constellation of satellites that collected and processed imagery for commercial purposes. Specifically, these satellites were tasked with capturing images in response to client requests.

The challenge presented to participants involves crafting schedules for individual satellites within the fleet. The goal is to secure image collections that match customer preferences. These collections are characterized by geographical coordinates on Earth and specific time windows for image capture. Each satellite, originating from unique latitude and longitude coordinates and possessing a certain velocity, possesses the ability to make minor positional adjustments along both axes to access potential photography sites. The problem's score is determined by aggregating the points earned through the successful completion of customer collections. In this context, completion signifies capturing all images for a given collection within the designated time frame.

In essence, this problem falls into the categories of both an assignment and a maximum covering problem. It involves not only covering the maximum number of images with the available satellites to complete collections, but also making decisions about which satellites will capture each photo. Additionally, the simulation aspect is crucial as it directly affects scoring; images not taken within the specified time frame won't contribute to the collection, potentially influencing its completion and the overall score.

3.2.4 Hash Code 2017

Streaming Videos

In the era of online streaming services like YouTube, effectively distributing content to users is crucial. This challenge focuses on optimizing video distribution across cache servers to minimize transmission delays and waiting times for users. Contestants must strategise video placement within servers while considering space limitations.

With a roster of videos, each assigned a specific size, an collection of cache servers with designated space, and an index of endpoints initiating multiple requests for various videos, this challenge entails determining an optimal video assignment within servers. The time saved for each request is measured as the difference between data center streaming time and cache server streaming time with minimal latency. The overall score is computed by summing the time saved for each request, multiplied by 1000, and then divided by the total request count. It's important to note that the problem description offers transmission latencies between different nodes.

In general, we categorize this problem as a combination of assignment and knapsack problems. Contestants are tasked not only with determining the allocation of videos to servers but also with accounting for capacity limitations on the number of videos per server. It's worth noting that the calculation of time saved for each request may encounter a bottleneck effect, which can pose challenges when optimizing the overall score.

Router Placement

Strategically optimizing the placement of Wi-Fi routers to achieve optimal signal coverage is a challenge encountered by many institutions and individual users. This issue becomes particularly prominent in larger and complex buildings. Furthermore, in such scenarios, the task may involve setting up a wired connection to establish internet connectivity from the source point, facilitating the strategic positioning of routers for maximum coverage.

The challenge tasked participants with optimizing the arrangement of routers and fiber wiring within a building's cell-based layout, along with a designated backbone connection point. The aim was to strategically position routers and devise an effective wiring configuration. The primary goal encompassed achieving optimal coverage while adhering to a predefined budget. The problem's score comprised two components: the count of cells covered by routers, multiplied by 1000, and the remaining budget. Notably, the scoring approach emphasized both extensive coverage and economical budget allocation.

In essence, this problem falls under the category of a maximum covering problem, as the central aim is to ensure the coverage of as many cells as possible. Furthermore, considering the budget limitations and wiring arrangement, we observe that this challenge shares similarities with the Steiner Tree Problem. This likeness arises from the possibility of determining the optimal cost of wiring placement based on the router locations, which may hold significance for the problem's resolution.

3.2.5 Hash Code 2018

Self-Driving Rides

Daily car commuting is a ubiquitous practice globally, involving trips to homes, schools, workplaces, and more. As a means of travel, cars remain a common choice, with ongoing efforts to enhance safety through the advancement of self-driving technology. In this challenge, contestants assume the role of managing a fleet of self-driving cars within a simulated setting. The goal is to ensure commuters reach their destinations securely and punctually.

With a fleet of cars at disposal and a roster of rides defined by their starting and ending intersections on a square grid representing the city, along with the earliest start time and the latest end time to ensure punctuality, the task is to allocate rides to vehicles. The aim is to maximize the number of completed rides before a predefined simulation time limit is reached. The scoring is determined by the summation of the individual ride scores. A ride's score is computed as the sum of a value proportional to the distance covered during the ride, augmented by a bonus if the ride commences precisely at its earliest allowed start time.

Generally, this problem can be categorized as an assignment and vehicle routing problem with time windows. This classification arises from the necessity to assign rides to cars within specific time constraints. Notably, the car's route is determined by the sequence of rides assigned to it. Moreover, this challenge falls under the simulation category, as it directly impacts the scoring mechanism and cannot be simplified or abstracted.

City Plan

With the world's population increasingly concentrating in urban areas, the demand for expanded city infrastructure is on the rise. This entails not only residential buildings but also the incorporation of essential public facilities and services to cater to the growing populace. This challenge mirrors a scenario where participants are tasked with planning a city's building layout, involving both the selection of building types and their strategic placement.

For this challenge, participants receive building projects with specific width and height dimensions, covering both residential and utility structures. The city is a square grid of cells. Overall, the goal is to create buildings from these plans, arranging them within the city to optimize space and create a balanced mix of structures. This minimizes residents' walking distance to reach essential services. The overall score is the sum individual residential building scores, calculated by multiplying the number of residents and the number of utility building types within walking distance of that building. Notably, the walking distance parameter is specific to each problem instance.

Essentially, this problem belongs to the category of packing problems. The core objective revolves around determining how to fit buildings within the city layout. Importantly, there is no predetermined limit on the number of buildings that can be constructed for each plan, granting contestants the flexibility to make choices accordingly.

3.2.6 Hash Code 2019

Photo Slideshow

Given the surge in digital photography and the vast number of images traversing the internet daily, this challenge delves into the interesting concept of crafting picture slideshows using the available photo pool.

In this scenario, participants were tasked with creating a slideshow composed of pictures, which could be oriented either vertically or horizontally in the slides. Notably, a slide could contain two photos if they were arranged vertically. Additionally, these photos could be tagged with multiple descriptors corresponding to their subjects. The scoring of this problem revolves around the slideshow's appeal, determined by a calculated value that depends on consecutive slide pairs. This value is computed as the minimum between the tags count of the first picture, the second picture in the sequence and the count of the common tags shared between the two images.

Overall, this challenge can be categorized as a scheduling problem, to be precise, a single-machine job scheduling problem. If we liken the jobs to photos, the goal is to sequence them to optimize a specific objective function in this context, the "appeal" factor. Additionally, the interactions between slides introduce elements resembling a grouping problem.

Compiling Google

Given Google's extensive codebase spanning billions of lines of code across numerous source code files, compiling these files on a single machine would be time-consuming. To address this, Google distributes the compilation process across multiple servers.

This challenge tasks participants with optimizing compilation time by strategically distributing source code files across available servers. Notably, the compilation of a single code file can depend on other files being compiled prior to it, involving dependencies. Given a certain number of available servers and specific deadlines for compilation targets, the problem's score is calculated by summing the scores for the completion of each compilation target. These scores are determined by a fixed value for meeting the deadline, with an additional bonus if the compilation is completed ahead of the expected time.

This problem can be categorized as a scheduling problem, as the primary objective involves distributing compilation tasks (jobs) among different machines while adhering to dependencies between files. In essence, this problem resembles a variation of the classical job-shop scheduling problem.

3.2.7 Hash Code 2020

Book Scanning

Google Books is project that aims to create a digital collection of many books by scanning them from libraries and publishers around the world. In this challenge, contestants are put in the position of managing the operation of setting up a scanning pipeline for millions of books.

Given a dataset describing libraries and available books, the objective of this challenge is to select books for scanning from each library within a specified global deadline. Each library has a distinct sign-up process duration before it can commence scanning, and only one library can be signed up at a time. Moreover, each library has a fixed scanning rate for books per day, and each scanned book contributes to the final score. The problem's goal is to maximize the overall score, which is calculated as the sum of the scores for unique books scanned within the given deadline.

This problem exhibits a combination of characteristics from classical scheduling, assignment, covering, and knapsack problems. It resembles a scheduling problem as the order in which libraries are signed up needs to be determined. It involves assignment, since libraries can share books, necessitating a decision on which libraries will scan each book. The covering aspect is apparent in the scoring mechanism, where the aim is to maximize the number of unique books scanned. Lastly, the problem also incorporates a knapsack-like element. While the time-related simulation factor exists, it can be abstracted into a knapsack scenario where the goal is to optimize the overall score by considering the number of books a library can scan until the deadline as its capacity.

Assembling Smartphones

Constructing smartphones is a intricate process that entails assembling a multitude of hardware components. This challenge delves into the concept of creating an automated assembly line for smartphones, employing robotic arms to streamline the manufacturing process.

Contestants are tasked with placing robotic arms within a workspace depicted as a rectangular cell grid. The objective is to optimize the arrangement of these arms to allow the execution of assigned tasks. Each task involves specific movements that a robotic arm must perform, essentially traversing a designated number of cells to accomplish the task. Notably, robotic arms cannot cross each other, necessitating precise task assignment and arm positioning to ensure unobstructed task execution for all arms. The problem's score is derived from the summation of scores obtained by successfully completing tasks within the constraints.

In summary, this challenge falls under the category of both assignment and scheduling problems since it encompasses the assignment of robotic arms to suitable positions and the scheduling of tasks across these arms to optimize the completion of tasks.

3.2.8 Hash Code 2021

Traffic Signaling

This challenge delves into the optimization of traffic light timers to enhance the travel experience in a city. While traffic lights inherently contribute to road safety, their built-in timers are important in regulating traffic flow. The focus here is to fine-tune these timers with the aim of optimizing overall travel time for all commuters within the city.

Contestants are presented with a city layout, complete with intersections housing traffic lights. The task is to strategically allocate time intervals to these traffic light timers, optimizing traffic flow to ensure the maximum number of car trips are successfully completed within a predefined simulation time limit. The problem's score is the cumulative sum of scores assigned to each completed trip. These scores comprise a fixed value for trip completion and a bonus proportional to how early the trip concludes relative to the simulation's time limit. While the challenge may seem complex due to its detailed rules and operational aspects, its core objective revolves around this fundamental optimization process.

In summary, this challenge can be categorized as a simulation problem. It's worth highlighting that this problem aligns closely with the Signal Timing problem in the literature of Control Optimization.

Software Engineering at Scale

This challenge addresses the complexity of managing Google's vast monolithic codebase, which has grown significantly alongside the expanding number of engineers. To overcome the hurdles of effective feature deployment, participants are tasked with creating a solution that optimally schedules feature implementation work among engineers.

In this challenge, there are three primary components to be considered: features, services, and binaries. Each feature may require certain services, which can be present in specific binaries. The main objective is to efficiently assign features to engineers, considering that their implementation might entail additional tasks such as service implementation, binary relocation, new binary creation, or waiting for a designated time. The challenge revolves around optimizing this workflow to minimize delays caused by multiple engineers working in the same service. The scoring is based on the sum of scores awarded for feature completion. Each completed feature's score is determined by the product of the number of users benefiting from it, as specified in the problem statement, and the number of days between the maximum day (also defined) and the day the feature was launched.

In essence, this challenge falls within the realm of classic scheduling problems. It involves assigning tasks (jobs) to engineers with the aim of optimizing a quantity influenced by the order in which each engineer performs their tasks and the interactions of tasks among multiple engineers.

3.2.9 Hash Code 2022

Mentorship and Teamwork

Santa Tracker

3.2.10 Outline

In summary, this section provided an overview and description of the key aspects of the Hash Code problems. Furthermore, a categorization that links these problems to topics commonly found in combinatorial optimization literature was presented. The Table 3.1 shows a summary of the analysis conducted.

Problem	**Categories**						
	Assignment	Knapsack	Coverage	Vehicle Routing	Simulation	Scheduling	Packing
Street View Routing			✓	✓			
Optimize a Data Center	✓	✓					
Loon			✓	✓	✓		
Delivery				✓			
Satellites	✓		✓		✓		
Streaming Videos	✓	✓					
Router Placement			✓				
Self-Driving Rides	✓			✓	✓		
City Plan							✓
Photo Slideshow						✓	
Compiling Google						✓	
Book Scanning	✓	✓	✓			✓	
Assembling Smartphones	✓					✓	
Traffic Signaling					✓		
Software Engineering at Scale						✓	
Mentorship and Teamwork						✓	
Santa Tracker				✓			

Table 3.1: Categorization of Google Hash Code Problems

3.3 Instance Design

3.4 Concluding Remarks

Chapter 4

Principled Modelling Framework

Chapter 5

Optimize a Data Center Problem

Chapter 6

Book Scanning Problem

Chapter 7

Conclusion

Acronyms

AC	Adaptive Computation. ii
ALGO	Algorithms and Optimization Laboratory. ii
CISUC	Centre for Informatics and Systems of the University of Coimbra. ii
CO	Combinatorial Optimization. 2, 3, 5
FCT	Foundation for Science and Technology. ii

Bibliography

[1] Ana Vieira. “Uma plataforma para a avaliação experimental de meta-heurísticas”. Doctoral Thesis. University of Algarve, Portugal, 2009. URL: https://sapientia.ualg.pt/handle/10400.1/518 (visited on 01/10/2023).

[2] Carlos M. Fonseca. *Nasf4nio*. Oct. 27, 2021. URL: https://github.com/cmfonseca/nasf4nio (visited on 01/15/2023).

[3] Samuel Barroca do Outeiro. “An Application Programming Interface for Constructive Search”. Msc Thesis. University of Coimbra, Portugal, Nov. 9, 2021. URL: https://estudogeral.sib.uc.pt/handle/10316/98261 (visited on 01/10/2023).