FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Pedro Miguel Duque Rodrigues

Principled Modeling 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 OptimizationLaboratory 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

List of Algorithms

Chapter 1

Introduction

"Begin at the beginning", the King said gravely, "and go on till you come to the end; then stop."

— Lewis Carroll

1.1 Motivation

Optimization problems are ubiquitous in real-world scenarios. When considering, for example, the task of planning a road trip, we are quickly faced with several optimization challenges arising from a seemingly simple task. For example, finding the best route, considering factors as fuel efficiency, travel time, and budget limitations or to efficiently packing luggage. Solving such optimization problems entails finding "good" solutions in the most time-efficient manner.

Tackling an optimization problem can typically be seen as a two-phase process. First, we start by understanding the problem and modeling its details. Then, we can apply, or develop, a solver to find one or more solutions taking into account the given model. This approach serves, for example, as the foundation of most linear optimization software packages which are widely used to solve real-world problems. In particular, such packages expect a mathematical formulation (model) describing the problem as a linear objective function and a set of linear constraints, and then use one or more algorithms designed to solve such linear optimization problems. This clear separation of concerns leads to some advantages. Notably, practitioners that want to solve a particular problem can focus on developing the model for that problem and easily use existing solvers to find solutions without needing to implement state-of-the-art algorithms themselves. Meanwhile, solver developers can take advantage of

existing problems to test and enhance their solvers's performance.

In this work, we are interested in tackling Combinatorial Optimization (CO) problems using a similar separation of concerns. Generally speaking, CO consists of finding an optimal solution, according to some objective function, from a discrete set of solutions. Several generic approaches have been developed to solve CO problems exactly, *i.e.*, to find an optimal solution. However, many CO problems are NP-Hard, meaning that, the time required to solve them via (problem-specific) exact approaches grows exponentially with the problem size, and consequently generic exact methods will also grow exponentially. In practice, this means that exact methods are often ineffective to solve "real-world" CO problems which have large problem sizes.

As a result, there has been a growing interest in the development of methods that can find "good" solutions for such problems. In this work, we focus on heuristic and meta-heuristic methods. Heuristic methods are search procedures, often problem-specific, that attempt to quickly solve a problem and provide a "rule of thumb" for attaining decent solutions, albeit without optimality guarantees. Meta-Heuristic (MH) methods employ several high-level strategies to construct and improve solutions. It is worth noting that such high-level strategies often depend on problem-specific details, *e.g.*, the neighborhood structure and search tree definition. However, meta-heuristics do not require knowledge about these problem-specific details and instead use the high-level strategies in a black-box fashion. As such, MH approaches are problem-independent and can be applied to a broad range of problems.

Given the nature of MH methods and the inherent diversity of problems, crafting universal MH solvers is a challenging task made harder due to the difficulty in separating the problem-specific details required by the high-level strategies from the MH problem-independent solving process. In fact, the abundance of MH optimization software that provides specific frameworks for implementing evolutionary, local or constructive search meta-heuristics for CO problems [4, 3, 8] and the lack of a unifying framework supporting all approaches can be regarded as a symptom of the difficulty of this endeavor. Still, it is worth remarking the works by Vieira [6] and Outeiro [13], which partially looked at the formalization of this objective.

The development of a unifying framework would standardize problem-solving approaches, facilitate the reuse of MH methods, and distinctly separate the tasks of problem modelling and solver development. Moreover, it would provide researchers and practitioners with a valuable tool to experimentally assess the performance of MH methods across a range of diverse problems.

Simultaneously, alongside the development and application of MH strategies to address CO problems, there exists a community interest in constructing a collection of benchmark optimization problems that hold both theoretical and practical significance [11]. 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 using any tools, (online) 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 and can be classified as "open" research problems.

Given the pertinence of these problems, and the wide range of challenges they present from both a theoretical and practical standpoint, they serve as apt benchmarks for the evaluation of meta-heuristics. Furthermore, they offer a suitable approach to assess the feasibility of the aforementioned unifying framework on more realistic and challenging problems beyond the ones commonly found in the literature.

1.2 Goals & Scope

The main goal of this work is the implementation and evaluation of meta-heuristic solution approaches for the Google Hash Code problems, using a principled approach that separates the modeling of the problems from the solvers.

In particular, we aim to expand upon the modelling approach for meta-heuristics that has been partially explored in previous research [6, 12, 13]. The objective is to solidify existing concepts while introducing additional functionality, both in conceptual understanding and practical application.

Furthermore, we aim to construct models for the Google Hash Code problems. These models will not only be described and discussed in this thesis but will also serve as illustrative examples documenting the modelling concepts. Furthermore, they will enable a critical evaluation of the merits and shortcomings of this principled approach in comparison to more ad-hoc and traditional methods of problem-solving.

Finally, the implementation of state-of-the-art meta-heuristic solvers is a vital component of our work as it will enable us to assess the performance and quality of solutions found for the models of the Google Hash Code problems

as well as the feasibility of the modelling approach for meta-heuristic solver development.

In summary, 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 [6, 13] and produce a practical implementation, potentially contributing with new features?

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

R3. Can Google Hash Code problems be solved effectively using this 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 modelling for meta-heuristics [6, 12, 13], 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 endeavour to synthesize the existing ideas concerning modelling for constructive and local search techniques.

C2. We implemented 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. 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. These models provide a practical example on how to model relatively complex problems and also allows us to think critically about the framework capabilities.

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 (TODO)

S2. Models and Experiments Code (TODO)

1.5 Outline

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

Chapter 2

Background

> *"If I have seen further than others, it is by standing upon the shoulders of giants."*
>
> — Isaac Newton

This chapter presents a comprehensive literature review of optimization, metaheuristics and modelling. Additionally, it provides a background review of the state-of-the-art regarding the principled modelling approach. In particular, Section 2.1 describes fundamental CO concepts deemed relevant for better understanding this work. **??** discusses multiple well-known techniques for solving CO problems. **??** describes MH methods and presents an extensive review of MH solvers. Finally, **??** delves into the details of the modelling approach and describes the existing implementations.

2.1 Optimization Concepts

Optimization, as defined by Papadimitriou and Steiglitz [2], is the task concerning the search for the optimal configuration or set of parameters that maximizes or minimizes a given objective function. In other words, optimizing involves finding the "best" solution to a given problem among a set of feasible solutions. Formally, an optimization problem can be defined as follows:

Definition 2.1.1 (Optimization Problem [2]). *An optimization problem is a tuple $(\mathcal{S}, f)$, where $\mathcal{S}$ is a set containing all feasible solutions, and f is an objective (cost) function, with a mapping such that:*

$$f : \mathcal{S} \longrightarrow \mathbb{R} \tag{2.1}$$

That is, each solution $s \in \mathcal{S}$, is assigned a real value representing its quality, with the highest quality solution $s^* \in \mathcal{S}$ being referred to as the (globally) optimal solution.

Definition 2.1.2 (Globally Optimal Solution [1, 2]). *Assuming, without loss of generality an optimization problem with a maximizing objective function a globally optimal solution $s^* \in \mathcal{S}$ is expressed by:*

$$\forall s \in \mathcal{S} : f(s^*) \geq f(s) \tag{2.2}$$

Since Google Hash Code problems [15] have a single-objective maximizing objective function, we will only consider maximization in this work. However, it is possible to reformulate problems with a minimizing objective function for maximization [5] using the identity:

$$\max -f(s) = \min f(s) \tag{2.3}$$

2.1.1 Combinatorial Optimization

Optimization problems can be divided into *discrete* and *continuous* based on the domain of their variables. In problems with *discrete* variables, solutions are defined on a finite, possibly countably infinite, set of values [2]. As for problems with *continuous* variables, the solutions take on any value on a continuous (infinite) subset of real numbers. Nevertheless, there are problems that involve both *discrete* and *continuous* variables, commonly denominated as mixed [5].

Combinatorial Optimization (CO) problems are a subset of optimization problems characterized by a discrete solution space that typically involves different permutations, groupings, or orderings of objects that satisfy some problem-specific criteria [2, 6]. As such, solutions for these problems are represented objects related to the combinatorics *e.g.* integers, permutations sets and graphs [16, 7]. Thus, regarding the previous definition of an optimization problem, a CO can be formally defined as follows:

Definition 2.1.3 (Combinatorial Optimization Problem [2]). *A combinatorial optimization problem is an optimization problem (2.1.1) where the set $\mathcal{S}$ of feasible solutions is finite or countably infinite.*

Typical examples of CO problems include network flow, matching, scheduling, shortest path and decision problems. Notably, the Knapsack Problem (KP) is a well-known [14, 7, 10] example of a CO problem where the goal is to find the subset of items with the highest total profit that can fit in a knapsack without exceeding its maximum capacity.

Due to the inherent discreteness of decision spaces within CO optimization problems, solutions can be understood as compositions of objects (components) selected from a finite set that including all elements capable of contributing to a solution. This set, commonly known as the "ground set", can be defined as shown:

Definition 2.1.4 (Ground Set [13, 10, 9]). *The ground set $\mathcal{G}$ of a CO problem is a finite set of containing all possible components for the problem.*

$$\mathcal{G} : \{c_1, c_2, c_3, \ldots, c_i\} \tag{2.4}$$

Hence, within the context of CO problems, a feasible solution constitutes a subset of the ground set, denoted as $s \in \mathcal{S} \subseteq 2^{\mathcal{G}}$, where the included components satisfy problem-specific constraints. Moreover, a partial solution is defined by its former capacity to incorporate additional components from the ground set. In contrast, a complete solution is incapable of accepting further components without violating feasibility, as opposed to the former which can be infeasible.

To illustrate these concepts, let's consider the practical example of the KP. In this context, the ground set is the set of all the available items (components). As such, a feasible solution is one in which items placed within the knapsack do not exceed its capacity limit. A partial solution is one where the knapsack is not yet full and additional items can still be accommodated. Notably, in this case, the partial solution is still feasible. Finally, a complete solution is a feasible solution where no further items can be added due to capacity constraints.

In essence, since CO problems involve choosing a combination of objects any algorithm that is able to enumerate the entirety of the solution space can be used to solve these problems. However, finding an optimal solution can be difficult, and exhaustive search strategies may not be able to solve many of these problems, which are often NP-Hard [7, 10] and thus not approachable by algorithms in a reasonable amount of time. In these cases, approximation, heuristic and MH methods present themselves as effective alternatives to be considered.

2.1.2 Global and Local Optimization

Chapter 3

Google Hash Code Competition

"understanding a question is half an answer"

— Socrates

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, **??** 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 History & Format

The Google Hash Code competition, organized by Google from 2014 to 2022, consisted of two main phases: a qualifying round and a final round. During this competition, teams of 2-4 skilled individuals were tasked with solving complex problems that mirrored real-world engineering challenges faced by Google's own engineers. The primary aim of the competition was to attract talented individuals to the company. In the qualifying round, participants worldwide engaged in a 4-hour problem-solving session. Subsequently, around 40-50 select teams advanced to the final round, which took place at a Google headquarters. Additionally, participants gathered at designated hubs globally during the qualifying round, fostering a competitive environment.

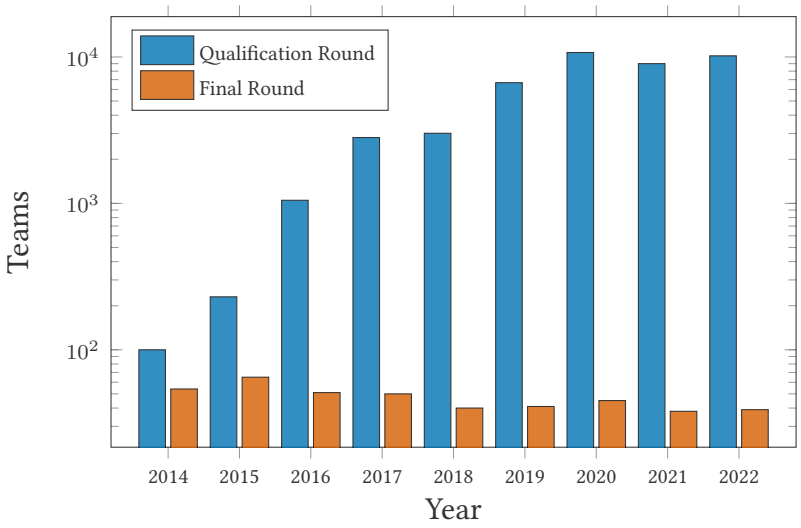

Figure 3.1: Google Hash Code Competition Attendance 2014-2022

In the early years of the competition, it was only open to teams from France. In the subsequent three years, it was open to teams from Europe, Africa, and the Middle East before becoming a worldwide competition. Therefore, it is expected that there will be an increase in the number of results available in the later years and more challenging problems due to the increase in competition. Furthermore, the number of participants kept growing throughout the years which highlights the importance of this event as illustrated in Figure 3.1

Unfortunately, this year Google decided to cease all its coding competitions including Hash Code. Nevertheless, the competition generated a diverse collection of attempts at solving the problems, resulting in a valuable wealth empirical data accessible to the community. As the official coding competition website is no longer accessible, Google created a repository containing all problem statements and instances distributed under an open source license [15]. However, it's worth noting that the scores achieved by participants are not integrated into this repository. Instead, they are documented across various blog posts and third-party websites [17].

It is worth noting, that due to the nature of the problems and format of the competition, participants frequently made use of heuristic and meta-heuristic strategies to solve the problems as best as possible in the alloted time. Moreover, the majority of competition problems are structurally different from one other, which makes it demanding to write general-purpose heuristic solvers that can be easily reused. Hence, it's a common practice for competitors to use solvers that are easily implementable or readily available online, given that internet access is permitted.

3.2 Problems

Its main objective is to provide the reader with a comprehensive understanding of the relevant details of the problems that drive connections with well-studied topics in combinatorial optimization literature instead of and exhaustively review the problem details, as the problem statements consulted in the coding [15]

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 optimise 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

Optimise a Data Center

The optimisation 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 ‘optimising 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 optimisation process more difficult.

Notably, the problem of optimisation 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 co-

ordinates 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 optimising 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 optimising the overall score.

Router Placement

Strategically optimising 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 optimising the arrangement of routers and fibber 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 optimise 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 optimise 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 optimising 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 optimise 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 optimise 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 optimise the completion of tasks.

3.2.8 Hash Code 2021

Traffic Signaling

This challenge delves into the optimisation 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 optimising 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, optimising 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 optimisation 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 Optimisation.

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 optimising 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 optimising 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

This challenge delves into the concept of a teamwork environment, where knowledge sharing among peers and collaborative efforts are central to task completion. In this challenge, participants are tasked with orchestrating a team comprising individuals with diverse backgrounds to successfully execute projects that demand a variety of skills.

The main goal is to efficiently assign a list of contributors, each possessing specific skills and the potential to improve them through project involvement, mentoring, or being mentored. The challenge involves allocating contributors to projects with skill requirements to ensure timely completion. Notably, contributors can participate in multiple projects concurrently. The key factor here is the order in which contributors develop or enhance their skills, a decision that significantly impacts the overall project completion process. The score in this challenge is the sum of project scores achieved by completing them before the defined overall deadline. A project's score comprises a fixed value for completion, minus penalty points if it surpasses the deadline but is still within a tolerance window. Projects exceeding the deadline or tolerance won't add to the score but will still contribute to workers' training.

In summary, this challenge shares similarities with a scheduling problem, as it involves assigning projects to contributors while considering the order in which they are completed to maximize the overall score achieved through project completion.

Santa Tracker

The *Google Santa Tracker* is a project that visualizes the route taken by the famous *Santa Claus* character during his December gift distribution to children

globally. In this challenge, participants were tasked with optimising the delivery route to enhance the efficiency of gift distribution.

The challenge scenario revolves around a 2D cell grid with no friction, symbolizing the world. Within this grid, children are located, and two types of items, carrots (providing speed boosts) and gifts, can be picked up by the cart. While the cart maintains its speed on the frictionless grid, the total weight affects the impact of carrot consumption. Thus, the main goal consists in devising a route that efficiently delivers the most gifts within the time constraints. The scoring metric for this problem involves summing the scores of successfully delivered items.

In summary, this challenge can be categorized as a type of Vehicle Routing Problem, specifically a Capacitated with Pick up and Delivery Vehicle Routing Problem. This is due to the presence of capacity constraints on the cart and the need to pick up and deliver items throughout the cart's journey.

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 optimisation 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			✓	✓			
Optimise 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 Instances

In the competition context, in combination with the problem statements, test case instances are provided to participants with the primary aim of providing a mechanism for scoring teams, thus quantitatively assessing the efficacy of their

strategies. These instances are carefully generated to conform to the stipulated limits and constraints inherent to the challenge, as described in upon in the problem statement.

The initial instance, commonly denoted as the "example", is routinely included within the problem statement for contestants' reference. This instance is included in the problem statement for contestants' reference, but is not solved optimally. Its purpose is to illustrate the input and output format for the instance and solution. However, the example is intentionally designed with small dimensions, making it approachable via exact brute force methodologies.

Subsequent instances are typically designed to push the boundaries of the problem. These instances are intentionally large and design to discourage exact methods and general heuristics, aiming to thoroughly examine various aspects of the problem and avoid that (non-exact) greedy approaches find an optimal solution. The ruggedness of their objective space introduces challenges for solvers, potentially rendering some of them ineffective or even unusable within the available time budget.

In the competition context, teams are allowed to provide unique solutions for each instance, thus becoming a common practice among participants to conduct thorough cross-instance analysis. This practice proves valuable in revealing patterns that can offer insights into tackling the challenge with greater efficiency. As such, participants have the flexibility to develop focused strategies for each instance. This can in fact be interesting for the study of general-purpose meta-heuristics, to understand whether they can achieve comparable results to instance-specific approaches.

Finally, given the articulated problem statements and the transparent instance generation process, participants can create customized test instances. This capability proves valuable for debugging purposes in a competition setting and further advocates these problems as interesting benchmarks for black-box optimization [11].

3.4 Concluding Remarks

In this chapter, we conducted a comprehensive exploration of the Google Hash Code competition, delving into its structure, problem descriptions, and instances. In particular, we established links between the challenges presented and well-known CO problems. Furthermore, we highlighted common techniques employed by participants, drawing from our own engagement over several years.

We consider this analysis to be an important step that not only facilitated a deeper comprehension of the challenges, but also guided our choice of two specific problems (Optimise a Data Center and Book Scanning) for detailed exploration in this study. The particular choice of these problems is mainly motivated by the range of combinatorial optimization topics covered, leaving only vehicle routing and simulation as subjects to address in future work.

Moreover, based on the conducted analysis, we once again emphasize the importance of these problems as promising candidates for black-box optimization [11]. However, we believe that for this potential to be realized, it is essential to establish a repository containing the scores achieved across various problems and instances. Ideally, this repository should be accompanied by the corresponding source code for reproducibility purposes. From the standpoint of experimentally evaluating meta-heuristics for these problems, we consider it vital to generate a diverse set of instances, eventually generated through different methods.

Having understood the Google Hash Code problems, in the ensuing chapters, we will discuss our modelling approach to solve them, analyze the chosen problems, and conclude with a reflection on the work carried out.

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 OptimizationLaboratory. ii

CISUC Centre for Informatics and Systems of the University of Coimbra. ii

CO Combinatorial Optimization. 2, 3, 5–8, 23

FCT Foundation for Science and Technology. ii

KP Knapsack Problem. 8

MH Meta-Heuristic. 2, 3, 6, 8

Bibliography

[1] J.-B. Hiriart-Urruty. "Conditions for Global Optimality". In: *Handbook of Global Optimization*. Ed. by Reiner Horst and Panos M. Pardalos. Nonconvex Optimization and Its Applications. Boston, MA: Springer US, 1995, pp. 1–26. ISBN: 978-1-4615-2025-2. DOI: 10.1007/978-1-4615-2025-2_1.

[2] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Courier Corporation, Jan. 1, 1998. 530 pp. ISBN: 978-0-486-40258-1.

[3] Luca Di Gaspero and Andrea Schaerf. "EasyLocal++: An Object-Oriented Framework for the Flexible Design of Local-Search Algorithms". In: *Software—Practice & Experience* 33.8 (July 10, 2003), pp. 733–765. ISSN: 0038-0644. DOI: 10.1002/spe.524.

[4] S. Cahon, N. Melab, and E.-G. Talbi. "ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics". In: *Journal of Heuristics* 10.3 (May 2004), pp. 357–380. ISSN: 1381-1231. DOI: 10.1023/B:HEUR.0000026900.92269.ec.

[5] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. ISBN: 978-0-387-30303-1. DOI: 10.1007/978-0-387-40065-5.

[6] Ana Vieira. "Uma plataforma para a avaliação experimental de metaheurísticas". Doctoral Thesis. University of Algarve, Portugal, 2009.

[7] "Combinatorial Optimization". In: *Introduction to Evolutionary Algorithms*. Ed. by Xinjie Yu and Mitsuo Gen. Decision Engineering. London: Springer, 2010, pp. 263–324. ISBN: 978-1-84996-129-5. DOI: 10.1007/978-1-84996-129-5_7.

[8] Juan J. Durillo and Antonio J. Nebro. "jMetal: A Java Framework for Multi-Objective Optimization". In: *Advances in Engineering Software* 42.10 (Oct. 1, 2011), pp. 760–771. ISSN: 0965-9978. DOI: 10.1016/j.advengsoft.2011.05.014.

[9] Rafael Martí, Mauricio G. C. Resende, and Celso C. Ribeiro. "Multi-Start Methods for Combinatorial Optimization". In: *European Journal of Operational Research* 226.1 (Apr. 1, 2013), p. 2. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2012.10.012.

[10] P. Festa. "A Brief Introduction to Exact, Approximation, and Heuristic Algorithms for Solving Hard Combinatorial Optimization Problems". In: *2014 16th International Conference on Transparent Optical Networks (ICTON)*. 2014 16th International Conference on Transparent Optical Networks (ICTON). July 2014, pp. 1–20. DOI: 10.1109/ICTON.2014.6876285.

[11] Thomas Bartz-Beielstein et al. *Benchmarking in Optimization: Best Practice and Open Issues*. July 7, 2020.

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

[13] Samuel Barroca do Outeiro. "An Application Programming Interface for Constructive Search". Msc Thesis. University of Coimbra, Portugal, Nov. 9, 2021.

[14] Valentina Cacchiani et al. "Knapsack Problems — An Overview of Recent Advances. Part I: Single Knapsack Problems". In: *Computers & Operations Research* 143 (July 2022), p. 105692. ISSN: 03050548. DOI: 10.1016/j.cor.2021.105692.

[15] Google LLC. *Coding-Competitions-Archive*. Aug. 19, 2023. URL: https://github.com/google/coding-competitions-archive (visited on 08/20/2023).

[16] Christian Blum. "Metaheuristics in Combinatorial Optimization". In: *ACM Computing Surveys* 35.3 ().

[17] *Google Coding Competitions Archive*. URL: https://zibada.guru/gcj/#hc (visited on 08/20/2023).