1 2 9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE Ð
COIMBRA

Department of Informatics Engineering

Pedro Miguel Duque Rodrigues

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

Intermediate Report

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.

January 2023

Abstract

The Google Hash Code programming competition is an yearly held event that challenges teams to solve complex engineering problems within a limited time frame using any necessary tools. These problems, which are inspired by real-world issues and can be approached from both practical and theoretical perspectives, are of particular interest to this work. In the context of this thesis, we hope to solve some of these problems in a principled manner, with a particular focus on the modelling aspect and the clear separation between the concept of models and solvers. Additionally, there is interest in exploring the impact of this strategy on the development of general-purpose meta-heuristic solvers that can tackle these problems in a black-box fashion, making them more accessible for practitioners, researchers and developers.

Keywords

Meta-Heuristics • Modelling • Local Search • Constructive Search • Combinatorial Optimization • Intelligent Systems • Software Engineering

Resumo

A competição de programação Hash Code da Google é um evento organizado anualmente onde equipas são convidadas a resolver problemas de engenharia complexos num curto espaço de tempo, usando qualquer ferramenta necessária. Estes problemas, que são inspirados em questões do mundo real e podem ser abordados tanto sob um ponto de vista prático como teórico, são de particular interesse para este trabalho. Esta tese visa resolver alguns destes problemas de uma forma fundamentada, com particular ênfase na vertente de modelação e na clara separação entre o conceito de modelos e algoritmos (solvers). Além disso, há interesse em explorar o impacto desta estratégia no desenvolvimento de algoritmos (solvers) meta-heurísticos genéricos que consigam atacar estes problemas numa perperspetiva "black-box", tornando-os mais acessíveis para profissionais, investigadores e programadores.

Palavras-Chave

Meta-Heurísticas • Modelação • Procura Local • Procura Construtiva • Otimização Combinatória • Sistemas Inteligentes • Engenharia de Software

Contents

List of Figures

List of Tables

Acronyms

BBO	Black Box Optimization.
CO	Combinatorial Optimization.
GBO	Glass Box Optimization.
GO	Global Optimization.
KP	Knapsack Problem.
LO	Local Optimization.
TSP	Travelling Salesman Problem.

Chapter 1

Introduction

This chapter presents the motivation behind this project 1.1, the main contributions 1.2 we hope to make with this work and a brief outline 1.3 on this document's structure.

1.1 Motivation

The Hash Code programming competition is a yearly event held by Google where teams are asked to solve complex and challenging engineering problems using any tools and programming languages of their choice in four hours available. The problems are typically inspired by issues arising in real-world situations, such as vehicle routing, task scheduling, and Wi-Fi router placement. They are posed as "open" research problems for which there exists a variety of solutions of different qualities. In fact, the great majority of these are essentially Combinatorial Optimization (CO) problems concerning the search for the best solutions among a potentially large set of candidate solutions, thus being of utmost importance the adoption of efficient search strategies that take the available time budget into account. Moreover, in the context of the competition, the contestants must also read, understand the problem and find a suitable representation for it, which is to say that, not all the available time will be spent in the solution optimization stage.

For solving CO problems, a variety of algorithms exist that often offer a compromise between the time and quality of the solutions found. The heuristics are a set of procedures, often problem-specific, that attempt to quickly solve a problem and provide a helpful "rule of thumb" for achieving decent results, generally in a greedy fashion. A superset of these algorithms is called meta-heuristics, and contrary to regular heuristics, these are generic and can be applied to a broad range of problems. Natural processes and phenomena, such as collective behavior, natural selection, and some physical properties of materials, inspire several meta-heuristics search processes making them flexible and adaptable, although more computationally intensive than heuristics.

A suite of optimization tools of different natures is available to practitioners for solving CO problems, thus constituting a challenge to enumerate them all. In particular, there exist a set of tools that rely upon more mathematical and exact approaches yielding optimal solutions although not being used in a competition context due to their poor time performance on challenging problems, such as the Hash Code challenges.

In the context of the Google Hash Code competition, competitors frequently make use of greedy and heuristic strategies tailored to the challenge. Moreover, meta-heuristics in this situation are not as popular due to the time constraints imposed by the competition format. At first instance, the majority of optimization problems are structurally different from each other, which makes it demanding to write general-purpose heuristic solvers that can be easily reused. On the other hand, it might not be worth the effort spent in the implementation of such solvers and provided that the benefit of using a simple heuristic strategy might outweigh the development and the running cost of meta-heuristic solvers e.g., evolutionary algorithms.

Algorithms such as meta-heuristics usually follow some notion of an optimization strategy that guides the procedure in the search for solutions. Some of the main strategies are constructive and local search. Constructive search algorithms work by starting with an empty or partially complete solution and building a complete and feasible solution by iteratively adding components based on the solution's current state and the problem's constraints. In contrast, local search algorithms develop a given initial solution to the problem by introducing small changes that aim to improve it to a local optimum rather than a global one. A common usage of these strategies in competition is to sequence them, starting with a constructive search stage, improving solutions to a certain stagnation point and then taking advantage of a local search strategy in an attempt to enhance the solutions even further.

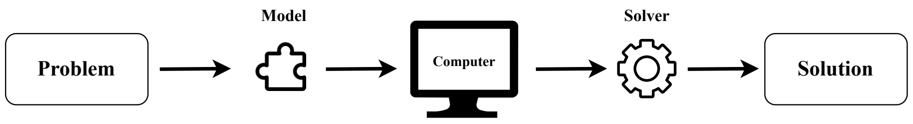

Figure 1.1: Modelling and Problem Solving

It is evident that algorithms or solvers that utilize these search strategies are highly dependent on the specific problem they are trying to solve. The solver plays a crucial role in the problem-solving process as it is responsible for finding a solution. However, without a comprehensive model that can provide the solver with relevant information about the problem, the solver's effectiveness may be impaired.

The model serves as a means of presenting the various aspects of the problem to the computer, which will subsequently utilize a solver to find a solution. When constructing a model, it is important to include relevant features such as the representation of the problem and solution, a description of how components can be added or removed from the solution, and methods for evaluating the solution, calculating bounds, and obtaining other heuristic information.

A good model should encode in it all the relevant information from the problem and ask the right questions to obtain it, adhering to the philosophy that – “*understanding the question is half the answer*”. Additionally, if a model was to be built in a standardized way, this would allow the development of a suite of generic and reusable (meta-heuristic) solvers that could find solutions in a black-box fashion. This idea has already been explored to some extent in previous work [10, 12] and will be further deepened.

It is worth noting that the modelling aspect in this field has often been neglected by the community, which has been primarily focused on the development of meta-heuristics algorithms (solvers). This discrepancy can be contrasted with the mathematical perspective, where practitioners and researchers have emphasized the importance of clearly separating the model and the algorithm (solver), and have placed a significant emphasis on the modelling perspective. This gap highlights the importance of considering the modelling aspect in this field.

There has also been a recent growing interest within the community in the development of optimization benchmark problems that are both relevant in practical applications and amenable to theoretical analysis. The Hash Code problems may be suitable candidates for this purpose, as they pose significant challenges from a modelling perspective while also being easily describable. Furthermore, these problems have already been partially solved in a competitive setting and have a wealth of empirical data available on the most effective known solutions. Overall, these factors make the Hash Code problems an ideal testing ground for both modelling and the evaluation of meta-heuristics.

1.2 Contribution

The main goal of this work is to develop and implement effective heuristic and meta-heuristic approaches for solving Hash Code problems, with a particular focus on the modelling aspect and the clear separation between solvers and models. By doing so, we hope to develop more structured and efficient problem-solving strategies that can effectively address a range of challenges. Additionally, some effort will be made to address other key areas that are crucial to the success of this work, namely:

- Development and refinement of the frameworks that separate models from solvers currently materialized in an Application Programming Interface (API) designed only for constructive search [10]. The main goal is to optimize and “fine-tune” this API in order to improve its efficacy and utility, by using the Hash Code problems as benchmarks.
- Expansion of the aforementioned API to support local search strategies in a problem-independent manner. This will allow for its application to a wider range of problems and contexts.
- Implementation of a small set of general-purpose meta-heuristic solvers and utilities that can be used to not only generate and test solutions

for the multiple Hash Code benchmark problems, but also to verify the correctness of the results.

Last but not least, the objective of this work is to engage in a critical examination of the strengths and limitations of our proposed approach to problem modelling and solver development. This discussion will be relevant to meta-heuristic researchers, software developers, and practitioners alike. The analysis will consider various performance dimensions, including the effort required for problem modelling and solver development, the computational efficiency of the implemented software, and the quality of the solutions obtained.

1.3 Outline

The remainder of this document is structured as follows:

- **Chapter 2:** Provides some background on some essential aspects of optimization, search strategies, meta-heuristics, and modelling. Moreover, it presents the modelling frameworks and current state of the art of the API [10] that supports this work.
- **Chapter ??:** Gives some insight into the Google Hash Code competition and typical problems presented to contestants. Furthermore, it makes a brief categorization of all the problems from previous editions, with particular emphasis on the ones analyzed so far.
- **Chapter 3:** Analyzes the main objectives that we hope to achieve, along with the methodology required to successfully accomplish them. In addition, a work plan is presented outlining the tasks to be carried out in the upcoming semester. Finally, some comments are made regarding the usability and ease of use of the tools that will be utilized, given their current state.
- **Chapter 4:** Focuses on the work that was completed during the first semester. Specifically, it describes the modelling of the problems examined and the results obtained.
- **Chapter 5:** Presents a summary of the work completed and some observations about the next steps to be taken.

Chapter 2

Background

This chapter presents a literature review of various optimization concepts relevant to the analysis of Hash Code problems as well as the framework detailed in [10], which will be used and potentially improved upon throughout this work. Additionally, it provides background on essential concepts such as modelling and meta-heuristics, which will be further discussed in subsequent chapters. In particular, Section 2.1 presents a series of combinatorial optimization concepts relevant to this work, whilst Section 2.3, delves into the definition of meta-heuristics and provides key concepts and examples. Finally, Section 2.4 presents the concept of modelling and offers insight into current frameworks and API [10, 12] based around this idea.

2.1 Optimization Concepts

Optimization involves finding the best solution to a given problem among a set of feasible solutions. Specifically, considering the single objective case involves finding the optimal configuration or set of parameters that maximize or minimize an objective function, possibly subject to constraints on its variables [9].Given that, and as defined by Papadimitriou and Steiglitz [11], an optimization problem can formally be described as follows:

Definition 2.1.1 (Optimization Problem [11]): *An optimization problem is a collection $\mathcal{I}$ of instances, typically generated in a similar manner. An instance ι of an optimization problem consists of a pair $(\mathcal{S}, f)$, 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, for each solution $s \in \mathcal{S}$, a real value is assigned to indicate the quality of the solution. Thus, the problem consists in finding a global optimal solution $s^* \in \mathcal{S}$, for each instance ι.

Definition 2.1.2 (Global Optimal Solution [6, 11]): *Assuming, without loss of generality, an optimization problem with a maximizing objective function*

$f(s)$, a global optimal solution s^* is expressed by:

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

Given that the Hash Code problems are designed with a maximizing objective function, only maximization problems will be considered in this work. Nonetheless, a minimizing objective function $f(s)$ can be reformulated for maximization by using the identity $\max -f(s) = \min f(s)$.

2.1.1 Combinatorial Optimization

There are two main categories of optimization problems based on the domain of the variables: discrete and continuous. In problems with discrete variables, the solutions are defined on a finite, or countably infinite, set of values. In contrast, for problems with continuous variables, the solutions take on any value on a continuous (infinite) subset of real numbers. Nonetheless, there are also problems that involve both categories commonly denominated as mixed [9].

Definition 2.1.3 (Combinatorial Optimization Problem [11]): *An instance ι of a combinatorial optimization problem is an instance of an optimization problem (2.1.1) where the set $\mathcal{S}$ of feasible solutions is finite or countable infinite.*

Combinatorial Optimization (CO) problems 2.1.3 are a subset of discrete optimization problems characterized by a discrete solution space that typically involves different permutations, groupings, or orderings of objects that satisfy certain criteria [12, 11]. As such, solutions for these problems are discrete objects related to the combinatorics e.g integers, permutations sets and graphs. [2, 3]

Typical examples of CO problems include network flow, matching, scheduling, shortest path and decision problems. Some representative examples of such problems are the Travelling Salesman Problem (TSP) and the Knapsack Problem (KP) [3]. In the case of the KP, given a set of items, each with a given weight and profit, and a knapsack with a given maximum weight capacity, the goal is to find the subset of items with the highest total profit that can be placed in the knapsack without exceeding its capacity. As for the KP, the goal is to find the shortest possible route that visits each city exactly once whilst returning to the starting city i.e an Hamiltonian cycle.

Definition 2.1.4 (Ground Set [10]): *The ground set $\mathcal{G}$ is a finite set of elements that represents all possible components that may integrate a feasible solution for a given instance ι of a CO problem.*

$$\mathcal{G} : \{c_1, c_2, c_3, \ldots, c_i\} \tag{2.3}$$

Thus, a feasible solution is a subset of the ground set, denoted as $s \in \mathcal{S} \subseteq 2^{\mathcal{G}}$, and which may be encoded as a binary string indicating the presence (1) or absence (0) of a given component c_i.

Given the discreteness of the decision space in CO problems, the solutions are constructed by combining objects present in a finite set containing all the individual components that fully characterize a solution. This set, commonly denominated "ground set" [10, 5, 8], can be defined as shown in 2.1.4:

Regarding the definition of a CO problem 2.1.3 and alluding to the solution representation as a binary string, we can then define both the feasible solution set $\mathcal{S}$ and the ground set $\mathcal{G}$ for the KP and the TSP.

- *Knapsack Problem*: The set of feasible solutions $\mathcal{S}$ consists of all possible subsets of items that can be placed in the knapsack without exceeding its weight capacity. The ground set $\mathcal{G}$ is a set of all possible items that may be included in a feasible solution [5]. As such, a solution $s \in \mathcal{S}$ can be represented as a binary string, where the i^th position of the string contains a value of 1 or 0 depending on whether the i^th item in $\mathcal{G}$ is present or absent in the final solution.

- *Travelling Salesman Problem*: The set $\mathcal{S}$ consists of all possible Hamiltonian cycles representing the order in which cities should be visited, typically represented as a permutation. The ground set $\mathcal{G}$ in this case can be thought of as a set containing all possible paths between two distinct cities, i.e. edges in a graph [8, 5]. Hence, a solution $s \in \mathcal{S}$ can be represented as a binary string where the i^{th} position of the string contains a value of 0 or 1 depending on whether a given edge connecting two cities is present or absent in the final solution.

In summary, 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 optimal solutions can be difficult, and exhaustive search strategies may not be able to solve many of these problems, which are often NP-Hard [3, 5] and thus not approachable by algorithms in a reasonable amount of time. In these cases, approximation or heuristic/meta-heuristic methods present themselves as effective alternatives to be considered.

2.1.2 Global and Local Optimization

With regard to the search of solutions for optimization problems, there are two primary strategies: Global Optimization (GO) and Local Optimization (LO).

GO refers to the process of identifying a global optimal solution (2.1.2) to a given problem, regardless of its location within the solution space. In contrast, LO focuses on finding the best solution among those that are proximate in some sense. The concept of proximity is related to the definition of a neighborhood, which for a given solution is specified by a particular neighborhood structure defined as follows:

Definition 2.1.5 (Neighborhood Structure [11, 2]): *A neighborhood struc-*

ture for an optimization problem with instances $(\mathcal{S}, f)$ *is a mapping:*

$$\mathcal{N} : \mathcal{S} \longrightarrow 2^{\mathcal{S}} \tag{2.4}$$

Such that, a set of neighboring solutions $\mathcal{N}(s) \subseteq \mathcal{S}$ *is assigned for each solution* $s \in \mathcal{S}$. *This referred to as the neighborhood of* s.

In general, the neighborhood structure refers to the set of rules that must be applied to a solution in order to generate all of its neighbors. Additionally, we consider the following definition of local optimal solution.

Definition 2.1.6 ((Strictly) Local Optimal Solution [2, 9]): *Assuming maximization without loss of generality, a solution* $\hat{s}$ *is local optimal with respect to a given neighborhood structure* $N(\hat{s})$ *iff:*

$$\forall s \in \mathcal{N}(\hat{s}) : f(\hat{s}) \geq f(s) \tag{2.5}$$

Furthermore, $\hat{s}$ *is considered strictly global optimal iff*

$$\forall s \in \mathcal{N}(\hat{s}) \setminus \{\hat{s}\} : f(\hat{s}) > f(s) \tag{2.6}$$

As an illustrative example, consider the objective function $f(s)$ shown in figure 2.1. With respect to the definitions 2.1.2 and 2.1.6 s^2, s^3 are (strictly) local optimal.

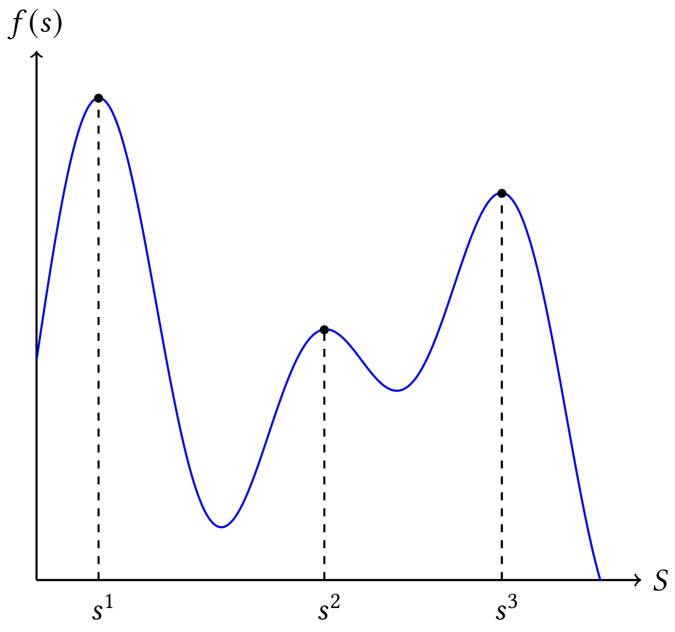

Figure 2.1: Global and Local Optimal Solutions

In practice, the decision to use either a global or local optimization strategy is often influenced by factors such as the available time budget and the preferences of the decision maker. While global optimization aims to find the optimal solution to a problem, the search process may be time-consuming or, in some cases, computationally infeasible due to the size of the search space. On the other hand, local optimization, while lacking the optimality guarantees of global optimization, is able to quickly generate good solutions that may be acceptable to the decision maker. Nonetheless, the quality of the solutions may be poor due to the ruggedness of the objective function fitness landscape (many local optima). Ultimately, the performance of both methods is closely tied to the knowledge of problem-specific features.

In a setting such as the Hash Code competition, the problems presented are designed to resemble real-world situations. Additionally, due to the time constraints, it is not in the interest of the contestants to use global optimization methods, as they are unlikely to finish on more complex problem instances. Instead, a balance between global and local optimization is typically employed, with contestants utilizing the advantages of both methods. This approach involves establishing a baseline with global optimization methods that explore unseen regions of the search space for potentially good solutions, and then utilizing local optimization methods for exploiting solutions that have already been found.

2.1.3 Black-Box and Glass-Box Optimization

In the field of optimization, two settings are commonly recognized: Black Box Optimization (BBO) and Glass Box Optimization (GBO).

In BBO optimization settings there is no information about the landscape of the function being optimized or constraints defining the set of feasible solutions [1] or the objective function is too complex to be approached from an analytical perspective. As such, algorithms for achieving solutions for these problems do so only by interacting with the problem through the evaluation of potential candidate solutions [4]. Meta-Heuristics, as will be later detailed are examples of methods that follow this approach for finding/improving solutions. By contrast, Glass Box Optimization (GBO) optimization, also known as “white box” optimization, there is a good understanding of the problem instance being optimized and the objective function properties [4]. Hence, the algorithms used may take advantage of more analytical properties of the problem since they are transparent to the optimizer.

For the purpose of clarification, with regard to the previously mentioned TSP, a black-box strategy would entail the utilization of a search heuristic such as simulated annealing [7] to obtain solutions. This is because the algorithm only necessitates knowledge of how to evaluate the quality of solutions through the objective function and not any specific information about the function being optimized. Alternatively, if the problem were to be formulated as an Integer Linear Programming problem [9, 11], it would become amenable to a white-box approach, as the objective function would be accessible, and additional information about the problem could be inferred from it and provided to the algorithm.

In the context of the Hash Code competition, contestants typically engage with a problem from a black-box perspective, as the underlying objective function of the problem is not disclosed, and/or is too complex to formalize. Additionally, the process of formalization would be time-consuming and, as a result, the usage of white-box methods post-formalization would not be justified, given that they may be computationally slower. Despite this, it is important to note that white-box methods should not be overlooked as they construct a model of the problem in order to solve it, which could be applied to a certain extent in the context of black-box approaches.

2.2 Optimization Strategies

2.2.1 Exact and Approximation Approaches

2.2.2 Constructive Search

2.2.3 Local Search

2.2.4 Bounds

2.3 Meta-Heuristics

2.4 Modelling

2.4.1 Frameworks

Chapter 3

Approach and Objectives

3.1 Objectives

3.2 Timeline

Chapter 4

Preliminary Work

4.1 The Hash Code Problems

The Hash Code competition problems are modeled on complex engineering challenges faced by Google. Given the scale of the enterprise, it is evident that they represent complex engineering challenges that mirror real-world scenarios prevalent in the industry. As such, these problems not only function as valuable benchmark problems, but also provide an opportunity to establish a robust baseline for the development and testing of modeling based metaheuristic approaches. This is due to the fact that many individuals with diverse backgrounds have attempted to solve these problems, and their scores and some of the solving approaches are readily accessible.

When examining the available information about the solutions for the problems, it is essential to take into consideration a few key points. Firstly, since only a select number of teams from the qualification round advance to the finals, the number of solutions available tend to be limited. However, it is reasonable to expect that the solutions found by the top teams will be excellent representatives of the best problem-solving processes, as they have undergone the selection process. Secondly, in the early years of the competition, it was only open to teams from Paris. 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.

In order to fulfill the objectives of this work, it is crucial to gain a thorough understanding of the problem at hand before proceeding with the formulation of the model. This includes identifying the objective, classifying the problem type, understanding the constraints, and most importantly, determining its relationship to problems previously studied in literature. Through this analysis, valuable insights into the appropriate approach can be gained. Therefore, a comprehensive analysis of the Hash Code problems was conducted and a summary of the key aspects of each problem evaluated in this first semester is presented in the following sections

4.1.1 Google Hash Code 2014

Final Round: "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 (time) 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 visited 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.

This problem bears a significant resemblance to a set covering problem (cite) where in this specific instance, the objective is to schedule the routes for the cars in such a way that the combination of all the sets of streets visited by each car encompasses the entirety of the city streets, while doing so in the most efficient manner possible.

4.1.2 Google Hash Code 2015

Qualification Round: "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 a such a way as to maximize the minimum guaranteed capacity for all resource pools. This metric serves as the criterion for evaluating solutions to this problem. The minimum 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 entire row of servers.

Overall, by examining the problem from a modeling perspective, it can be classified as an assignment problem. Furthermore, this challenge bears resemblance to the Knapsack Problem (cite) to some extent. In this case, the knapsack, which can be thought of as the set of all rows in the server, is divided into distinct sections, and the placement of servers in these sections is important, although still contributing to the overall capacity of the knapsack. Additionally, nuances must also be taken into account when evaluating the score as the assignment of servers to specific resource pools within these sections is also crucial for the problem's scoring.

Final Round: "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 categorized as both a simulation and a covering problem, based on the properties previously outlined. Additionally, this problem can be represented in a three-dimensional graph, where edges symbolize altitude changes and lateral movement (wind) for a specific time instant, which define the components of a solution.

4.1.3 Google Hash Code 2016

Qualification Round: "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 the objective is to maximize this overall score.

Through this description we can observe that this problem can be reduced to a variant of the Vehicle Routing Problem (cite) that takes into account the pickup and delivery of items, the time window for delivery and the multiple routes that each vehicle may do.

4.1.4 Concluding Remarks

In summary, this section provided an overview and description of the key aspects of the Hash Code problems studied in the first semester. Furthermore, a categorization that links these problems to topics commonly found in combinatorial optimization literature was presented. The table 4.1 shows a summary of the analysis conducted.

	Categories				
Problem	Assignment	Covering	Vehicle Routing	Simulation	Sum
Street View Routing		✓			1
Optimize a Data Center	✓				1
Loon		✓		✓	2
Delivery			✓		1

Table 4.1: Categorization of Google Hash Code Problems

4.2 Modelling "Optimize a Data Center"

In this section, we will analyze the model developed for the Hash Code 2015 qualification round problem, entitled "Optimizing a Data Center." To begin, in section 4.2.1 will provide a detailed description of the problem and key concepts outlined in the problem statement, which are essential for understanding the context. In section 4.2.2, we will examine the problem's objective, focusing on the objective function and its properties. In section 4.2.3, we will delve into the model developed for this problem, highlighting the representation and the constructive search strategy employed, which are aligned with the concepts outlined in [12, 10]. Finally, in section 4.2.5, we will present a brief overview of the results and in section 4.2.5, we will offer some final thoughts and observations on the problem and the model developed for it.

4.2.1 Problem Description

As previously discussed in Section 4.1.2, the problem at hand entails optimizing the placement of servers in a data center. The data center is modeled as a series of rows, each containing slots in which servers can be placed. However, it should be noted that certain slots may be unavailable due to other installations within the data center. The servers available (to be continued...)

4.2.2 Objective

4.2.3 Model

Representation

Constructive Search

4.2.4 Results

4.2.5 Concluding Remarks

Chapter 5

Conclusion

References

[1] Stéphane Alarie et al. “Two Decades of Blackbox Optimization Applications”. In: *EURO Journal on Computational Optimization* 9 (Jan. 1, 2021), p. 100011. ISSN: 2192-4406. DOI: 10.1016/j.ejco.2021.100011.

[2] Christian Blum. “Metaheuristics in Combinatorial Optimization”. In: *ACM Computing Surveys* 35.3 ().

[3] “Combinatorial Optimization”. In: Xinjie Yu and Mitsuo Gen. *Introduction to Evolutionary Algorithms*. Red. by Rajkumar Roy. Vol. 0. Decision Engineering. London: Springer London, 2010, pp. 263–324. ISBN: 978-1-84996-128-8 978-1-84996-129-5. DOI: 10.1007/978-1-84996-129-5_7.

[4] Carola Doerr. “Complexity Theory for Discrete Black-Box Optimization Heuristics”. In: *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*. Ed. by Benjamin Doerr and Frank Neumann. Natural Computing Series. Cham: Springer International Publishing, 2020, pp. 133–212. ISBN: 978-3-030-29414-4. DOI: 10.1007/978-3-030-29414-4_3.

[5] 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.

[6] 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.

[7] Sean Luke. *Essentials of Metaheuristics*. second. Lulu, 2013.

[8] 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.

[9] 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.

[10] Samuel Barroca do Outeiro. "An Application Programming Interface for Constructive Search". MA thesis. Nov. 9, 2021.

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

[12] Ana Vieira. "Uma plataforma para a avaliação experimental de meta-heurísticas". Doctoral Thesis. 2009.