



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE
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 a 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 perspetiva “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

1	Introduction	1
1.1	Motivation	1
1.2	Contribution	3
1.3	Outline	4
2	Background	5
2.1	Optimization Concepts	5
2.1.1	Combinatorial Optimization	6
2.1.2	Global and Local Optimization	6
2.1.3	Black-Box and Glass-Box Optimization	6
2.2	Optimization Strategies	6
2.2.1	Local Search	6
2.2.2	Constructive Search	6
2.2.3	Bounds	6
2.3	Meta-Heuristics	6
2.4	Modelling	6
2.4.1	Frameworks	6
3	The Google Hash Code Problems	7
4	Approach and Objectives	8
5	Preliminary Work	9
6	Conclusion	10
A	Hash Code Problem Description	11
	References	12

List of Figures

1.1	Modelling and Problem Solving	2
-----	---	---


List of Tables

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 behaviour, 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.

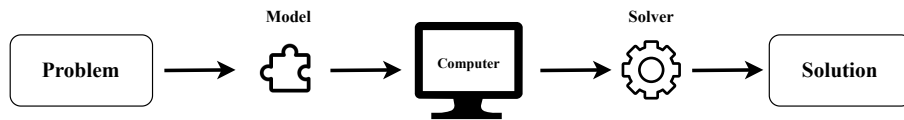


Figure 1.1: Modelling and Problem Solving

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 [1, 2] 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 [1]. 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 [1] that supports this work.
- **Chapter 3:** 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 4:** 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 5:** 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 6:** 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 [1], 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, 2.4 presents the concept of modelling and offers insight into current frameworks and API [1, 2] 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 in-**volves finding the optimal configuration or set of parameters that maximize or minimize an objective function, possibly subject to constraints on its variables [3]. Given that, and as defined by Papadimitriou and Steiglitz [4], an optimization problem can formally be described as follows:

Definition 2.1.1 (Optimization Problem [4]) *An optimization problem is a collection I of instances, typically generated in a similar manner. An instance ι of an optimization problem consists of a pair (S, f) , where S is the set containing all feasible solutions, and f is an objective (cost) function, with a mapping such that:*

$$f : S \longrightarrow \mathbb{R}$$

That is, for each solution $s \in S$, a real value is assigned to indicate the quality of the solution. Thus, the problem consists in finding the solution $s^ \in S$, for each instance ι that satisfies:*

$$\forall s \in S, f(s^*) \geq f(s)$$

Such a solution s^* is called a (globally) optimal solution to the given instance ι

This definition only applies to maximization problems. However, minimization problems can be reformulated for maximization by using the identity $\min f(x) = \max -f(x)$ and applying the same transformation to the constraints, if any. Given that the Hash Code problems are designed with the the objective of maximizing the score, without loss of generality, only maximization problems will be considered in this work.

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.

Definition 2.1.2 (Combinatorial Optimization Problem [4]) *An instance ι of a combinatorial optimization problem is an instance of an optimization problem (2.1.1) if the set S of feasible solutions is finite or countable infinite.*

Combinatorial Optimization (CO) problems 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 (2.1.2).

2.1.2 Global and Local Optimization

2.1.3 Black-Box and Glass-Box Optimization

2.2 Optimization Strategies

2.2.1 Local Search

2.2.2 Constructive Search

2.2.3 Bounds

2.3 Meta-Heuristics

2.4 Modelling

2.4.1 Frameworks

Chapter 3

The Google Hash Code Problems

Chapter 4

Approach and Objectives

Chapter 5

Preliminary Work

Chapter 6

Conclusion

Appendix A

Hash Code Problem Description

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

References

- [1] Samuel Barroca do Outeiro. “An Application Programming Interface For Constructive Search”. Msc Thesis. Pólo II - Pinhal de Marrocos, 3030-290 Coimbra: University Of Coimbra, Faculty Of Sciences and Technology, Department Of Informatics Engineering, 2021. URL: <http://hdl.handle.net/10316/98261>.
- [2] Ana Cristina do Carmo Cardoso Vieira. “Uma Plataforma para a Avaliação Experimental de Meta-heurísticas”. Phd Thesis. Campus de Gambelas, Edifício 8, 8005-139 Faro: Universidade do Algarve, Faculdade de Ciências e Tecnologia, 2009. URL: <http://hdl.handle.net/10400.1/518>.
- [3] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. ISBN: 9780387400655. URL: <https://books.google.pt/books?id=VbHYoSye1FcC>.
- [4] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Books on Computer Science. Dover Publications, 1998. ISBN: 9780486402581. URL: <https://books.google.pt/books?id=u1RmDoJqkF4C>.