Universitat Oberta de Catalunya

MSc in Computational and Mathematical Engineering

PEC1 - Definition and Work Plan

Author:
Manuel Canedo Tabares

Reviewer:
Dr. Antonio Burgera
Burgera

An assignment submitted for the subject:

TFM - Artificial Intelligence

February 20, 2023

Contents

1 Title and Keywords

1.1 Title

Optimization Heuristics Performance Analysis and Implementation Guidelines

1.2 Keywords

- Optimization Heuristics
- Python
- Julia
- Permutation Flowshop Scheduling Problem (PFSP)
- Nawaz, Enscore, and Ham (NEH) Heuristic
- Vehicle Routing Problem (VRP)
- Clarke and Wright Savings (C&W) Heuristic
- Profiling
- Performance Optimisation
- Performance Guidelines

2 Context and Justification of the Work

Optimisation heuristics are crucial tools in many AI applications. These algorithms solve complex optimisation problems often encountered in real-world scenarios. For example, they can be used to optimise resource allocation in transportation, logistics, and manufacturing or to improve decision-making processes in healthcare, finance, and other industries. They are also often used as components of larger AI systems, such as deep learning networks, reinforcement learning algorithms, and expert systems. Developing high-performance implementation guidelines for these algorithms makes them more efficient and effective, improving the overall performance of AI systems and leading to more accurate predictions, efficient decision-making, and effective problem-solving.

This thesis also aims to democratise AI by making it more accessible to a broader range of users. High-performance implementation guidelines can enable non-experts to use heuristics more effectively without requiring specialised optimisation expertise. This empowers individuals and organisations with limited resources to benefit from AI and drive innovation in their respective fields.

3 Description

This thesis aims to develop high-performance implementation guidelines for heuristics and metaheuristics by translating existing Python heuristics to Julia and optimizing them using high-performance computing principles. The study will then benchmark these heuristics and conclude how implementation quality affects the validity of results. The goal is to develop generalizable guidelines and rules of thumb for future researchers to ensure that their heuristic implementations perform well by default. The study will be carried out in collaboration with the ICSO META research group, which has developed a range of heuristics for various optimization problems.

4 Objectives

4.1 General objectives

The general objective of this thesis is to develop high-performance implementation guidelines for optimization heuristics.

4.2 Specific objectives

1. Translate existing Python heuristics to Julia: The first objective of this thesis is to translate the existing Python heuristics developed by the ICSO META research group to Julia. This will enable us to utilise Julia's performance and low-level optimisation capabilities.

2. Optimise heuristics using high-performance computing principles: The second objective is to profile and optimise these heuristics using high-performance computing principles.

3. Benchmark heuristics and conclude: The third objective is to benchmark the optimised heuristics and draw conclusions on how the implementation quality affects the results' validity. I will evaluate the accuracy and efficiency of the heuristics and compare them to the original unoptimised Julia and Python versions.

4. Develop generalisable guidelines: The final objective is to develop generalisable guidelines and rules of thumb that future researchers can use to ensure that their heuristic implementations perform well by default. I will distil the findings into a set of best practices that can be applied to various optimisation problems.

5 Approach and Method

5.1 Julia Language

To achieve the first objective of this thesis, the heuristics will be translated from Python to Julia. Julia is a high-performance programming language specifically designed for

scientific computing and optimization. Compared to Python, Julia offers several key advantages in terms of performance, including:

- Just-in-time (JIT) compilation: Julia uses a JIT compiler, which means the code is compiled at runtime, allowing for dynamic optimization based on the specific input data. This can result in significant performance gains, especially for compute-intensive tasks.
- Type system: Julia has a sophisticated type system that allows for multiple dispatches, which means the code can be optimised based on the specific types of input data. This can lead to significant performance gains compared to Python, which has a more limited type of system.
- Low-level control: Julia provides low-level control over memory allocation and other performance-critical details, which can be used to achieve maximum performance.

By using Julia, I can take advantage of these features to achieve faster and more efficient heuristic implementations.

5.2 Profiling

Once the heuristics have been translated to Julia, profiling will be done to identify potential performance bottlenecks in the code. I will use built-in Julia profiling tools such as @time and @profile to measure the execution time and memory allocations of different parts of the code and identify areas that may be causing bottlenecks. In addition, I will use external tools such as FlameGraphs to visualise the profiling data and better understand the code's performance characteristics. By identifying the parts of the code slowing down the program, I can focus the optimisation efforts on those areas to achieve the best possible performance.

5.3 Optimisation

With a better understanding of the performance characteristics of the code, I will use low-level performance principles to optimise the execution. This will involve various techniques.

These techniques include preallocating memory for data structures, preventing the creation of conditional branches in computationally intensive code paths, using optimised built-in functionalities of the language and avoiding unnecessary work and more.

Optimising the code at a low level can achieve significant performance improvements that would not be possible with a higher-level approach.

5.4 Benchmark

To evaluate the effectiveness of the optimisations, I will benchmark the optimised heuristics against the original Python versions. This will involve running a series of

experiments on a range of optimisation problems and measuring the heuristics' performance in accuracy and efficiency. By comparing the results of the optimised and unoptimised implementations, I can determine the impact of the optimisation efforts on the overall quality of the heuristics.

6 Planning

6.1 Tasks

- Review the existing Python heuristics developed by the ICSO META research group and identify which ones to translate to Julia for this study.
- Conduct a literature review on high-performance computing principles and identify which techniques are relevant for optimising heuristics in Julia.
- Translate the selected Python heuristics to Julia and ensure they produce the same results as the original Python versions.
- Develop a profiling plan to identify performance bottlenecks in the Julia implementations.
- Profile the Julia heuristics using built-in profiling tools such as @time and @profile.
- Visualise profiling data using external tools like FlameGraphs to understand performance characteristics better.
- Analyse profiling data to identify areas of the code that may benefit from optimisation.
- Develop an optimisation plan and apply relevant techniques to improve the performance of the heuristics.
- Benchmark the optimised Julia heuristics against the original Python versions to measure performance improvements.
- Evaluate the accuracy and efficiency of the optimised heuristics and compare them to the original unoptimised versions.
- Translate the findings into generalisable guidelines and best practices for future researchers to ensure high-performance heuristic implementations.
- Obtain feedback from the ICSO META group researchers and apply their suggestions.
- Write up a comprehensive report.

6.2 Schedule

- Translate existing Python heuristics to Julia (2 weeks)
  - Understand the existing Python code (1 week)
  - Implement and benchmark the code in Julia (1 week)
- Profile the heuristics (2 weeks)
  - Learn to use Julia built-in profiling tools (1 week)
  - Profile and identify performance bottlenecks in the code (1 week)
- Optimise the heuristics (6 weeks)
- Benchmark heuristics and conclude (2 weeks)
- Develop generalisable guidelines and apply feedback (2 weeks)
- Finalise report write-up and prepare a presentation (2 weeks)

6.3 Milestones

1. A selection of heuristics is implemented in Julia and benchmarked.
2. The heuristics are profiled and optimised.
3. The guidelines are generalised.
4. The report is finalised.

6.4 Risks

Several risks have been identified, and more may reveal themselves along the way:

- Scope-creep: This thesis implementation goals are ambitious, but I trust my ability to deliver them. However, new necessities and objectives can arise along the way, risking the completion of some pursuits. If this were an issue, I would reduce the number of heuristics under study for the upcoming phases.
- Low optimisation margins: Encountering extremely thin performance margins would force revising the project idea and roadmap. If this were an issue, I would identify why the margin for optimisation is so low, using the discoveries of what was done right to formulate guidelines. I would also target other heuristic implementations that offer more margin for optimisation.
- Work-life-studies difficulties: While I am used to balancing these elements while keeping my mental and physical health, an unexpected life event could risk my ability to deliver in any of these aspects. If that were the case, I could either reduce the scope of the work or keep the size as is and accept to present later.

7 Expected Results

- A high-quality Work Plan.
- A high-quality Work Report.
- A selection of optimised Julia heuristics
- A deep understanding of the performance implication of different implementation techniques.
- Results that justify the need for high-quality implementations in the literature.
- A comprehensive set of guidelines and rules of thumb that researchers can apply without specialised low-level knowledge.
- An engaging virtual presentation.