# Chapter 1

# Evaluation

## 1.1 Measures of Success

The success of the project can be measured by comparing the project results with objectives. We will also review the design choices in constructing the tool. The review will highlight the practical outcomes from using argumentation to explain scheduling.

Arguably, the most important outcome of this project is that the tool is functionality correct. This means the tool is required to explain schedules for feasibility, efficiency and satisfaction with respect to user decisions.

One objective is to implement an accessible tool. To measure accessibility, we can refer to the tractability complexity from explanations. The length of explanations either in the number of words or characters may be correlated with understandability. Alternatively, we can conduct a survey targeted towards potential users. Because the understandability of explanations are difficult to measure, we will attempt to quantity this using multiple choice questions. To carefully evaluate explanations, one would need to refer to cognitive science, which is beyond the scope of this project. In addition, we can measure performance to reflect responsiveness and scalability of the tool. This may be achieved by using profiling utilities to measure performance metrics such as time to generate explanations and memory consumption.

We can also use the survey to ask how well our tool performs with respect to applicability and knowledge transfer. We can also infer the practical limitations from the tool to measure the applicability.

## 1.2 Theoretical Results

| Algorithm | Computational | Memory | Tractability |
|---|---|---|---|
| CONSTRUCT-FEASIBILITY | $\mathcal{O}(m^2n^2)$ | $\mathcal{O}(m^2n^2)$ | |
| CONSTRUCT-EFFICIENCY | $\mathcal{O}(m^2n^2)$ | $\mathcal{O}(m^2n^2)$ | |
| CONSTRUCT-SATISFACTION | $\mathcal{O}(m^2n)$ | $\mathcal{O}(m^2n^2)$ | |
| COMPUTE-UNATTACKED | $\mathcal{O}(m^2n^2)$ | $\mathcal{O}(mn)$ | |
| COMPUTE-PARTIAL-CONFLICTS | $\mathcal{O}(mn)$ | $\mathcal{O}(mn)$ | |
| EXPLAIN-STABILITY | $\mathcal{O}(m^2n^2)$ | $\mathcal{O}(m^2n^2)$ | |
| EXPLAIN-FEASIBILITY | $\mathcal{O}(mn^2)$ | $\mathcal{O}(mn)$ | $\mathcal{O}(mn)$ |
| EXPLAIN-EFFICIENCY | $\mathcal{O}(mn^2\log(mn^2))$ | $\mathcal{O}(mn^2)$ | $\mathcal{O}(mn^2)$ |
| EXPLAIN-SATISFACTION | $\mathcal{O}(mn)$ | $\mathcal{O}(mn)$ | $\mathcal{O}(mn)$ |
| FULL-PRECOMPUTATION-EXPLAIN | $\mathcal{O}(m^2n^2\log(mn^2))$ | $\mathcal{O}(m^2n^2)$ | $\mathcal{O}(mn^2)$ |
| PARTIAL-PRECOMPUTATION-EXPLAIN | $\mathcal{O}(m^2n^2\log(mn^2))$ | $\mathcal{O}(mn^2)$ | $\mathcal{O}(mn^2)$ |

Figure 1.1: Computational, memory and tractability complexity of algorithms using argumentation

| Algorithm | Computational | Memory | Tractability |
|---|---|---|---|
| NAIVE-EXPLAIN-FEASIBILITY | $\mathcal{O}(mn)$ | $\mathcal{O}(mn)$ | $\mathcal{O}(mn)$ |
| NAIVE-EXPLAIN-EFFICIENCY | $\mathcal{O}(mn^2\log(mn^2))$ | $\mathcal{O}(mn^2)$ | $\mathcal{O}(mn^2)$ |
| NAIVE-EXPLAIN-SATISFACTION | $\mathcal{O}(mn)$ | $\mathcal{O}(mn)$ | $\mathcal{O}(mn)$ |
| NAIVE-EXPLAIN | $\mathcal{O}(mn^2\log(mn^2))$ | $\mathcal{O}(mn^2)$ | $\mathcal{O}(mn^2)$ |

Figure 1.2: Computational, memory and tractability complexity of algorithms without using argumentation

Using an naive explanation approach only improves the computational complexity of verifying the feasibility property.

## 1.3 Practical Results

I will be able to demonstrate the tool, and its functionality. This will be reflected in the report through a user documentation guide.

### 1.3.1 Full versus Partial Precomputation

We will compare two algorithmic approaches to AAF construction with the naive approach without argumentation. The algorithms are implemented, then profiled for elapsed time and maximum allocated memory. The time and memory are measured with the Python's `cProfile` and `memory-profiler` modules respectively. The tool was executed on Department of Computing's virtual machines, with the specification of dual-core CPU at 2GHz with 2GiB RAM.

Time comparison:

- Elapsed time measurements are noisy.
- For less than 100 jobs, all approaches have approximately equal timings.

- From profiling, the tool takes 0.4 seconds on average to startup.
- Partial precomputation is 3% faster than full precomputation on average, excluding startup time.
- Naive is 18% faster than partial precomputation on average, excluding startup time.
- Both graphs hints at quadratic complexity.

Memory comparison:

- For less than 40 jobs, all approaches have approximately equal memory usage.
- From profiling, the tool uses 52MiB on average to startup.
- For a large number of jobs, partial-precomputation is 7 times more efficient than full precomputation excluding startup memory.
- Naive and partial-precomputation have less than 1% memory usage difference.
- Both graphs hints at quadratic complexity.

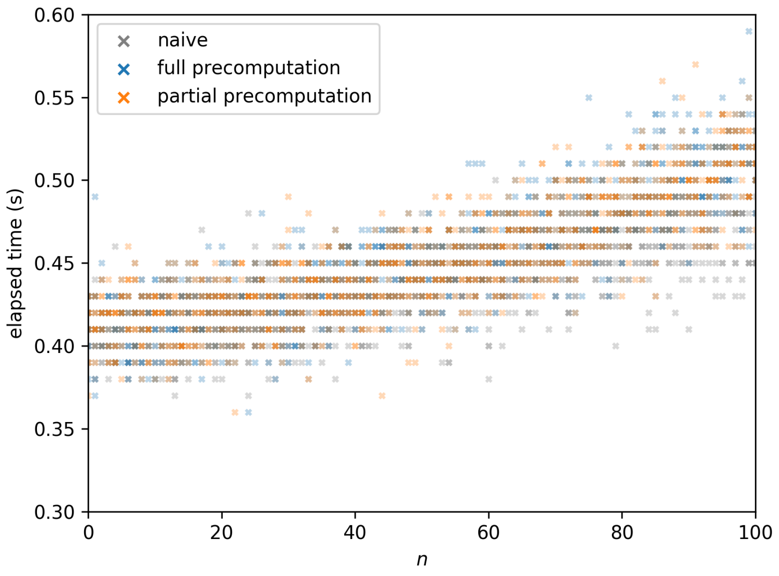

Figure 1.3: Elapsed time comparison where $m = 10$ and $0 \leq n \leq 100$ with 1000 samples.

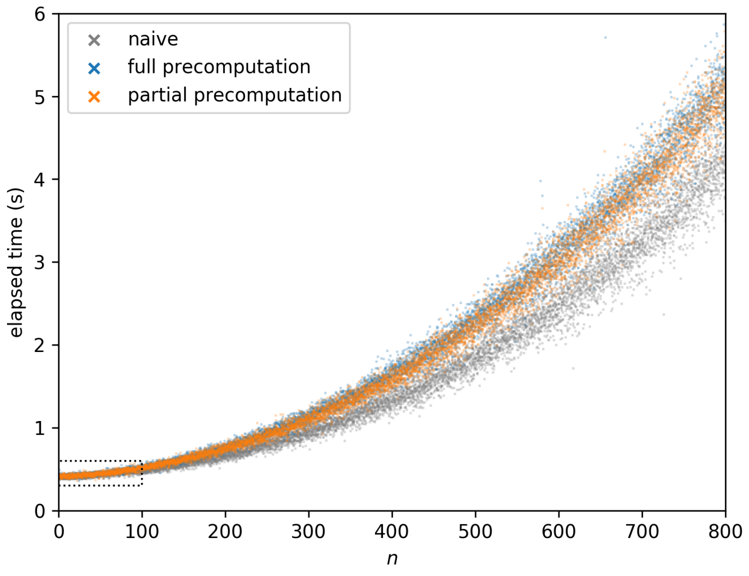

Figure 1.4: Elapsed time comparison where $m = 10$ and $0 \leq n \leq 800$ with 8000 samples.

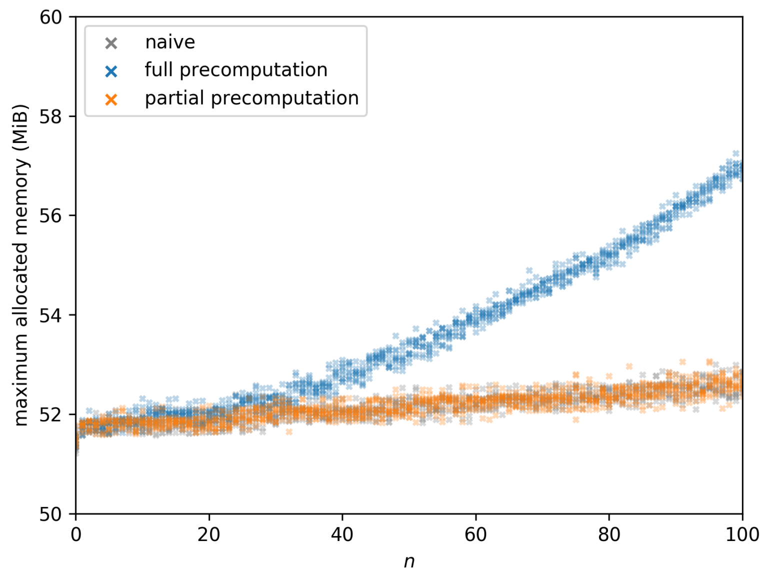

Figure 1.5: Maximum allocated memory comparison where $m = 10$ and $0 \leq n \leq 100$ with 1000 samples.

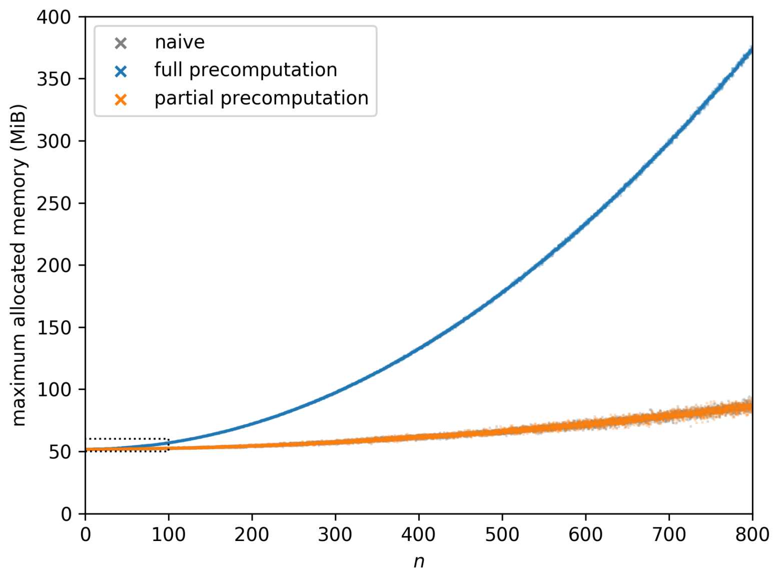

Figure 1.6: Maximum allocated memory comparison where $m = 10$ and $0 \leq n \leq 800$ with 8000 samples.

## 1.4 Feedback

During development of the tool, I asked my colleagues and friends for advice about the usability of my tool.

- In early versions of the tool, the GUI did not arrange textboxes and buttons into groups such as problem, schedule and explanation. Instead, all buttons and their functionality were exposed as an menu. This was confusing to new users because it was not evident which textboxes were input or output.
- The paper [1] represented machine job pair assignments as a pairs of integer indices. The tool internally uses indices to compute explanations, however exposing both machines and job as integer indices was confusing. Therefore, jobs are now represented alphabetically.

# Bibliography

[1] K. Cyras, D. Letsios, R. Misener, and F. Toni, *Argumentation for Explainable Scheduling.* 2018.