Chapter 1

Conclusion

1.1 Limitations

Argumentation interfaces between optimisation solvers and explanation generators. By computing satisfaction of extensions, such as stability, solvers are exposed as white-box algorithms, transforming argumentation semantics into human-understandable explanations. However, this application of argumentation is flawed in practice.

- **Memory performance:** Abstract argumentation operates on directed graphs, whose data structure storage suffers from quadratic space complexity. Using the direct implementation of the paper [1], namely, the full-precomputation approach, at 256 machines and jobs, each framework requires 4GiB of memory. To solve this scalability issue, the tool instead operates on partitioned directed graphs, resulting in optimal linear space complexity. However, optimal complexity does not yield optimal results, as at best, the partial-precomputation approach uses at least same amount of memory as the non-argumentative approach, from our empirical results.
- **Computational performance:** In order for argumentation semantics to correspond to schedule properties, such as feasibility, extensions must be global to capture the space of the problem's linear constraints. Local extensions such as conflict-freeness and admissibility are insufficient to model makespan schedules. Hence, we use the stability extension, which has quadratic computational complexity. This is inefficient compared to an non-argumentative approach, where makespan feasibility can be computed in linear complexity. Therefore, any argumentation approach is less scalable than an non-argumentative approach.
- **Abstracted interface:** For argumentation to be exploited as a white-box, users should interact with the underlying argumentation. This is possible for small schedules where the number of machine job pairs and their attacks are limited. But for realistic schedule sizes of at least tens of machines and jobs, argumentation frameworks become too busy to be visualised clearly. As such, the argumentative interface of the tool is abstracted, as a black-box, where users of the tool may ignore the underlying argumentation semantics.

- **Functionality equivalence:** In a black-box comparison of an argumentative versus a non-argumentative approach, both approaches result in identical explanations given identical problems and schedules. This is verified with the tool by comparing results with and without the `--naive` flag over many test cases. In practical settings, users will favour using the non-argumentative approach for CPU and memory performance.
- **Implementation complexity:** To optimise the performance computational and memory of argumentative approaches, the source-code becomes more difficult to understand. This is highlighted that the argumentative approaches are written in approximately 300 lines of Python while the non-argumentative approach is written in 100 lines. In conjunction with the above functional equivalence statement, we can interpret the non-argumentative approach as a refactoring of the argumentative approaches.

While argumentation offers, soundness, completeness, polynomial tractable and computational complexity, a non-argumentative approach can offer these too.

1.2 Future Work

- **Space of stability-modellable linear programs:** We have shown that makespan and interval scheduling are stability-modellable. An interesting direction would be to explore the space of linear programming problems that are stability-modellable, or more generally, extension-modellable.
- **Implementation of interval scheduling:** We have shown that interval scheduling feasibility is stability-modellable. Due to the time constraints of this project, we have not been able to integrate makespan and interval scheduling into a single graphical tool.
- **Web GUI:** The Python GUI is clear for users from an academic environment. But in comparison with commercial tools, their interface allows users to reschedule by intuitive drag-drop interactions. Although this could have been implemented, writing a full software package is an unlikely outcome of an university project.

1.3 Summary

Argumentation with scheduling is
practically possible but not practically feasible.

Bibliography

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