Explaining Infeasibility in the Test Laboratory Scheduling Problem

Aida Aliu, Florian Mischek, and Nysret Musliu

DBAI, TU Wien, Vienna, Austria
{aida.aliu, florian.mischek, nysret.musliu

Keywords: Infeasibility Explanations, Test Laboratory Scheduling Problem, Counterfactual Explanations, Constraint Programming

1 Introduction

Scheduling problems typically consist of numerous constraints that are often conflicting. This may prevent scheduling systems from generating feasible solutions, resulting in scheduling infeasibility. Scheduling infeasibility in automated scheduling systems refers to a situation in which the system cannot generate a valid schedule that satisfies all specified constraints and requirements. Several factors can contribute to scheduling infeasibility, including resource constraints, temporal constraints, complexity of constraints, and incomplete or incorrect input data

The lack of explanations for failures to generate feasible solutions undermines the primary purpose of automated scheduling systems designed to support human users. In such cases, users are left with no solution and have to manually check for reasons behind the infeasibility, which can be very time inefficient. Therefore, a vital challenge of these systems is generating reasons that explain why the system cannot find a feasible solution under certain specifications.

We investigate generating infeasibility explanations for a specific scheduling problem, namely the Test Laboratory Scheduling Problem (TLSP), introduced by [6]. The TLSP is an NP-complete scheduling problem, a real-world scheduling problem inspired by the operational needs of industrial test laboratories. It extends the classical Resource-Constrained Project Scheduling Problem (RCPSP) [2] by introducing additional complexity through task grouping, diverse resource types, and real-world constraints like employee qualifications and equipment availability. Solvers for the TLSP have already been successfully deployed in real-world industrial labs. However, in practice, users encountered challenges due to the lack of explainability when schedules turned out to be infeasible, making it difficult to understand and resolve conflicts.

By adapting existing approaches from the literature [5, 4, 3], we developed an infeasibility explainer for the TLSP, which to the best of our knowledge, is the first explainer specifically designed for this problem. Our approach integrates two distinct explainers, each supporting different types of explanations. These explainers are designed to generate human-readable outputs that effectively identify the constraints causing infeasibility while also providing valuable

insights for restoring feasibility. The first explainer is based on conflict detection, utilizing minimal unsatisfiable sets (MUS) of constraints and maximal satisfiable sets (MSS) to describe the source of infeasibility. Restoring feasibility is achieved through minimal correction sets (MCS). The second explainer supports counterfactual explanations, suggesting minimal changes required to restore feasibility, and, building on an existing approach, introduces novel weighting and blocking techniques. The explanations were collaboratively constructed with a real-world TLSP domain experts, incorporating their insights on common sources of conflicts.

Our approach was implemented using a tailored TLSP constraint programming model in OR-Tools. Additionally, we developed a configurable interface that improves usability by allowing users to configure the explainer's settings and run multiple configurations. This practical solution simplifies the diagnosis of infeasibility and provides suggestions for restoring feasibility, significantly reducing the need for time-consuming trial-and-error methods.

We evaluated the explainer on real-world settings, assessing its efficacy in identifying common sources of infeasibility and evaluating its utility in practical settings. The experiments conducted included both qualitative analysis and efficiency assessments. We demonstrated various use cases, presenting the types of explanations generated and evaluating their effectiveness and computational performance. We refer the reader to [1] for extensive experimental evaluation and details on the explanations.

The infeasibility explainer has been successfully deployed in an industrial setting, where it effectively aids in resolving real-world scheduling infeasibilities.

References

1. Aliu, A.: Investigating explanations of infeasibility for test laboratory scheduling problems. Master Thesis, TU Wien (2025), https://doi.org/10.34726/hss.2025.115143
2. Brucker, P., Drexl, A., Möhring, R.H., Neumann, K., Pesch, E.: Resource-constrained project scheduling: Notation, classification, models, and methods. Eur. J. Oper. Res. **112**(1), 3–41 (1999), https://doi.org/10.1016/S0377-2217(98)00204-5
3. Gupta, S.D., Genc, B., O'Sullivan, B.: Finding counterfactual explanations through constraint relaxations. CoRR **abs/2204.03429** (2022), https://doi.org/10.48550/arXiv.2204.03429
4. Leo, K., Tack, G.: Debugging unsatisfiable constraint models. In: CPAIOR 2017. LNCS, vol. 10335, pp. 77–93. Springer (2017), https://doi.org/10.1007/978-3-319-59776-8_7
5. Liffiton, M.H., Sakallah, K.A.: Algorithms for computing minimal unsatisfiable subsets of constraints. J. Autom. Reason. **40**(1), 1–33 (2008), https://doi.org/10.1007/s10817-007-9084-z
6. Mischek, F., Musliu, N.: A local search framework for industrial test laboratory scheduling. Ann. Oper. Res. **302**(2), 533–562 (2021), https://doi.org/10.1007/s10479-021-04007-1