Cable Tree Wiring - Benchmarking Solvers on a Real-World Scheduling Problem with a Variety of Precedence Constraints (Extended Abstract)[⋆]

Jana Koehler[1,2], Josef Bürgler[3], Urs Fontana[3], Etienne Fux[3], Florian Herzog[3], Marc Pouly[3], Sophia Saller[2], Anastasia Salyaeva[2], Peter Scheiblechner[3], and Kai Waelti[3]

[1] Universität des Saarlandes, Saarland Informatics Campus, Saarbrücken, Germany
[2] DFKI, Saarland Informatics Campus, Saarbrücken, Germany
[3] Lucerne University of Applied Sciences and Arts, Lucerne, Switzerland
`firstname.lastname@dfki.de / firstname.lastname@hslu.ch`

The Cable Tree Wiring Problem (CTW) is studied and formalized in the paper corresponding to this abstract. The CTW problem addresses the task of finding a valid plugging sequence for the cables in a cable tree into their corresponding cavities given atomic, soft atomic and disjunctive precedence constraints. Cable trees are used in industrial products to transmit energy and information between different product parts. To this date, they are mostly assembled by humans and only few automated manufacturing solutions exist using complex robotic machines. For these machines, the wiring plan has to be translated into a wiring sequence of cable plugging operations to be followed by the machine, which is the origin of the CTW problem. The CTW problem can be modeled as a traveling salesman problem with atomic, soft atomic, and disjunctive precedence constraints as well as tour-dependent edge costs. We prove the NP-hardness of this problem by a reduction from the Maximum Acyclic Subgraph problem. The proof heavily relies on the presence of soft atomic constraints, however we conjecture that the CTW problem is NP-hard even with atomic constraints only.

The Modeling of the CTW problem closely follows the formalization of the problem to provide a base line for an empirical study with various solvers. Our models are original work by the authors and are based on a so-called quadratic permutation representation of the TSP as described in [4]. As our objective function, we define the weighted sum of four criteria, where the weights are chosen to ensure that solvers prioritize optimizing for the criteria based on a fixed ranking order. First, a non-dual model **M** and a dual model **DM** were written in the OPL language [6]. In the **M** model, an array decision variable *position_for_cavity (pfc)* is introduced to represent the permutation sequence. This array uses the cavity number into which a cable end is to be inserted as index and stores the position in the plugging sequence as value. An *allDifferent* constraint is added for the *pfc* permutation sequence. The constraint model **DM** follows the approach from [5] and uses two permutation sequences *cavity_for_position (cfp)* and *position_for_cavity (pfc)*. Whereas the *pfc* permutation uses the cavity number as

[⋆] The paper corresponding to this extended abstract has been accepted to appear in the Springer Constraints Journal.

index and stores the position as value, the *cfp* permutation assigns cavity identifiers to positions. An additional array decision variable *cfp* is thus added to the model. The dual permutation representations are linked via a channeling constraint using the built-in *inverse* constraint in OPL. Furthermore, a (now) redundant *allDifferent* constraint is added for both permutation sequences. In both cases, the atomic, soft atomic and disjunctive precedence constraints are then expressed as constraints over the *pfc* array. For each solver, variants of the models **M** and **DM** are created in different languages as there is no single modeling language which all solvers would support. For the tested MIP solvers, am additional model is developed which makes use of a big-M reformulation for the disjunctive constraints [8].

Solver-specific model variants in different languages are kept in close similarity and not further tuned to achieve the best possible model for a specific solver. As such, the experiments described in the paper provide a base line for the comparison of different solvers on the CTW problem, but they do not generalize to wider conclusions about the scalability or performance of a solver beyond the specific model or CTW problem.

The CTW Benchmark Set comprises 205 real-world and 73 artificial instances of cable tree wiring problems. The complete benchmark set with all models and instance data is available on github (https://github.com/kw90/ctw_toolchain) and was included in the MiniZinc challenge 2020. Each instance is defined by the number of cable ends k which need to be plugged, the number of two-sided cables b, and its constraint sets. The benchmark set contains 20 instances with a permutation length smaller than 10, 239 instances with a permutation length between 10 and 100, and 19 instances with a length of over 100 and up to 198. The length of real-world instances mostly ranges between 20 and 50 with an average permutation length of 43.

The Benchmarking illustrated a varying performance of the tested state-of-the-art constraint programming (CP) [1,6,7], optimization modulo theories (OMT) [2,9], and mixed-integer programming (MIP) [3,6] solvers on the CTW benchmark set with a 5 minute time limit. Since no one tool supports all these different solvers, an elaborate tool chain supporting the conversation of models and data was developed by the authors and is made available for reuse via the github repository. In our experiments the IBM Cplex CP and the Google OR-Tools CP-SAT solver showed impressive results, with Cplex being the only tested solver to find solutions for each instance in the benchmark set and OR-Tools finding more optimal solutions than any of the other solvers we tested. In general, the CP solvers managed to solve more instances and solved these faster than the tested MIP solvers. The MIP solvers, however, outperformed the tested OMT solvers on our benchmark set. Selected experiments to further tune the CP solvers are also included in the paper, but we believe that the future will be in automatic, rather than human-provided search strategy selection. In addition, we are convinced that rewriting models has some more potential, in particular, for improving the performance of MIP solvers. It was furthermore very interesting to compare the performance of the non-dual with that of the dual model on the different solvers. It appears that modern CP-SAT solvers are less sensitive to duality and redundant constraints in models.. As an example, for Chuffed, using the non-dual or the dual models makes no difference at all — it solves exactly the same instances in the subsets of optimal or suboptimal solutions.

References

1. Chu, G., Stuckey, P.J., Schutt, A., Ehlers, T., Gange, G., Francis, K.: Chuffed: A lazy clause solver, https://github.com/chuffed/chuffed
2. De Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: International conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer (2008)
3. Gurobi. http://www.gurobi.com/
4. Gutin, G., Punnen, A.P. (eds.): The Traveling Salesman Problem and its Variations. Springer (2007)
5. Hnich, B., Smith, B.M., Walsh, T.: Dual modelling of permutation and injection problems. JAIR **21**, 357–391 (2004)
6. IBM: Cplex. https://www.ibm.com/products/ilog-cplex-optimization-studio/
7. Google OR-tools. https://developers.google.com/optimization/
8. Ruiz, J.P., Grossmann, I.E.: Global optimization of non-convex generalized disjunctive programs: a review on reformulations and relaxation techniques. Journal of Global Optimization **67**(1), 43–58 (2017)
9. Sebastiani, R., Trentin, P.: Optimathsat: A tool for optimization modulo theories. In: Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I. pp. 447–454 (2015)