CP Solver Design for Maximum CPU Utilization

Petr Vilím
ScheduleOpt, Nový Knín, Czech Republic

Abstract

In this talk, I explain how to improve the performance of a solver without focusing on algorithms, search, propagation or parallelism. Performance is achieved instead with better CPU utilization, efficient code and more precise design of the solver itself.

In the words of Fedor G. Pikus [1], the time of "performance taking care of itself" is over. In today's hardware the number of cores is increasing while the CPU clock speed has reached a plateau. Main memory access is slow in comparison to the CPU. And despite multiple memory cache levels, the CPU can easily become idle waiting for data from the memory, slowing down the computation considerably. Unfortunately, those trends are probably not going to change in the near future.

For those reasons we are witnessing revived interest in efficient code and performance-centered software design, especially in areas where the performance is critical: computer games, compilers, internet browsers, language interpreters (e.g. JavaScript or Python), etc.

The good news is that many of the tricks used in the above-mentioned areas, can be used in constraint programming as well. The bad news is that the performance has to be taken into account from the very beginning of the design. It is not possible to add it easily later. Sometimes, better performance can be achieved only by radical shifts in the design such as from object-oriented to data-oriented programming.

The design of a CP solver is not an exception in this regard. Without the efficient core of the CP solver, it is not possible to write truly efficient propagation or search algorithms. On the other hand, all algorithms in the solver must take the design of the solver into account and leverage it.

In this talk, I will describe what I consider the most important aspects of the design of *ScheduleOpt Optal* solver. I will concentrate on the performance, but I will also mention other aspects such as ease of use, maintainability, and testing.

2012 ACM Subject Classification Mathematics of computing → Solvers; Theory of computation → Constraint and logic programming

Keywords and phrases Constraint Programming, Software Design, Efficient Code

Digital Object Identifier 10.4230/LIPIcs.CP.2023.5

Category Invited Talk

References

1 F.G. Pikus. *The Art of Writing Efficient Programs: An advanced programmer's guide to efficient hardware utilization and compiler optimizations using C++ examples.* Packt Publishing, 2021.