Paracooba Enters SAT Competition 2022

Maximilian Levi Heisinger
Institute for Symbolic Artificial Intelligence
Johannes Kepler University Linz
Linz, Austria
maximilian.heisinger@jku.at

PARACOOBA is a parallel, distributed, Cube-and-Conquer (CnC) SAT solver based on KISSAT [1] and CADICAL [2]. It tries to split problems into sub-problems by recursively setting variables to concrete values, i.e. applying assumptions to the original formula. These sub-problems are then solved with the incremental version of CADICAL, combined with an (optional) timer to stop the solving process and re-split the formula again. Next to this parallel solving process, an instance of KISSAT is also running sequentially to stop bad splits from having too large impacts. The PARACOOBA solver can be found on GitHub using the URL below.

github.com/maximaximal/paracooba

I. FORMULA SPLITTING

Splitting formulas is based on our implementation of tree-based lookahead [3] part of CADICAL. If the splitting with lookahead takes too long, we fall back to the number of occurrences of variables. The chosen variable x is then deemed to be the most decicive one and used to split the formula ϕ into $\phi \wedge \neg x$ and $\phi \wedge x$. Now, if both sub-formulas are unsatisfiable, the whole formula is also unsatisfiable. If one is satisfiable, the whole formula is also satisfiable with the same assignment. This process is repeated until a predefined cube-tree-depth is reached, which defaults to saturate the locally available cores. Sub-problems usually vary greatly in their hardness and are then solved in individual solver threads or offloaded to other worker nodes in the network.

II. COMMUNICATION AND TASK SCHEDULING

The communication between nodes uses a custom binary protocol transported via TCP. After starting, a fully interconnected network of workers is created. Every worker receives utilization information of all other worker nodes and is thus able to decide locally, whether to offload work from the local queue to other nodes in the network. The dynamic offloading of tasks ensures that the highly different task hardness is mitigated by always saturating as many cores as possible.

To start, paracooba does not need the addresses of it's workers. The main node can start on it's own and either immediately start solving (if local worker threads were enabled) or idle until worker nodes connect to it. The main node listens on a port for incoming connections from workers (both the port and listen address are configurable). After connecting, workers receive all other known peers from the peer they connected to, in turn forming the fully connected network explained above. Workers may solve multiple problems at once, problems sharing the available worker threads on a given node.

III. EXTENDING PARACOOBA

PARACOOBA is built to be highly modular, also offering a QBF solving module. The software is MIT-licensed and can be found on GitHub. More detauls about the implementation, the employed algorithms and the expandability can be found in [4] and [5]. Possible extension points are the communication stack, the solver module, the offloading / scheduling mechanism, and the local task runner. Extensions may be loaded from provided shared object files at start-up. Automated testing of modules is done using integration tests, also provided in the repository.

REFERENCES

[1] A. Biere, K. Frazekas, M. Fleury, M. Heisinger, CaDiCaL, Kissat, Paracooba,
[2] A. Biere, CaDiCaL at the SAT Race 2019, SAT Race 2019
[3] M. Heule, M. Järvisalo, A. Biere, Revisiting hyper binary resolution, International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Springer, Berlin, Heidelberg, 2013
[4] M. Heisinger, M. Fleury, A. Biere, Distributed Cube and Conquer with Paracooba, SAT2020
[5] M. Heisinger, Distributed SAT & QBF Solving: The Paracooba Framework, 2021 Plingeling and Treengeling Entering the SAT Competition 2020, SAT 2020