Overview

State-of-the-art algorithms for hard computational problems often expose many parameters that can be modified to improve empirical performance. However, manually exploring the resulting combinatorial space of parameter settings is tedious and tends to lead to unsatisfactory outcomes. Recently, automated approaches for solving this algorithm configuration problem have led to substantial improvements in the state-of-the-art for solving various problems.

Automated procedures for solving this algorithm configuration problem are useful in a variety of contexts. Their most prominent use case is to optimize parameters on a training set of instances from some application (“offline” as part of algorithm development) in order to improve performance when using the algorithm in practice (“online”). Algorithm configuration thus trades human time for machine time and automates a task that would be performed manually.

The algorithm configuration problem can be formally stated as follows: given a parameterized algorithm $A$ (the target algorithm), a set (or distribution) of problem instances $I$ and a cost metric $c$, find parameter settings of $A$ that minimize $c$ on $I$. The cost metric $c$ is often based on the runtime required to solve a problem instance, or, in the case of optimization problems, on the solution quality achieved within a given time budget.

$$
\theta^* \in \arg \min_{\theta \in \Theta} c(\theta)
$$

Tasks

Recently, we proposed sorting algorithms that are multi-level generalizations of the known algorithms sample sort and multiway mergesort. In particular, our sample sort scales up to $2^{15}$ MPI processes with outstanding performance in particular for medium sized inputs. Of course algorithm performance depends on system and instance features such as input size, number of cores used or even the hierarchy of the super computer. The main task of this thesis is challenging but also rewarding: integrate algorithm configuration tools to automatically select algorithm and parameters to get the best performance possible.

Requirements

- Interest in algorithms and data structures
- Excellent programming skills in C++ and experience with MPI