ETH zürich

Master Thesis Proposal

# Compiler Flag Optimization by Maximization of a Fourier-sparse Poset Function

Advisor: Chris Wendler
Professor: Markus Püschel

March 18, 2021

Given a compiler that can apply optimizations $N = \{1, \dots, n\}$ to a program, what is the most effective subset of optimizations and what is the best order to apply them? Optimizations might influence each other.

Motivated by our recent advances in poset signal processing we propose the following Ansatz:

1. Define a poset of optimizations,
2. learn a Fourier-sparse approximation of the poset function mapping each optimization $(x_1, x_2, \dots, x_\ell)$ with $x_1 \neq x_2 \neq \cdots \neq x_\ell$ to its score (e.g., speedup),
3. maximize the learned function to find a good optimization.

**Poset of optimizations** As a starting point we provide one construction for the poset of optimizations. Formally, we define an optimization by the pair $(A, a)$, where $A \subseteq N$ is a subset and $a \in \Pi(A)$[1] is a permutation of the elements of $A$. Let $P = \{(A, a) : A \subseteq N, a \in \Pi(A)\}$ denote the set of all possible optimizations. Now, we can define a partial order on the optimizations: $(A, a) \leq (B, b)$ iff $A \subseteq B$ and $inv(a) \subseteq inv(b)$, where $inv(a) = \{(i, j) \in A \times A : i < j \wedge a(i) > a(j)\}$ is the inversion set of $a$. Intuitively, the $(A, a) \leq (B, b)$ iff $b$ applies more optimization techniques (i.e., optimizations plus variations from the default order) than $a$.

[1] We denote the set of all permutations of the elements in $A$ by $\Pi(A)$.

**Reduction mapping and preimage** [1] outlines a construction of the reduction mapping and its preimage for posets $P$ with a minimal element ($(\emptyset, ())$ in our case) and a total order $\leq_T$. Luckily, we can define a total order: $(A, a) \leq_T (B, b)$ iff $A \leq_{lex} B$ and $inv(a) \leq_{lex} inv(b)$, where $\leq_{lex}$ denotes the lexicographic order on sets. Now the reduction mapping $f$ is simply defined by $f((A, a)) = (B, b)$, where $(B, b)$ is the smallest element (w.r.t. $\leq_T$) covered by $(A, a)$. Consequently, the preimage is

$$\begin{aligned} f^{-1}((B, b)) &= \{(A, a) \in P : f((A, a)) = (B, b)\} \\ &= \{(A, a) : (B, b) \text{ is the smallest element covered by } (A, a)\}. \end{aligned}$$

**Covers relation** An optimization $(A, a)$ is covered by $(B, b)$ iff either $A = B$ and there exists a transposition $(i, i+1)$ such that $(i, i+1)a = b$ or if $A \subset B$ with $|A| = |B| - 1$, $inv(a) \subseteq inv(b)$ and there is no $c \in \Pi(B)$ with $inv(a) \leq inv(c) \leq inv(b)$. For $B = A \cup \{x\}$, those $b$ are the ones agreeing with $a$, i.e.,

$$b = (a_1, \ldots, a_i, x, a_{i+1}, \ldots, a_{|A|}),$$

that introduce the least number of new inversions.

Thus, given an optimization $(A, a)$ we can compute the optimizations $b \in \Pi(A \cup \{x\})$ that cover $a$ in $O(n)$ by making use of the identity

$$|inv(a_1, \ldots, a_i, x, a_{i+1}, \ldots, a_{|A|})| = |\{a_j : j \leq i \wedge a_j > x\}| + |\{a_j : j > i \wedge a_j < x\}|,$$

where the right hand side can be computed for all values of $i$ in linear time.

**Your contribution** With the poset structure in place, our recent results for learning and maximizing Fourier-sparse poset functions can be used to find good program optimizations. The goals of this project are

1. the implementation of our recent poset function learning algorithm for the poset of optimizations,
2. the implementation of our recent poset function maximization algorithm for the poset of optimizations,
3. the evaluation of the implementations on real compiler flag optimization tasks.

**Deliverables**

*Final report:* The final report may be written in English or German. It must contain an abstract written in both English and German. It should include an introduction, an analysis of related work, and a complete documentation of all used software tools and mathematical derivations. Three copies of the final report must be delivered to the supervisor.

*Reproducible experimental setup:* Implementations, configuration scripts and instructions to reproduce the results reported in the thesis must be delivered in electronic form.

*Presentation:* The results of the thesis must be presented during a software-group seminar. The presentation is capped to 30 minutes and should give an overview as well as the most important details of the work.

**Contact** If you are interested in pursuing this master thesis, please contact wendlerc@ethz.ch or pueschel@ethz.ch.

# References

[1] S. Nowozin. *Learning with structured data: applications to computer vision.* PhD thesis.