EVALUATING STOKE

Przemysław Kowalewski[1] supervised by Soham Chakraboty[1] and Dennis Sprokholt[1]

[1]EEMCS, Delft University of Technology, The Netherlands

1. Problem Summary

- Superoptimizer is a program that given a function and a set of instructions of a processor it traverses through a space of programs that compute a given function and tries to find the shortest one[3].
- STOKE[4] introduces a **cost function** and then it uses **Markov Chain Monte Carlo sampler** to try to find the solution with the lowest cost value.
- The basic principle of this Superoptimizer is **correctness preservation** of the optimized program while **performance improvement** is desired but not required.

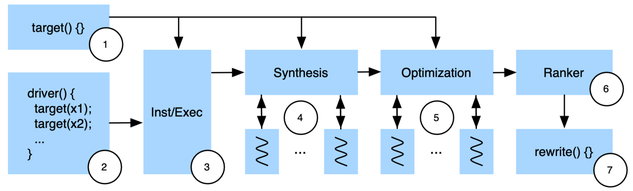

Figure 1. STOKE's pipeline[4]

2. Research goal

Authors in the original paper claim that can generate programs a few times faster than the compiler without any optimizations would. They also state that the new program will be **at least as efficient** as programs produced by gcc -O3 and in some cases **faster than expert handwritten assembly**. The goal of this research is to

- verify claims made by the Authors,
- identify classes of programs that STOKE may handle particularly well and any class of programs that stochastic optimization might not be able to handle.

3. Methodology

Experiments and benchmarks from the original paper have been repeated and results have been compared. New programs have been written to check how STOKE handle different program classes. Each measurement was done 10 times and average time and speed-up was calculated as a 95% confidence interval.

4. Experimental Setup

All optimizations have been performed on a personal computer not on computational cluster as in the original paper. The computer is MSI GS65 with i7-8750H 6 Core, 8GB RAM, Ubuntu 14.04 and GCC 4.9 installed.

References

[1] Christof Ebert, James Cain, Giuliano Antoniol, Steve Counsell, and Phillip Laplante. Cyclomatic complexity. *IEEE Software*, 33(6):27–29, 2016.

[2] Merrill M. Flood. The traveling-salesman problem. *Operations Research*, 4(1):61–75, 1956.

[3] Henry Massalin. Superoptimizer: A look at the smallest program. *SIGARCH Comput. Archit. News*, 15(5):122–126, oct 1987.

[4] Eric Schkufza, Rahul Sharma, and Alex Aiken. Stochastic superoptimization. *SIGPLAN Not.*, 48(4):305–316, mar 2013.

5. Original experiments verification

To verify the performance of programs optimized in the original paper[4] four programs from the original paper were optimized by configurations put on the STOKE's GitHub Repository and the execution times were compared.

Function name	gcc -O3	STOKE	Speed-up
p21	2.52	1.23	51%
p23	0.62	0.54	10%-14%
SAXPY	1.30	1.28	−1%-3%
Linked List	3.8	3.79	0%-1%

Figure 2. Results of the experiments

- The experiments confirmed the claim of the **performance improvement or matching programs produced by gcc -O3**. In case of p21 and p23 experiments a clear speed-up can be observed as STOKE managed to find more efficient solutions to the given problems.
- In case of SAXPY experiment STOKE managed to find a solution with greatly reduced number of CPU instructions, however speed-up achieved on this particular computer is basically non-existent compared to around 30% speed-up achieved in the original experiment.
- The experiment with program that traverses confirmed **the limitations of STOKE regarding the programs containing loops**. Despite that STOKE managed to find the solution performing comparably well to gcc -O3's.

6. Programs with high cyclomatic complexity

Programs with high cyclomatic complexity were tested. It is defined as a "metric measures the number of linearly independent paths through a piece of code"[1]. The aim of carrying out this experiment was to check if STOKE:

- can find ways to decrease numbers of conditional jumps,
- is able to get rid of code that will never be executed.

Function name	gcc -O3	STOKE	Speed-up
Program with nested if statements which can be simplified	0.74	0.67	−8%-−13%
Program with nested if statements containing unreachable branches	5.39	6.13	−13%-−15%

Figure 3. Results of the HCC experiments

STOKE managed to find solutions more efficient than a naive rewrite however their performance **didn't match the programs generated by gcc -O3**.

TU Delft

7. Programs accessing random memory regions

Programs that access random parts of memory multiple times in a random order were tested.

Function name	gcc -O3	STOKE	Speed-up
Program with random memory accesses to an array of arbitrary size	2.16	2.39	6%-13%
Program with random memory accesses to an array of size 10	1.78	1.86	−3%-−5%

Figure 4. Results of the RMA experiments

STOKE managed to find a successful rewrite however the result of the run time measurements turned out quite differently. This shows that **a slight difference in the program may cause STOKE find a completely different rewrite for them**.

8. Programs with high computational complexity

Programs with a significant **computational complexity** were tested. The goal of this experiment was to check if the claimed limitations of the program sequences containing loops still hold.

- First program solved **Traveling Salesman Problem**[2]. It is a $O((n-1)!)$ problem[2]. STOKE managed to find potentially better solutions but some errors had to occur in the optimization process as the **programs optimized by STOKE caused a Segmentation Error**.
- Two other experiments also gave **unsatisfactory results**. Despite trying multiple configurations of the search procedure it **wasn't able to find a solution**. Multiple cost functions, verification strategies, iteration timeouts, numbers of test cases were tried and all the search for better solution took combined around 24 hours.

9. Conclusions

- The experiments confirmed that **STOKE is able to produce programs at least as fast gcc -O3** in some cases.
- However it is not always in all cases, **STOKE's limitations when it comes to programs that contain loops** may cause a slow-down.
- It is able to **reduce numbers of multiple logical and mathematical operations**, decreasing number of processor cycles (e.g. p23 experiment)
- Struggles to find solutions or finds solutions slower than gcc -O3 in case of programs with loops
- Although some of the experiments conducted in this paper have not reduced positive results the other ones show **great potential that lies in Stochastic Optimization**.

10. Future Work

- STOKE is capable of finding really smart optimization as shown e.g. in p21 function experiment, however **finding optimal configurations of STOKE is challenging** and takes a lot of work. This process should be simplified.
- STOKE can also produce **programs that crash** as shown in the Travelling Salesman experiment. This needs prevention.