Auto-Tuning Libraries with MLKAPS

Lead Architect: Eric Petit[1]

Main Contributors: Mathys Jam[1*,2], Pablo Oliveira[2],

Greg Henry[1], Geoff Lowney[1], Suyash Bakshi[1], David Defour[3], William Jalby[2], Frank Schlimbach[1], Sohaib Ouzineb[1*]

[1]Intel-SEG-DSE, [2]University of Versailles, [3]University of Perpignan

*formerly

intel®

MLKAPS on MKL use-case

Context:
The AI new Deal for high performance software development

The new deal of machine learning

- LLM changed general perspective on ML
  - Increased acceptability e.g. HPC+AI, AI/ML in software development

- Where is this going?
  - Larger-scale project automation
  - Go beyond human capabilities on wider set of tasks
  - Machine learning keeps accelerating!

- Uncertainty in the output by design!
  - But humans are not perfect either…

*Times magazine

Long term What if…

- We can generate optimized code from specification?
- Tailor libraries to a specific application and cluster?
- Generate library optimized for custom design?
- Generate custom function optimized libraries?

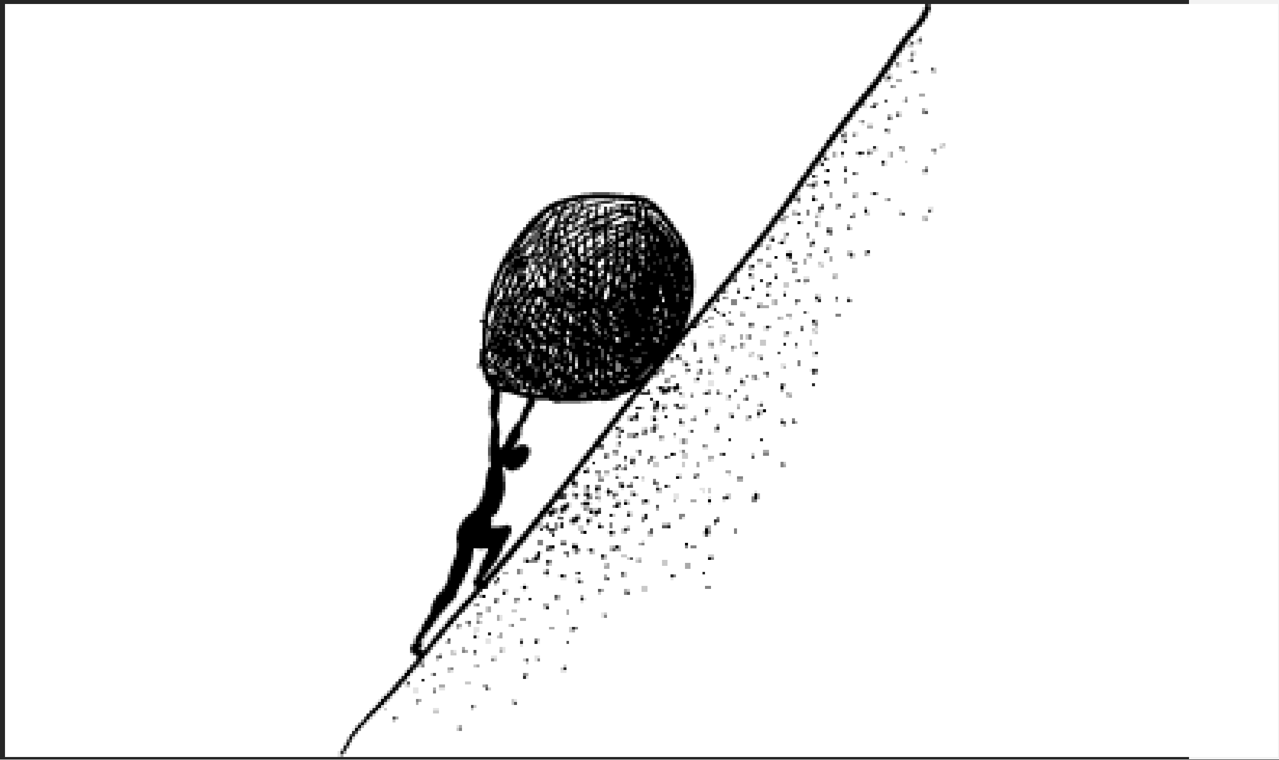
….

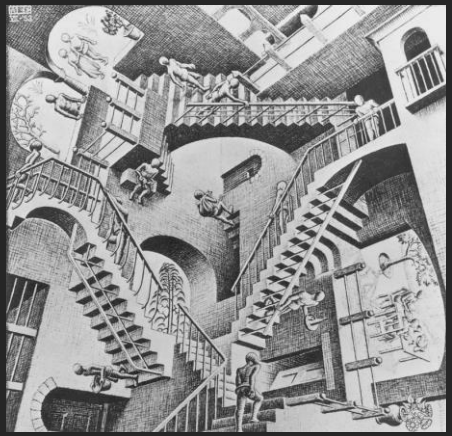
- These are way too costly to be generalized when human based
  - Only key domain (HPC, AI) or key large customers…
- But if AI can largely automatize the process, we can consider a completely different scale!
  - A new era of custom software democratization!

Unleashing performance tuning in software with ML

- Subtask: Hand tuning high-performance kernel
  - Tedious, error prone, biased
  - Perpetual

- Unfold productivity with AI/ML
  - What if tuning can go down from months to days to hours?
  - It is already proven to go beyond human capabilities
    - We aim to explore further
    - And generalize





MLKAPS overview

(one of) The MLKAPS project ultimate goal as a prompt

- "Here is the open source BLAS project, generate an optimized version for this new platform"
  - Given a description (source/doc), a target system, automatize as much as possible the process of getting highly optimized version

- Compete with ninja code
  1. Generate reference code
  2. Find and Generate optimized algorithms with design parameters
  3. Optimize for all potential input and generate runtime/decision tree
  4. Or single input super-optimizer
  5. Iterate from tools and users feedbacks

MLKAPS

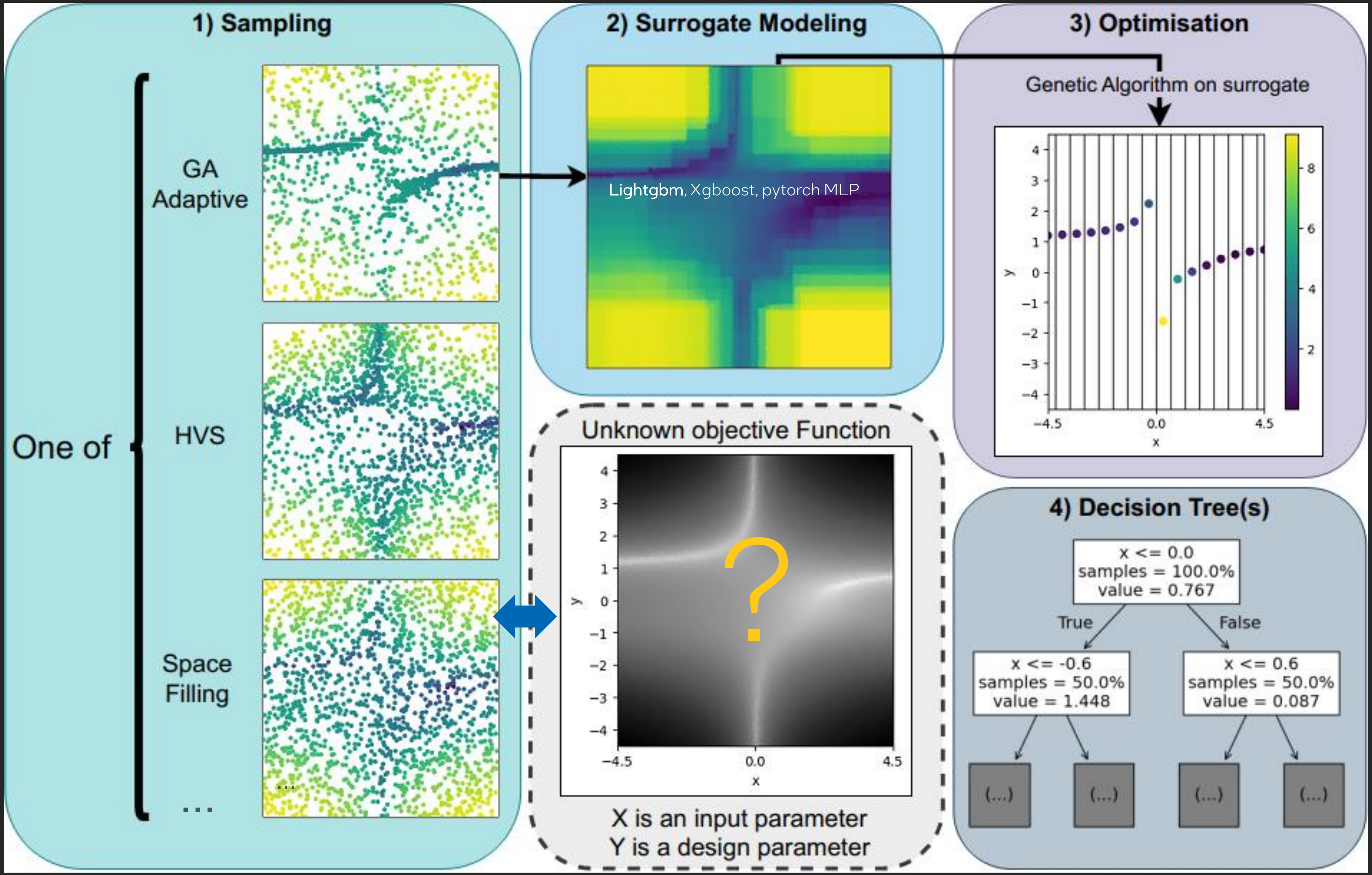
- Multi-objective modeling and optimization framework
  - Blackbox approach
  - Use a surrogate model to enable costly optimization phase
    - Sampling from the model has unsignificant cost
  - Optimize sampling to maximize information gain for the model
    - ‘Enough’ high quality information
    - Minimize number of sample required
  - Design parameters choice and code generation based on decision trees
    - Powerful model, easy to turn into code, cheap to evaluate
  - Based on opensource packages
    - We are releasing everything upstream
  - Used for hardware unit design, performance regression analysis, tuning libraries

MLKAPS Key design focus

- Scalability over complexity
  - Handle ‘large enough dimensionality’ to be useful
  - Handle large enough number of samples
  - Accuracy is scaling with the number of samples
- Separate input (I) and design (D) parameters
  - F(I)=D, find a ‘good’ F
  - MLKAPS predict best design as a function input parameters
  - If it was just about design parameters, plenty of existing optimizers
    - Mlkaps is wrapping some of them

Resources

- The code is published opensource (BSD3)
  - MLCGO/MLKAPS: MLKAPS: Machine Learning for Kernel Accuracy and Performance Studies
  - Test on examples
    - synthetic2D: (~2min) a dummy and fast example we will look today
    - openBlas: (~30min) tuning number of threads of outer product of openBlas
- Publication
  - MLKAPS: Machine Learning and Adaptive Sampling for HPC Kernel Auto-tuning
    Mathys Jam (LI-PaRAD, UVSQ), Eric Petit (intel), Pablo de Oliveira Castro (LI-PaRAD, UVSQ), David Defour (LAMPS, UPVD), Greg Henry(intel), William Jalby (LI-PaRAD, UVSQ)

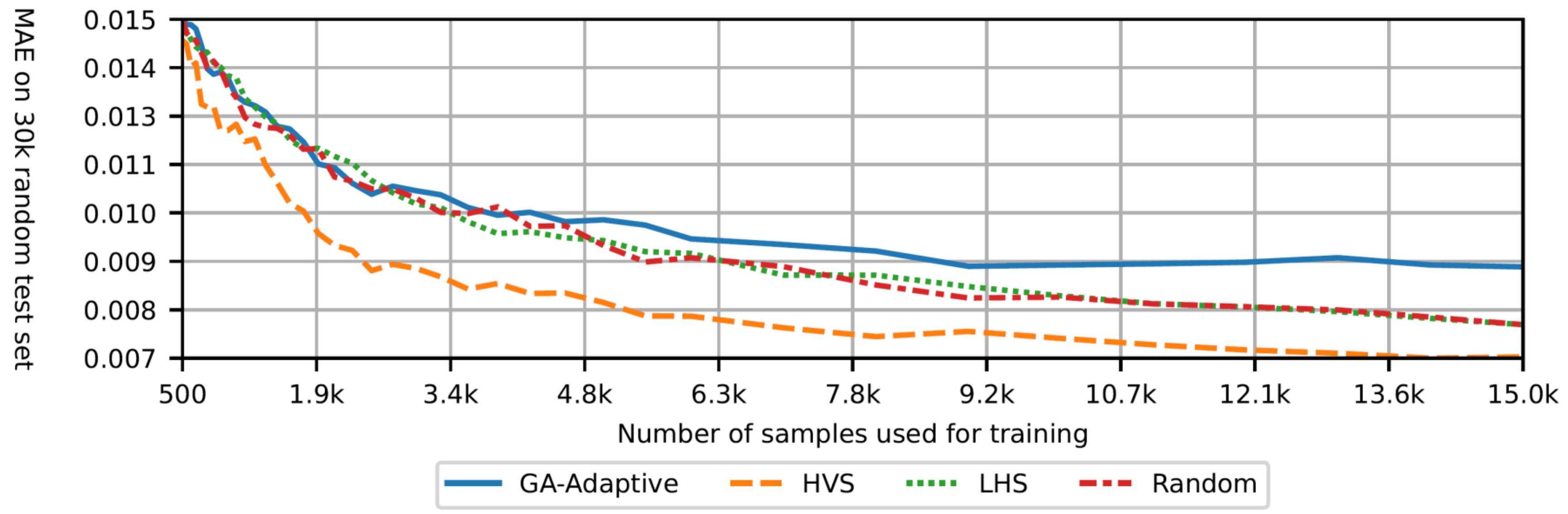


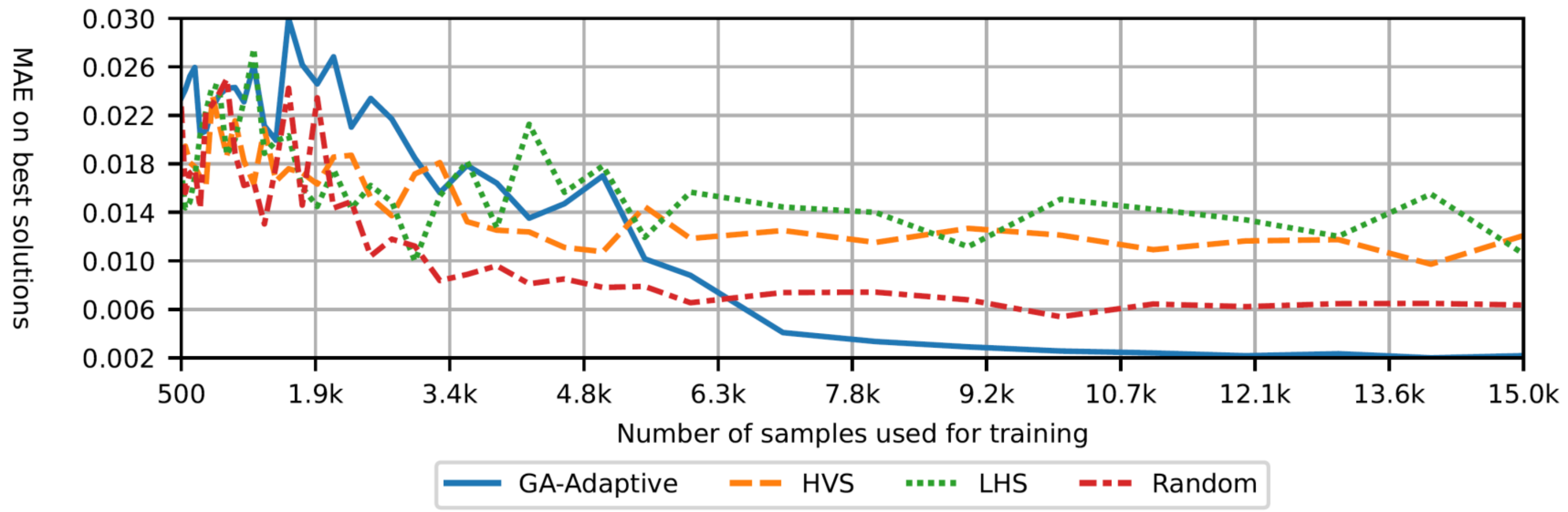
1) Sampling
One of
GA Adaptive
HVS
Space Filling
...
2) Surrogate Modeling
Lightgbm, Xgboost, pytorch MLP...
3) Optimisation
Genetic Algorithm on surrogate
Unknown objective Function
?
X is an input parameter
Y is a design parameter
4) Decision Tree(s)
x <= 0.0
samples = 100.0%
value = 0.767
True
False
x <= -0.6
samples = 50.0%
value = 1.448
x <= 0.6
samples = 50.0%
value = 0.087
(...)
(...)
(...)
(...)

About adaptive sampling

- Different potential bias to compare to random sampling:
  - Space filling: LHS
    - Make sure to maximize the exploration of each parameters
    - Bias toward exploring all value for each parameters: "a little bit of everything"
  - From ASK: HVS(r)
    - Sample based on variance estimator vs recursive tree region (valid) area
    - Bias toward places where "something is happening"
  - New: bias for optimization: GA-Adaptive
    - Model aim to be used for optimization, with clustering:
      - We need to have good knowledge of the 'good' solutions
    - After each iteration, build surrogate model and bias next samples toward region of predicted minima
    - Bias toward "be accurate where it matters"
  - Soon: true and scalable Bayesian for input selection

GA-adaptive improves model where it matters!





Modeling

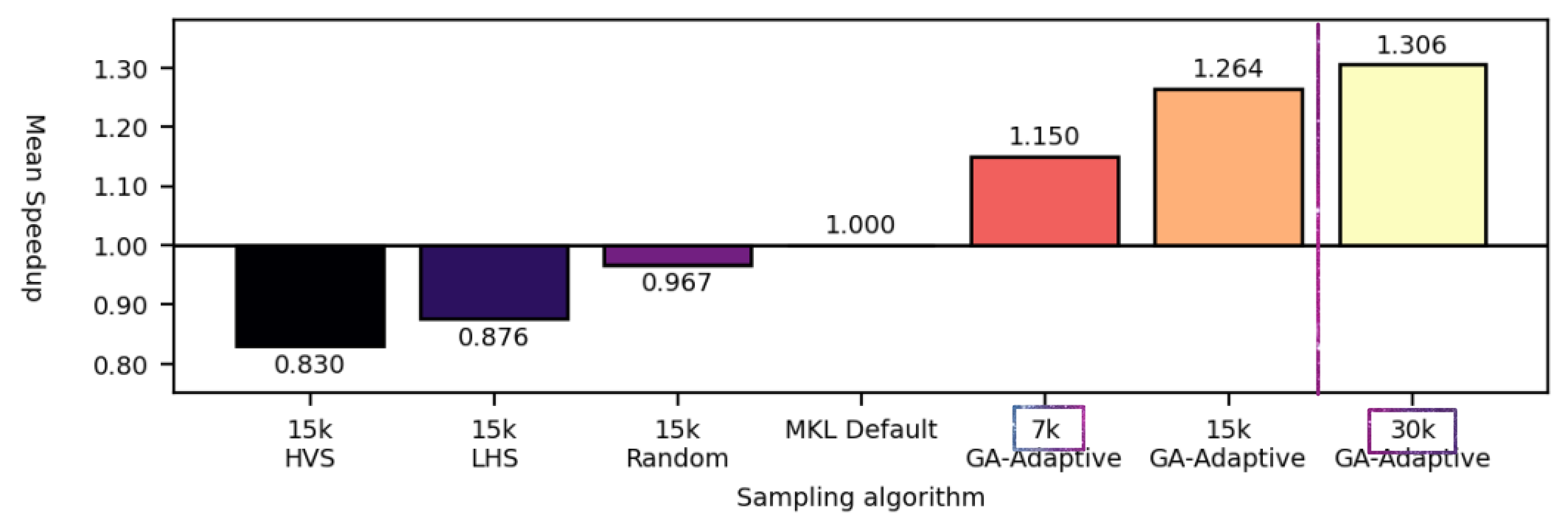
- Boosting trees
  - Lightgbm is currently our main model
  - Also support Xgboost
- Custom MLP-NN approach under experimentation
  - With a twist ;)
- Tuning model hyperparameters
  - We have a reasonable base configuration
  - Tuning it requires some expertise…
  - Optuna option to tune the model hyperparameters
    - But it can overdo it…. Overfitting

Optimization

- Currently preferred optimizer Pymoo NSGAII
  - Can use Optuna, random, or even exhaustive search

- Point selection can be grid based or random
  - More advanced algorithm currently in research phase
  - Need more => improve optimization cost
  - Or at least place them where it matters => adaptive on input

Information qualitydrastically impact optimization performance



oneMKL DGETRF as a baseline: expert hand-tuned version, beating it is the real deal!

Decision trees

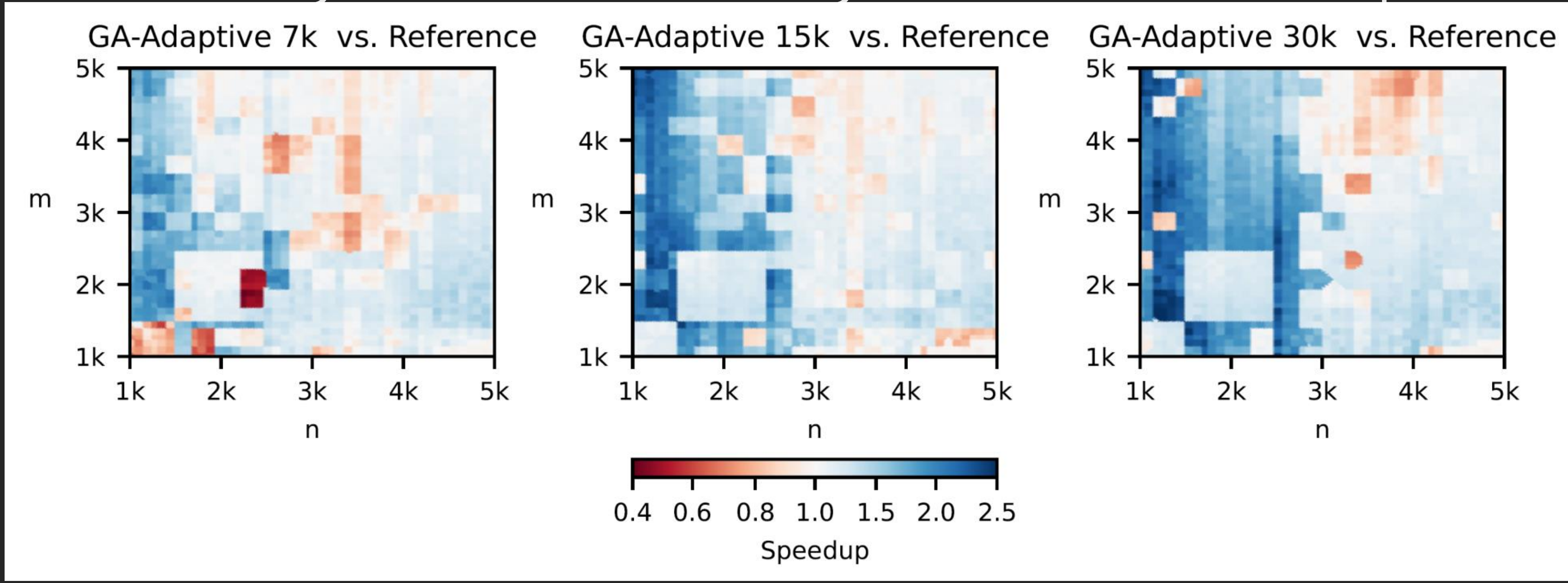
- Currently Integrated in MLKAPS
  - Regression trees from scikitlearn
    - One for each parameters
    - Very simple code and easily human readable
- Soon released (used in MKL and publication)
  - Expert trees
  - Idea:
    - Avoid regression compared to previous reference
    - Running multiple independent run in CI will add speedup overtime
  - Integrate reference implementation knowledge prior to build the tree and keep best of both world
  - Designed for MKL-like use case where a reference optimization exist

MKL Lapack autotuning case on LU and QR

Dgetrf (LU) and Dgeqrf(QR) case description

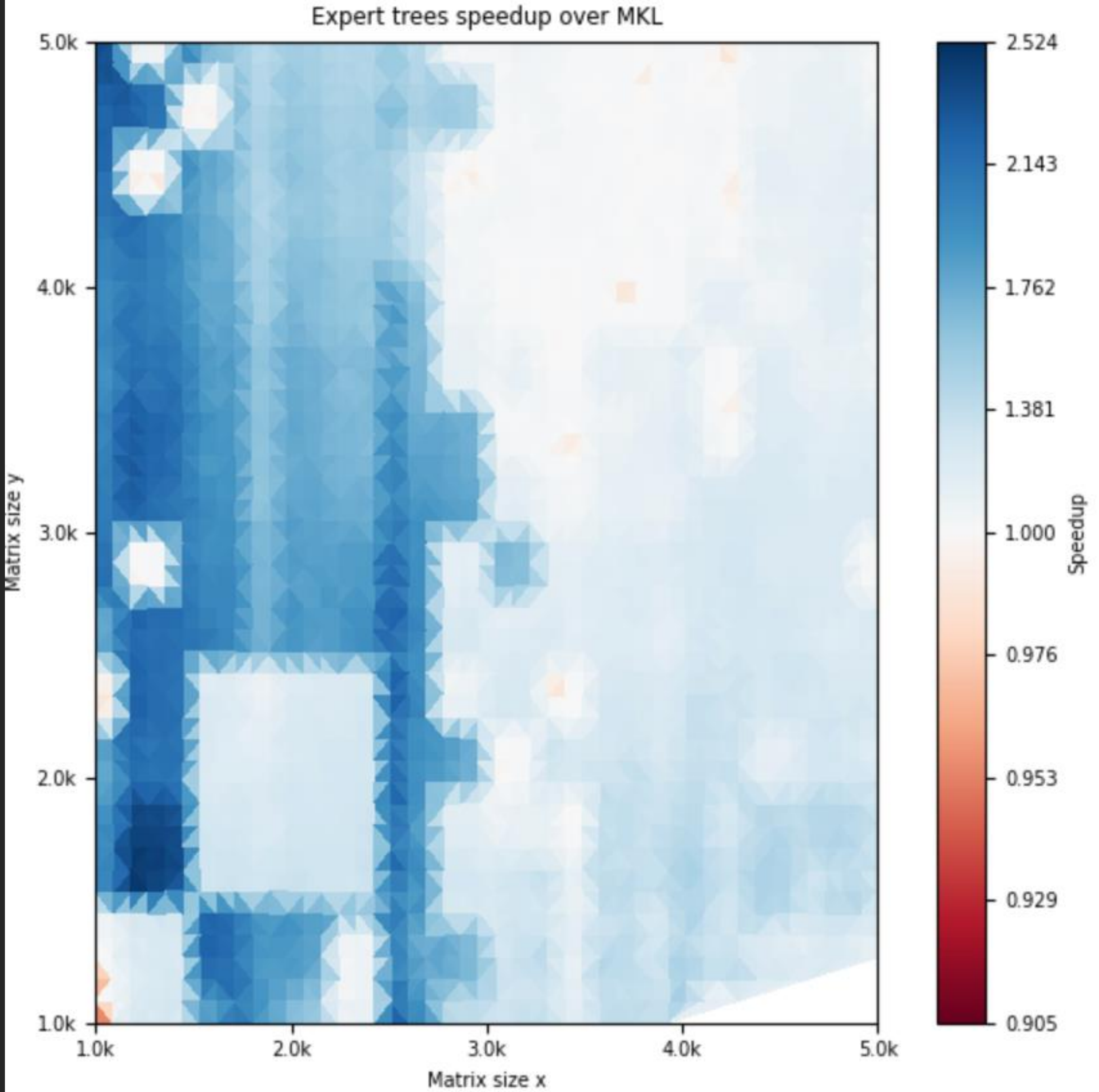
- 2 input parameters: M,N
  - Results based on [1000:5000]X[1000:5000] input intervals
  - New one on a much larger input domain not shown here
- 8 Design Parameters
  - And some constraints
- Results shown are 56 core runs

MLKAPS LU results with scikitlearn regressor trees
Scaling with increasing number of sample



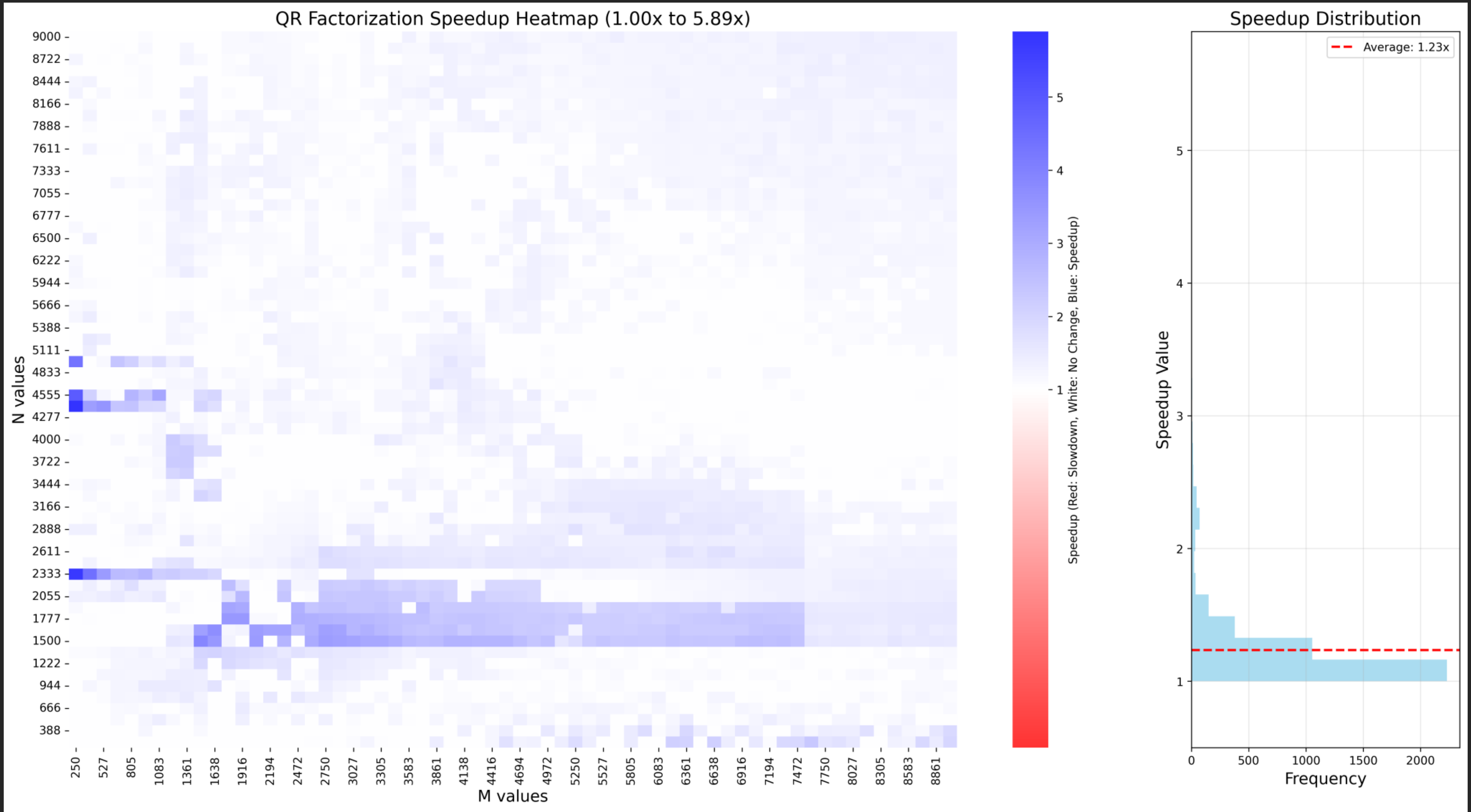
- GA-Adaptive 7k vs MKL: GA-Adaptive 7k is better on 0.77 of the input space, mean speedup of 1.18
  - GA-Adaptive 7k has an average speedup of 0.89 for regressions and 1.27 for progressions
- GA-Adaptive 15k vs MKL: GA-Adaptive 15k is better on 0.85 of the input space, mean speedup of 1.31
  - GA-Adaptive 15k has an average speedup of 0.947 for regressions and 1.38 for progressions
- GA-Adaptive 30 vs MKL: GA-Adaptive 30 is better on 0.86 of the input space, mean speedup of 1.39
  - GA-Adaptive 30 has an average speedup of 0.91 for regressions and 1.47 for progressions

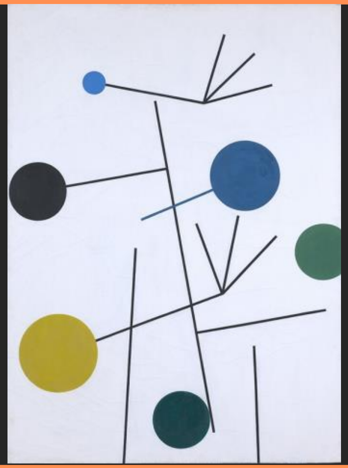
LU with expert trees, merging MKL and MLKAPS



- Expert trees is better on 0.99 of the input space, mean speedup of 1.46

e.g. QR on a larger domain, 64 threads, with expert trees





Conclusion and discussion

Some limitations

- What is a good range?
  - No extrapolation model in mlkaps
  - Very large sizes does not allow 'heavy' statistical methods
    - Inductive bias: search the steady state, gray box approach
    - Be wiser and cost aware when choosing a point to mitigate
  - Small sizes/dimensions often requires different code path
    - Search Tall and Skinny frontier
- When input space dimension increases,
  - Number of optimization point explodes
  - Random input exploration is not enough in sampling
  - Same as design space explosion
    - In progress:
      - adaptive selection of optimization point
      - Smarter choice of sample input coordinate

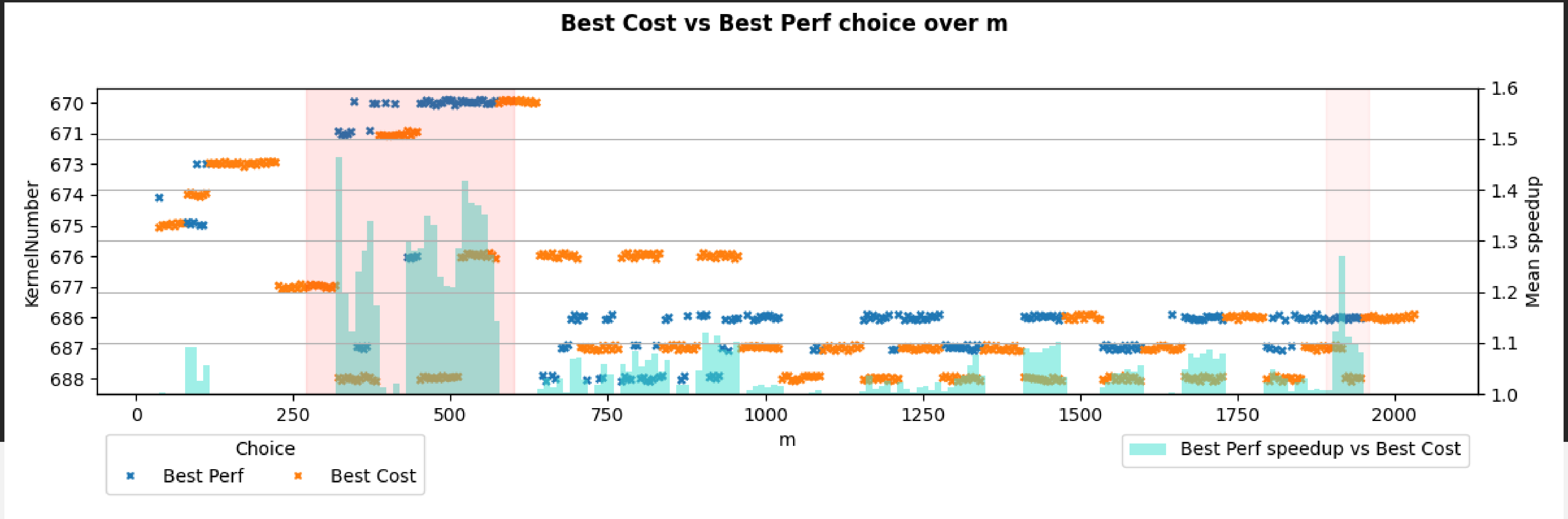
Some related ongoing and future work

- Improve MLKAPS software
  - Continuous integration updating the library seamlessly
  - Replace more and more tedious expert tuning in existing libraries
  - Checkpoint restart remote run about to be released (see public PR)
  - Next big thing will be full pythonic interface for custom worflows
- Improve MLKAPS methodology
  - Gray box approach: integrate more expert knowledge
    - Integrate/discover analytical models
    - Better sampling on input parameters and optimization point
  - Better/different models
    - DNN/MLP
  - Better decision trees
- Front-end research activities
  - LLMs
  - MLIR
- More use cases! (help needed)

intel®

GEMMs kernel on PVC

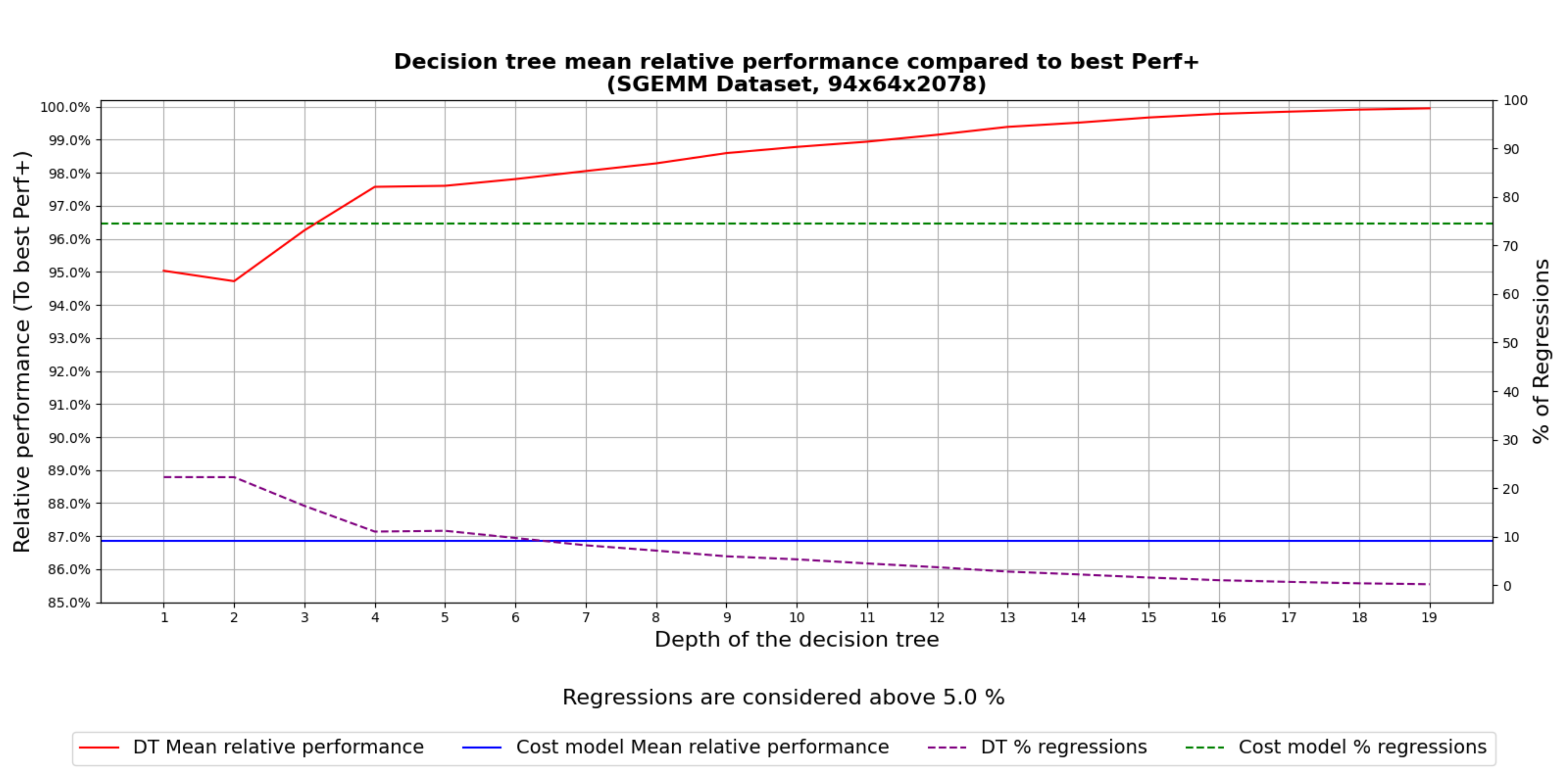
- GEMM = Matrix Multiply
- On PVC/iGPU, oneDNN/MKL uses a assembly-kernel generator
  - For each (m,n,k) find the best microkernel among 5 to 15 **preselected**
  - sgemm, dgemm…
  - And find best mapping function f(m.n.k)=kenel_ID
    - Currently a fitted analytical a cost model (current solution)
    - Can we do better?



Results on decision tree and regression

A lot remain to be done, but it is highly encouraging

- MLKAPS rightfully predict kernel that were not preselected for a **given size** (>100% of best perf => 'bestrperf+')
- MLKAPS tree have less regression and better SU



```
static int Pbest_8[] = {191, 169};
static int Pbest_9[] = {169, 232, 118};
static int Pbest_10[] = {147, 236, 117};
static int Pbest_11[] = {9, 108, 78};
static int Pbest_12[] = {142, 98};
static int Pbest_13[] = {37, 157, 149, 124};
static int Pbest_14[] = {151, 103, 104};
static int Pbest_15[] = {160, 36, 80, 219};

int decision_tree(float n, float m) {
    float y;
if (-0.015504753071330315 * n + -5.074066167232161e-17 * m + 46.506566
    if (-0.1881603818289878 * n + -0.10978173858676729 * m + 1087.8541
        if (-0.14254403476119454 * n + -0.12680213896911824 * m + 1085
            // depth = 3 ; bestnorm = 0.9866353932389675
            y = -2.002122115467538e-05 * n + -0.0005948999967214964 *
            return Pbest_8[floor(y)];
        } else {
            // depth = 3 ; bestnorm = 0.974406033s9899819
            y = 0.0003385844333697469 * n + -0.00022492166450271842 *
            return Pbest_9[floor(y)];
        }
    } else {
        if (-0.15496124606992043 * n + 0.04648837382097097 * m + 465.2
            // depth = 3 ; bestnorm = 0.9520015333258025
            y = 0.00024971879438375106 * n + -0.0005603488466896405 *
            return Pbest_10[floor(y)];
        } else {
            // depth = 3 ; bestnorm = 0.9728991767898622
            y = -5.606648294478492e-05 * n + 0.00042117389001227774 *
            return Pbest_11[floor(y)];
```

Bibliography ML-KAPS

[ASK13] *Adaptive sampling for performance characterization of application kernels*, Asma Farjallah Pablo de Oliveira, William Jalby, *Concurrency and Computation, Practice and Experience, 2013*
[SKL11] *Scikit- learn: Machine learning in Python.* F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Pas- sos, D. Cournapeau, M. Bruche, M. Perrot, and E. Duchesnay. *Journal of Machine Learning Research, 2011*
[NSGA18] *The multi-objective network design problem using minimizing externalities as objectives: comparison of a genetic algorithm and simulated annealing framework.* Bastiaan Possel, Luc Wismans, Eric van Berkum, and Michiel Bliemer. In Transportation, 2018.
[LGBM17] *Lightgbm: A highly efficient gradient boosting decision tree.* Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, et al., In Advances in neural information processing systems, 2017
[AML15] *Efficient and robust automated machine learning*. Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. In Advances in Neural Information Processing Systems, volume 28, 2015
[XGB16] *XGBoost: A Scalable Tree Boosting System*. Chen T, Guestrin C. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016
[GYM22] *CompilerGym: Robust, Performant Compiler Optimization Environment for AI Research*. C. Cummins, B. Wasti, J. Guo, C. Cui, J. Ansel, S. Gomez, International Symposium on Code Generation and Optimization, 2022
[Numa20] *Modeling and optimizing NUMA effects and prefetching with machine learning*. I. S. Barrera, D. Black-Schaffer, M. Casas, M. Moreto, A. Stupnikova, and M. Popov. In Proceedings of the 34th ACM International Conference on Supercomputing, 2020
[CERE15] *CERE: LLVM Based Codelet Extractor and REplayer for Piecewise Benchmarking and Optimization.* P. Oliveira, C. Akel, E. Petit, M. Popov, W. Jalby, ACM Transaction on Architecture and Code Optimization, 2015
Milepost GCC: Machine Learning Enabled Self-tuning Compiler. G. Fursin, Y. Kashnikov, A. Memon, Z. Chamski, O. Temam, M. Namolaru, B. Mendelson, A. Zaks, E. Courtois & F. Bodin, François,P. Barnard, E. Ashton, E. Bonilla, J. Thomson, C. Williams, M. O'Boyle, International Journal of Parallel Programming, 2011
[AT22] *Discovering faster matrix multiplication algorithms with reinforcement learning.* Fawzi, A., Balog, M., Huang, A. *et al., Nature*, 2022
[AF21] *Highly accurate protein structure prediction with AlphaFold.* Jumper, J., Evans, R., Pritzel, A. *et al. Nature*, 2021
[Lus23] *Combining multitask and transfer learning with deep Gaussian processes for autotuning-based performance engineering, Luszeck et al. The International Journal of High Performance Computing Applications, 2023*
[Liu21] GPTune: multitask learning for autotuning exascale applications Y. Liu et al. in Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '21)