# Automating HPC Model Selection on Edge Devices

**Abrar Hossain and Kishwar Ahmed**
Electrical Engineering and Computer Science
The University of Toledo

## Motivation

- Edge devices are crucial for HPC at the edge due to their ubiquity and computational capabilities. Figure 1 demonstrates the basic block diagram to run HPC applications on edge devices.
- Parameter search optimization is essential for optimal HPC performance on edge devices, considering resource and time constraints.
- The multi-armed bandit (MAB) model efficiently explores the search space, dynamically allocates resources, and converges toward near-optimal solutions in edge environments [1].

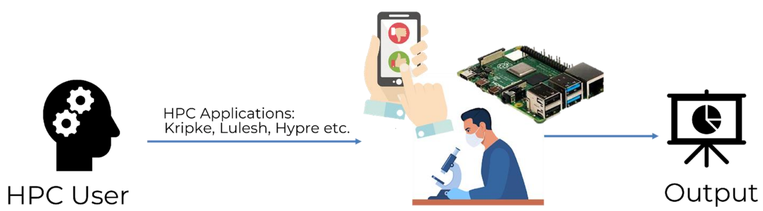


**Figure 1**: HPC on edge ecosystem

## Our Contribution

**HPEE (HPC Parameter Exploration on Edge):** An algorithm using random sampling, resource allocation to promising configurations, and discarding underperforming ones.

HPEE optimizes parameters by adapting configurations and resource allocation based on the round index. The algorithm iterates over rounds, determining configurations and resource allocation for each stage [2].

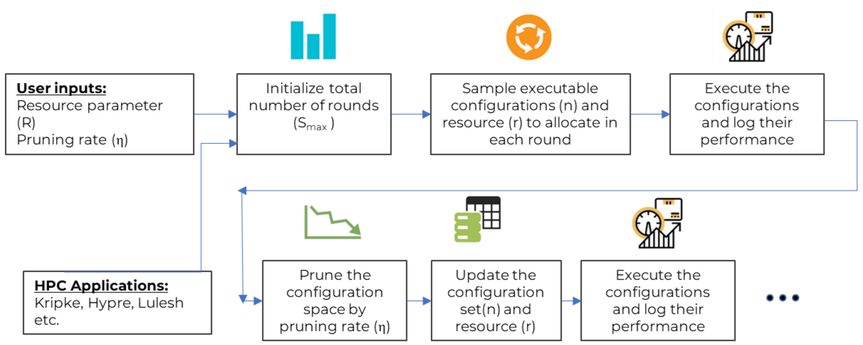


**Figure 2**: HPEE block diagram

Figure 2 describes the process that includes sampling parameter configurations, evaluating them based on execution time, and retaining top-performing configurations. The choice of parameters resource (R) and pruning rate (η) influences convergence, efficiency, exploration, and exploitation in the algorithm.

| Application | Application Parameters (all parameters are discrete integers unless otherwise stated) | Size of Search Space (# Configs.) |
|---|---|---|
| Kripke | # OpenMP Threads, Nesting Order, Group Set, Direction Set | 1, 112, 832 |
| Clomp | # OpenMP Threads, Parts/Thread, Zones/Part, Zone Size, Flop Size | 618, 240, 000 |
| AMG | Solver Type (categorical), Smoother Type, Interpolator Type, Coarsening Type, Elements/Row | 5, 873, 280 |
| Hypre | Solver Type (categorical), Smoother Type, Interpolator Type, Coarsening Type, Elements/Row | 3, 297, 280 |
| Lulesh | Elements in a Mesh, Materials in a Region | 49, 680 |

**Table 1:** Parameter space for different HPC applications [3]

Table 1 demonstrates how prohibitively large the parameter search space is for different HPC applications.

## Problem Formulation

HPC parameters vary in type and have dependencies. Uniformly sampling configurations, we evaluate performance with metric $t\ (k,\ x)$, where $k$ is resource allocation and $x$ is a configuration from set $X$. We aim to find the optimal choice by defining $R$ as the distributed resource parameter and $t*$ as the limit of the performance metric. We frame the problem as an MAB problem, searching for a configuration $x$ that minimizes $(t*\ -\ \nu*)$ in the OPS optimization problem. Mathematically,

$$\text{OPS}: \underset{x}{\text{minimize}} \quad t^*(x) - \nu^*$$
$$\text{subject to} \quad p(k,x) < p^*$$

## Algorithm

**Algorithm 1** HPEE

**Input:** Maximum computational iteration ($R$), prune factor ($\eta$)

**Initialization:** Initialize maximum stage number, $s_{\max} = \log_\eta R$ and budget, $B = (s_{\max}+1)\cdot R$;
**for** $s \in \{s_{\max}, s_{\max}-1, \ldots, 0\}$ **do**
  $n = \frac{B}{R}\frac{\eta^s}{s+1}$, $r = R\eta^{-s}$;
  **for** $i \in \{0, \ldots, s\}$ **do**
    $n_i = n\eta^{-i}$, $r_i = r\eta^i$;
    $L = \{\text{find the execution times}\}$;
    $T = \text{top\_k}(T, L, \frac{n_i}{\eta})$;
  **end for**
**end for**
**return** configuration with the smallest intermediate loss seen so far

- HPEE algorithm optimizes parameters by iteratively refining configurations based on performance evaluation, focusing on the most promising settings.
- Choice of parameters R and η plays a crucial role: R determines convergence and efficiency-accuracy balance, while η controls exploration-exploitation tradeoff.
- The algorithm allows customization for specific needs and constraints, making it suitable for resource-constrained edge devices.

## Evaluation

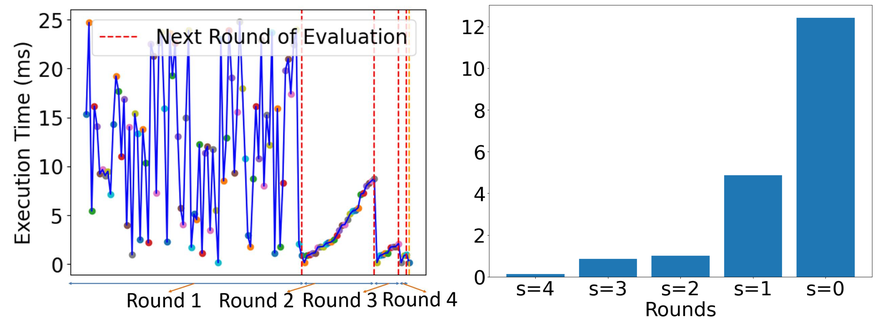


**Figure 3**: Evaluation of the model with R = 243 and $\eta$ = 3.

Figure 3 shows evaluation results highlight effective sampling approaches and iterative pruning for identifying the most optimal configuration. Multiple rounds of sampling conducted, including a focused phase ($s$ = 4) for optimal configuration exploration.

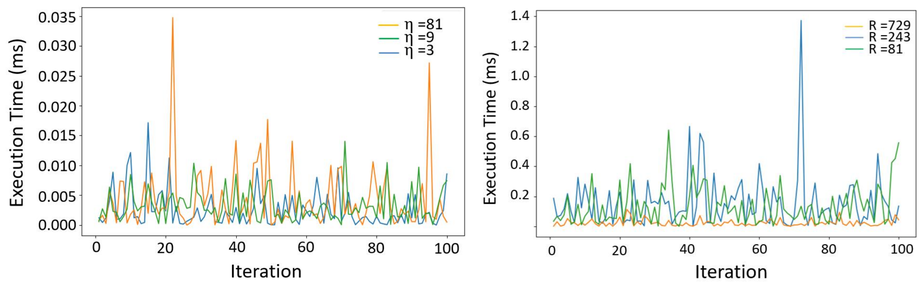


**Figure 4**: Performance of HPEE by customizing $R$ and $\eta$

Figure 4 illustrates model's behavior assessed by tuning hyperparameters $R$ and $\eta$, showing variations in performance. It is evident that increasing $R$ yields more optimal results.

## Conclusion and Future Work

- Proposed HPEE approach optimizes parameter search space in HPC applications on edge devices using MAB framework, random sampling, and resource allocation.
- HPEE efficiently explores the configuration space with customization options for performance optimization.
- Results demonstrate the effectiveness of HPEE, and future research can focus on broader device validation and advanced MAB strategies for improved optimization

## Acknowledgements

This work is supported in parts by the US National Science Foundation under grant CNS-2300124.

## References

**[1]** Steven L Scott. 2010. A modern Bayesian look at the multi-armed bandit. Applied Stochastic Models in Business and Industry 26, 6 (2010), 639–658.
**[2]** Xingfu Wu, Michael Kruse, Prasanna Balaprakash, Hal Finkel, Paul Hovland, Valerie Taylor, and Mary Hall. 2022. Autotuning PolyBench benchmarks with LLVM Clang/Polly loop optimization pragmas using Bayesian optimization. Concurrency and Computation: Practice and Experience 34, 20 (2022), e6683.
**[3]** Roy, Rohan Basu, Tirthak Patel, Vijay Gadepally, and Devesh Tiwari. "Bliss: auto-tuning complex applications using a pool of diverse lightweight learning models." In Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, pp. 1280-1295. 2021.