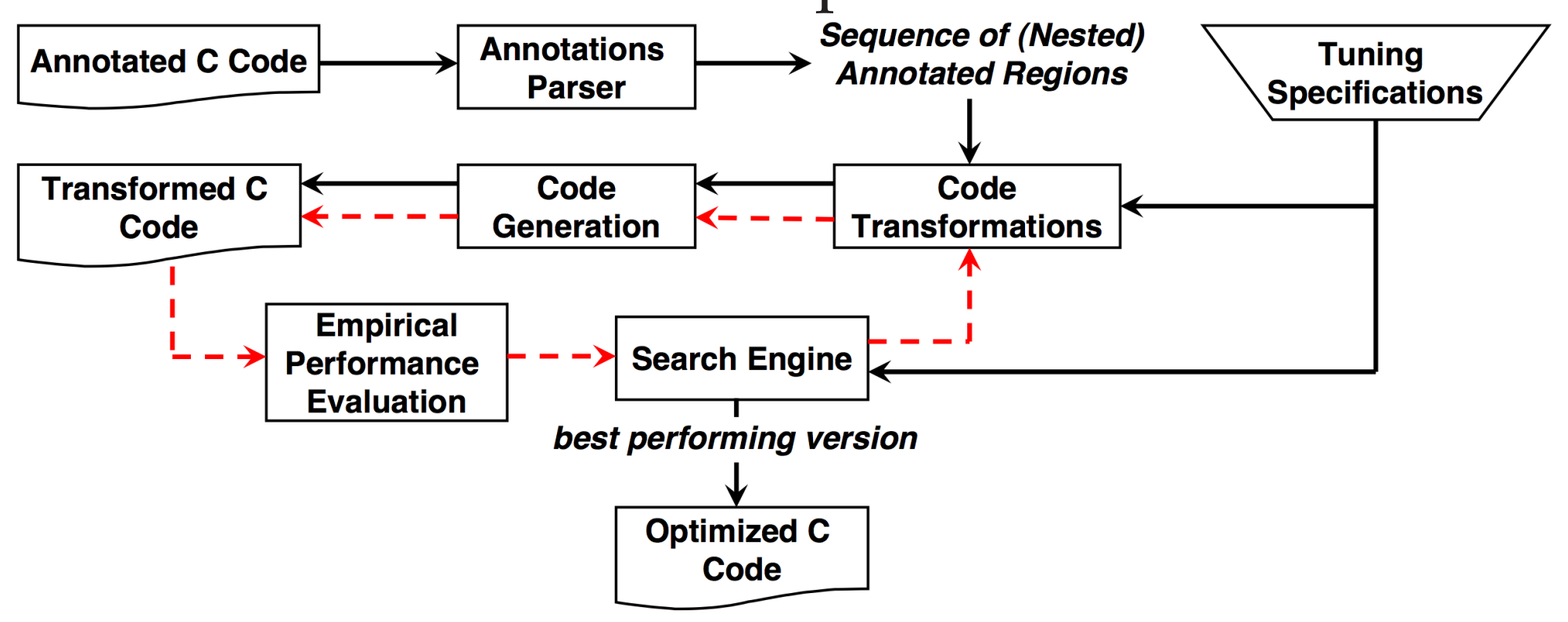
Model-Based Optimization Algorithms for Empirical Performance Tuning

Prasanna Balaprakash and Stefan M. Wild
Mathematics and Computer Science Division, Argonne National Laboratory, IL
{pbalapra,wild}@mcs.anl.gov

Problem

Increasing complexity of modern computer architectures presents obstacles for achieving high-performance of scientific codes.
Empirical tuning is an attractive approach for the performance quest.

Example:


Annotated C Code → Annotations Parser → *Sequence of (Nested) Annotated Regions* → Code Transformations ← Tuning Specifications
Code Transformations → Code Generation → Transformed C Code → Empirical Performance Evaluation → Search Engine → Code Transformations
Search Engine → *best performing version* → Optimized C Code

Computation time is a major bottleneck for large-scale performance tuning:

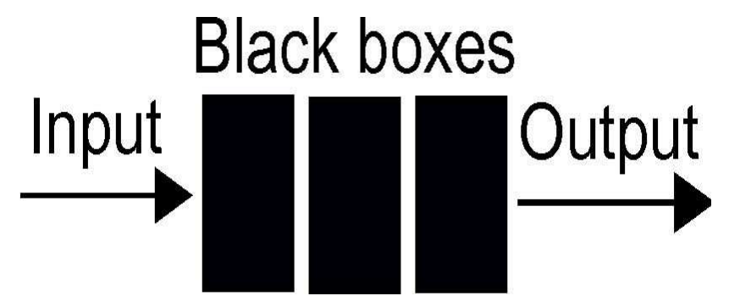
- Number of code variants to test grows exponentially with the parameters

Contributions

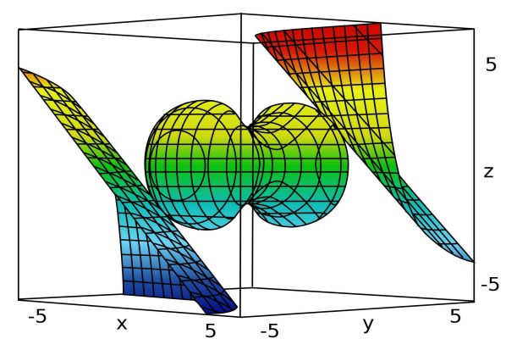
1. Formulated the search in tuning as a mathematical optimization problem
2. Developed SPAPT test suite for benchmarking optimization algorithms
3. Designed a model-based optimization algorithm for performance tuning

Challenges

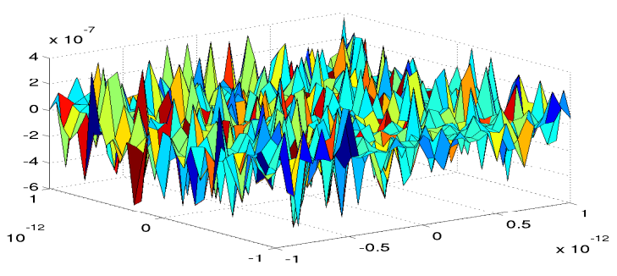
no derivatives



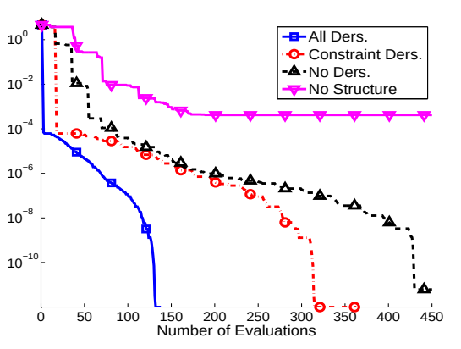
discontinuities



noise



difficult



Modeling and Formulation

Mixed-integer, nonlinear optimization problem

$$\min_{x} \{ f(x) : x = (x_{\mathcal{I}}, x_{\mathcal{B}}, x_{\mathcal{C}}) \in \mathcal{D} \subset \mathbb{R}^n \}$$

x: a parameterization of the code

- $x_{\mathcal{I}}$: integer parameters (cache tiling, unroll jam, . . .).
- $x_{\mathcal{B}}$: binary parameters (multicore parallelization, compiler types, . . .).
- $x_{\mathcal{C}}$: continuous parameters (tolerance for an iterative solver . . .).

$f(x)$: empirical performance metric of a code variant such as FLOPS, power, or run time

- noisy $f(x)$: mean, median, . . .

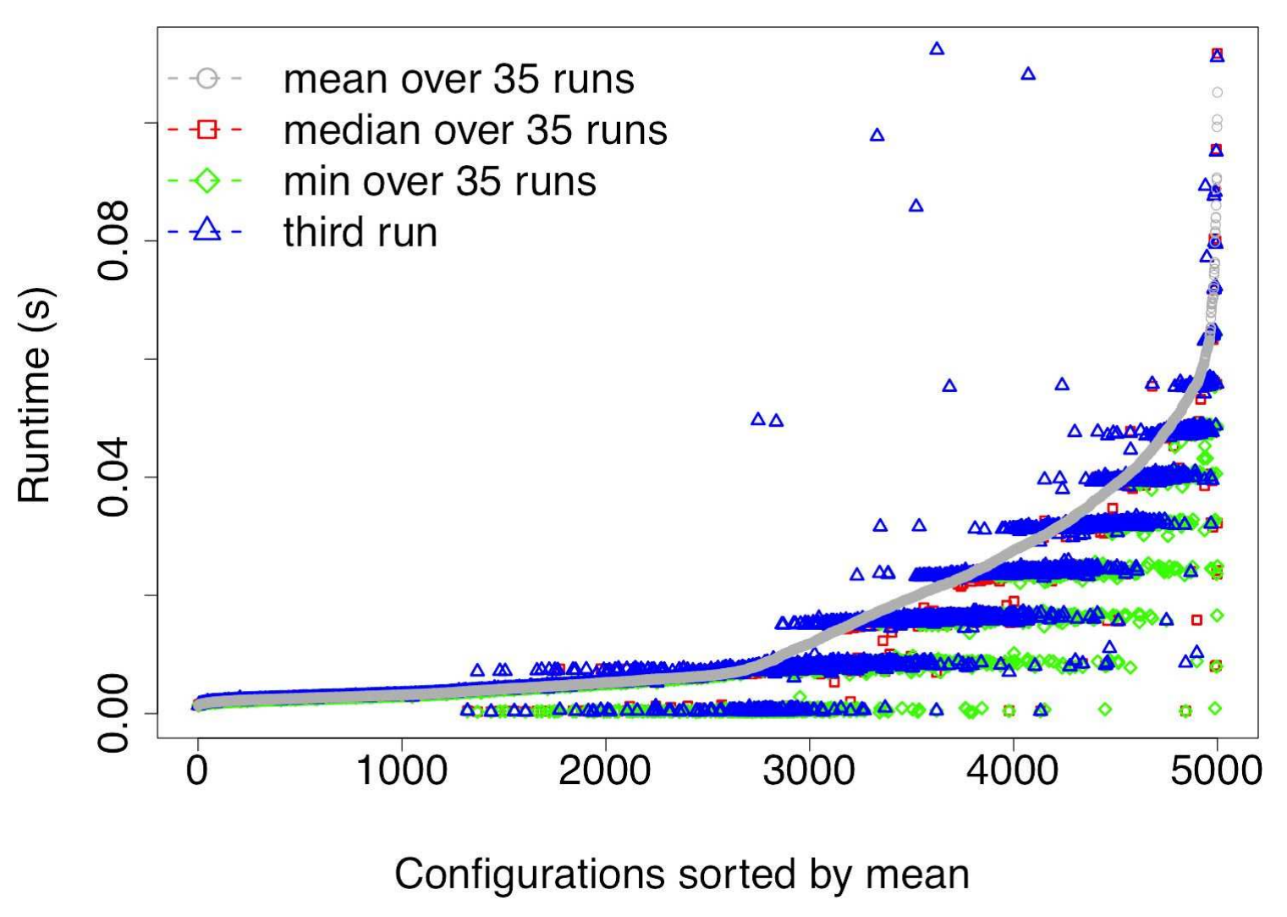
subject to constraints:

- bound: unroll = [1. . . 30], RT = [1,8,32].
- known: $RT_I * RT_J \leq 150$ (cheap)
  power consumption < 90 W (expensive).
- hidden: transformation errors (relatively cheap), compilation errors (expensive), and run time errors (very expensive).

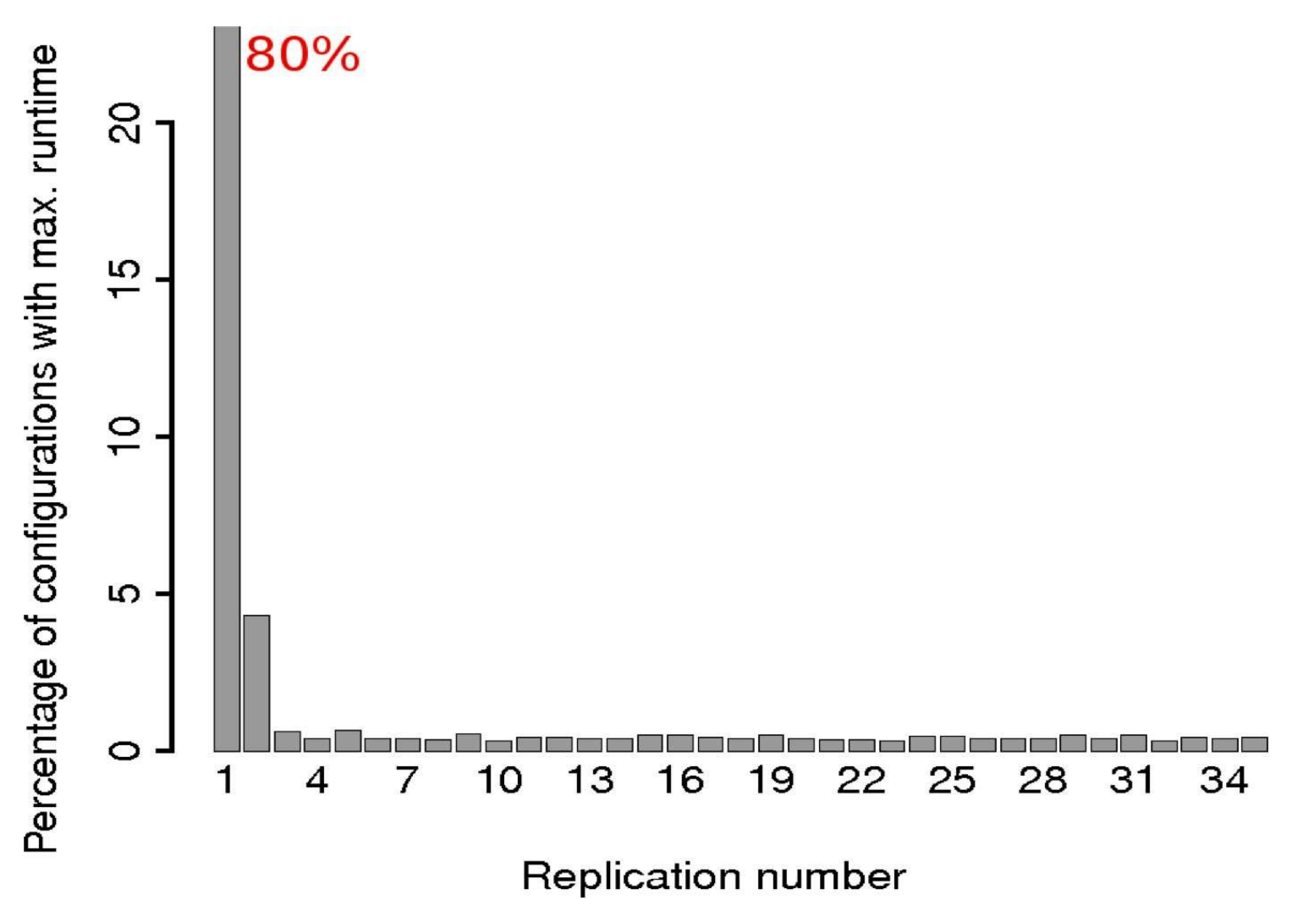
SPAPT test suite

- 72 problems from 18 serial scientific computation kernel codes
- A SPAPT problem = code + set of transformations + parameter specifications + constraints + input size (+ machine)
- 10 to 50 parameters with search space of $1e08$ to $2e30$ code configurations

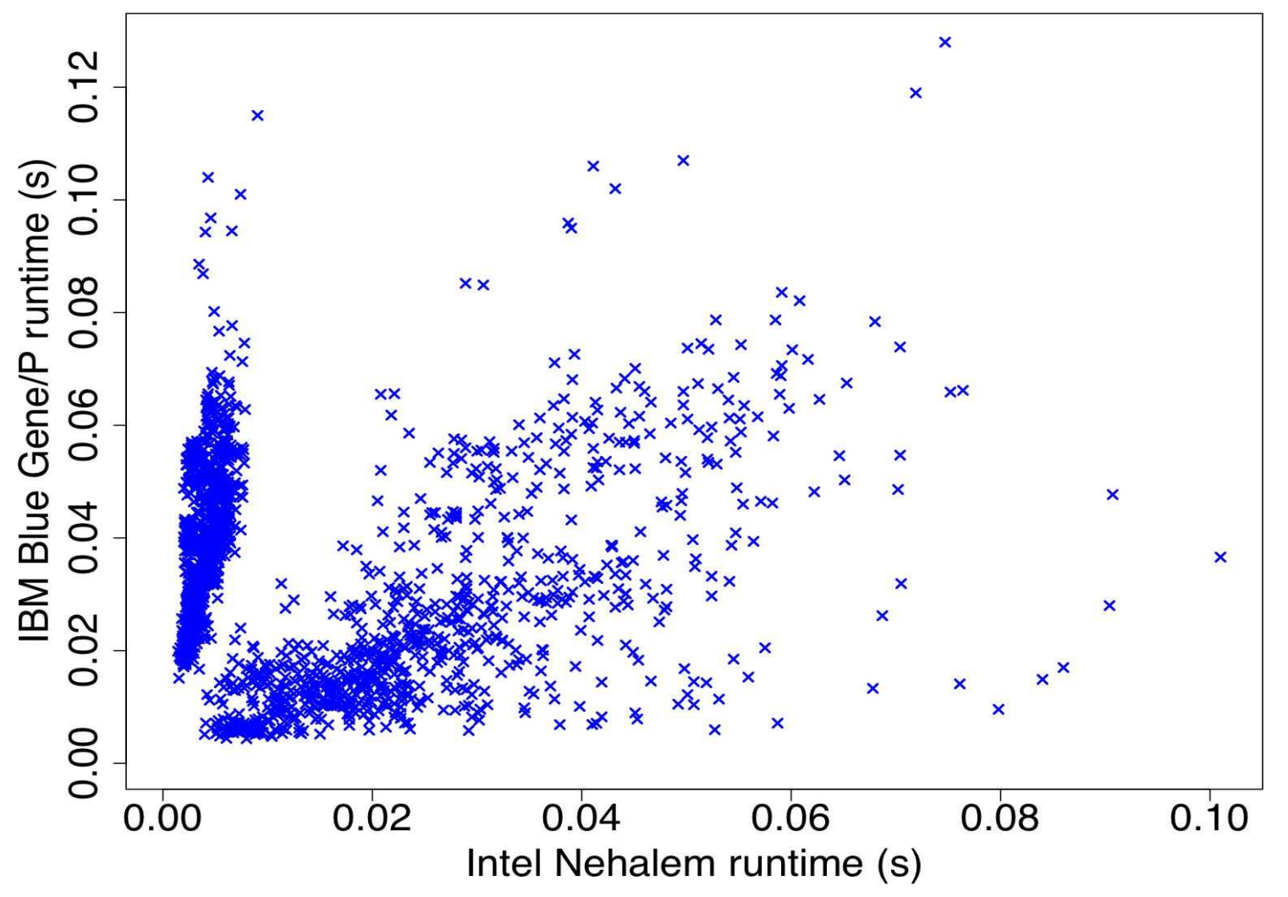
choice of statistics:



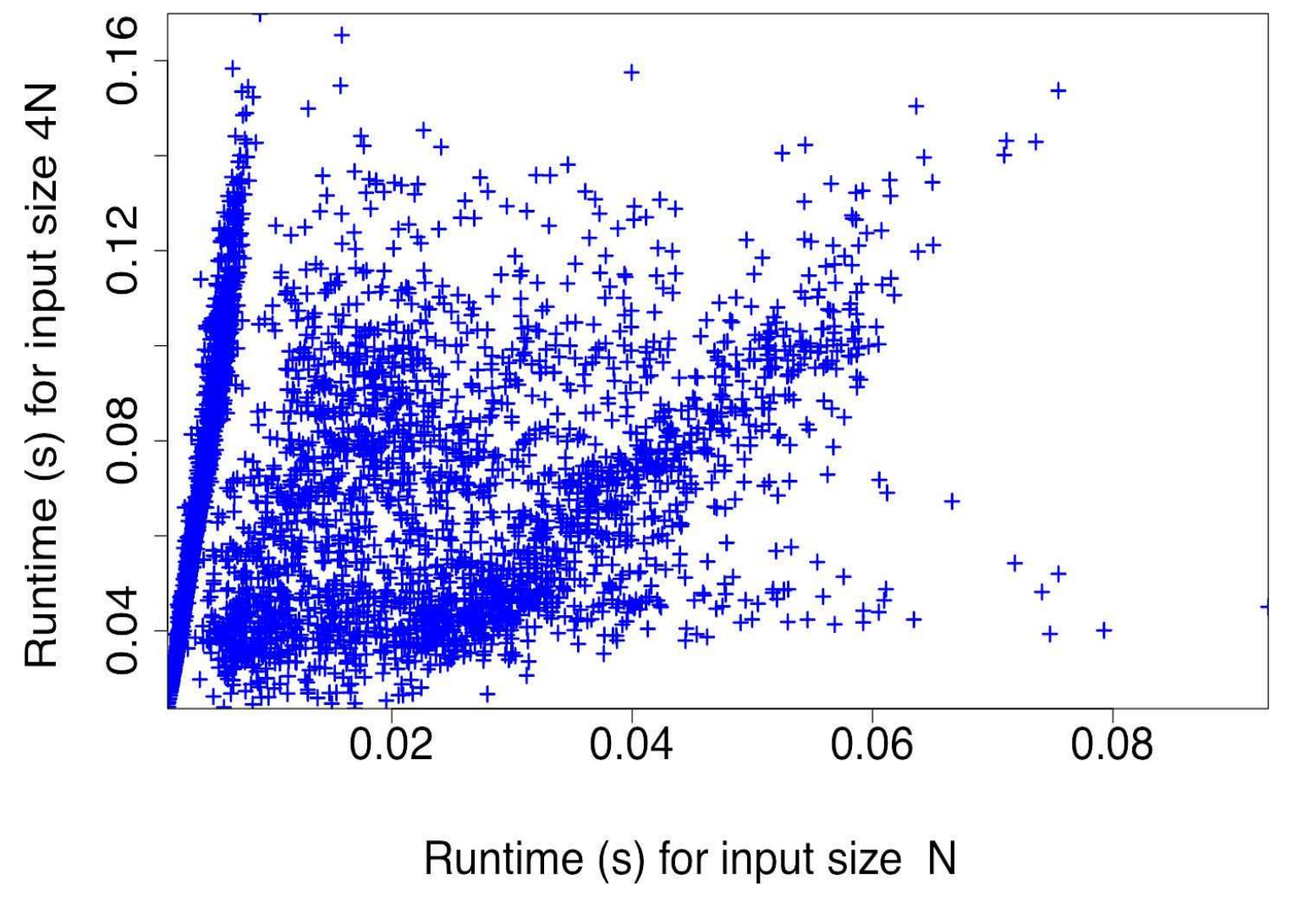
effect of cache misses:



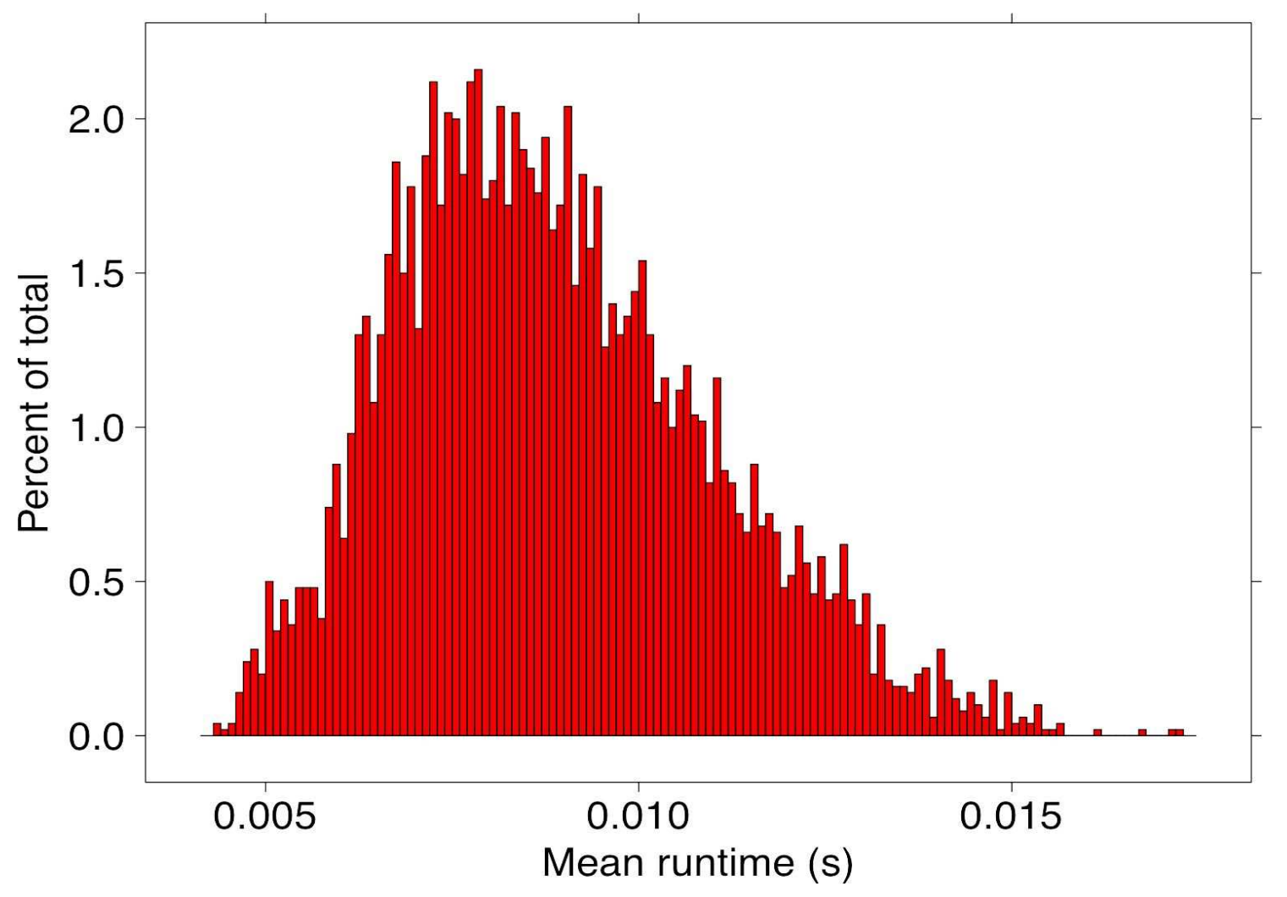
impact of target machine:



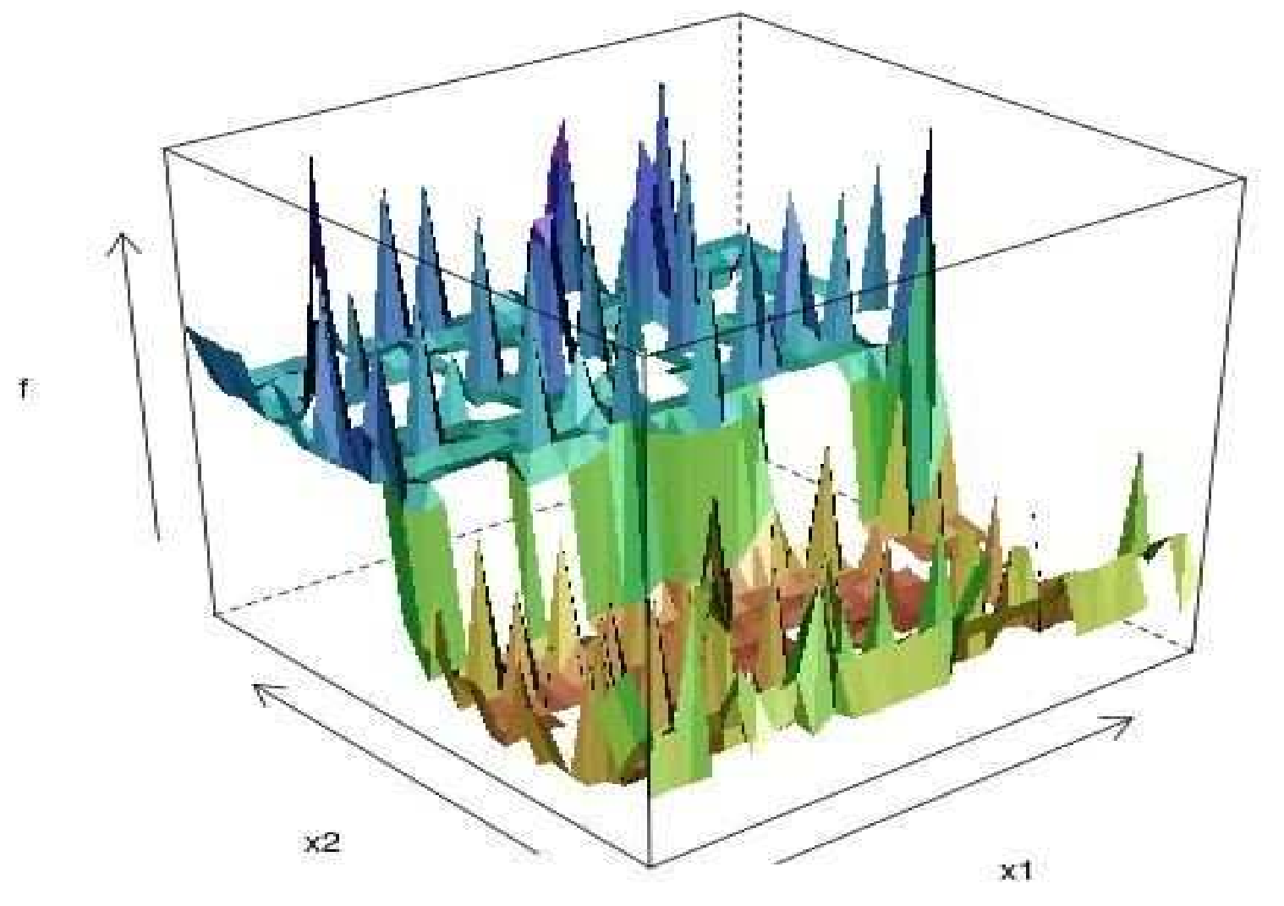
impact of input size:



objective density:



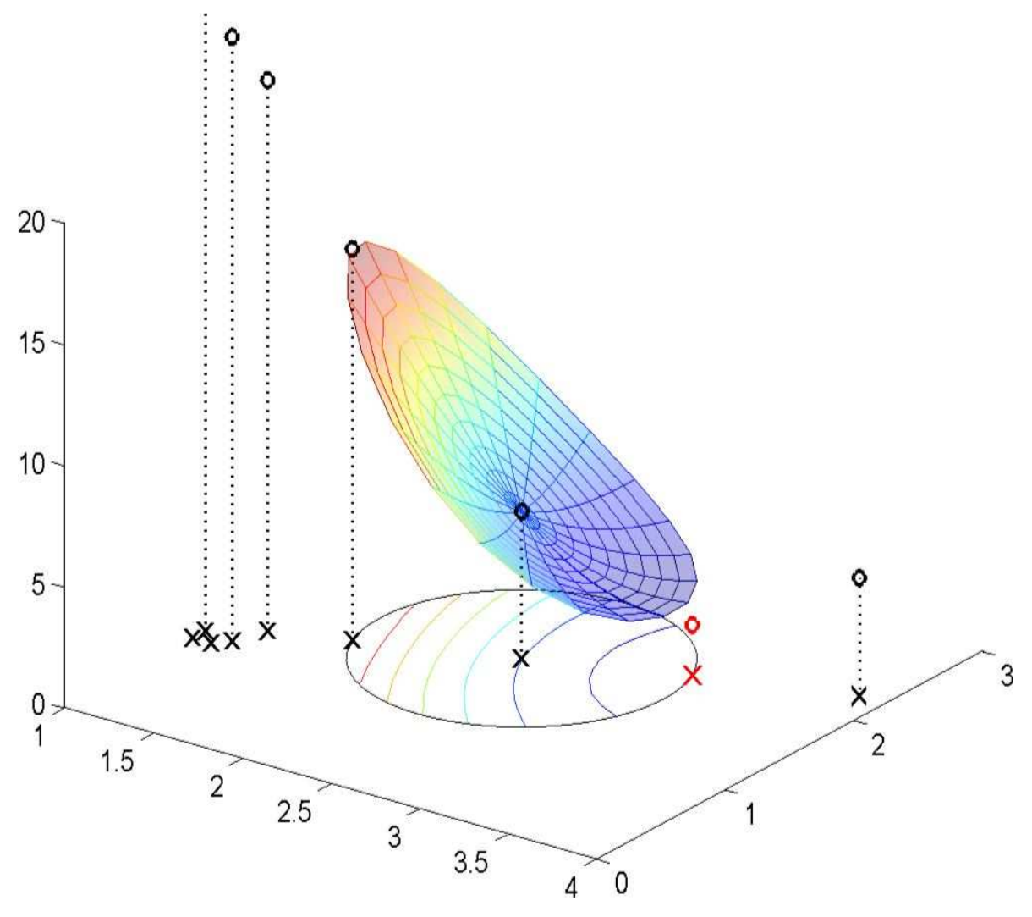
structure:



Model-based derivative-free method

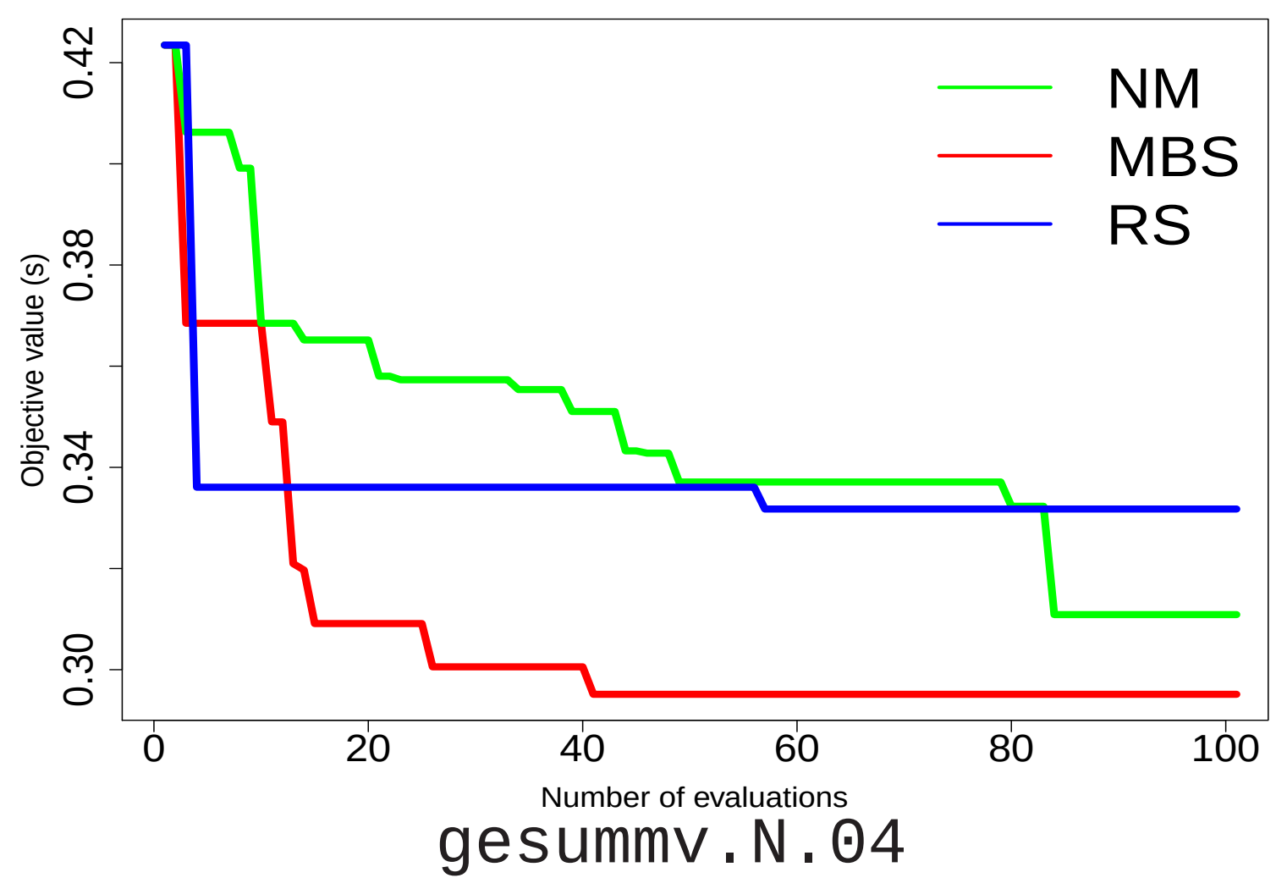
A straw-man trust region algorithm at iteration k:

- construct a quadratic model q_k
- minimize quadratic q_k locally to find x_c
- replace x_c with the best neighbor point x_b using q_k when x_c is evaluated before
- compute $f(x_c)$
  1. sufficient decrease: update x_k; increase trust region radius;
  2. no improvement: decrease radius or improve sampling.

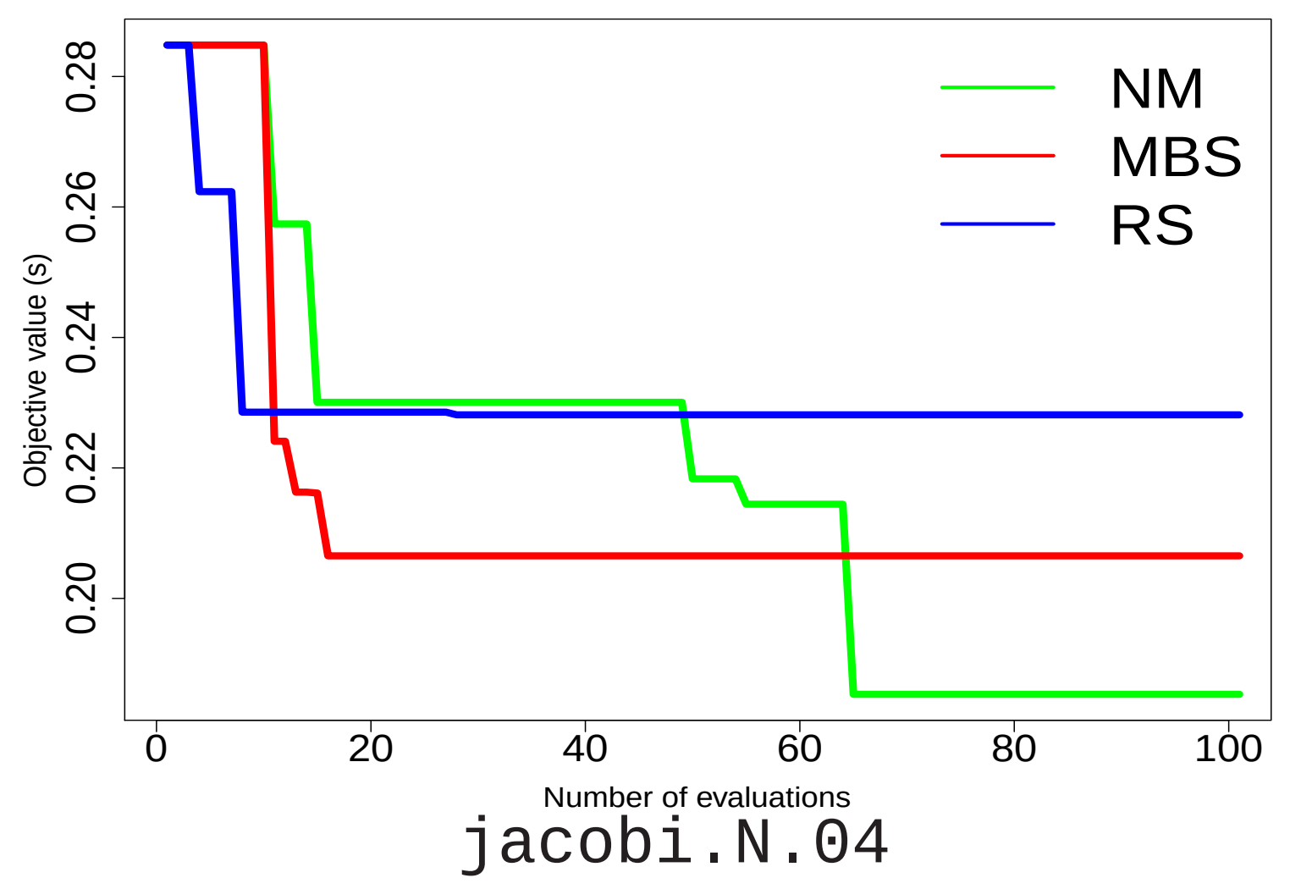


Initial results

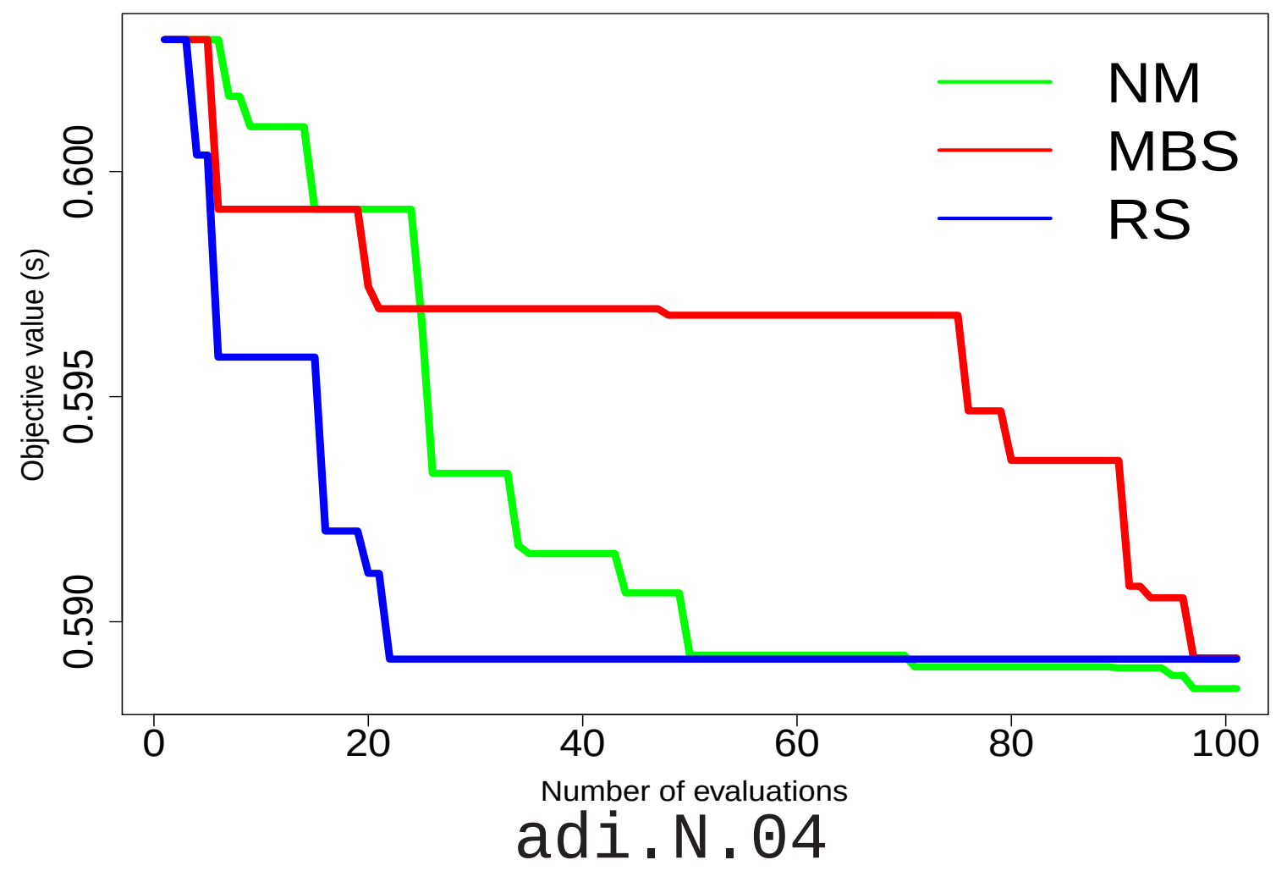
- Three implementations: Random search (RS), modified Nelder Mead (NM), Model-based search (MBS)
- SPAPT problems
- Each evaluation consists of 35 runs; Objective: mean run time



gesummv.N.04



jacobi.N.04



adi.N.04

Winner depends on the problem characteristics
Continuous optimization algorithms demand careful customization

Conclusions

- Search in performance tuning is a derivative-free optimization problem
- Novel optimization algorithms offer potential to find high-quality configurations in a short time
- Problem characteristics can significantly impact the effectiveness
- Algorithms need to exploit tuning problem characteristics

Future work

- Search space characterization
- Customization of algorithms to handle contraints, binary parameters, and cache misses
- Developing parallel optimization algorithms
- Tuning communication avoidance and hiding kernels

Acknowledgments

This work was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357. We are grateful to Boyana Norris, Paul D. Hovland, and Aswin Kannan for helpful discussions and to the National Energy Research Scientific Computing Center, and Argonne National Laboratory's Leadership Computing Facility and Laboratory Computing Resource Center.