FZ HPC

universität innsbruck
Department of Computer Science

DPS
Distributed and Parallel Systems

Towards Multi-objective, Region-based Auto-tuning for Parallel Programs

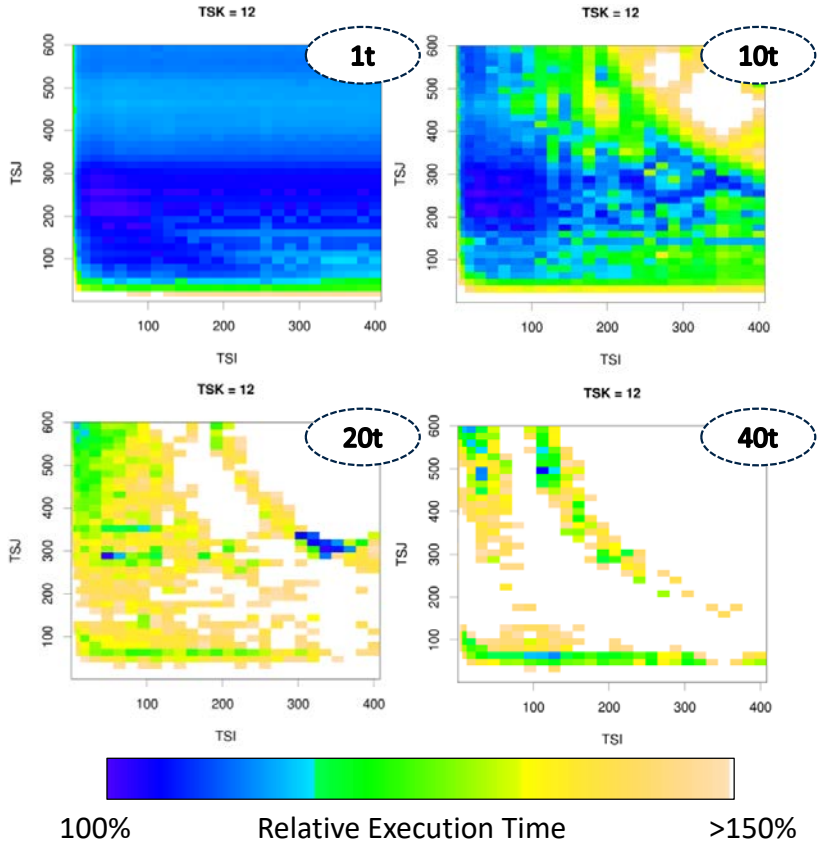
Philipp Gschwandtner
Juan Durillo, Klaus Kofler, Herbert Jordan, Peter Thoman, Thomas Fahringer
Research Center HPC / Distributed and Parallel Systems Group, University of Innsbruck

Why is it so Hard to Optimize Codes for Parallel Systems?

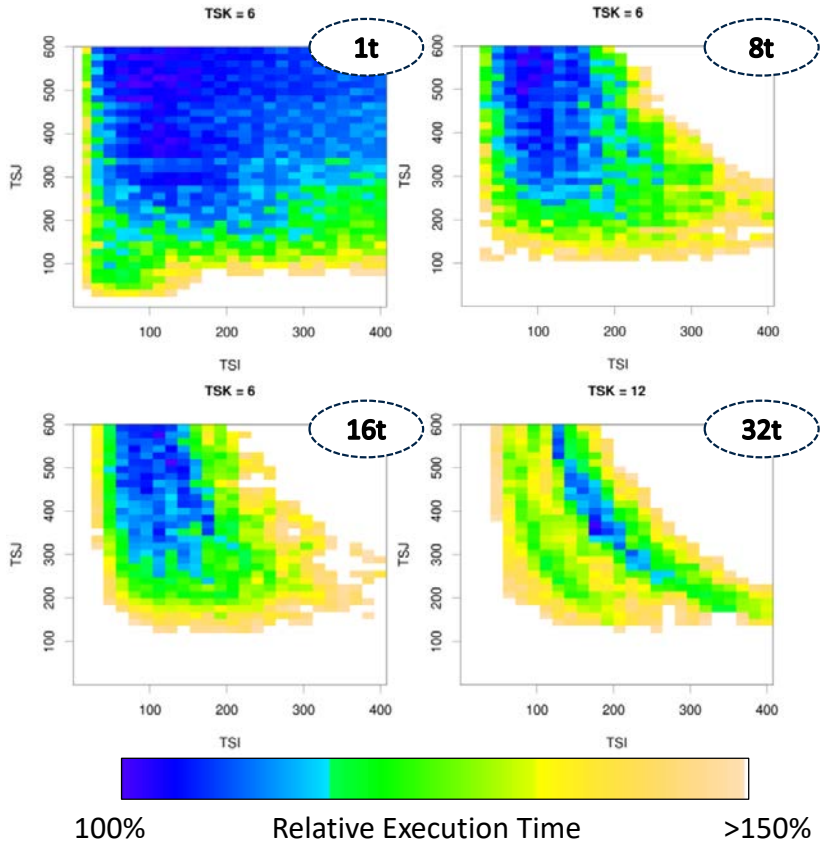
- If you ran your code on different hardware how would you alter your code?
  - I/O and process scheduling, cache sizes and policies, interconnect, …
- Complexity, non-determinism, effort to predict program and system behavior
  - dynamic reallocation of cores and memory, clock speed, external load and resource contention, etc.
  - operating system, queueing systems, software libraries, etc.
- Modern system architectures are too complex for humans beings to manually find the best code transformation sequences for program optimization.

Impact of Parallelism on Loop Tiling of Matrix Multiply

Intel Westmere



AMD Barcelona



Search Space for Code Optimizations and Parallelization Strategies

- code transformations
  - loop tiling, scheduling, interchange, data transpose, data distribution, …
- library settings
  - MPI tuning parameters (several hundred)
- compiler selection and flag settings
  - optimization flags (gcc: approx. 200)
- target machine configuration
  - number of cores, multi-threading settings, frequencies & turbo boost, …
- huge search space to explore for manual optimization

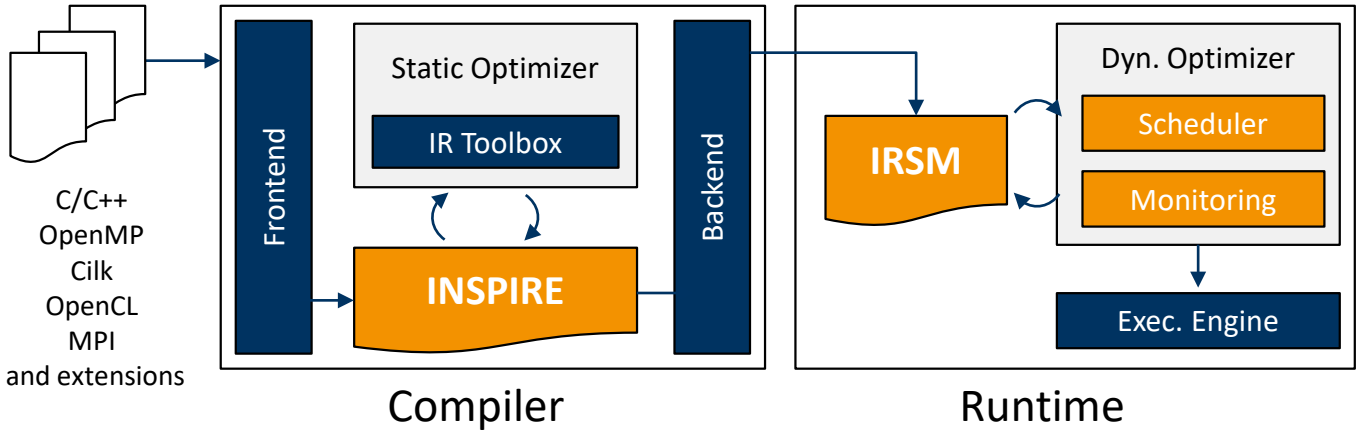
  ➔ use **auto-tuning** instead

Automatic Optimization Landscape

- Able to re-write your application? Use a domain-specific language!
  - toolchain will/should/might take care of optimization for you
  - facilitates performance portability and separation of concerns
  - EU H2020 Project AllScale – www.allscale.eu
- Unable to re-write, but you can manually annotate?
  - manually mark interesting code regions, leave rest up to toolchain
- Unable to manually annotate?
  - use a compiler!
  - fully automatic

Insieme – Compiler and Runtime Research Platform

- Insieme Source-to-source compiler
  - C/C++, OpenMP, MPI, OpenCL, Cilk
  - uniform intermediate representation
  - analysis and transformation framework (polyhedral model, pattern matching, etc.)
- Insieme runtime system
  - scheduling & runtime auto-tuning
  - compiler-aided decision making process

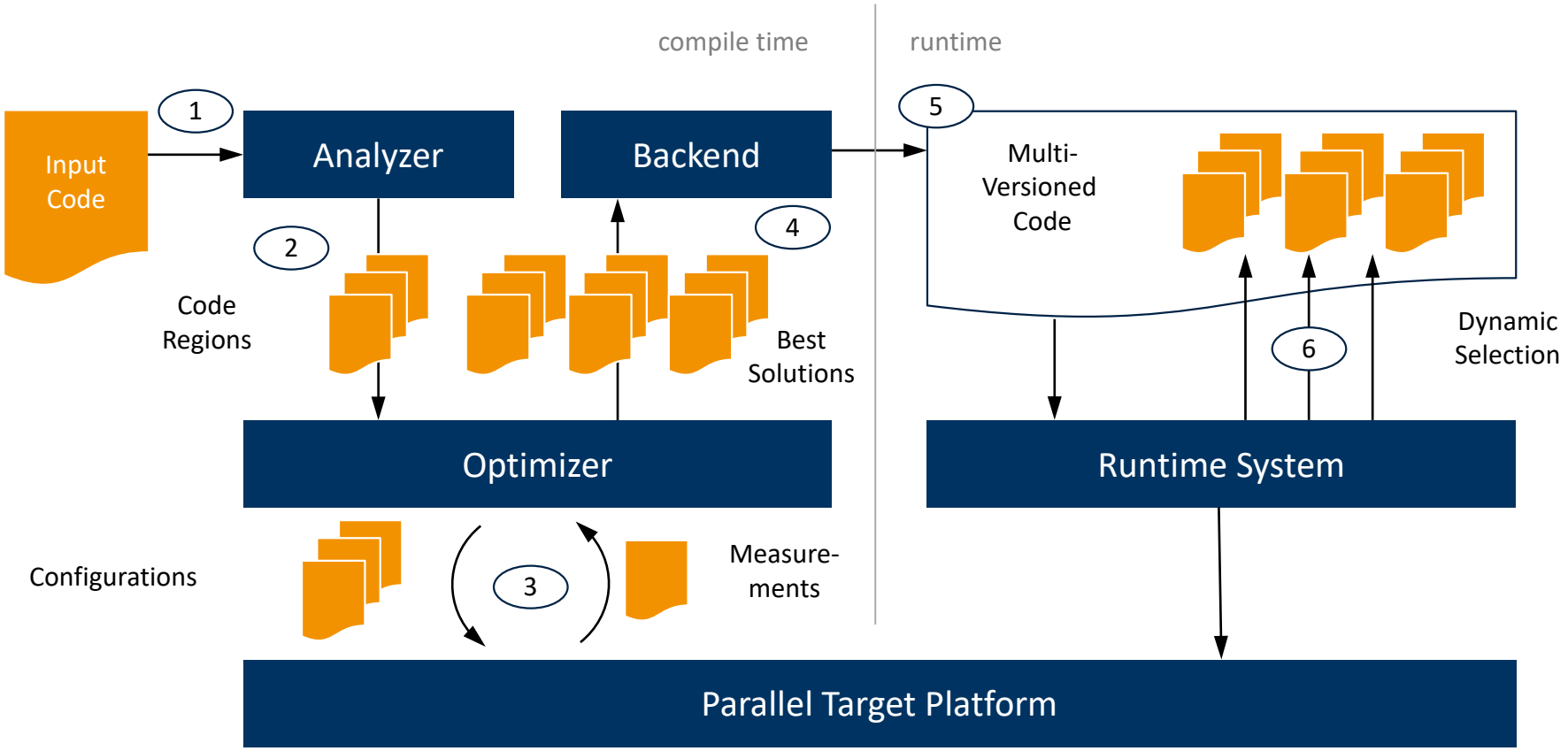


Insieme – Compiler and Runtime Research Platform

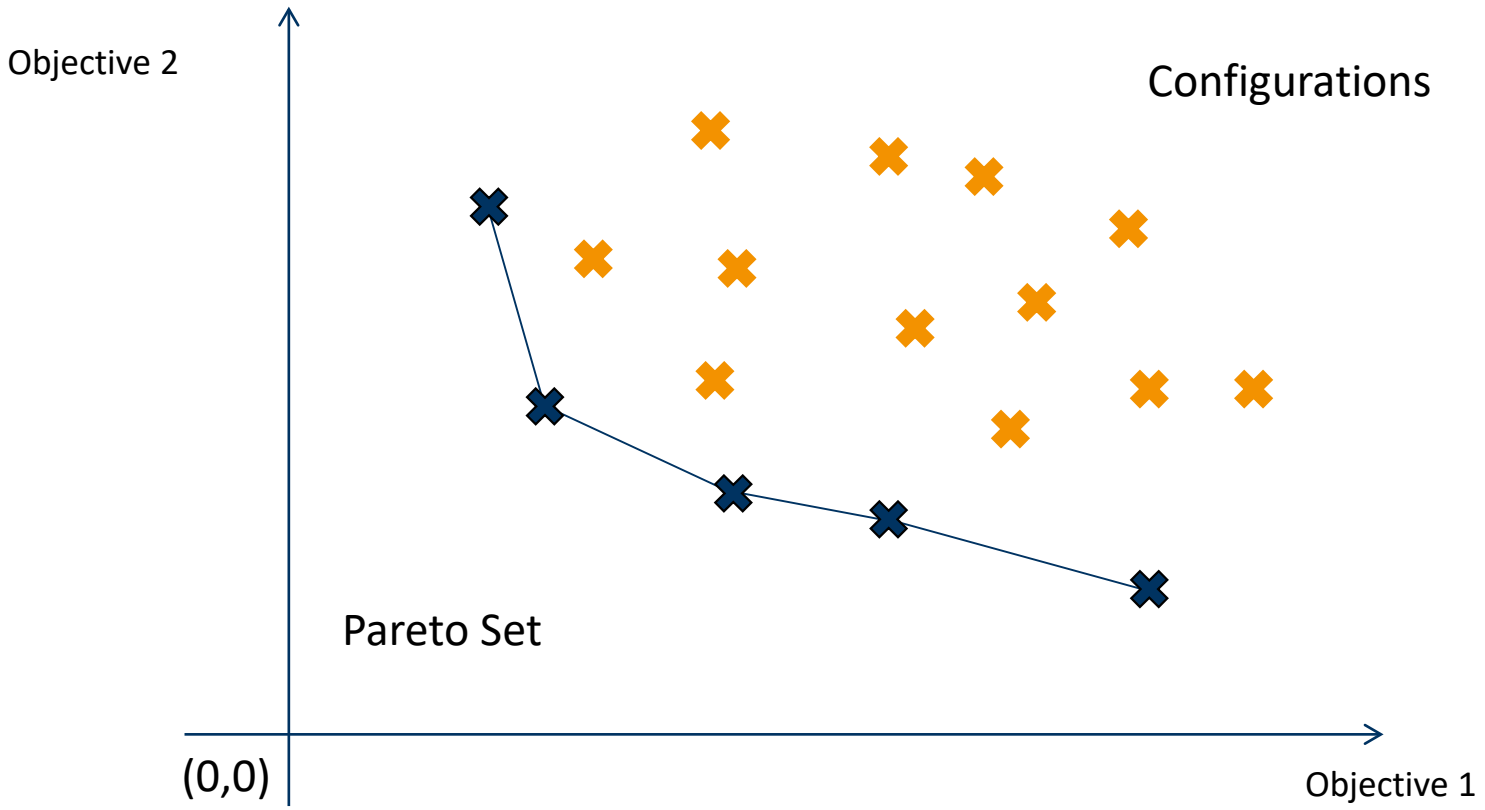
- Foundation for research in
  - parallel languages and extensions
  - optimizing parallel applications
    - static, dynamic, and hybrid approaches
    - search- and ML-based auto-tuning
  - constraint-based program analysis
  - **region-based multi-objective optimization**

- Further Information:
  - https://www.insieme-compiler.org

Insieme Auto-tuning Architecture



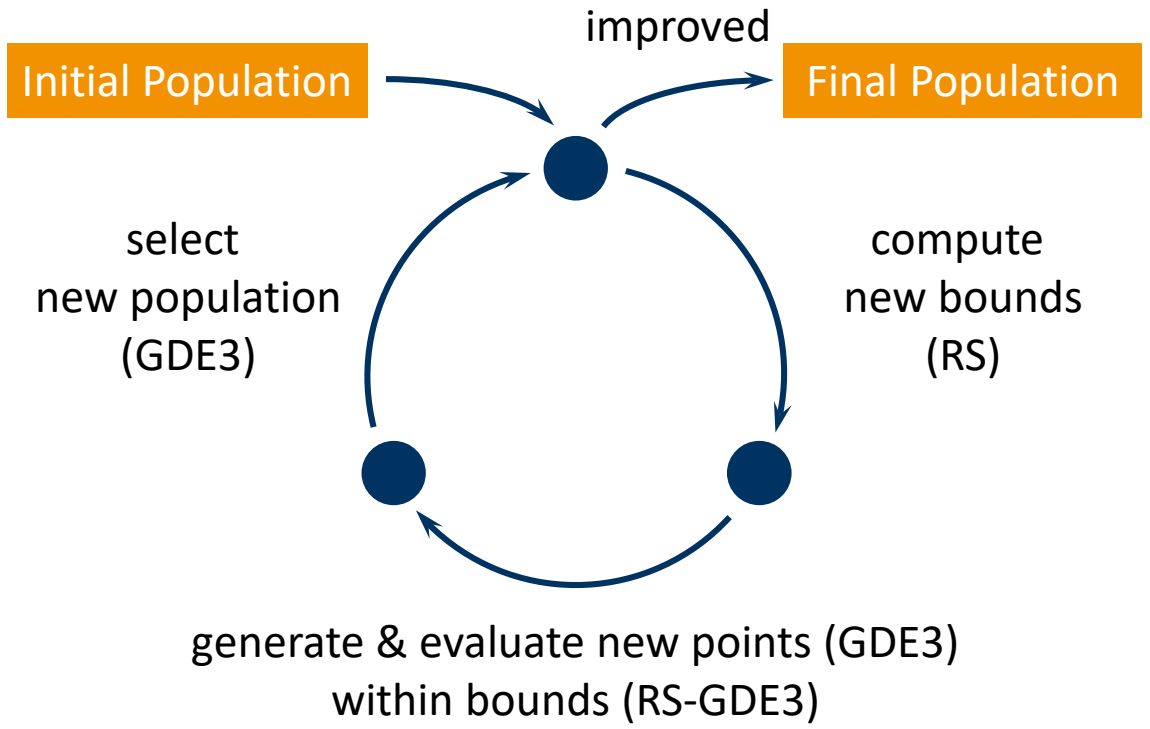
Multi-objective Optimization Using Pareto Optimality



Optimization Algorithm

- Our algorithm: RS-GDE3

- Combination of
  - metaheuristics
    - generalized differential evolution (conducts the actual iterative search)
  - search space reduction based on rough sets
    - domain independent



Objective Space and Problem Space

- Optimize 3 objectives simultaneously
  - runtime (measured)
  - efficiency ($\mathrm{cost}(\mathrm{x}) = x \times t_p(x)$)
  - energy (measured e.g. via Intel RAPL)

- Several tuning parameters
  - 2D/3D loop tiling
  - degree of parallelism (number of threads)
  - clock frequency (DVFS)

Experimental Setup

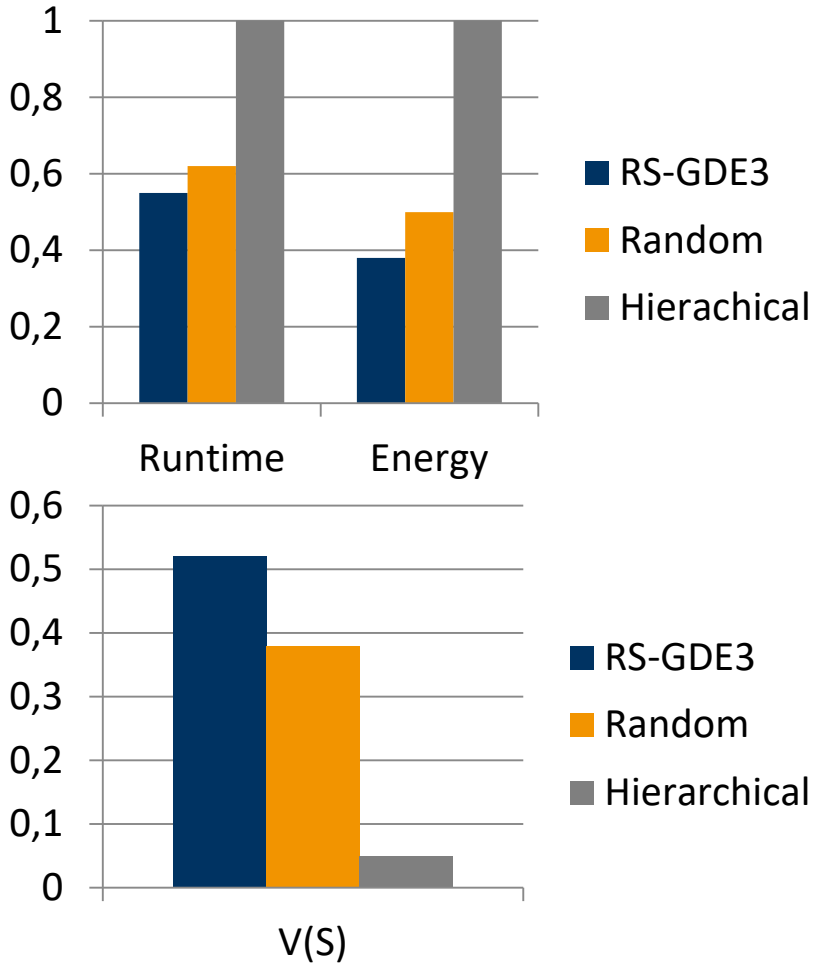
- Target architecture:
  - 4x Intel Xeon Sandy Bridge with 32 cores total
  - available clock frequencies (1.2-2.6 GHz)
- Tuning algorithms:
  - hierarchical search (regular grid)
  - random search
  - RS-GDE3 (our algorithm)
- Multiple codes with various kernels
  - matrix multiply, stencil, n-body, multi-grid, etc.
- Quality of solution S
  - hypervolume: $\mathrm{V}(\mathrm{S}) \in [0, \ldots 1]$
  - number of points: $|S|$
  - degree of dominance: $|S|'$
- Efficiency of algorithms
  - number of evaluated configurations N
  - time-to-solution
  - energy-to-solution
- $\overline{\mathrm{V}(\mathrm{S})}$, $\overline{\mathrm{S}}$, $\overline{\mathrm{E}}$ for stochastic algorithms

Performance Comparison

Benchmark	Hierarchical Search				Random Search				RS-GDE3			
	N	|S|	|S|‘	V(S)	N	|S|	|S|‘	V(S)	N	|S|	|S|‘	V(S)
mm	18432	18	2%	0.00	15000	4.4	0%	0.33	956.2	23.4	98%	0.48
dsyrk	18432	21	5%	0.00	15000	2.2	11%	0.17	1149.6	24.8	98%	0.32
jacobi-2d	15876	31	78%	0.69	15000	17.2	5%	0.55	1243.6	29.8	75%	0.76
3d-stencil	15876	30	22%	0.75	15000	24.8	60%	0.61	981.4	28.2	77%	0.76
n-body	15876	26	0%	0.50	15000	30.0	17%	0.70	1801.4	29.6	87%	0.77

Highlights

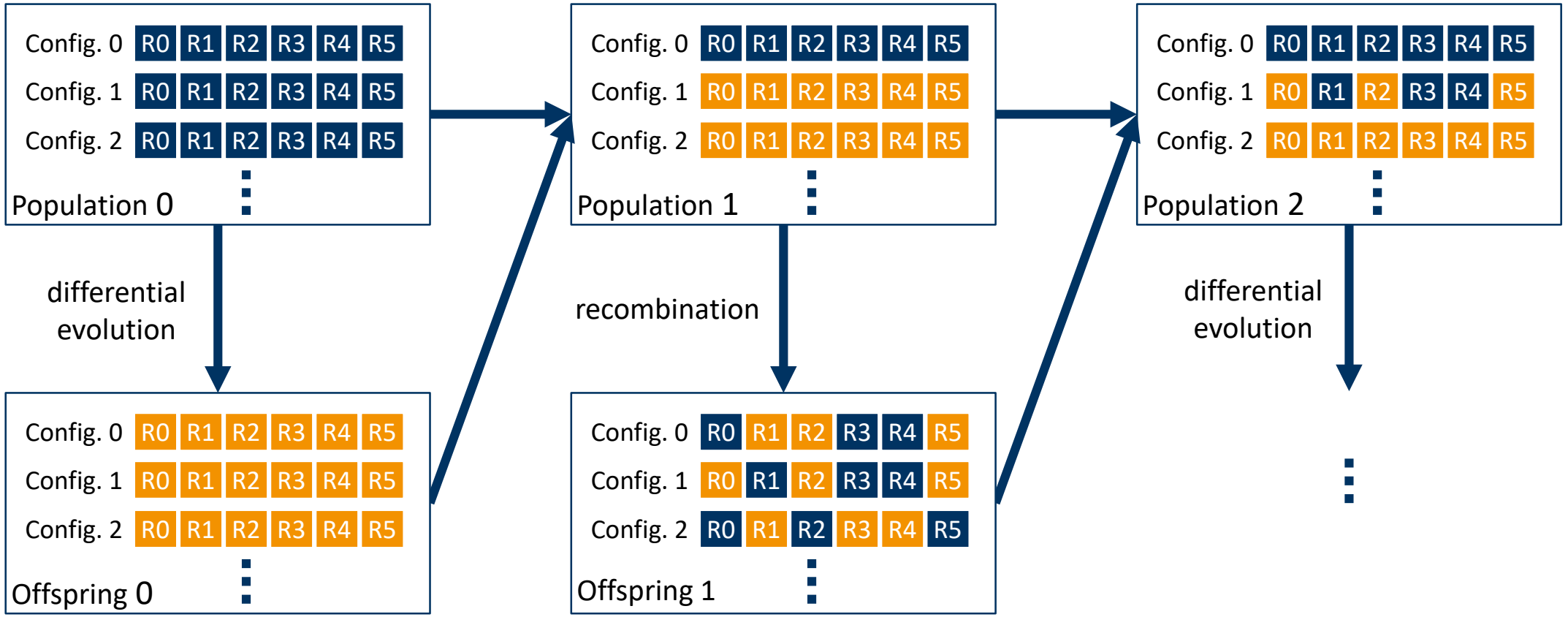
- RS-GDE3 evaluates less than 10^{-7} of all possible configurations and only 7 % of the configurations required by random search
- Comparing the fastest solutions
  - 70 % more energy efficient than hierarchical, 45 % faster
  - 24 % more energy efficient than random search, 14 % faster
- The hypervolume also shows the higher quality of the Pareto fronts computed by RS-GDE3



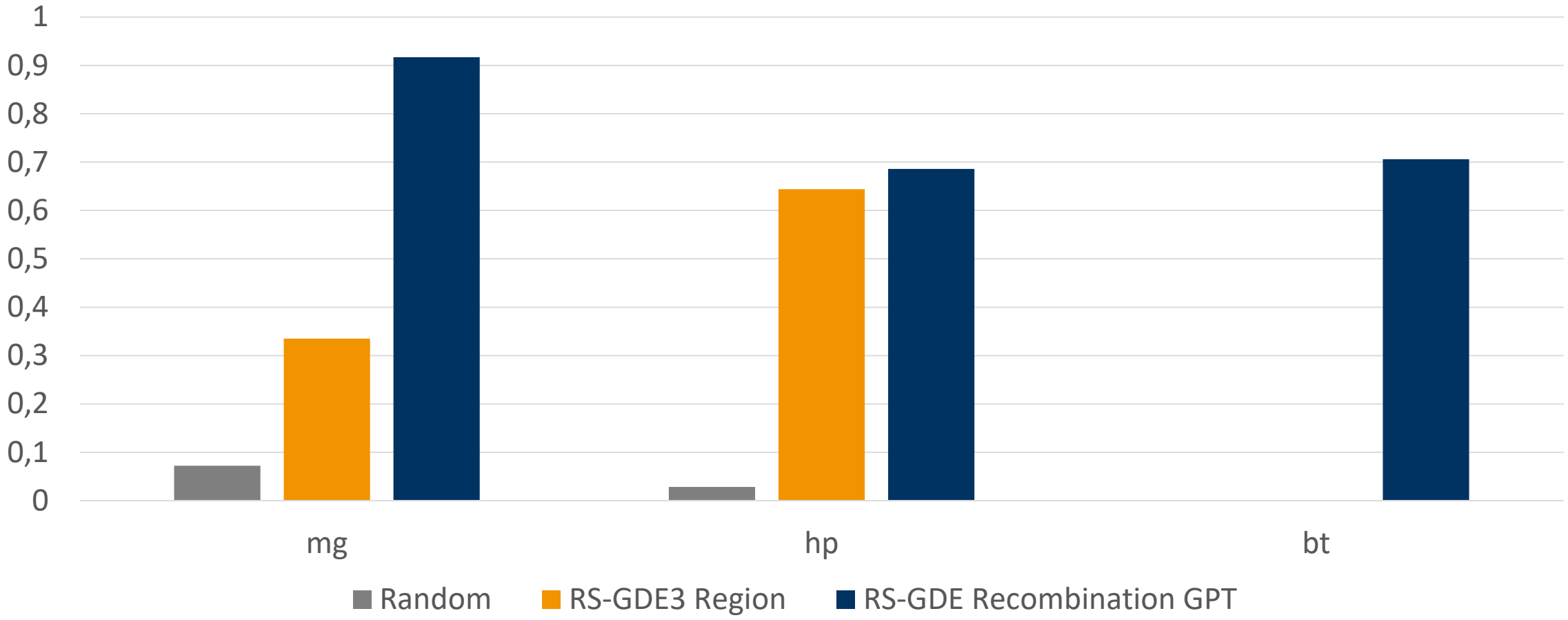
Region-aware Auto-tuning

- Most existing auto-tuners
  - global tuning finds fixed parameter values for the entire program
  - assumes that fixed parameter values achieve best performance across all regions
- Programs may consist of many code regions
  - code region specific parameter values may yield best performance
  - changing parameter values implies overhead
  - region performance may be inter-dependent
- Region based auto-tuning increases search space

Recombination Algorithm



Region-based Hypervolume Comparison



Conclusion

- motivated **multi-objective** auto-tuning with compilers
- discussed a suitable **generic optimizer** (RS-GDE3) for arbitrary numbers of objectives
- demonstrated its **effectiveness** on tuning multiple parallel code regions for three objectives (runtime, efficiency, energy)
- Region-aware auto-tuner code versions
  - up to **7x faster,**
  - up to **10x less energy consuming**, or
  - up to **61x more efficient** than the parallel, unoptimized version

DPS
Distributed and Parallel Systems

universität innsbruck
Department of Computer Science

FZ HPC

- www.uibk.ac.at/fz-hpc
- https://www.insieme-compiler.org
- https://dps.uibk.ac.at

- Ref: Juan J. Durillo, Philipp Gschwandtner, Klaus Kofler, Thomas Fahringer: ***Multi-objective region-aware optimization of parallel programs***. J. of Parallel Computing, Vol. 83, p. 3-21, Elsevier, 2019.