Parallel Algorithm Configuration

Frank Hutter, Holger Hoos, Kevin Leyton-Brown

University of British Columbia, Vancouver, Canada

Algorithm configuration

Most heuristic algorithms have parameters

- E.g. IBM ILOG CPLEX:
  - Preprocessing, underlying LP solver & its parameters, types of cuts, etc.
  - 76 parameters: mostly categorical + some numerical

Automatically find good instantiation of parameters

Related work on parameter optimization

Optimization of numerical algorithm parameters

- Population-based, e.g. CMA-ES [Hansen et al, '95-present]
- Model-based approaches: SPO [Bartz-Beielstein et al., CEC'05]
- Experimental Design: CALIBRA [Adenso & Laguna, OR'06]

General algorithm configuration (also categorical parameters)

- Racing: I/F-Race [Birattari et al., GECCO'02, MH'07, EMOAA'09]
- Iterated Local Search: ParamILS [Hutter et al., AAAI'07 & JAIR '09]
- Genetic algorithms: GGA [Ansotegui et al., CP'09]
- Model-based approaches: SMAC [Hutter et al., LION'11]

Algorithm configuration works

Problem	Algorithm name and # parameters	Speedups	Reference
SAT	Spear (26)	× 4.5 – 500	[Hutter et al., '07b]
SAT	SATenstein (41)	× 1.6 – 218x	[KhudaBukhsh et al., '09]
Most probable explanation (MPE)	GLS+ (5)	≥ × 360	[Hutter et al., '07a]
MIP	CPLEX (76)	× 2 – 52	[Hutter et al., '10]
AI Planning	LPG (62)	× 3 – 118	[Vallati et al., '11]

Can parallelization speed up algorithm configuration?

Multiple independent runs of the configurator

- Our standard methodology for using ParamILS
- Here: first systematic study of this technique's effectiveness

Parallelism within a single configuration run

- GGA [Ansotegui et al, CP'09]
  - Evaluates 8 configurations in parallel & stops when one finishes
- BasicILS variant of ParamILS [Hutter et al, JAIR'09]
  - Distributed target algorithm runs on a 110-core cluster
- Here: a new distributed variant of SMAC: d-SMAC

Overview

- Multiple independent configuration runs: an empirical study
- Distributed variant of model-based configuration: d-SMAC
- Conclusions

Overview

- Multiple independent configuration runs: an empirical study
- Distributed variant of model-based configuration: d-SMAC
- Conclusions

Parallelization by multiple independent runs

Many randomized heuristic algorithms have high variance

- Some runs perform much better than others (different random seeds)

We can exploit that variance!

- Multiple independent runs in sequence: random restarts
- Multiple independent runs in parallel
  - Run multiple copies of an algorithm in parallel & return best result
  - Perfect speedups for exponential runtime distributions [Hoos & Stützle, AIJ'99]
  - Can reduce expected runtime even on a single core [Gomes & Selman, AIJ'01]

Multiple independent runs of configurators

Our standard methodology for using ParamILS

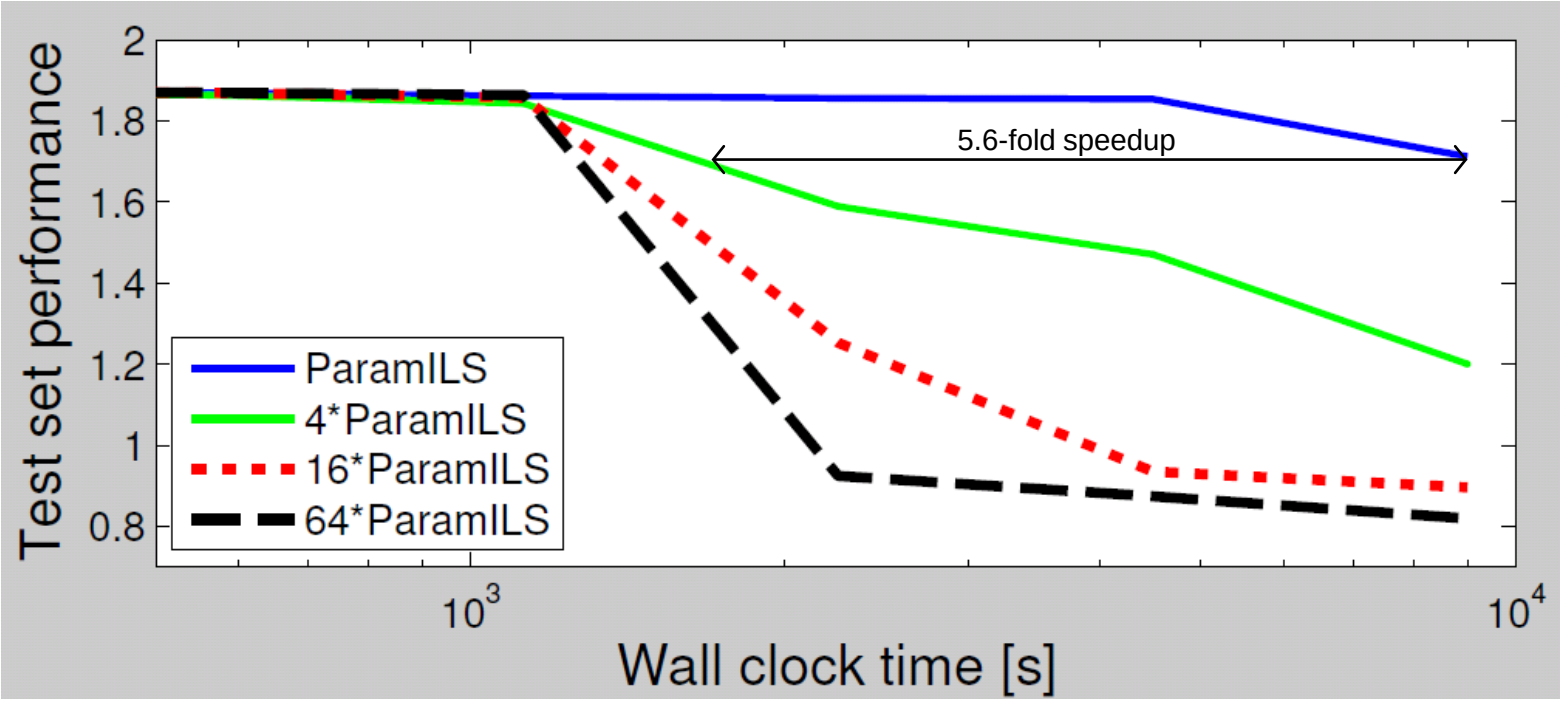
- Perform 10 to 25 parallel ParamILS runs
- Select the run with the best training (or validation) performance

How much do we gain by performing these parallel runs?

Experimental Setup

- 5 configuration scenarios from previous work
  [Hutter et al., CPAIOR'10]
  - Optimize solution quality that CPLEX achieves in a fixed time limit
  - 2010: ParamILS achieved substantial improvements
  - Side effect of this paper: SMAC & d-SMAC even a bit better
- We studied k × ParamILS, k × SMAC, k × d-SMAC
  - 200 runs for each underlying configurator on each scenario
  - To quantify performance of one run of (e.g.) k × ParamILS:
    - Draw bootstrap sample of k runs from the 200 ParamILS runs
    - Out of these k runs, pick the one with best training performance
    - Return its test performance

Example speedups for k × ParamILS



Configuration scenario: Regions 200

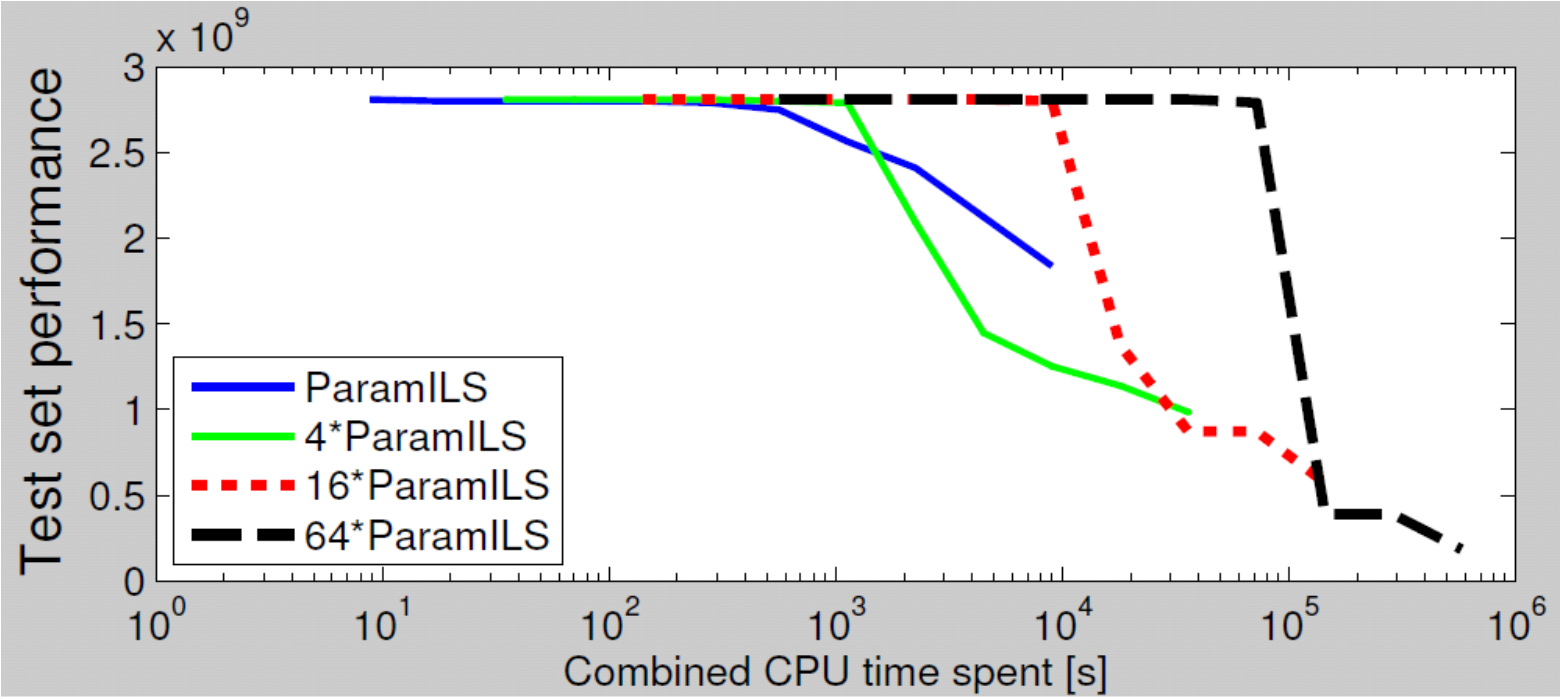
Small (or no) speedup for small time budgets

- Each run starts with the default configuration

Substantial speedups for large time budgets

- 5.6-fold speedup from 4-fold parallelization?

Utilization of total CPU time spent



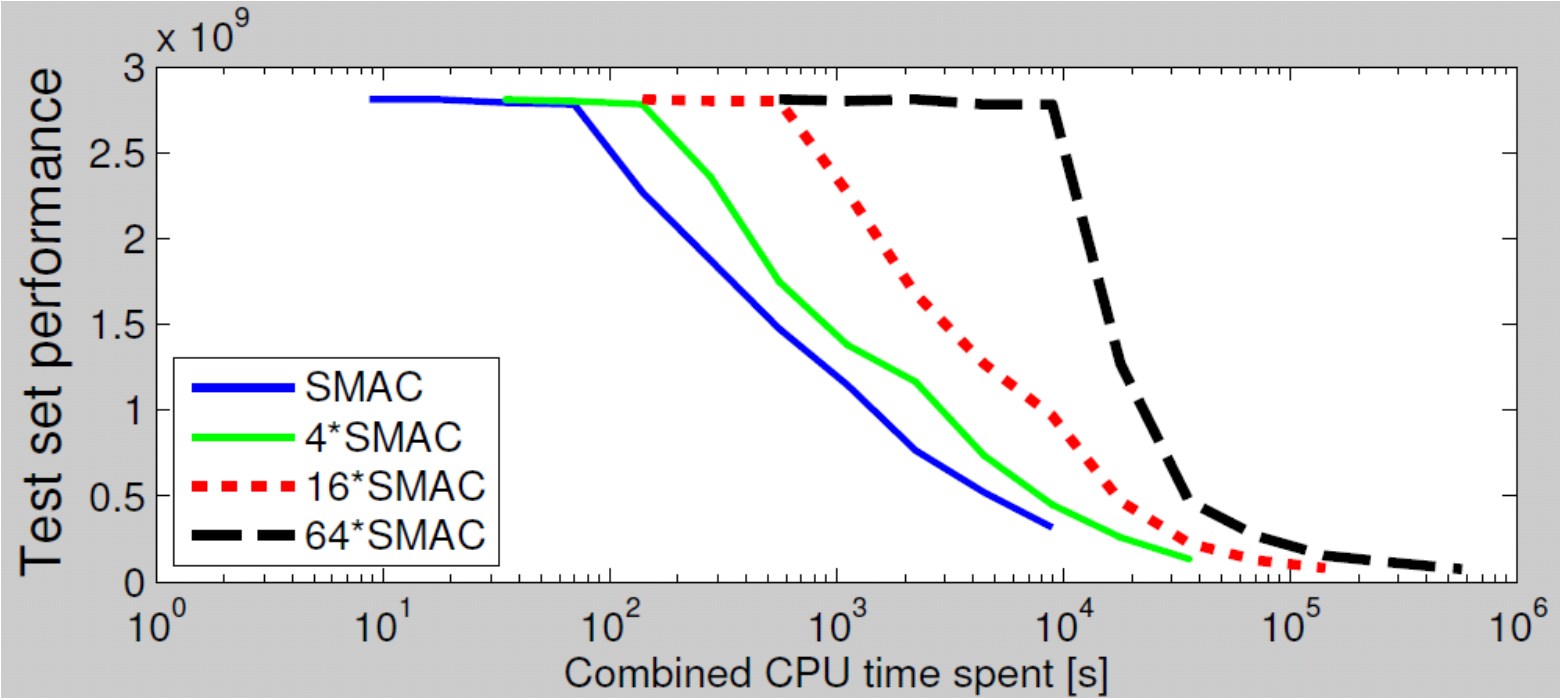
Configuration scenario: CORLAT

4×ParamILS often better than 1×ParamILS, even on a single core

- I.e. > 4-fold wall clock speedups with k=4

Almost perfect speedups up to k=16; then diminishing returns

Utilization of total CPU time spent



Configuration scenario: CORLAT

Multiple independent runs are not as effective in SMAC

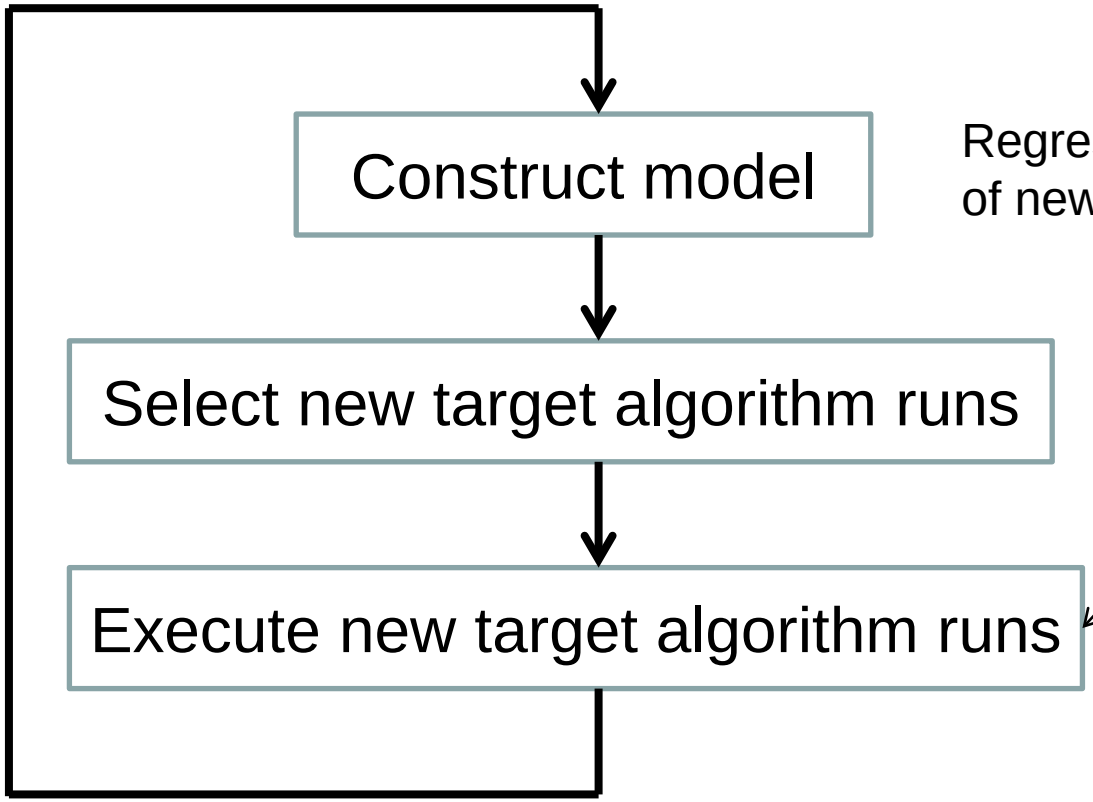
- SMAC is more robust [Hutter et al., LION'11]
- It has lower variance
- Parallelization by independent runs doesn't help as much

Overview

- Multiple independent configuration runs: an empirical study
- Distributed variant of model-based configuration: d-SMAC
- Conclusions

SMAC in a nutshell

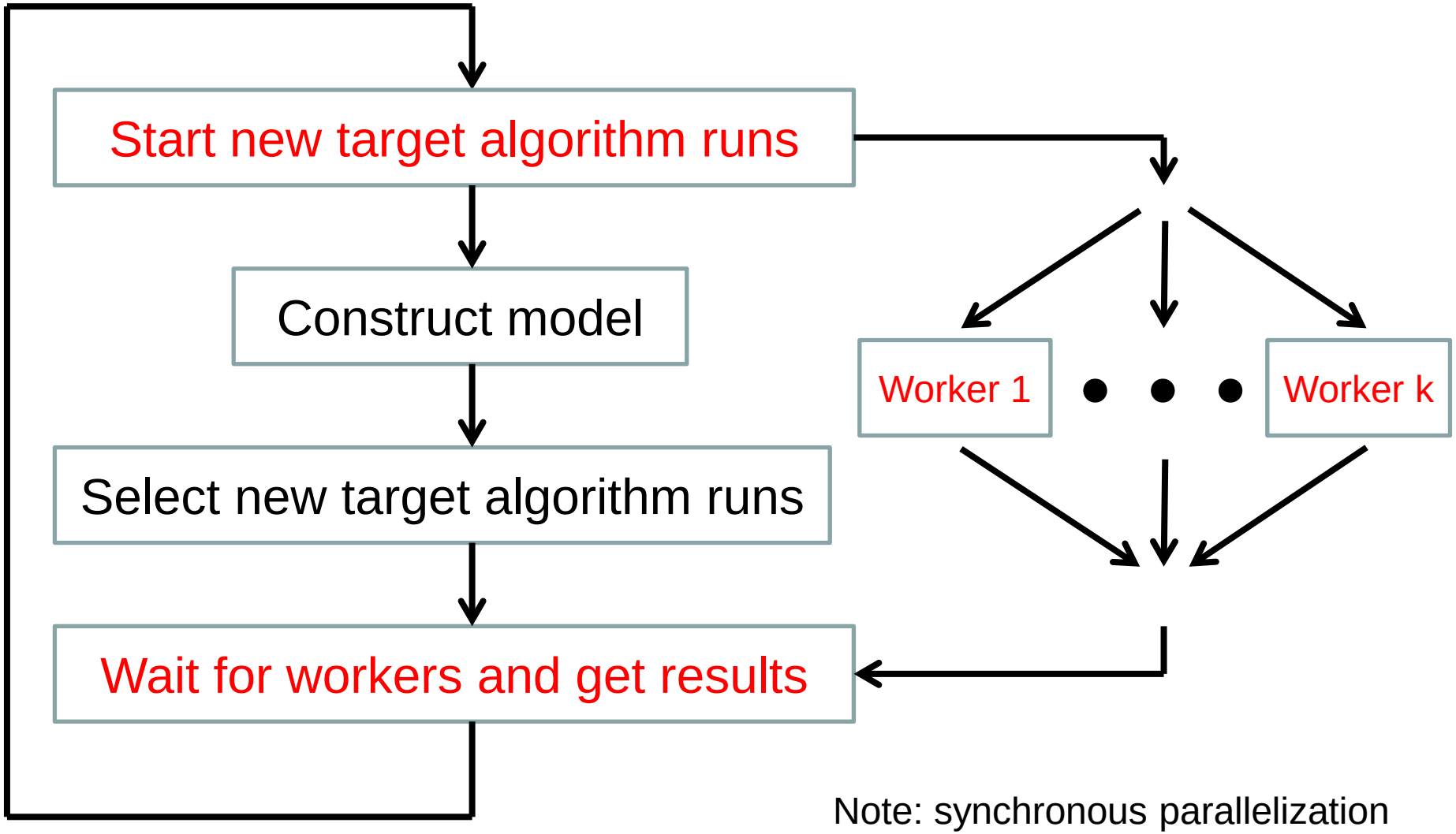
Algorithm performance data:
(configuration, instance, performance) tuples



Regression model predicts performance of new (configuration, instance) pairs

Could parallelize this stage. But the two other sequential steps would become chokepoints

Control flow in distributed SMAC



Note: synchronous parallelization

Selecting multiple promising configurations

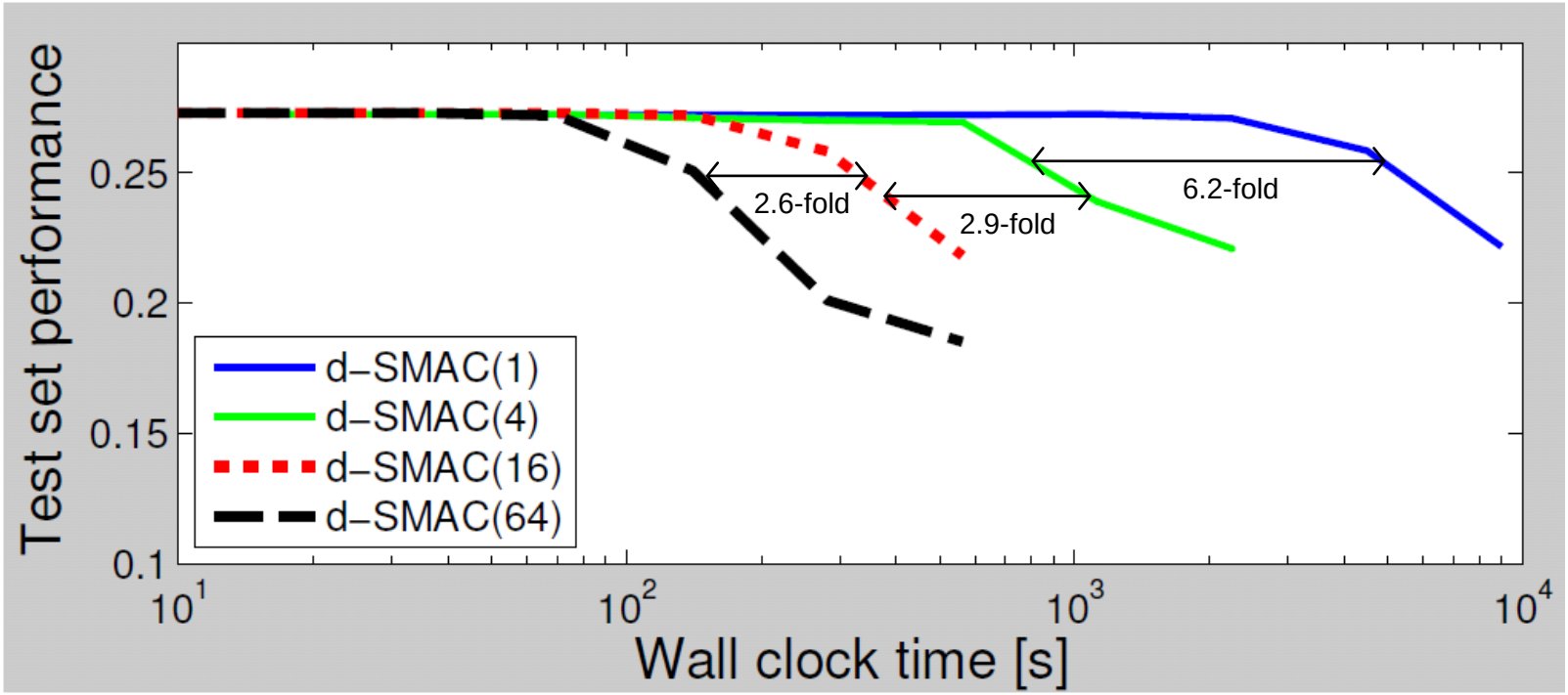
We leverage existing work on parallelizing model-based optimization

- Simple criterion from [Jones, '01]
  - Yields a diverse set of configurations

  (detail for experts only: we minimize μ-$\lambda\sigma$ with sampled values of λ)

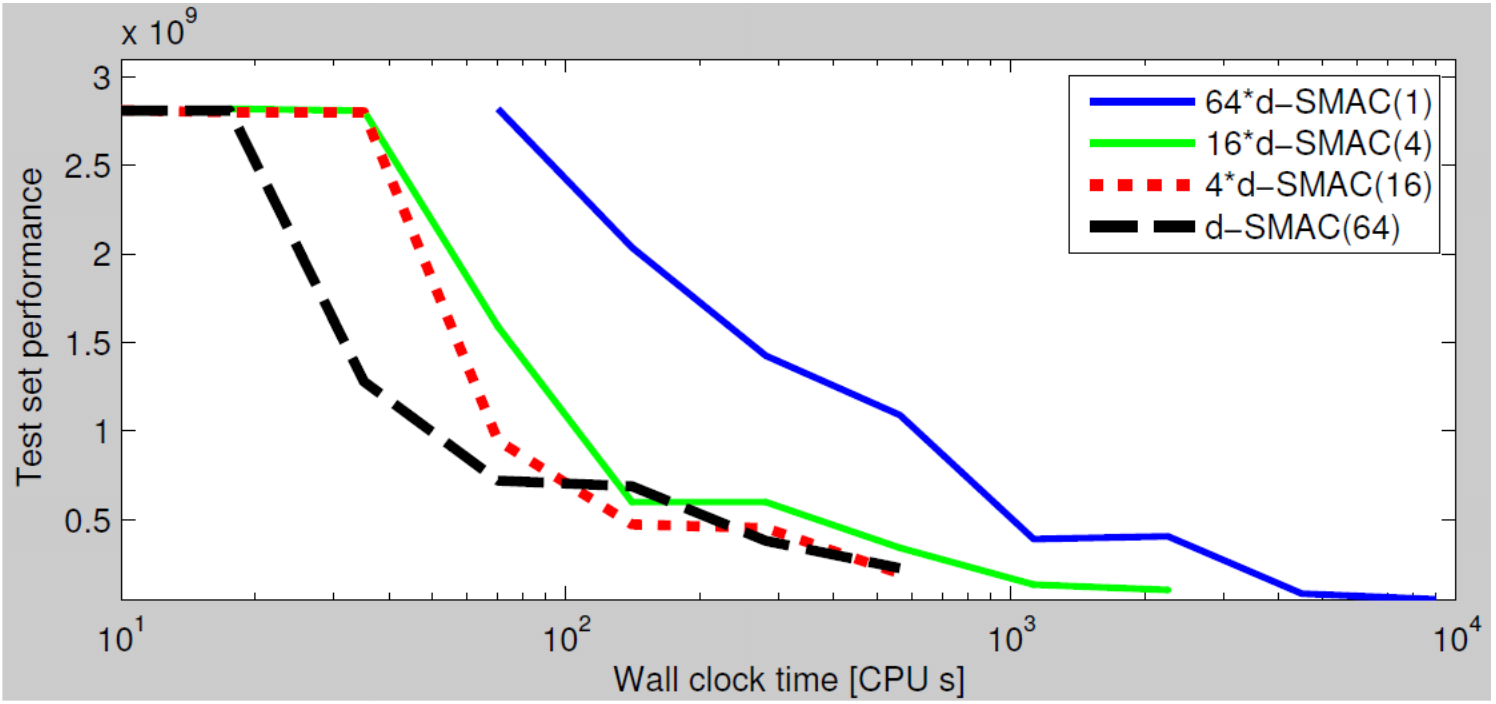
- Other approaches could be worth trying
  - E.g. [Ginsbourger, Riche, Carraro, '10]
  - Very related talk tomorrow @ 11:55am:
    Expected improvements for the asynchronous parallel global optimization of expensive functions: potentials and challenges
    Janis Janusevskis, Rodolphe Le Riche, and David Ginsbourger

d-SMAC with different numbers of workers



- Speedups even for short runs!
- Almost perfect speedups with up to 16 workers
- Overall speedup factor with 64 workers: 21× – 52×
  - Reduces 5h run to 6 – 15 min

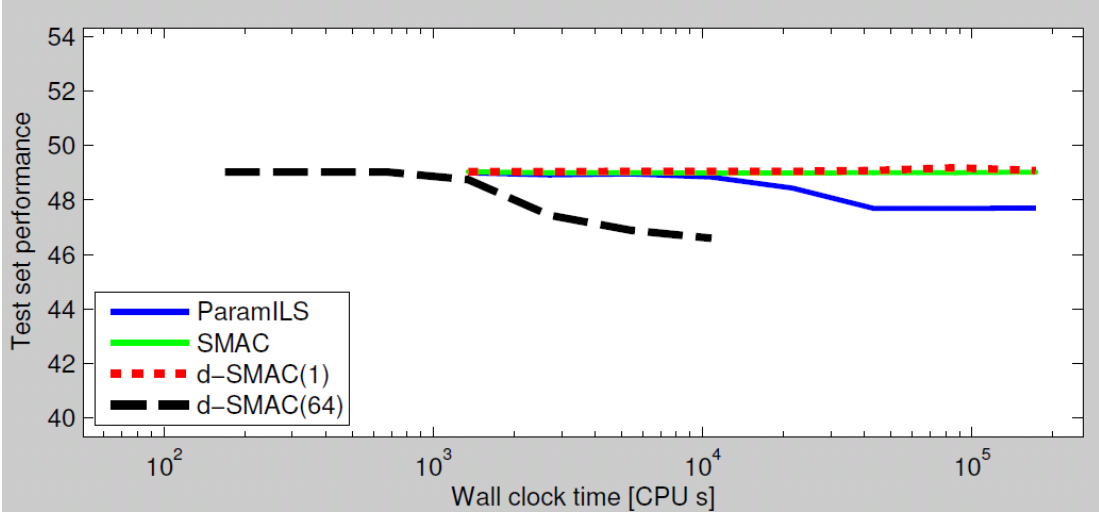
Should we perform independent runs of d-SMAC?



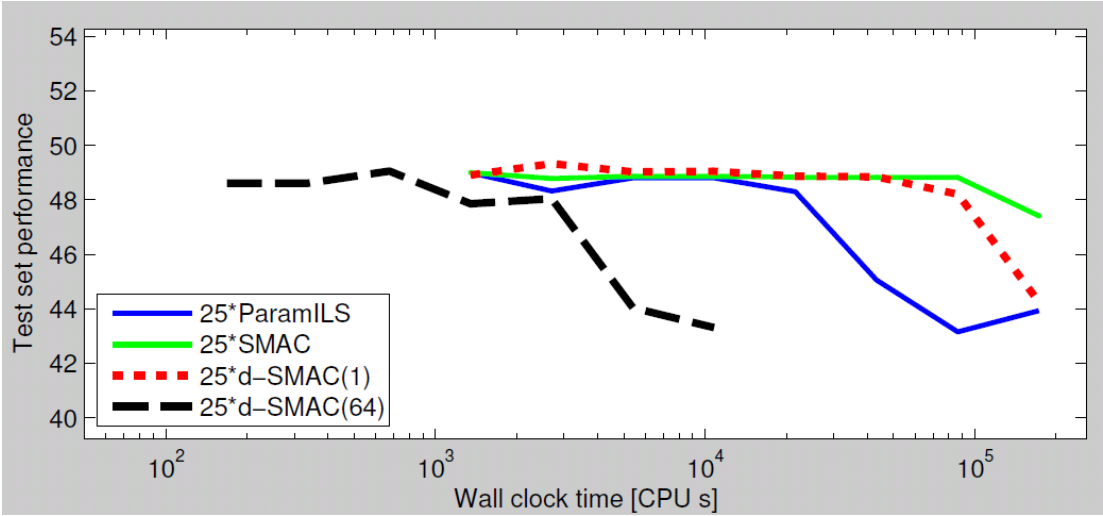
- Typically best to use all cores in a single d-SMAC(64) run
- 4 × d-SMAC(16) comes close:
  no statistical difference to 1 × SMAC(64) in 3 of 5 scenarios

Experiments for a harder instance distribution

d-SMAC(64) takes 40 minutes to find better results than the other configurators in 2 days



25×d-SMAC(64) takes 2 hours to find better results than 25×ParamILS in 2 days



Conclusion

Parallelization can speed up algorithm configuration

- Multiple independent runs of configurators
  - Larger gains for high-variance ParamILS than lower-variance SMAC
  - 4 × ParamILS better than 1 × ParamILS even on a single CPU
  - Almost perfect speedups with up to 16 × ParamILS
  - Small time budgets: no speedups
- Distributing target algorithm runs in d-SMAC
  - Almost perfect speedups with up to 16 parallel workers
    - Even for short d-SMAC runs
  - Up to 50-fold speedups with 64 workers
    - Reductions in wall clock time: 5h → 6 min -15 min
      2 days → 40min - 2h

Future Work

Asynchronous parallelization

- Required for runtime minimization,
  where target algorithm runs have vastly different runtimes

Ease of use

- We needed to start cluster workers manually
- Goal: direct support for clusters, Amazon EC2, HAL, etc.