Automatic Tuning of Compiler Options using irace

Manuel López-Ibáñez
manuel.lopez-ibanez@manchester.ac.uk

University of Manchester, UK

Leslie Pérez Cáceres, Federico Pagnozzi, Alberto Franzin, Thomas Stützle

IRIDIA, Université libre de Bruxelles, Belgium

GNU Tools Cauldron 2018, Manchester, UK

Who am I?

- Lecturer (Assistant Professor) at University of Manchester

Business Analytics, Operations Research, Decision-Making, Mathematical Optimisation, Evolutionary Computation, Machine Learning, Routing, Planning, Scheduling, Manufacturing, ...

Nothing to do with compilers or code optimisation

Who am I?

- 2006-04-24 19:16 Google SoC Project proposal: `-Wcoercion` option
- 366 commits to `/trunk`
  - `-Wconversion`
  - `-Wnull-dereference`
  - `-Wpedantic`
  - `c-c++-common/` testsuite
  - Fix `dg-error` and `dg-warning`
  - Move `gfortran` to common diagnostics
  - `LangEnabledBy`, `EnabledBy` in `.opt` files
  - Remove %H and %J from diagnostics
  - `permerror()`
  - First implementation of caret line (`-fdiagnostics-show-caret`)
  - `-fdiagnostics-color` (with Jakub)
  - `-fdiagnostics-show-option` by default
  - `-fshow-column` by default (with Aldy)
  - ... lots of other diagnostic fixes
  - https://gcc.gnu.org/wiki/ FAQ, EasyHacks, BetterDiagnostics, DiagnosticsGuidelines, ClangDiagnosticsComparison, GettingStarted (Contributing to GCC in 10 easy steps)

Tuning of GCC Options

- Optimisation options:

  171 `-flags` (categorical) + 193 `--params` (numerical)

- Given the source code of an executable (target algorithm) what is the best configuration of these options to compile it?
- Speed-up measured over a number of different *instances*:

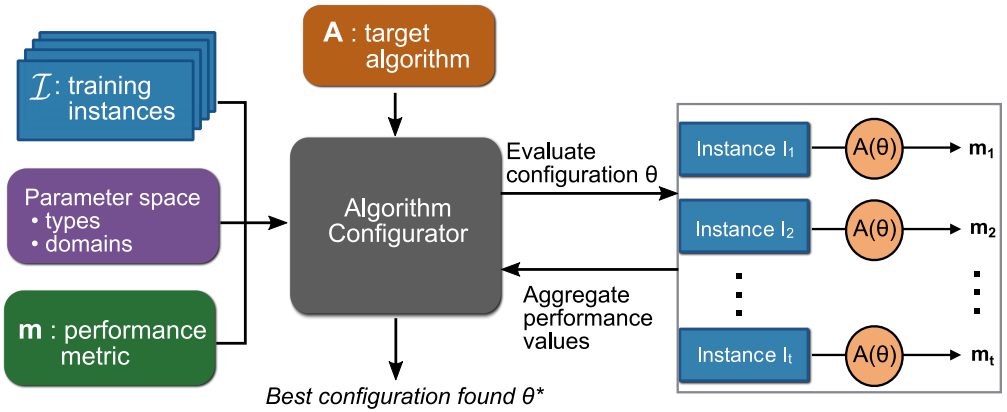
  random seeds, inputs, workloads, machines, etc.

A mathematical optimisation problem! ☺ ☺ ☺

A problem with many names:

automatic algorithm configuration, parameter tuning, hyper-parameter optimisation (AutoML), hyper-heuristics, programming by (mathematical) optimisation, ...

Automatic algorithm configuration [Birattari, 2009]



The irace Package

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari.
The irace package: Iterated Racing for Automatic Algorithm Configuration.
Operations Research Perspectives, 3:43–58, 2016. doi:10.1016/j.orp.2016.09.002
http://iridia.ulb.ac.be/irace

- Implementation of (Elitist) Iterated Racing in R
  - Goal 1: Flexible
  - Goal 2: Easy to use
- GPL package available at CRAN (GNU/Linux, Windows, OSX)
- Use it through the command-line: (see `irace --help`)

```
irace --max-experiments 1000 --param-file parameters.txt
```

✔ No knowledge of R needed

- Google group: https://groups.google.com/forum/#!forum/irace-package

Other compiler tuning methods

- ACOVEA [Ladd, 2000]
- TACT [Plotnikov et al., 2013]
- OpenTuner [Ansel et al., 2014]
- https://github.com/ctuning/ck/wiki/Compiler-autotuning
- ...

Benefits over irace

- Easier to specify build setup
- Error handling
- Out-of-the-box support for multiple target-boards
- ... and more specific features for compiler tuning

Benefits of irace

✔ **Elitist racing**:
- Automatically adjust number of instances needed
- Generalize over instances (best mean, best rank, ...)

✔ Plain-text definition of categorical and numerical parameters
+ conditional parameters: `--foo only if --bar == "x"`
+ forbidden configurations `--foo == "x" || --bar == "y"`

✔ **Parallel evaluation**: MPI, multiple cores, batch job clusters (SGE, PBS, Torque, Slurm)

✔ Adaptive capping of long running executions

✔ Still improving ⇒ current version 3.1

Up to 1.40× speed-up over -O3 [Pérez Cáceres et al., 2018]

References I

J. Ansel, S. Kamil, K. Veeramachaneni, J. Ragan-Kelley, J. Bosboom, U. M. O'Reilly, and S. Amarasinghe. Opentuner: An extensible framework for program autotuning. In *Proceedings of the 23rd International Conference on Parallel Architectures and Compilation*, pages 303–315. ACM New York, NY, USA, 2014. doi: 10.1145/2628071.2628092.

M. Birattari. *Tuning Metaheuristics: A Machine Learning Perspective*, volume 197 of *Studies in Computational Intelligence*. Springer, Berlin/Heidelberg, Germany, 2009. doi: 10.1007/978-3-642-00483-4.

S. R. Ladd. ACOVEA (Analysis of compiler options via evolutionary algorithm). https://github.com/Acovea/libacovea, 2000.

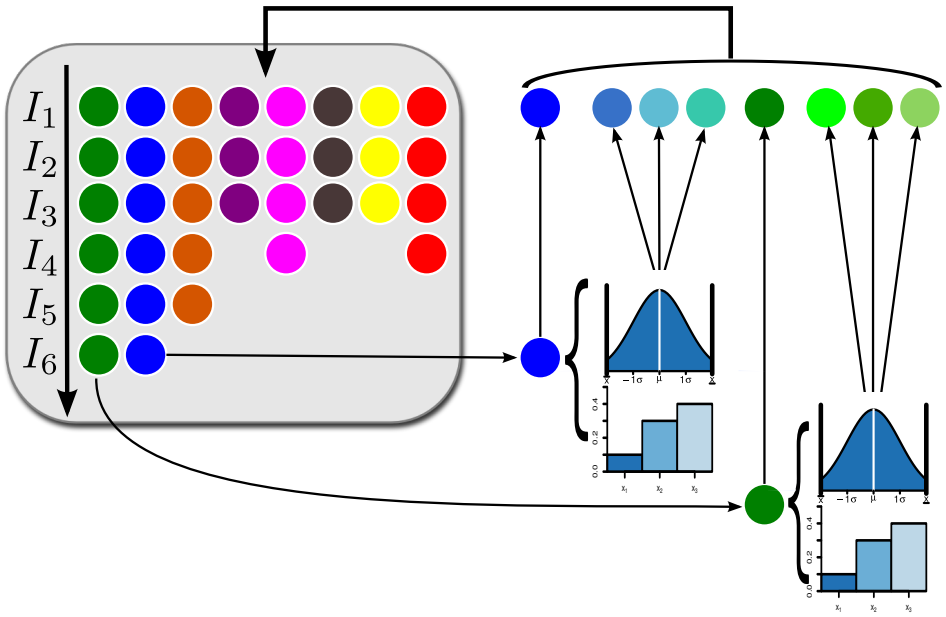
M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, and M. Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016. doi: 10.1016/j.orp.2016.09.002.

L. Pérez Cáceres, F. Pagnozzi, A. Franzin, and T. Stützle. Automatic configuration of GCC using irace. In E. Lutton, P. Legrand, P. Parrend, N. Monmarché, and M. Schoenauer, editors, *EA 2017: Artificial Evolution*, volume 10764 of *Lecture Notes in Computer Science*, pages 202–216. Springer, Heidelberg, Germany, 2018.

D. Plotnikov, D. Melnik, M. Vardanyan, R. Buchatskiy, R. Zhuykov, and J.-H. Lee. Automatic tuning of compiler optimizations and analysis of their impact. In V. Alexandrov, M. Lees, V. Krzhizhanovskaya, J. Dongarra, and P. M. Sloot, editors, *2013 International Conference on Computational Science*, volume 18 of *Procedia Computer Science*, pages 1312–1321. Elsevier, 2013. doi: 10.1016/j.procs.2013.05.298.

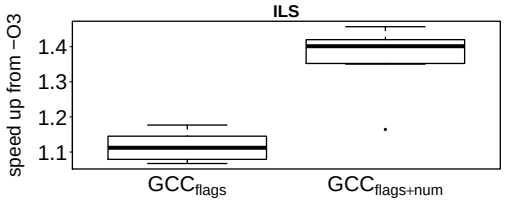
irace

[López-Ibáñez et al., 2016]

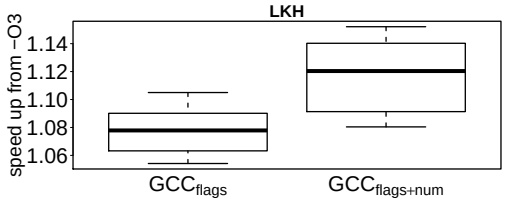


Experimental results

[Pérez Cáceres et al., 2018]



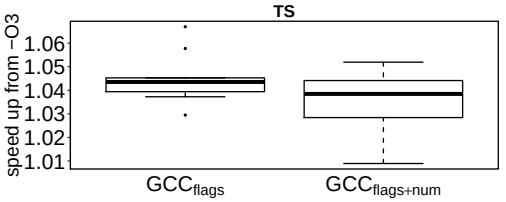
Iterated local search for the TSP (C)



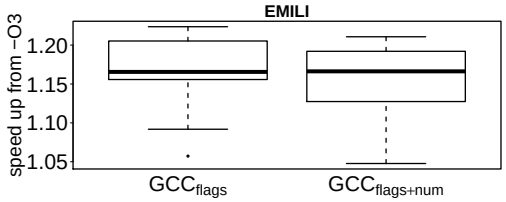
LKH heuristic for the TSP (C)

Experimental results

[Pérez Cáceres et al., 2018]



Tabu search applied to the QAP (C)



Moderately large application (C++)