The **irace** Software Package

Manuel López-Ibáñez

manuel.lopez-ibanez@ulb.ac.be

COMEX Workshop on Practical Automatic Algorithm Configuration

The irace Package

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. **The irace package, Iterated Race for Automatic Algorithm Configuration.** *Technical Report TR/IRIDIA/2011-004*, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.
http://iridia.ulb.ac.be/irace

- Implementation of Iterated Racing in R

  Goal 1: Flexible

  Goal 2: Easy to use

The irace Package

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. **The irace package, Iterated Race for Automatic Algorithm Configuration.** *Technical Report TR/IRIDIA/2011-004*, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.
http://iridia.ulb.ac.be/irace

- R package available at CRAN:

  http://cran.r-project.org/package=irace

```
R> install.packages("irace")
```

The irace Package

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. **The irace package, Iterated Race for Automatic Algorithm Configuration.** *Technical Report TR/IRIDIA/2011-004*, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.
http://iridia.ulb.ac.be/irace

- Use it from inside R ...

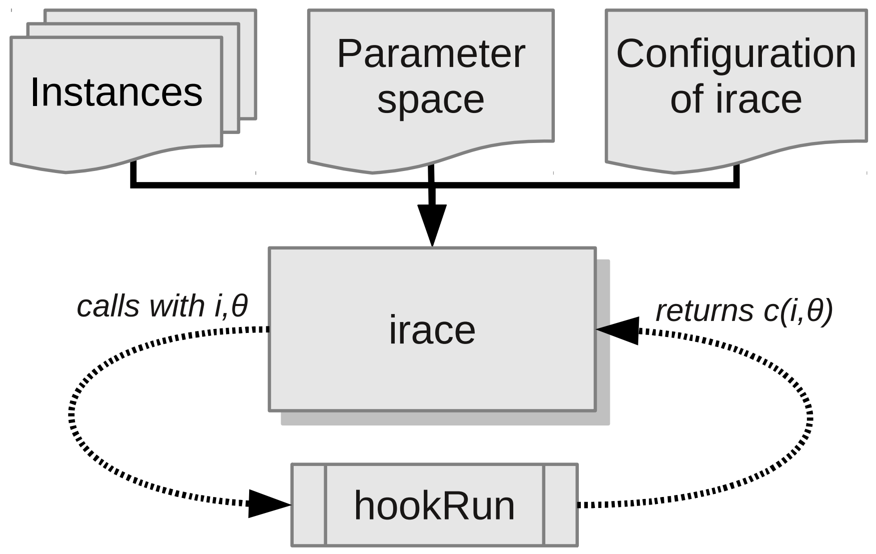
```
R> result <- irace(tunerConfig = list(maxExperiments = 1000),
                   parameters = parameters)
```

- ...or through command-line: (See `irace --help`)

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

✔ No knowledge of R needed

The irace Package



The irace Package: Instances

- TSP instances

```
$ dir Instances/
3000-01.tsp 3000-02.tsp 3000-03.tsp ...
```

- Continuous functions

```
$ cat instances.txt
function=1 dimension=100
function=2 dimension=100
...
```

- Parameters for an instance generator

```
$ cat instances.txt
I1 --size 100 --num-clusters 10 --sym yes --seed 1
I2 --size 100 --num-clusters 5 --sym no --seed 1
...
```

- Script / R function that generates instances
  ☞ if you need this, tell us!

The irace Package: Parameter space

- Categorical (c), ordinal (o), integer (i) and real (r)
- Subordinate parameters (| condition)

$ cat parameters.txt

```
# Name       Label/switch        Type   Domain            Condition
LS           "--localsearch "     c     {SA, TS, II}
rate         "--rate="            o     {low, med, high}
population   "--pop "             i     (1, 100)
temp         "--temp "            r     (0.5, 1)          | LS == "SA"
```

- For real parameters, number of decimal places is controlled by option *digits* (--digits)

The irace Package: Options

- *digits*: number of decimal places to be considered for the real parameters (default: 4)
- *maxExperiments*: maximum number of runs of the algorithm being tuned (tuning budget)
- *testType*: either F-test or t-test
- *firstTest*: specifies how many instances are seen before the first test is performed (default: 5)
- *eachTest*: specifies how many instances are seen between tests (default: 1)

The irace Package: hook-run

- A script/program that calls the software to be tuned:

```
./hook-run instance candidate-number candidate-parameters ...
```

- An R function:

```
hook.run <- function(instance, candidate, extra.params = NULL,
                     config = list())
{
  ...
}
```

Flexibility: If there is something you cannot tune, let us know!

Example: ACOTSP

Thomas Stützle. **ACOTSP: A software package of various ant colony optimization algorithms applied to the symmetric traveling salesman problem**, 2002.
http://www.aco-metaheuristic.org/aco-code/

- Command-line program:

```
./acotsp -i instance -t 5 --mmas --ants 10 --rho 0.95 ...
```

Goal: find best parameter settings of ACOTSP for solving random Euclidean TSP instances with $n \in [1000, 3000]$ within 1 CPU-second

Example: ACOTSP

```
$ cat parameters-acotsp.txt

# name       switch              type  values      conditions
algorithm    "--"                c (as,mmas,eas,ras,acs)
localsearch  "--localsearch "    c (0, 1, 2, 3)
alpha        "--alpha "          r (0.00, 5.00)
beta         "--beta "           r (0.00, 10.00)
rho          "--rho  "           r (0.01, 1.00)
ants         "--ants "           i (5, 100)
nnls         "--nnls "           i (5, 50)     | localsearch %in% c(1,2,3)
dlb          "--dlb "            c (0, 1)      | localsearch %in% c(1,2,3)
q0           "--q0 "             r (0.0, 1.0)  | algorithm == "acs"
rasrank      "--rasranks "       i (1, 100)    | algorithm == "ras"
elitistants  "--elitistants "    i (1, 750)    | algorithm == "eas"
```

Example: ACOTSP

```
$ cat hook-run

#!/bin/bash
INSTANCE=$1
CANDIDATENUM=$2
CAND_PARAMS=$*
STDOUT="c${CANDIDATENUM}.stdout"
FIXED_PARAMS=" --time 1 --tries 1 --quiet "
acotsp $FIXED_PARAMS -i $INSTANCE $CAND_PARAMS 1> $STDOUT
COST=$(grep -oE 'Best [-+0-9.e]+' $STDOUT |cut -d' ' -f2)
if ! [[ "${COST}" =~ ^[-+0-9.e]+$ ]] ; then
   error "${STDOUT}: Output is not a number"
fi
echo "${COST}"
exit 0
```

Example: ACOTSP

```
$ cat tune-conf

  execDir <- "./acotsp-execdir"
  instanceFile <- "./training.txt"
  maxExperiments <- 300
  digits <- 2
```

✔ Good to go:

```
$  make -C ACOTSP-1.04-tuning all
$  mkdir acotsp-execdir
$  irace
```

Questions

Exercise #1

- Setup irace for your own problem

OR

- Run the ACOTSP example:

  http://iridia.ulb.ac.be/~manuel/comex_workshop/acotsp-example.tar.gz

  - Add another default configuration and make irace use it
    See `default.txt`
  - Add another forbidden configuration and make irace use it
    See `forbidden.txt`
  - Use `--debug-level 1` to see what irace is executing

ACOTSP-VAR Example: Tuning for anytime

- Like ACOTSP, but with more parameters
- The output is now a Pareto front of (time, quality) pairs
  - Use a common reference point $(2.1, 2.1, \dots)$
  - Normalize the objectives range to $[1, 2]$ per instance without predefined maximum / minimum

☞ We need all Pareto fronts for computing the normalization!

✘ We cannot simply use `hook-run`

✔ We use `hook-evaluate` !
  - `hook-evaluate` ≈ `hook-run`
  - Executes *after* all `hook-run` for a given instance
  - Returns the cost value instead of `hook-run`

```
./hook-evaluate instance candidate-number total-candidates
```

Exercise #2

- Continue setting up irace for your own problem

OR

- Run the ACOTSP-VAR (anytime) example:

http://iridia.ulb.ac.be/~manuel/comex_workshop/acotspvar-example.tar.gz

Exercise #2

1. Compile ACOTSPvar, hv and nondominated
2. Add training instances: use the ones from the ACOTSP example
3. Examine all irace files
   - Note the use of fixed parameters in the parameters file
   - Note that hook-run does not return a value
   - Note that tune-conf needs adjusting
4. Run irace and try to understand what it is doing

Exercise #3

Analyzing the results of irace:

```
 $ R
R> load("acotsp-execdir/irace.Rdata")
R> names(tunerResults)
R> print(tunerResults$experiments)
R> print(tunerResults$allCandidates)
```