CSE/ECE 848 Introduction to Evolutionary Computation

Module 3, Lecture 13, Part 3

Metaheuristics and Automatically Tuned EC Methods

Kalyanmoy Deb, ECE

Koenig Endowed Chair Professor

Automatic Algorithm Configuration: irace

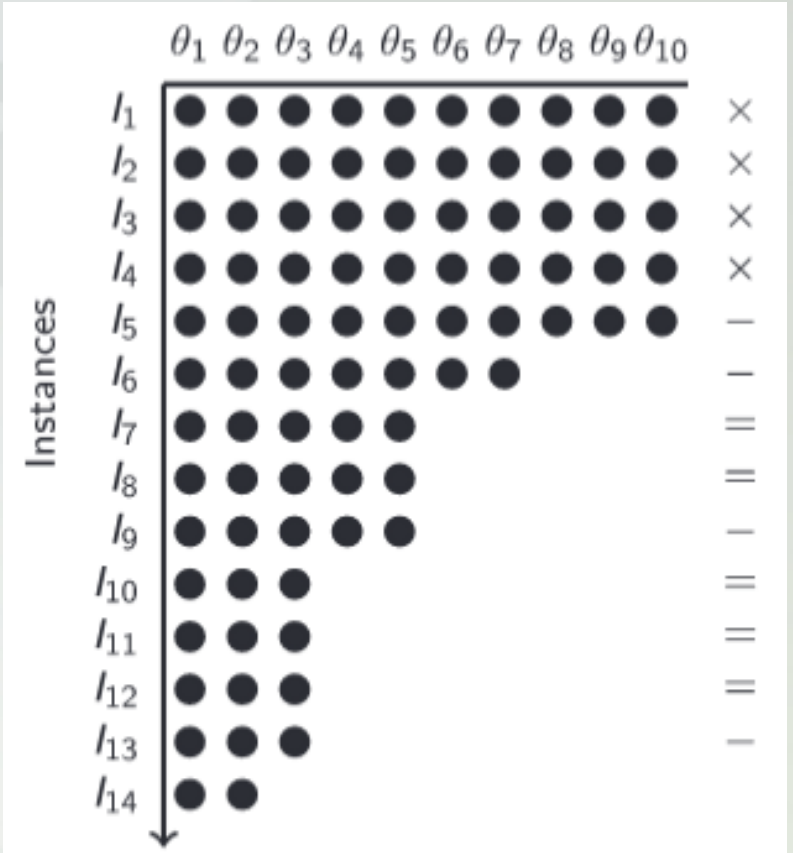
- Part 2:
  - Generic Automatic Parameter Tuning in EC Methods
- Part 3 (This Part):
  - A specific approach – irace in detail
  - Hyperopt, OPTUNA

Past Approaches

- Numerical Optimization
  - CMA-ES (Hansen and Ostermeier, 2001), MADS (Audet and Orban, 2006)
- Heuristic Search
  - ParamLS (Hutter et al., 2009), GP (Fukunaga, 2002), Gender-based EA (Sellman et al., 2010)
- ANOVA, experimental design
  - CALIBRA (Adenso-Diaz and Laguna, 2006)
- Response Surface Methods
  - SPO (Bartz-Beielstein, 2006), SMAC (Hutter, Hoos, Leyton-Brown, 2011)
- Sequential Statistical Testing
  - F-Race (Birratari et al., 2002)

Racing Approach and F-Race

- Start with initial candidates (θ)
- Consider a stream of instances (π)
- Sequentially evaluate candidates
  - x: no statistical test performed
  - -: test discarded at least one θ
  - =: test did not discard any θ
- Discard inferior candidates that are statistically (Friedman test) worse than at least one θ
- Repeat until a winner is selected or until computation time limit is met



$T_{first} = 4, T_{each} = 1$

Reference: M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle. (2010). F-race and iterated F-race: An overview. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, (Eds.) Experimental Methods for the Analysis of Optimization Algorithms, pages 311–336. Springer, Berlin.

Iterated Race (irace)

- Input: $\mathfrak{T}, \mathcal{X}, \mathcal{C}, \mathcal{B}$
- #Iter: $N^{iter} = \lfloor 2 + \log_2 N^{param} \rfloor$
- Each iteration uses a budget
  - $B_j = (B - B^{used})/(N^{iter} - j + 1)$
  - Starts with parameters in Θ_j
  - $|\Theta_j| = \lfloor B_j/(T_{first} + T_{each} \min(5, j)) \rfloor$
  - More budget with less config with j
- *Race* selects best config. based on paired t-test
- *Sample* a new config around Θ^{elite} using truncated normal distr. with reducing std-dev
  - Discrete and categorical parameters are handled differently

Require: $I = \{I_1, I_2, \ldots\} \sim \mathcal{I}$,
parameter space: X,
cost measure: $C(\theta, i) \in \mathbb{R}$,
tuning budget: B

1: $\Theta_1 = \text{SampleUniform}(X)$
2: $\Theta^{\text{elite}} = \text{Race}(\Theta_1, B_1)$
3: $j = 1$
4: **while** $B^{\text{used}} \leq B$ **do**
5: $\quad j = j + 1$
6: $\quad \Theta^{\text{new}} = \text{Sample}(X, \Theta^{\text{elite}})$
7: $\quad \Theta_j = \Theta^{\text{new}} \cup \Theta^{\text{elite}}$
8: $\quad \Theta^{\text{elite}} = \text{Race}(\Theta_j, B_j)$
9: **end while**
10: **Output:** Θ^{elite}

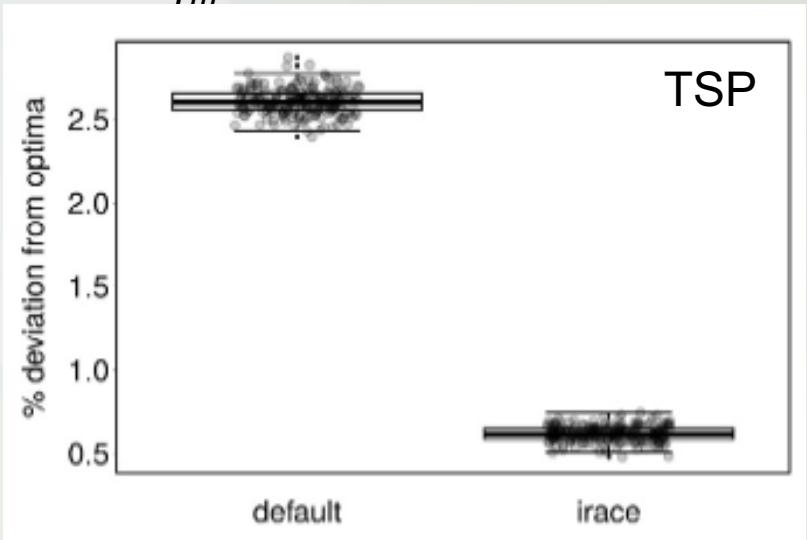
Iterated method is an optimization procedure, uses a restart

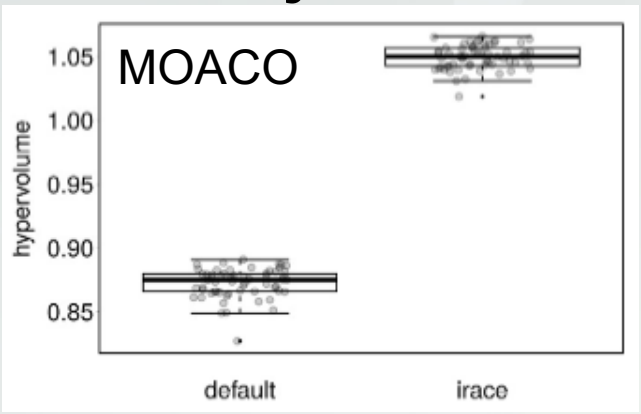
M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, and M. Birattari (2016). The irace package: Iterated Racing for Automatic Algorithm Configuration. *Operations Research Perspectives*, *3*. 43–58.

Iterated Race Features and Some Results

1. When premature convergence, use a soft-restart
2. Elitist method to keep best configurations
3. Parallel evaluation of configurations
4. Use of Forbidden Configuration supplied by user (e.g. small popsize and high tournament size)
5. Conditional parameters modified logically
   - If Polynomial Mutation is chosen, consider η_m

- Example: TSP using ACOTSP versus irace
- Multi-objective MOACO vs. irace





irace Details

- It is written in R, need to install R-package
- Irace package: https://cran.r-project.org/web/packages/irace/vignettes/irace-package.pdf
- Some features:
  - Link with any optimization algorithm
  - An algorithm can be tuned for multiple problems
  - Multiple conditional parameters can be tuned
  - Windows, Linux, MacOS
  - Parallel-irace version available
- irace descriptions
  - Target algorithm parameter description
  - Target algorithm runner (which algo?)
  - Training instances list (problems, #var)
- irace options:
  - Initial config (param)
  - Forbidden config
  - Target algo. Evaluator
    - Delayed eval.
    - Single eval. function

Other Algorithm Configurators

- Hyperopt (http://hyperopt.github.io/hyperopt/)
  - Python-based, nested approach
  - Algorithms used: (i) Random search, (ii) Tree of Parzen estimators (TPE) and (iii) Adaptive TPE
  - Parallel version available
- OPTUNA (https://optuna.org/)
  - Also in python
  - Algorithms used: (i) Parzen estimator, (ii) CMA-ES, (iii) Grid search, (iv) Random search
  - Also has the option of pruning
  - Parallel version available

References: 1. Bergstra, J., Yamins, D., Cox, D. D. (2013) Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. To appear in Proc. of the 30th International Conference on Machine Learning (ICML 2013).
2. Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta,and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In KDD.

End of Module 3, Lecture 13, Part 3

- Discussed irace method in detail
- Flexible to be used with almost any optimization algorithm
- Other algorithm configurators are available

- Before finalizing an EC, it is advised to find optimal hyperparameters using one of these automated methods