marcelo.desouza@udesc.br
http://souzamarcelo.github.io

# Capping strategies based on performance envelopes for the automatic design of meta-heuristics

**Marcelo de Souza[1,2], Marcus Ritt[1], Manuel López-Ibáñez[3]**

[1]Institute of Informatics, Federal University of Rio Grande do Sul, Brazil
[2]Department of Software Engineering, Santa Catarina State University, Brazil
[3]Alliance Manchester Business School, University of Manchester, United Kingdom

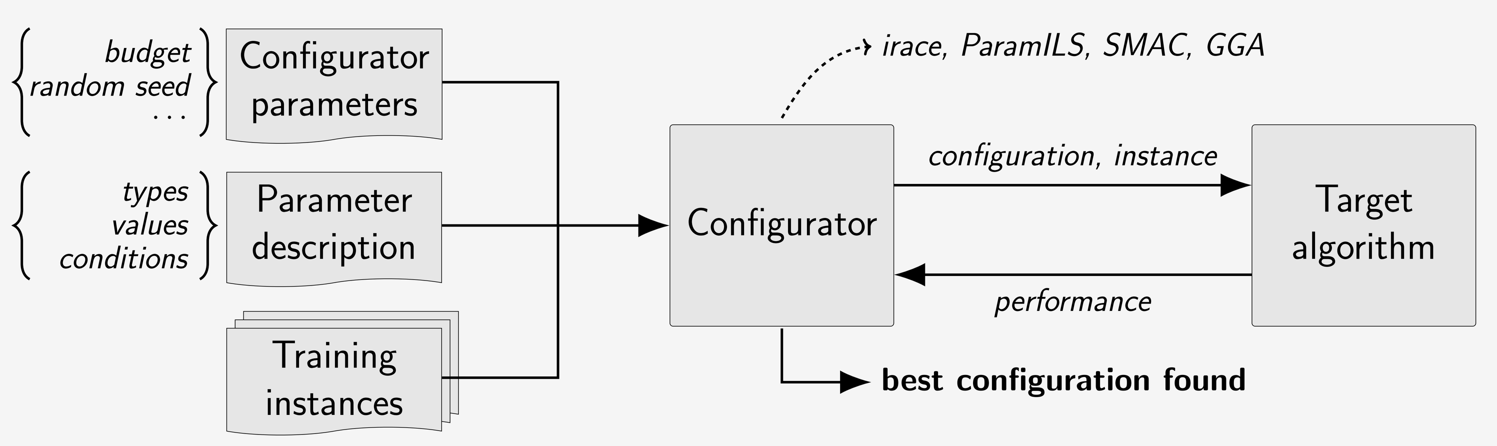
## Contextualization

We show **how to speed up parameter tuning** with no modification to the tuned software by 75%. This is achieved by **eliminating unpromising configurations early**. We analyze the performance of such **capping strategies** on different scenarios and show that they can also help to find better parameter values.

Previous work focus on capping for scenarios minimizing running time (decision algorithms):

- Hutter et al. [1]: capping strategies for ParamILS and SMAC configurators.
- Cáceres et al. [2]: capping strategies for the irace configurator.
- In a nutshell: the running time required by the best found configurations is used to calculate a cutoff time for the execution of new configurations.
- Problem: techniques not suitable for scenarios optimizing solution quality.

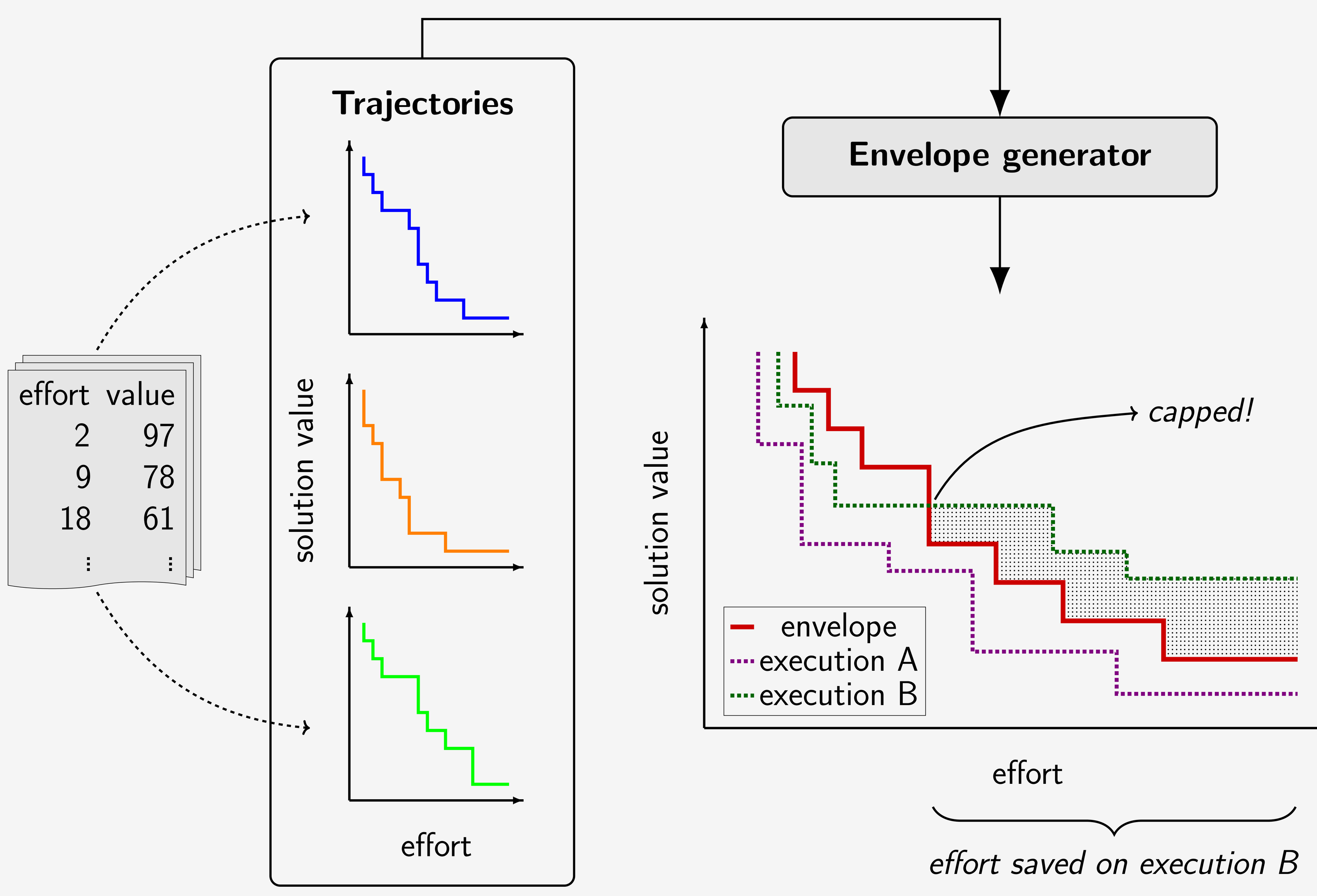
## Automatic algorithm configuration




### Using irace as configurator (iterated F-race)

- Iteratively executes F-race to evaluate candidate configurations:
  1. Sampling.
  2. Experiments.
  3. Statistical test.
  4. Update probabilistic model.
- **What if we use knowledge from previous iterations to speed up the process?**

## Envelope-based capping mechanism

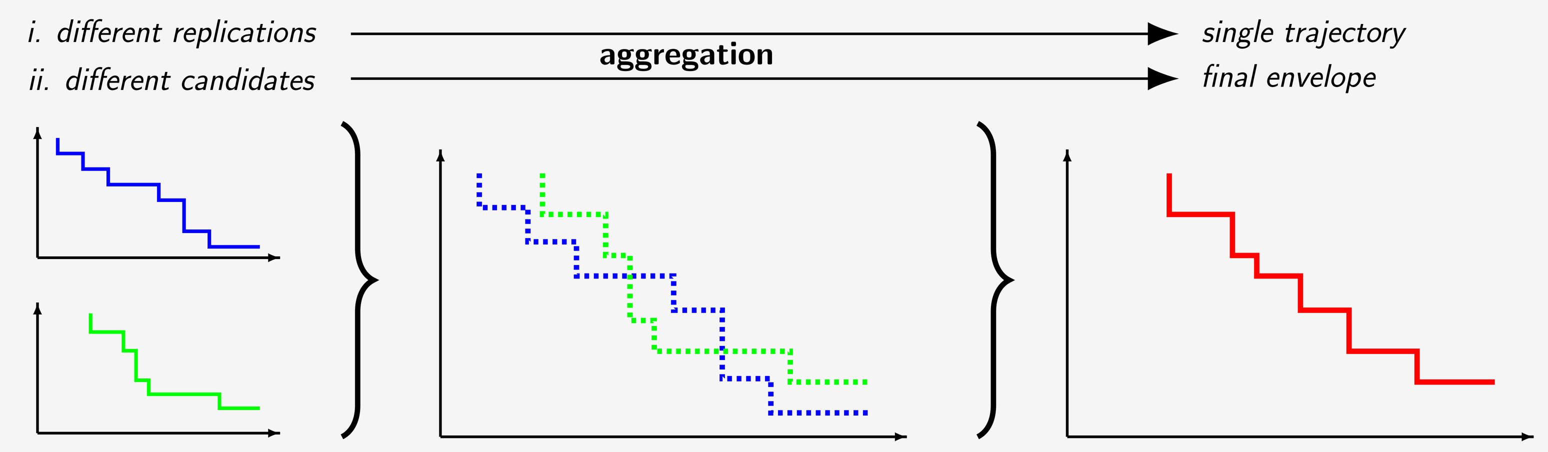



**Algorithm 1:** Envelope generator
**Input** : Trajectories $T_{ij}(t)$ for candidates $i \in [n]$ and replications $j \in [m_i]$.
**Output:** Performance envelope $E(t)$.

$R \leftarrow$ `select-references`$([n])$
**foreach** $i \in R$ **do**
  $T_i \leftarrow$ `aggregate-replications`$(T_{ij})$  $\forall j \in [m_i]$
$E \leftarrow$ `aggregate-candidates`$(T_i)$  $\forall i \in R$
**return** *performance envelope* $E$

- $[n]$ is the set of best candidate configurations found so far.
- $[m_i]$ is the set of replications of candidate $i$ on the instance at hand.
- We usually select all reference candidates $[n]$ in `select-references` procedure.
- The performance envelope $E$ defines the minimum expected quality for future executions.
- **How do we aggregate replications and candidates to determine the performance envelope?**

## Aggregation strategies




Basic strategies: for each effort value we can use the quality found by the **best**/**worst**/**average** performer. For aggregating replications, we can also use the exponential model.

### Exponential model

- Assumption: the time $t$ required by the algorithm to find a solution as good as some target value follows an **exponential distribution** [3], given by $f(t, \lambda) = 1 - e^{-\lambda t}$.
- We determine $\hat{\lambda} = \sum_{j \in [m_i]} m_i / t_{jq}$, where $t_{jq}$ is the time used to reach quality $q$ on replication $j$. If $q$ is not reached, we assume $t_{jq} = t_f \times \alpha$, where $t_f$ is the running time cutoff and $\alpha$ is a penalization constant.
- For each quality value $q$, the model gives the time required by some fraction $p$ of the replications to reach $q$ as $1 - e^{-\lambda t} = p$, so $t = -\ln(1-p) \times \bar{t}$.

## Experimental evaluation

### Configuration scenarios

| scenario | description | effort | | parameters | | | | instances |
|---|---|---|---|---|---|---|---|---|
| | | type | limit | int | cat | real | cond | |
| ACOTSP | Ant colony algorithm for TSP | time | 10 s | 4 | 3 | 4 | 5 | 50 |
| HEA-COL | HybridEA algorithm for graph coloring | checks | $10^9$ | 4 | 2 | 1 | 0 | 27 |
| AAC-UBQP | Recombination heuristic for UBQP | time | 100 s | 10 | 3 | 1 | 7 | 20 |

### Saved effort and algorithm performance

| scenario | algorithm | performance | | config. effort | |
|---|---|---|---|---|---|
| | | avg. dev | best dev | total | saved |
| ACOTSP | original | 0.9 | 0.7 | - | - |
| | AAC | 0.7 | 0.5 | 354 | - |
| | AAC-worst | **0.5** | **0.3** | 270 | 23.7 |
| | AAC-best | 1.1 | 0.8 | **200** | **43.5** |
| | AAC-exp | 0.9 | 0.6 | 275 | 22.3 |
| HEA-COL | original | 4.9 | 4.6 | - | - |
| | AAC | 4.1 | 3.8 | 3.01 | - |
| | AAC-worst | 4.2 | 3.9 | 2.55 | 15.1 |
| | AAC-best | **3.8** | **3.6** | **2.22** | **26.2** |
| | AAC-exp | 4.1 | 3.9 | 2.63 | 12.5 |
| AAC-UBQP | original | 282.5 | 100.8 | - | - |
| | AAC | 204.9 | **72.7** | 3380.1 | - |
| | AAC-worst | 373.4 | 167.7 | 1606.7 | 52.5 |
| | AAC-best | 405.9 | 248.2 | **827.6** | **75.5** |
| | AAC-exp | **246.9** | 186.3 | 1803.9 | 46.6 |

We studied the original and automatically configured algorithms for each scenario (with and without capping). AAC-worst method aggregates replications and candidates using worst strategy (analogously the same for AAC-best). AAC-exp method uses the exponential model ($p = 0.7$ and $\alpha = 10$) to aggregate replications and the worst strategy to aggregate candidates. Column "performance" presents the average and best deviations from the best known solutions. In scenario AAC-UBQP we report the absolute deviation, while in the others we report the relative deviation. The total effort of the configuration process is measured in minutes (ACOTSP and AAC-UBQP) or trillions of checks (HEA-COL). Column "saved" present the effort savings in percentage.

## Conclusion and future work

Capping **mechanisms perform well** on the evaluated scenarios.

- Reduce up to 75% of the configuration effort.
- Can help the search for good configurations.
- Applicable to any configuration scenario for optimization problems.

Next steps:

- Explore other scenarios (e.g., larger search spaces, design choices).
- Apply quality-based capping methods for scenarios minimizing running time.
- Analyze the behavior of capping in irace: changes in the search and decision mistakes.

## Acknowledgements

Authors would like to acknowledge the ELAVIO organizing committee for the travel scholarship, and the Federal University of Rio Grande do Sul for the funding to attend COSEAL Workshop.

## References

[1] Hutter et al. ParamILS: an automatic algorithm configuration framework. *JAIR*, 36:267–306, 2009.
[2] Cáceres et al. An experimental study of adaptive capping in irace. *LION*, pages 235–250, 2017.
[3] Hoos and Stützle. *Stochastic local search: Foundations and applications.* Elsevier, 2004.

XXIII Latin-American Summer School in Operations Research (ELAVIO 2019), Spain.
7th Configuration and Selection of Algorithms Workshop (COSEAL 2019), Germany.