Evaluating anytime performance on NAS-Bench-101

Carlos Vieira
IMD, Universidade Federal
do Rio Grande do Norte
Natal, RN, Brazil
carlosv@ufrn.edu.br

Leslie Pérez Cáceres
Escuela de Ingeniería Informática
Pontificia Universidad Católica de Valparaíso
Valparaíso, Chile
leslie.perez@pucv.cl

Leonardo C. T. Bezerra
IMD, Universidade Federal
do Rio Grande do Norte
Natal, RN, Brazil
leobezerra@imd.ufrn.br

Appendix A
Benchmarking irace hyperparameters

Given that irace has not being applied to tackle NAS-Bench-101, we study its performance with different setups for a fair comparison with the other NAS-Bench-101 techniques. We note that when dealing with known benchmarks, ACs can benefit from tailored setups as they can exploit known characteristics of the benchmarks.

We performed 20 repetitions of irace setting 10 million TPU seconds as total configuration budget and evaluate the performance obtained by the following setup options:

Estimation budget Budget assigned for the initial estimation of the average execution time of an evaluation. We perform experiments with 2% and 0.2% of the total configuration time for budget estimation.

First test Number of evaluations required to perform the first elimination test in a race. We perform experiments with 2 and 3 evaluations for applying the initial test.

For further details of these setup options we refer to [1]. Figure 1 gives the ECDFs of the final quality obtained by the different setups of irace. The best overall performance is obtained by setting the estimation budget to 2% of the total configuration time and allowing 3 evaluations before the first elimination test (mean test regret 0.002837).

Appendix B
High-performing configurations plots

Here we present parallel categories plots representing the architectures selected by the different NAS techniques. Figure 2 shows the configurations selected when using a fixed number of nodes, while Figure 3 shows those using a variable number of nodes. As in the paper, only NAS hyperpameters related to the node label list are given (op_1-op_5), corresponding to each vertex of the DAG model, i.e. each layer of the selected neural network, for legibility's sake. Furthermore, color scaling (on the same range for all plots) reflects the mean test regret, also depicted as a discretized variable in the left-most column of the plots. Do note that, since topology is defined by the adjacency matrix, no layer order or architecture size should be assumed.

Appendix C
Runtime statistics

Here we give runtime (wallclock) statistics for all three algorithms in their different experimental setups in Figure 4. Abbreviations for experimental setups are used as in the paper: O for original; VS for variable-sized; C for caching; and CVS for caching and variable-sized. Runtime along the y axis is given in hours:minutes:seconds format.

References

[1] M. López-Ibánez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle, "The irace package: Iterated racing for automatic algorithm configuration," *Oper. Res. Perspect.*, vol. 3, pp. 43–58, 2016.

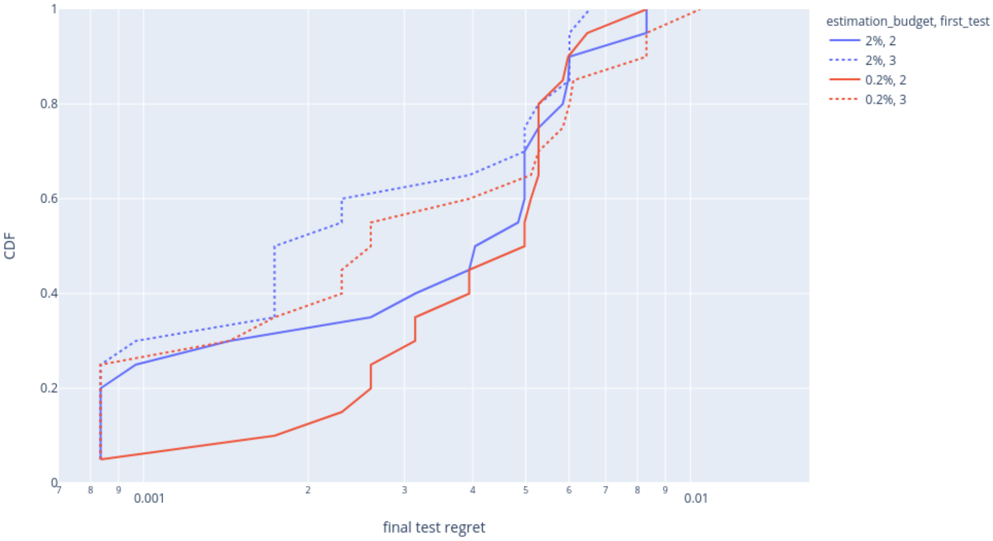


Fig. 1. Empirical cumulative distribution function (x-axis) of the final regret (y-axis) from 20 runs of each irace setup.

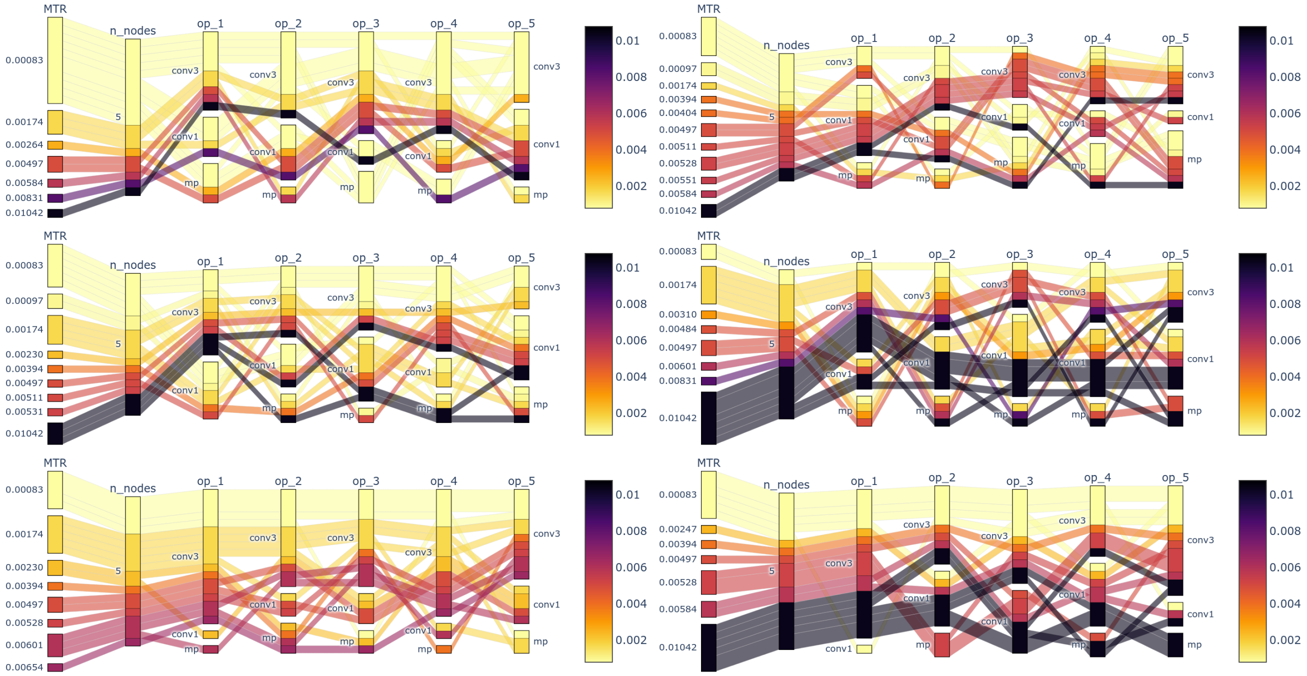


Fig. 2. Parallel categories plots of the 20 architectures selected by SMAC (top), RE (middle), and irace (bottom) with fixed number of nodes, no caching (left), caching (right).

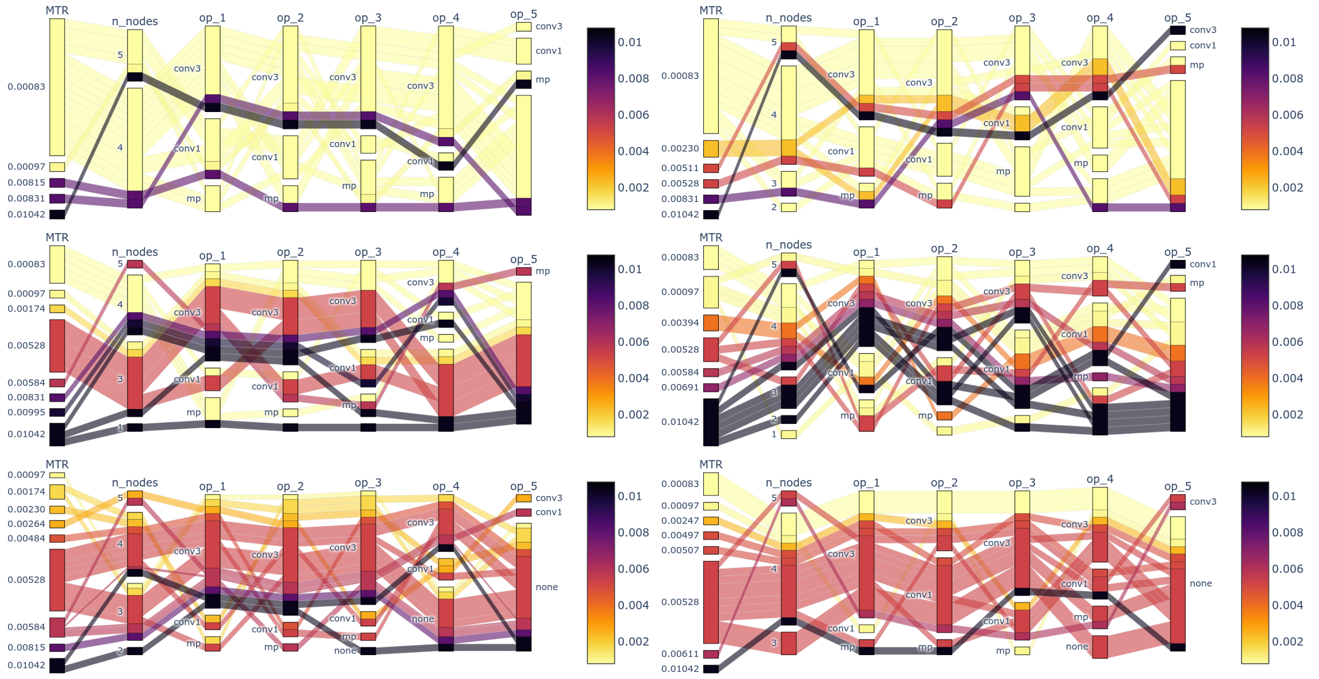
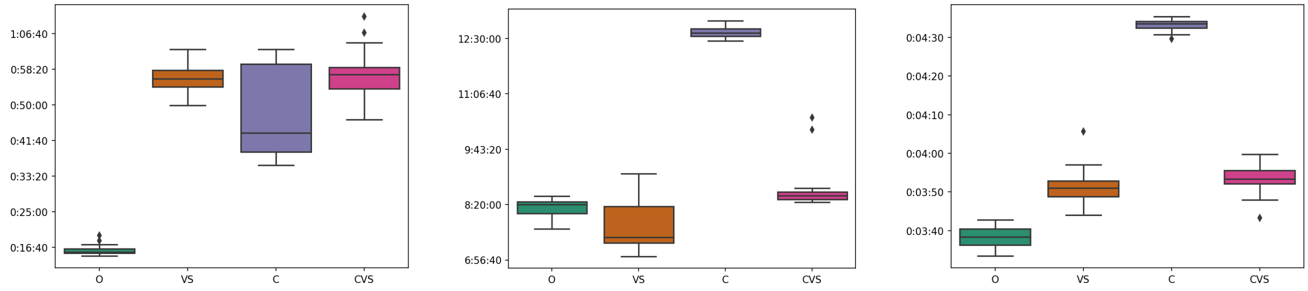


Fig. 3. Parallel categories plots of the 20 architectures selected by SMAC (top) RE (middle) and irace (bottom) with variable number of nodes, no caching (left), caching (right).



(a) irace

(b) SMAC

(c) RE

Fig. 4. Boxplots representing wallclock runtimes (over 20 runs) of each algorithm under different experimental setups