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

Marcelo de Souza[1,2], **Marcus Ritt**[1]
[1] Federal University of Rio Grande do Sul – UFRGS, Brazil
[2] Santa Catarina State University – UDESC, Brazil
marcelo.desouza@udesc.br, marcus.ritt@inf.ufrgs.br

The methods of automatic algorithm design reduce the human effort in the development of new meta-heuristics, minimize the bias in the selection of algorithmic components, and allow the researcher to focus on creativity-related activities. First, the researcher defines a set of algorithmic components and represent them as a parameterized framework, which describes all the possible combination of components and the possible values to their parameters. Second, the researcher applies an automatic configuration tool to explore this space and find the best combination of components and parameter values. Although there are several works on literature reporting the success of applying these techniques on different optimization problems, the exploration of the space of components takes a considerable amount of time. Each candidate evaluation implies its execution on some input instances, which is usually time-consuming. In this work, we propose and study a set of capping strategies to reduce the total time of the automatic design process. These strategies analyze each candidate execution to determine the quality of the algorithm. If a weak candidate is identified, the execution is stopped and it is discarded, i.e., the capping strategy minimizes the time spent with poor performers.

We use irace in the automatic design of meta-heuristics and apply the capping strategies from the second iteration. For each execution of candidate c on instance i, we select one of the elite candidates of the previous iteration as the reference candidate r. We propose four different strategies for this selection: the overall best/worst elite candidate, or the best/worst elite candidate on instance i. Then, we create a performance envelope based on the performance of the best/worst execution of the pair $\{r, i\}$, and use it to evaluate the quality of c on i. The envelope is represented by a set of points (t, q), where q is the minimum expected quality of the best found solution on time t. If in some instant t of the execution of c, the quality of the best found solution is worse than q, the execution can be halt, and candidate c can be discarded. In order to relax the capping aggressiveness and allow candidates near the expected performance to be accepted, we define an acceptable gap in the performance envelope. This gap is implemented by a shift on the time (e.g., we give 10% more time on t to the candidate reach the expected quality q), on the quality (e.g., we accept at most 10% of deviation from q at each time t), or both at the same time.

We performed computational experiments to evaluate the performance of the different capping approaches according to the type of shift, the strategy for selecting the reference candidate, and the size of the gap. We identified that the shift on time is too aggressive, especially in the final seconds of the execution. Its combination with the shift on quality is a better strategy, avoiding the capping of promising candidates. We applied the proposed capping strategies on the automatic design of meta-heuristics for several problems, including binary quadratic programming, maximum cut, test-assignment, and flow shop scheduling. Experiments show that using our capping strategies the time spent with the automatic design process can be reduced up to 34%, with a slight reduction of at most 8% in the quality of the resulting algorithms. Thereby, we show the potential of the proposed capping strategies to improve the efficiency of the automatic algorithm design and configuration techniques.

As a future work, we aim at study capping strategies which determine the size of the gap according to the previous executions of the reference candidate. Assuming that the time required for the reference candidate to find a determined solution quality is exponentially distributed, we can estimate the distributions for different quality values based on the known previous executions. Then, we can create a performance envelope by applying a p-value to this distribution to determine the time required for each quality value.