Predicting parameter configurations for tuning effective algorithms on very large instances

Franco Mascia Mauro Birattari
Thomas Stützle
IRIDIA, Université Libre de Bruxelles
{fmascia,mbiro,stuetzle}@ulb.ac.be

1 Introduction

Automatic algorithm configuration has become an essential tool for tuning stochastic local search (SLS) algorithms that, once configured, are able to solve effectively hard optimisation problems [1]. Regardless of the impressive advances of the last years, configuring an SLS algorithm for very large instances remains difficult. The main reason for that is that the computation time of each run of the algorithm scales with the size of the instances being tackled. In fact, the computational cost of a single search step scales with instance size; moreover, larger instances usually also require a much larger number of search steps to find good quality solutions. Even if a limited number of algorithm configurations are tested during the tuning of the algorithm, and therefore the tuning time scales linearly with the computation time, the sheer amount of time required to test a single configuration on a very large instance renders the whole tuning process impractical.

2 Method

In this work, we propose an experimental protocol to tune an algorithm on a set small instances $s \in S$ and extrapolate the obtained configuration on very large target instances $s^* \in S$. This is done by optimising the free variables of the experimental setting we defined. As a proof of concept, we present a study on an iterated local search (ILS) [2] algorithm for the quadratic assignment problem (QAP). In this example, the two free variables being optimised are two prescriptive models: a cut-off-time policy $t(s)$, and a parameter configuration policy $\hat{m}(s, t(s))$. The first policy depends on the instance size and prescribes a cut-off time during the tuning phase on the small instances in S. The second policy depends on the instance size and the cut-off time, and it is used to actually extrapolate the parameter configuration for the algorithm on the very large target instance s^* given a time budget T^*.

The rationale behind having a policy also for the cut-off time is that a parameter configuration is a function of both the instance size and of the computation time. For example, in the case of the proof of concept on the ILS algorithm for QAP, there is only a parameter exposed to the algorithm configuration, and this

parameter acts on the balance between intensification and diversification. Given an instance size s, tuning with a short cut-off time leads to configurations that emphasise intensification, while tuning with long cut-off time leads to configurations that emphasise diversification. It is reasonable to assume that on very large instances the amount of diversification required is smaller due to the fact that the search space to be explored is already large and the algorithm will have to spend most of the time intensifying. In such cases a small cut-off time when tuning on small instances, can lead to algorithm configurations that imply stronger intensification and are therefore more suited for large instances. What is a long or a short cut-off time for a given size is left open to the policy and the cut-off time for each size is left as a free variable.

In order to optimise the free variables in our experimental setting, we cast the problem as a parameter estimation for the minimisation of a loss function. We first decide a parametric family for the cut-off time policy $t(s)$ and the parameter configuration policy $\hat{m}(s, t(s))$. The values prescribed by the policies will be determined by the parameters π_t and π_m, which number and type depend on the parametric families chosen. In the most general case, we also define a weighting policy $\omega(s)$, with a specific parametric family and parameters π_ω that allows us during the tuning on the small instances to weight instances differently depending on their size. The rationale here is that very small instances could be less useful than small to medium ones when extrapolating a parameter configuration for very large instances.

We then define the loss function as the difference between $C\hat{m}(s; t(s))$, which is the cost obtained when executing the algorithm with the parameter setting determined by $\hat{m}(s; t(s))$, and $CB(s; t(s))$, which is the cost function obtained when executing the algorithm with the best possible parameter setting $B(s; t(s))$, given the same maximum run-time $t(s)$. Then we try to determine

$$\arg\min_{\pi_\omega, \pi_{\hat{m}}, \pi_t} \sum_{s \in S} \omega(s) \left[C\hat{m}(s; t(s)) - CB(s; t(s)) \right]. \tag{1}$$

By finding the optimal settings for π_ω, $\pi_{\hat{m}}$, and π_t, we effectively find the best scaling of the examples in S, and the best cut-off time, which allows us to find the policy that best describes how the parameter setting scales with the sizes in S. The same policy can be used to extrapolate a parameter setting for a target instance size s^* and a target cut-off time T^*.

Experimental results on the ILS algorithm for QAP show that our method is able to extrapolate parameter settings that are extremely close to the best parameter settings on very large instances.

References

[1] Hoos, H.H.: Programming by optimization. Communications of the ACM **55**(2) (2012) 70–80

[2] Lourenço, H.R., Martin, O., Stützle, T.: Iterated local search: Framework and applications. In Gendreau, M., Potvin, J.Y., eds.: Handbook of Metaheuristics. Volume 146 of International Series in Operations Research & Management Science. 2 ed. Springer, New York, NY (2010) 363–397