Noname manuscript No.
(will be inserted by the editor)

Effect of Transformations of Numerical Parameters in Automatic Algorithm Configuration

Alberto Franzin · Leslie Pérez Cáceres · Thomas Stützle.

the date of receipt and acceptance should be inserted later

Abstract We study the impact of altering the sampling space of parameters in automatic algorithm configurators. We show that a proper transformation can strongly improve the convergence towards better configurations; at the same time, biases about good parameter values, possibly based on misleading prior knowledge, may lead to wrong choices in the transformations and be detrimental for the configuration process. To emphasize the impact of the transformations, we initially study their effect on configuration tasks with a single parameter in different experimental settings. We also propose a mechanism of how to adapt the transformation used and give exemplary experimental results with that scheme. We also propose a mechanism for how to adapt towards an appropriate transformation and give exemplary experimental results of that scheme.

1 Introduction

Automatic algorithm configuration has shown to be crucial for reaching high performing parameter settings of search algorithms [7] and a number of effective configurators are available [1, 9, 10, 12, 16, 4]. Modern configurators can handle a large number of parameters and different types of parameters including categorical, ordinal and numerical ones. In this article, we focus on the latter type of parameters.

Choosing the value of a parameter for an algorithm can be difficult due to three intertwined issues: (i) the wide range of possible values, (ii) the possibility of parameter interactions, and (iii) the impact a variation of a parameter value

This research has received funding from the COMEX project (P7/36) within the Interuniversity Attraction Poles Programme of the Belgian Science Policy Office. Thomas Stützle acknowledges support from the Belgian F.R.S.-FNRS, of which he is a Research Director.

Université Libre de Bruxelles, IRIDIA, 50, Av. F. Roosevelt, 1050 Brussels
E-mail: {afranzin, lperez, stuetzle}@ulb.ac.be

has. Automatic algorithm configuration techniques relieve algorithm designers and practitioners of the first two issues. For numerical parameters, the third issue depends in part on the parameter representation and the sensitivity of each single parameter during the configuration task. The representation issue is related to the question whether a parameter should be varied according to an additive or a multiplicative scale. For example, in a population based-algorithm it is intuitively clear that the sensitivity of the population size depends on its value: the effect of increasing the population size by a constant number of, say, ten has a very different impact if the increase is from 1 to 11, from 10 to 20, or from 100 to 110. Instead, it seems more intuitive to change population size by a specific factor. Choosing an additive or multiplicative scale is related to the question whether a parameter should be presented in a normal scale or in a logarithmic scale.

In this article, we study the effect of a transformation of the parameter values for automated algorithm configuration. We consider five transformations of a single parameter and study their effect under different configuration budgets. We show how the right transformation helps considerably to discover good parameter values; this impact is stronger for low configuration budgets. At the same time, we show how a wrong transformation, which may be chosen due to misleading prior knowledge, can instead be harmful. As a possible alternative to a manual choice of a transformation, we explore a scheme to adapt it at execution and we give some illustrative results of its possible effectiveness.

The implementation of the transformation depends on the particular configurator. For example, ParamILS [9] requires a discretization of numerical parameters and one possibility is therefore to discretize the numerical interval in normal, logarithmic, or another fashion. SMAC [10] includes the possibility of exploring a parameter range in a logarithmic way (e.g. $1, 10, 100, \dots$) by specifying a transformation in the parameter definition, which allows to scrutinize better on the lower part of the parameter range. A more general solution is to apply a transformation in the wrapper script used to execute the experiments. This possiblity is currently the only one available when using irace [12]; it can also straightforwardly be used in other configurators such as SMAC.

The article is structured as follows. In the next section, we define the transformations we studied. We empirically observe their impact in Section 3, and we propose an automatic selection scheme in Section 4, along with an experimental evaluation. Further experiments are available as supplementary material at `http://iridia.ulb.ac.be/supp/IridiaSupp2017-011`.

2 Parameter transformations

As a baseline, we consider the parameter space with no transformation, e.g.

$$\mathcal{I}(x) \triangleq f(x \in [a, b]) \mapsto x \in [a, b] \tag{1}$$

where $[a, b]$ is the range of possible values for parameter x. This is the default behaviour of configurators such as SMAC or irace. This baseline is well-suited for parameters whose effect reflects an additive scale. For a multiplicative scale, we may apply a logarithmic transformation of the parameter:

$$\mathcal{L}og(x) \triangleq f(x \in [\log a, \log b]) \mapsto 10^x \in [a, b], \tag{2}$$

that is, we sample the logarithm of the parameter under consideration. For ease of representation, we consider base-10 exponentials and logarithms. (Note that this is also the transformation that is natively provided by SMAC.) As for $a = 0$, $\log a$ is not possible, we use as lower bound the precision δ that we use when we sample real-valued parameters. For example, if $\delta = 4$ and the parameter range is $[0, 1]$, our sampling interval will be $[-4, 0]$.

The effect of a $\mathcal{L}og(x)$ transformation is a modification of the probability of sampling specific parameter ranges, favoring the lower part of the interval. For example, assuming uniform distributions, in a logarithmic range $[0, 4]$, corresponding to a final parameter value range of $[1, 10\,000]$, there is a probability of 0.25 of sampling a value between zero and one (three and four), which actually corresponds to a transformed parameter between one and ten (1 000 and 10 000) once rescaled, an interval whose probability of being hit in a non-transformed scale is 0.001 (0.9).

In case we are interested in exploiting the upper subrange of the interval, we define the reflection in the parameter range of the logarithmic transformation:

$$\mathcal{RL}og(x) \triangleq f(x \in [\log a, \log b]) \mapsto b - 10^x + a \in [a, b]. \tag{3}$$

This transformation is useful in the dual case of $\mathcal{L}og(x)$: consider a parameter that represents a probability that we want to take values very close to 1, with a precision of four decimal digits, resulting in a sampling interval $[-4, 0]$. As an example, with probability 0.25, we could sample a value between -4 and -3, which maps to the range $[0.9991, 1]$ after the transformation.[1]

Milder transformations are also possible, for example the power function with exponent greater than 1, that in an interval $[0, 1]$ magnifies the area close to 0. Here, we consider a quadratic transformation

$$\mathcal{S}(x) \triangleq f(x \in [a, b]) \mapsto \left(\frac{x - a}{b - a}\right)^2 \cdot (b - a) + a \in [a, b], \tag{4}$$

exploring the lower subrange; by reflecting it in the parameter range we obtain

$$\mathcal{RS}(x) \triangleq f(x \in [a, b]) \mapsto b - \left(\frac{x - a}{b - a}\right)^2 \cdot (b - a) \in [a, b] \tag{5}$$

that explores the upper subrange.

[1] Note that SMAC offers the possibility of a $\mathcal{L}og(x)$ transformation but not that of $\mathcal{RL}og(x)$.

Also powers with exponent between 0 and 1 can be useful to exploit the upper part of the interval; we consider the square root of the normalized sampled value

$$\mathcal{S}qrt(x) \triangleq f(x \in [a,b]) \mapsto \left(\frac{x-a}{b-a}\right)^{1/2} \cdot (b-a) + a \in [a,b]. \tag{6}$$

3 Empirical assessment of the transformations

We benchmarked the six transformations of Section 2 to observe their impact in different scenarios. All the experiments reported in this paper have been conducted on an Intel Xeon E5-2680 CPU with 2.4GB of RAM available for each experiment and running under Linux Rocks 6.2. The tuning was done with irace version 2.4 [12]. Each tuning was run 30 times, resulting in 30 tuned parameter values. We then computed the average percentage deviation (APD) obtained for each tuned value on the test instances.

In this section we show two artificial examples that employ different variants of Simulated Annealing (SA). SA is a popular metaheuristic, which moves through solutions accepting them using a probabilistic acceptance criterion. In the original formulation, SA uses the Metropolis criterion [13], which accepts a solution in a minimization problem with a probability given by

$$p = \begin{cases} 1, & \text{if } \Delta \leq 0 \\ \exp(-(\Delta/T_k)), & \text{otherwise} \end{cases} \tag{7}$$

where Δ is the difference between the cost of the new and the current solution and T_k is the temperature parameter after k cooling steps, that is, after k reductions of the temperature value. A solution is always accepted if it is better than or equal to the current incumbent solution; if it is worse, it is accepted with a probability that depends both on the relative difference in terms of objective function values between the two solutions and on the stage of the execution. The temperature parameter is usually decreased during the search following a so-called cooling scheme; the number of solutions evaluated at a same temperature value T_k is called temperature or epoch length. The effect of the temperature parameter is to gradually transition from an initial exploratory search phase, in which more uphill moves are accepted, to a final exploitative search phase, in which almost only improving moves are accepted.

We evaluate the transformations applying the SA algorithms to the Quadratic Assignment Problem (QAP) [2]. We use instances of size 100 facilities and locations, generated uniformly at random [15]; 25 instances are used for the tuning, and 25 for testing of the tuned parameter settings. For both tuning and testing we stop the algorithm after two seconds of CPU time. The tuning is done using three different budgets, namely 125, 500 and 2 000 experiments per tuning. The convergence of the parameter to the best value under the different transformations is reported in the supplementary material.

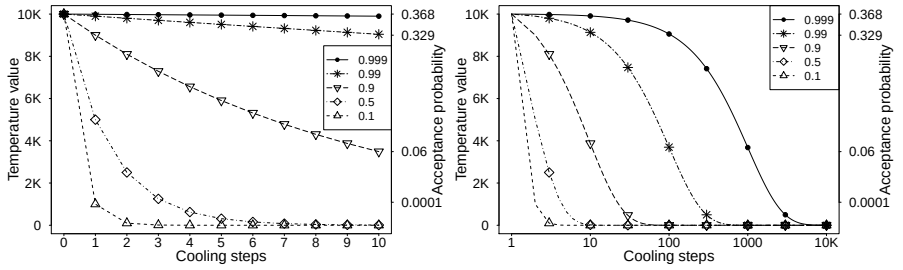

Fig. 1 Evolution of the temperature for different values of α, measured over 10 cooling steps (left plot) and 10 000 cooling steps (right plot, in logarithmic scale). On the right side is indicated the acceptance probability of a solution whose objective function value is worse than the optimal one by $\Delta = 10\,000$.

3.1 Geometric cooling in Simulated Annealing

We consider a SA with a geometric cooling as

$$T_k = \alpha \times T_{k-1}, \tag{8}$$

where $\alpha \in [0, 1]$ controls the decrease of the temperature value. Low values of α enforce a very quick decrease in the temperature, as shown in Figure 1; for example, for values of α equal to 0.1 or 0.5 the acceptance probability becomes negligible already after a few cooling steps, making the acceptance of worsening solutions very unlikely. As α gets closer to 1, many more cooling steps are needed to decrease the acceptance probability of worsening moves. Traditionally, in the SA literature α is chosen close to 1, to ensure that the algorithm does not converge too quickly.

We consider two cases, one in which the best value of α is close to 1, and one in which the best value is close to 0; such settings are obtained by fixing the temperature length either as very small or very large. These values are manually chosen to be 1 and 500 times the size of the neighbourhood, respectively; they guarantee to observe, in our scenario and computational environment, optimal values for α close to 1 and to 0, respectively. For small values of the temperature length the search shows an exploitative behaviour, updating the temperature very frequently and rejecting non improving moves more often; thus, the preferred values of α are close to 1, to reduce the decreasing rate of the temperature and maintain exploration. As the temperature length increases, the temperature gets updated less frequently, and the search loses exploitative potential; to compensate, the values of α tend to decrease to favour convergence. The other parameters are kept fixed: the initial temperature is given by the maximum gap between consecutive solutions observed during an initial random walk of 10 000 steps in the search space.

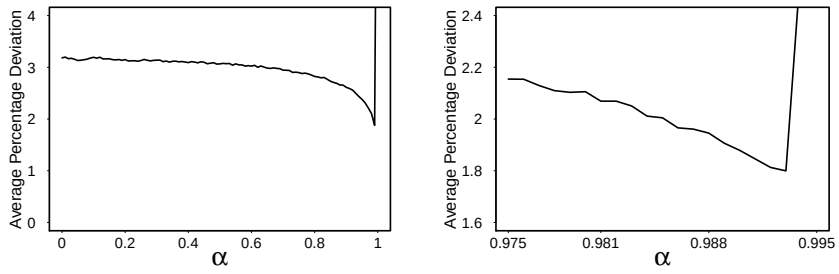

Fig. 2 APD in dependence of the value of parameter α for small temperature length; the left plot gives the full parameter range and the right plot the subinterval with the best-performing values.

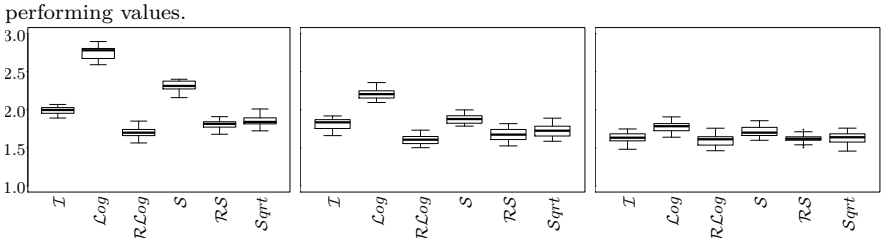

Fig. 3 APD (given on y-axis) obtained by the different transformations for a small temperature length. The three plots show the results obtained with a budget of 125, 500 and 2 000 experiments, respectively.

3.1.1 Short temperature length

The APD in dependence of the values for α obtained with a small temperature length is shown in Figure 2 for the whole parameter range (left plot) and for the best-performing value (right plot). There is a slow but clear improvement as α grows, until $\alpha = 0.993$; for $\alpha = 0.994$ the results are already very bad, as shown by the spike in the plots for values close to $\alpha = 1$. This is an extreme condition, where a tiny variation can make a huge difference but also a good benchmark as the very best parameter values stem from a rather narrow range.

In Figure 3, we show the results obtained by each transformation with a budget of 125, 500 and 2 000 experiments per tuning. It is quite evident how the normal sampling is outperformed by the transformations aimed to exploit the region of the range close to 1, which are $\mathcal{RL}og(x)$ and $\mathcal{RS}(x)$. The transformations $\mathcal{L}og(x)$ and $\mathcal{S}(x)$ obtain worse results, as they tend to exploit the wrong range of values. Analyzing more in detail the behaviour of the transformations aimed to exploit the upper subrange of parameter values, $\mathcal{RL}og(x)$ converges very quickly to the best values, while $\mathcal{RS}(x)$ and $\mathcal{S}qrt(x)$ exhibit a slower convergence. The remaining transformations are able to converge to a narrow, though suboptimal, subrange of values, but only for the high budget case.

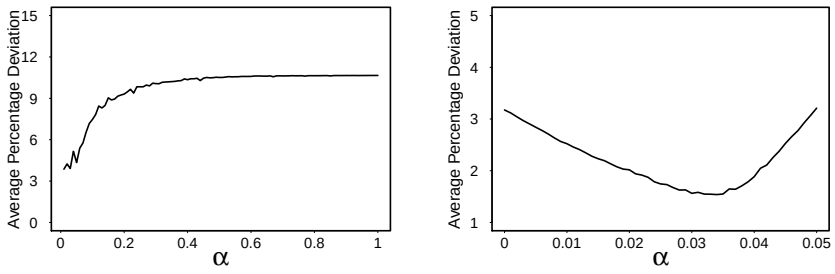

Fig. 4 APD in dependence of the value of parameter α for high temperature length; the left plot gives the full parameter range and the right plot the subinterval with the best-performing values.

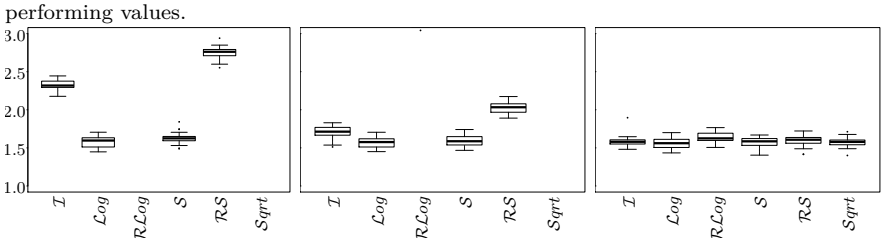

Fig. 5 APD (given on y-axis) obtained by the different transformations for a high temperature length. The three plots show the results obtained, respectively, with a budget of 125, 500 and 2 000 experiments.

3.1.2 High temperature length

In Figure 4, we show the APD values for a high temperature length in dependence of the parameter α over the full (left plot) and the best-performing (right plot) subrange. In this case, we have a smoother and relatively wider "good area" of parameter values than in the case of a short temperature length, roughly centered between $\alpha = 0.025$ and $\alpha = 0.035$.

Figure 5 shows the obtained results. As expected, the situation is reversed with respect to the previous situation. In this case, the $\mathcal{L}og(x)$ and $\mathcal{S}(x)$ transformations exhibit a slightly quicker convergence to the best values than the identity transformation, while the remaining transformations $\mathcal{S}qrt(x)$, $\mathcal{RS}(x)$ and $\mathcal{RL}og(x)$ obtain much worse results. These latter ones require a large number of experiments (here 2 000) to converge to good values. A more detailed examination shows that $\mathcal{L}og(x)$ and $\mathcal{S}(x)$ converge slightly faster than $\mathcal{I}(x)$ to the best values; however, here the differences are not very marked, as the range of very good values appears to be somewhat larger as for the case with low temperature length and the variability between similar values larger.

3.2 SA with a fixed temperature

In this example, we study a fixed temperature variant of the original SA formulation [3, 5], where the temperature is held constant during the whole runtime.

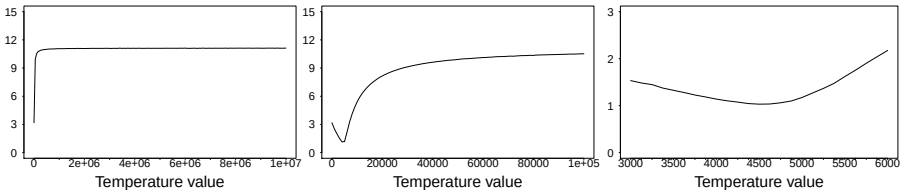

Fig. 6 Landscape of the results obtained by different fixed temperature values, in terms of APD (given on y-axis). Given are respectively, the full landscape, the landscape of the results obtained in the range $(0, 10\,000)$ (the lowest 1% of possible values) of the fixed temperature, and the subinterval close to optimal parameter values.

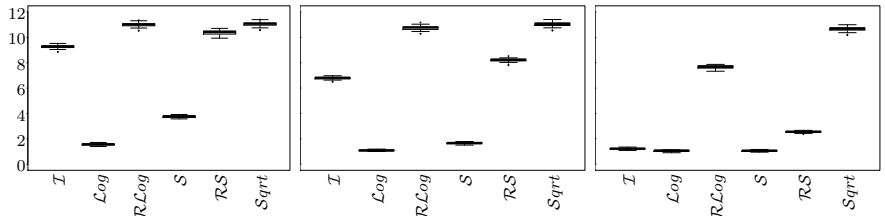

Fig. 7 APD values obtained by the different transformations. The three plots show the results obtained with a budget of 125, 500 and 2000 experiments, respectively.

The cooling scheme is therefore replaced by the formula

$$T_{k+1} = T_k = T_0 \quad \forall\, k, \tag{9}$$

where T_0 is the initial temperature. The acceptance probability of a worsening move depends therefore only on Δ. Hence, the only numerical parameter to be considered is T_0.

We allow the temperature T_0 to take any integer value in the area $[1, 10\,000\,000]$, which translates in an interval $[0, 7]$ for the logarithmic transformations $\mathcal{L}og(x)$ and $\mathcal{RL}og(x)$. In our instance set, the values that give the best results lie in a very narrow range close to the lower part of the interval, as shown in the plots of Figure 3.2. In this context, we expect the logarithmic transformation $\mathcal{L}og(x)$, and the quadratic one $\mathcal{S}(x)$, to outperform their counterparts, especially in the low budget case.

The results of the five transformations are shown in Figure 7, giving the results for the 30 tunings for each of the budgets of 125, 500 and 2000 experiments. As expected, the logarithmic transformation outperforms all the other ones, with very good results already with the smallest budget, followed by the quadratic transformation. The identity transformation is only able to catch up for the largest available interval.

3.3 Discussion

In Sections 3.1.2 and 3.2, we have shown two experiments in which the optimal parameter values lie in the lower part of the allowed range. In this case, the transformations that are designed to exploit that specific area, namely the logarithmic transformation $\mathcal{L}og(x)$ and the square transformation $\mathcal{S}(x)$, also show a faster convergence towards good values with respect to the identity transformation. Analogously, in Section 3.1.1 we see how their bounded reflections, namely the transformations $\mathcal{RL}og(x)$ and $\mathcal{RS}(x)$, outperform the identity transformation. The differences in convergence speed are more marked in the case of a large range of possible parameter values, as shown in Section 3.2, where the identity transformation needs eight times the budget needed by the logarithmic transformation to reach comparable results, even if only a single parameter is tuned.

However, a wrong transformation is detrimental for the tuning process. This is illustrated by the example in Section 3.1.2. Traditionally, the cooling coefficient α of the geometric cooling in Simulated Annealing is set to values close to 1, but in the example of Section 3.1.2 the best-performing values are actually close to zero. While this experiment is somewhat artificial, it is more important to note that in absolute terms, well-tuned parameter settings from the example in Section 3.1.2 perform better than well-tuned parameter settings from the example in Section 3.1.1. In fact, the solutions obtained are on average 1.55% and 1.59% from the best known solutions, respectively; while small, a pairwise Wilcoxon test shows that this difference is actually statistically significant. In a sense, while the definition and application of transformations is often intuitive if the parameter landscape is known, our results show that inaccurate assumptions may lead to wrong choices of transformations, and be harmful to the quality of the final results. Note that this does not only happen in case of artificial examples. In the supplementary material at `http://iridia.ulb.ac.be/supp/IridiaSupp2017-011` we also include some experiments with the SMAC configurator version 2.10 [10] applied to configure various scenarios that include 26 to 323 parameters of which for several numerical parameters a logarithmic transformation was manually chosen based on a priori information. However, our experimental results indicate that this manual choice of the transformation actually does not results in improved performance of SMAC, confirming that manually picking the transformations does not necessarily improve performance.

Hence, it is desirable to unburden the user of the choice of the right transformation to apply. In the context of Bayesian optimization (or, say, surrogate-assisted optimization), Snoek *et al.* proposed an automatic mechanism to adapt a parameter transformation [14], applying a Beta transformation $\textsc{Beta}(x;\alpha,\beta)$ to a numerical parameter x, where α and β are the control parameters that define the shape of the transformation. The control parameters α and β are in turn adapted using the same Bayesian optimization approach that is used also for parameter tuning. Interestingly, in [14] the authors also report observations about surprising choices of the right transformation to ap-

ply, which may contrast with intuition and established practices. As irace is not a method that works within a Bayesian optimization scheme, in the next section we propose a simple, heuristic scheme that is suitable for use in irace.

4 Heuristic transformation in irace

The irace package iteratively samples configurations that then undergo a racing process to identify the best performing candidates. We propose a method to automatically adapt the proper transformation when sampling numerical parameters in irace. Here, we consider only $\mathcal{L}og(x)$ and $\mathcal{RL}og(x)$ transformations alongside $\mathcal{I}(x)$, as they have dominated the corresponding polynomial transformations. Our idea is to sample each numerical parameter in the first iteration according to a uniform random sampling as done by default in irace. After the results of the first iteration are available we simulate a sampling according to the three possibilities for the following races. The simulated sampling of numerical parameters is done as follows. First, for each parameter value appearing in the configurations an estimate of the mean performance is computed as the average deviation from the best result across the already seen instances. These estimates are saved in an array, which is sorted according to the parameter values. Next, three subsets of results are extracted from this array by choosing positions that (i) simulate the generation of values close to the lower part of the parameter range (as it would happen when using $\mathcal{L}og(x)$), (ii) simulate the generation of values close to the upper part of the parameter range (as it would happen when using $\mathcal{RL}og(x)$), and (iii) simulate a sampling according to an identity transformation (that is, no transformation). If the mean of the first subset is significantly better than the others, $\mathcal{L}og(x)$ is applied; if the mean of the second subset is significantly better than the others, $\mathcal{RL}og(x)$ is used. Otherwise, no transformation is applied. This process is repeated for every new candidate configuration, that is, the same parameter may be subject to different transformations for different configurations.

This method is quite simple but effective, as we show in the next examples. In Figure 8 we show the results obtained by this automatic transformation selection on the scenarios we considered in the experiments of Sections 3.1.1, 3.1.2 and 3.2, respectively. For each benchmark we compare the automatically chosen transformation for a budget of 125, 250 and 500 experiments per tuning (the three central boxplots) with $\mathcal{I}(x)$ with a budget of 125 and 500 (leftmost boxplots) and the best transformations ($\mathcal{RL}og(x)$, $\mathcal{L}og(x)$ and $\mathcal{L}og(x)$, respectively) for each of these cases with a budget of 125 and 500 (rightmost boxplots). In all cases, the automatic selection scheme improves clearly over $\mathcal{I}(x)$ when considering the same budget. The automatic scheme also catches up with the a priori chosen best possible transformation at a budget of 125, though still lagging behind the results obtained with the "right" transformation at a budget of 500. While in low budget scenarios we cannot expect results as good as applying the right transformation from the beginning, the loss in terms of performance is limited.

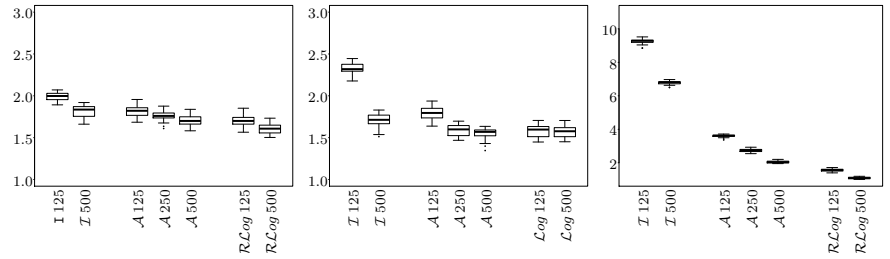

Fig. 8 Comparison of the convergence observed for the automatic transformation selection described in Section 4 for, left to right, experiments from Sections 3.1.1, 3.1.2 and 3.2, respectively. We compare the results in terms of average percentage deviation from the best known solutions (given on y-axis) for $\mathcal{I}(x)$, the automatic selection, and the "right" transformation using a budget of 125 and 500 evaluations (for the adaptive variant an additional budget of 250 evaluations is considered).

In the following experiments we compare the automatic selection only with $\mathcal{I}(x)$, that is, the original sampling scheme in irace. The results for both of them are reportd in Figure 9. In the first one (left plot) we show the results obtained when using a linear cooling scheme for an SA algorithm for budgets of 125, 250 and 500 experiments per tuning. As in the previous experiments, only the cooling scheme coefficient is tuned, chosen in the range $[0, 100000]$; the temperature length is one times the size of the neighbourhood. While $\mathcal{I}(x)$ with the given budget cannot make significant improvements, the automatic scheme obtains better results already with the lowest budget, and continues improving as more experiments are allowed. In the second experiment (right plot) we show a more realistic scenario, tuning the SA algorithm for the QAP proposed in [8], with budgets of 500 and 2000 experiments. This SA algorithm has four numerical parameters: the α coefficient for the geometric cooling scheme in the range $[0, 1]$; the temperature length coefficient in the range $[1, 100]$ (proportional to the size of the neighbourhood); the initial value for the temperature is chosen as a the the solution cost of a randomly generated solution rescaled by a value in the range $[0, 10]$; the temperature resets to its initial value when it reaches a certain minimum value, chosen in the range $[1, 10000]$. In this case, the improvement is not as strong as in the other examples reported, because of the combined effect of different parameters. Nonetheless, for both budgets SA algorithm tuned using the automatic transformation selection obtains better performance than the default irace.

In general, tuning an algorithm with several parameters is a complex task, with possibly surprising outcomes and also an expert may be misled by wrong assumptions and intuitions, with possibly detrimental effects on the search. Our proposed automatic selection scheme for parameter transformations relieves the user from the possibly very detrimental effect of a wrong choice. Additionally, as shown in the example above, it helps in the cases where a transformation is useful to speed-up convergence to good parameter values when compared to not using any transformation. Contrarily to [14], our scheme

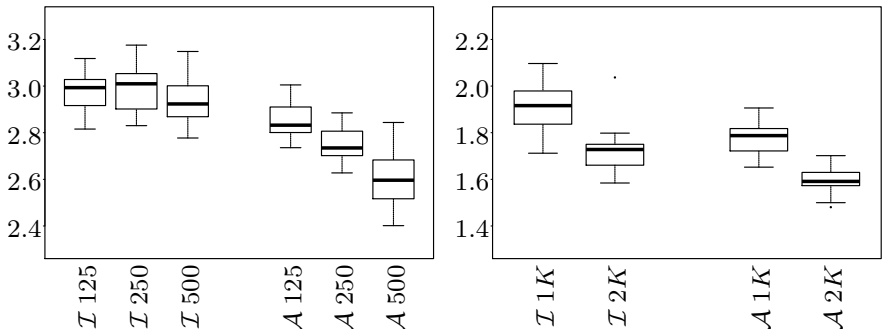

Fig. 9 Comparison of the convergence observed for the automatic transformation selection for a linear cooling scheme (left plot) and tuning an SA algorithm with four parameters (right plot). We compare the results in terms of average percentage deviation from the best known solutions APD (given on y-axis) of the automatic scheme with those of $\mathcal{I}(x)$.

does not require to specify a prior, as it makes its decisions solely based on the data observed in the already executed experiments. In summary, our automatic selection scheme is not meant to replace expert knowledge, or to prevent users from making decisions, but rather to provide an optional, robust data-driven support when said decisions are difficult to make.

5 Conclusions

Parameter space transformations have been considered before, but their impact on automatic configuration techniques has not been studied explicitly in the optimization community, unlike in other research communities such as medicine [6, 11] or machine learning [14]. The results reported in this paper illustrate the usefulness of knowledge about the parameter space in practice. Tuning a single parameter is a seemingly trivial task, but choosing the right transformation can make this task even less computationally expensive. This is highly desirable, as the hardness of the tuning task grows strongly with the number of parameters. However, a wrong transformation is detrimental for the tuning process, as we have also illustrated with several examples in this paper. In a sense, while the definition and application of transformations is often intuitive if the parameter landscape is known, our results show that inaccurate assumptions may lead to wrong choices of transformations, and be harmful for the quality of the final results. Hence, our recommendation would be to apply transformations of the parameter domain only if a careful deliberation indicates that such transformation may be useful as the best parameter values are expected to be at the boundary of a parameter domain. As an alternative one may apply adaptive schemes that choose appropriate transformations during the run of a configuration algorithm. Such approaches have previously been proposed in the context of Bayesian optimization [14]. In this paper, we have proposed another, heuristic strategy to choose an appropriate transformation

at run-time. We have shown that on various of the configuration scenarios we have considered in this paper, the proposed strategy is useful and improves performance when compared to not applying any transformation at all.

References

1. C. Ansótegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In I. P. Gent, editor, *Principles and Practice of Constraint Programming, CP 2009*, volume 5732 of *LNCS*, pages 142–157. Springer, Heidelberg, Germany, 2009.
2. E. Çela. *The Quadratic Assignment Problem: Theory and Algorithms*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.
3. H. Cohn and M. J. Fielding. Simulated annealing: Searching for an optimal temperature. *SIAM Journal on Optimization*, 9(3):779–802, 1999.
4. A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1 (1):19–31, 2011.
5. M. J. Fielding. Simulated annealing with an optimal fixed temperature. *SIAM Journal on Optimization*, 11(2):289–307, 2000.
6. N. Girerd, M. Rabilloud, P. Pibarot, P. Mathieu, and P. Roy. Quantification of treatment effect modification on both an additive and multiplicative scale. *PLOS ONE*, 11(4):1–14, 04 2016.
7. H. H. Hoos. Programming by optimization. *Communications of the ACM*, 55(2):70–80, February 2012.
8. M. S. Hussin and T. Stützle. Tabu search vs. simulated annealing for solving large quadratic assignment instances. *Computers & Operations Research*, 43:286–291, 2014.
9. F. Hutter, H H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, October 2009.
10. F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In C. A. Coello Coello, editor, *Learning and Intelligent Optimization, 5th International Conference, LION 5*, volume 6683 of *LNCS*, pages 507–523. Springer, Heidelberg, Germany, 2011.
11. M. J. Knol, T. J. VanderWeele, R. H. H. Groenwold, O. H. Klungel, M. M. Rovers, and D. E. Grobbee. Estimating measures of interaction on an additive scale for preventive exposures. *European Journal of Epidemiology*, 26(6):433–438, 2011.
12. M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, and M. Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.
13. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

14. J. Snoek, K. Swersky, R. Zemel, and R. P. Adams. Input warping for Bayesian optimization of non-stationary functions. In *Proceedings of the 31th International Conference on Machine Learning*, volume 32, pages 1674–1682, 2014.
15. É. D. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17(4-5):443–455, 1991.
16. Z. Yuan, M. A. Montes de Oca, T. Stützle, and M. Birattari. Continuous optimization algorithms for tuning real and integer algorithm parameters of swarm intelligence algorithms. *Swarm Intelligence*, 6(1):49–75, 2012.