Configuring irace using surrogate configuration benchmarks

Nguyen Dang
KU Leuven, CODeS, imec-ITEC
nguyenthithanh.dang@kuleuven.be

Leslie Pérez Cáceres
IRIDIA, CoDE, Université libre de Bruxelles
leslie.perez.caceres@ulb.ac.be

Thomas Stützle
IRIDIA, CoDE, Université libre de Bruxelles
stuetzle@ulb.ac.be

Patrick De Causmaecker
KU Leuven, CODeS, imec-ITEC
patrick.decausmaecker@kuleuven.be

ABSTRACT

Over recent years, several tools for the automated configuration of parameterized algorithms have been developed. These tools, also called configurators, have themselves parameters that influence their search behavior and make them malleable to different kinds of configuration tasks. The default values of these parameters are set manually based on the experience of the configurator's developers. Studying the impact of these parameters or configuring them is very expensive as it would require many executions of these tools on configuration tasks, each taking often many hours or days of computation. In this work, we tackle this problem using a meta-tuning process, based on the use of surrogate benchmarks that are much faster to evaluate. This paper studies the feasibility of this process using the popular irace configurator as the method to be meta-configured. We first study the consistency between the real and surrogate benchmarks using three measures: the prediction accuracy of the surrogate models, the homogeneity of the benchmarks and the list of important algorithm parameters. Afterwards, we use irace to configure irace on those surrogates. Experimental results indicate the feasibility of this process and a clear potential improvement of irace over its default configuration.

KEYWORDS

surrogate benchmarks, automated parameter configuration

1 INTRODUCTION

Good parameter settings are fundamental to obtain the best perfomance of parameterized algorithms and the process of finding performance-optimizing settings for them is often tedious and computationally expensive. For the last fifteen years, the idea of an automated configuration of algorithms has been studied [8] and several general-purpose automated algorithm configuration tools (*configurators*) such as *ParamILS* [11], *GGA* [2], *SMAC* [9] and irace [17] have been proposed and successfully been applied in several studies. Since configurators are also algorithms, they themselves have a number of parameters that must be set. Examples of these parameters are the initial population size of the genetic algorithm-based configurator *GGA*, or the perturbation strength in *ParamILS*. Users can use the default parameter values recommended by the configurator's authors, or modify them according to the characteristics of the configuration scenario, e.g. the configuration budget.

Given these parameters in configurators, the obvious question then becomes how to set these to optimize the performance of the configurator, that is, make it as high performing as possible when tackling configuration tasks. In this work, we focus on optimizing the parameter settings of irace, a publicly available iterated racing-based configurator; some parameters of irace have been studied in [20]. Our idea is to configure the parameters of irace for different configuration benchmarks by means of an algorithm configurator, which here is taken again as irace. Hence, in a sense, we use irace to optimize irace, a process we call in the following meta-tuning. As a side product of this process, we also analyse the importance of irace's parameters.

A first approach to this meta-tuning would be to define a set of configuration benchmarks, each of which consists of a parameter space of a target algorithm, a set of training instances of the optimization problem under concern, and a configuration budget. The configuration of irace on such a benchmark is usually very time consuming. As an example, let's consider the SPEAR-IBM [13] configuration benchmark, which involves the tree search solver SPEAR [3] and a set of Satisfiability (SAT) problem instances taken from the IBM Formal Verification Benchmarks Library [22]. One configuration process for SPEAR is often associated with about two days of CPU time for the execution of this solver [13]. Consequently, configuring irace with irace on this benchmark with a meta-tuning budget of 2000 evaluations would take about 11 CPU years on a single CPU. This amount of CPU time is not really feasible, even with the help of parallel processors.

An alternative is to replace the real benchmarks by surrogate benchmarks that ideally exhibit properties similar to the ones of real benchmarks, while providing inexpensive evaluations. Surrogate models have been extensively used in optimization [15, 16] and are particularly useful when evaluations are very expensive to compute due to lack of computational resources or time constraints. In this study, we perform the meta-tuning of irace using surrogate benchmarks that are created using performance data of real configuration benchmarks. As surrogate benchmarks we use the *empirical performance models (EPMs)* proposed in [14]. These EPMs, which are obtained by training a regression model on available performance data, constitute the surrogate model that is used to predict the performance of a target algorithm configuration on a problem instance. In this way, executing the target algorithm is avoided, thus, saving a significant amount of computation time. With this surrogate approach, an execution of irace on SPEAR-IBM only takes about 5 CPU minutes on our machines, and the meta-tuning using a budget of 2000 evaluations can be finished within about 7 CPU days on a single core. This strong saving of computation time makes the idea of meta-tuning attractive and computationally feasible. Of course, collecting the necessary algorithm performance datasets for building the regression models can be expensive, but this only needs to be done once and the surrogates can be used for any meta-tuning experiments afterwards.

Since the ultimate goal is to study the performance of irace on real-world configuration scenarios, there is an important question we need to answer before doing the meta-tuning on the surrogate benchmarks: can these surrogates represent the landscape of the

real algorithm configuration space for the configuration task? We study this question by analysing the surrogates using three different metrics: the prediction accuracy of the regression models, the homogeneity of the surrogates, and the contribution of algorithm parameters on the predicted performance dataset.

This paper is organized as follows: section 2 presents the related work, section 3 explains the parameters of irace, section 4 shows the surrogate configuration benchmarks used in our study and the analysis on the consistency between them and the original benchmarks, section 5 describes our meta-tuning experimental settings and results, section 6 analyses the importance of irace's parameters, section 7 shows results on a number of real configuration benchmarks, finally, section 8 gives conclusions and a discussion on future works.

2 RELATED WORK

Surrogate modeling approaches have been used in a variety of applications where the evaluation is very costly. In continuous optimization, the EGO algorithm is a well-known example [15]. More recent is the SPOT package [4], which has also been used for the configuration of algorithm parameters. Surrogate models are particularly useful in the context of automated algorithm configuration as they help to avoid expensive runs of algorithms on training instances and, thus, save considerable computation time. They have been used in several other applications of algorithm selection, automated algorithm configuration and algorithm parameter analysis. They are called *empirical hardness models* in algorithm selection and are used as the core of the state-of-art instance-based SAT algorithm selector SATzilla [21]. In automated algorithm configuration, they are used to guide the search of model-based algorithm configurators such as *SMAC* [9] and *GGA++* [1]. For analysing importance of algorithm parameters, an *EPM* is used to build functional data for the functional analysis of variance toolbox *fANOVA* [10]; or as a surrogate for an efficient ablation analysis [5].

From the machine learning's point of view, the meta-tuning can be considered as the hyper-parameter optimization problem. The first work that proposes using surrogate benchmarks for this task is [6], in which the performance of machine learning algorithms was modeled using different regression models and from which tree-based approaches were shown to be the best choice. The regression models in [6] receive a hyper-parameter configuration as input and predict its corresponding test performance on a machine learning task. This is equivalent to a single-instance scenario in the field of automated parameter configuration. For our study, we require surrogate benchmarks dealing with multiple problem instances to resemble typical real-world situations. For this reason, in this work we use the performance data and EPMs provided in [14].

The only work that so far has studied the parameters of irace used three real configuration benchmarks [20]. However, due to the extremely expensive nature of the approach, only a limited number of irace configurations were tested and only single-parameter effects were studied. The obtained results stressed out the importance of the statistical test chosen and its corresponding confidence level.

3 PARAMETERS OF IRACE

In this section, we briefly describe irace and its parameters. For a detailed description of irace, the readers are referred to [17] and the irace's user guide [18]. irace is a configurator, which supports the task of configuring (highly) parameterized optimization algorithms. The target algorithm, i.e., the one that needs to be configured, can be either deterministic or stochastic and its parameters can be categorical, ordinal, integer, continuous or a mix of those categories. irace is based on a machine learning technique called *racing* [19]. Given a fixed configuration budget, instead of running every algorithm configuration on every problem instance and taking the one with the best performance measure value, all configurations are first tested on a limited number of instances. Then a statistical test (e.g., Friedman-test, multiple t-tests with/without correction) is applied to eliminate significantly worse configurations. The remaining ("surviving") algorithm configurations are run on more problem instances before the statistical test is applied again to discard bad configurations. This procedure (*race*) continues until the configuration budget is reached, or until there are only one or a few remaining configurations. At each iteration of irace, a race with Friedman test as the default choice is done. The surviving configurations at the end of each race are used to update a sampling model, from which new configurations for the next race (i.e., the next iteration of irace) are generated.

Table 1 lists the parameters of irace, their types, domains and default values. The original domains of the numerical parameters can be larger than the ones speficied in Table 1, e.g., the domain of parameter *confidence_level* can be [0, 1], however, in this work, we set them in reasonable ranges, which will be used for the meta-tuning experiments in section 5. The first parameter, N^{iter}, defines the number of iterations during the configuration process. Its default value depends on the number of parameters of the target algorithm (denoted as n^{params}). The budget assigned to each iteration is decided based on the second parameter, μ. T^{first} is the number of instances all configurations must be evaluated on before the statistical test is applied the first time in a race. N^{min} is the minimum number of configurations required to continue the execution of a race. An iteration is terminated if either N^{min} is reached or its corresponding budget is exhausted. *test_type* is the statistical test used for elimination; it could be the Friedman test or the Student t-test with/without corrections. The confidence level of the test is set by the parameter *confidence_level*. The sampling model for generating new configurations for each race is updated at the end of each iteration based on the surviving configurations of the previous race. Sometimes this might lead to premature convergence. Therefore, an additional scheme called *soft restart* can be enabled by setting the parameter *enable_soft_restart* as $true$. The last parameter, *elitist_instances*, sets the number of unseen instances initially evaluated in the race and it is active only when the *elitist* strategy is enabled (setting the parameter *enable_elitist* as $true$). This strategy is available in the most recent version of irace and is used to ensure that the best configurations obtained are also the ones that have been evaluated on the largest number of training instances.

4 SURROGATE BENCHMARKS

In this section, we describe the surrogate configuration benchmarks used. These benchmarks are only useful if they can represent real configuration scenarios, i.e., they should show consistency with the original ones to a certain extent. Therefore, before using the surrogate benchmarks for the meta-tuning, we study their consistency with the real benchmarks. First, we check the prediction accuracy of the regression models. Then, we examine the homogeneity of the instance sets regarding the configurations. And finally, we compare the list of important algorithm parameters in both surrogate and

Parameter	Type	Domain	Default value
N^{iter}	integer	[1, 50]	$2 + log_2 n^{params}$
μ	integer	[1, 20]	5
T^{first}	integer	[4, 20]	5
N^{min}	integer	[1, 20]	$2 + log_2 n^{params}$
test_type	categorical	F-test, t-test, t-test-bonferroni, t-test-holm	F-test
confidence_level	real	[0.5, 0.99]	0.95
enable_soft_restart	categorical	true, false	true
enable_elitist	categorical	true, false	true
elitist_instances	integer	[1, 10]	1

Table 1: Parameters of irace

real benchmarks using the functional analysis of variance (*fANOVA*) [10].

4.1 Surrogate configuration benchmarks

In [14], the authors have shown that the running time of a number of algorithms can be well predicted using a regression model based on a random forest. The algorithm performance datasets used to train their prediction models and their model's source code are publicly available[1]. We use these prediction models as surrogate configuration benchmarks. They include four mixed integer linear programming benchmarks (CPLEX-BIGMIX, CPLEX-REG, CPLEX-CORLAT and CPLEX-RCW) and two Satisfiability benchmarks (SPEAR-IBM and SPEAR-SWV) with CPLEX and SPEAR [3] as the target algorithms. In [14], all continuous algorithm parameters were discretized before applying the prediction models. However, in this work, we convert them back to their original continuous domains so that we can study the configuration performance of irace on all parameter types. Due to the large amount of memory required for saving the models, we limit the number of instances to 1000. Also, we add a new configuration benchmark called ACOTSP. It configures the Ant Colony Optimization algorithm ACOTSP for solving the Travelling Salesman Problem using as cost metric the solution quality obtained after running the algorithm for 20 seconds. The seven surrogate configuration benchmarks used in this study are listed in Table 2. A configuration benchmark is composed of a parameterized algorithm, a training instance set, a cost metric and a performance measure. All configuration benchmarks in our study use the mean of the cost metric values over the problem instance set as the performance measure. The prediction models described in [14] are applied on log-transformed performance values since it is well-known that this transformation gives better prediction quality on running time data [21]. However, for the configuration benchmark on solution quality (ACOTSP), we use the original (un-transformed) performance values. All models are built using performance datasets of 1000 random configurations on all problem instances.

4.2 Analysis on prediction accuracy

The prediction accuracy measures the concordance between the real performance of configurations on the instances and their predicted values. High prediction accuracy implies high consistency between the real and the surrogate benchmarks. In [14], the authors have shown that the prediction quality of the random forest model for running time performance data is generally the best compared to

[1]http://www.cs.ubc.ca/labs/beta/Projects/EPMs/

Algorithm	Parameters	Instance sets	#Instances	Metric
CPLEX	76 parameters (int: 6, real: 4, categorical: 66, conditional: 4)	BIGMIX REG CORLAT RCW	1000 1000 1000 1000	Running time
SPEAR	26 parameters (int: 4, real: 12, categorical: 10, conditional: 9)	IBM SWV	765 604	Running time
ACOTSP	11 parameters (int: 4, real: 4, categorical: 3, conditional: 5)	TSP	50	Solution quality

Table 2: The seven configuration benchmarks used in our study.

Benchmark	*CC*	Benchmark	*CC*	Benchmark	*CC*
ACOTSP	0.999	CPLEX-BIGMIX	0.852	SPEAR-SWV	0.9
CPLEX-CORLAT	0.875	CPLEX-RCW	0.723	SPEAR-IBM	0.946
CPLEX-REG	0.701				

Table 3: Mean values of Pearson's correlation coefficient (*CC*) across 10-fold cross validation.

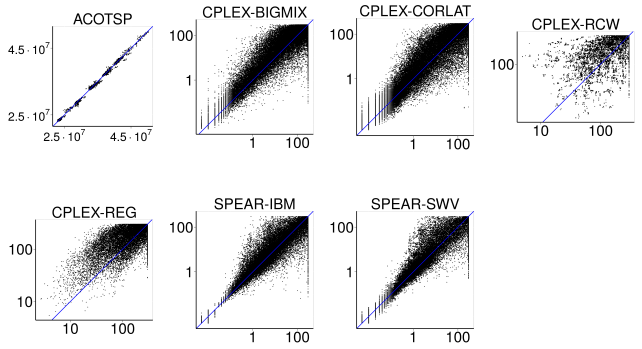


Figure 1: Scatter plots of real performance values (x-axis) and the corresponding prediction values (y-axis) of 100 algorithm configurations on the instance set of each benchmark.

other regression approaches. Since we make some changes in these benchmarks (continuous parameters, number of instances) and add a new one (ACOTSP), we re-examine their prediction accuracy using the *Pearson's correlation coefficient (*CC*)* together with scatter plots. The mean values of *CC* across 10-fold cross validation are shown in Table 3. The closer the value to one, the stronger the linear correlation between the predicted and the real performance values. The scatter plots of the test set in one of the 10 folds are given in Figure 1. The prediction quality in all CPLEX and SPEAR benchmarks is generally in line with results reported in [14]. For the new ACOTSP benchmark, the high value of the *CC* and the scatter plot indicate a good prediction accuracy, this is not surprising since for this specific benchmark we know that most of the variability in the algorithm performance can be captured by the scaling of objective function values by instance size.

4.3 Analysis on homogeneity

The homogeneity of a problem instance set measures the similarity of the hardness of the instances w.r.t. different configurations. The

Benchmark	m_{real}	$m_{prediction}$	$mean_{1000}$	std_{1000}
ACOTSP	0.9689	0.9724	0.9665	0.0029
CPLEX-BIGMIX	0.1323	0.1796	0.221	0.0046
CPLEX-CORLAT	0.3233	0.4192	0.4406	0.0112
CPLEX-RCW	0.28	0.2936	0.368	0.0097
CPLEX-REG	0.4295	0.5077	0.7029	0.0144
SPEAR-IBM	0.2414	0.267	0.3118	0.0067
SPEAR-SWV	0.176	0.2034	0.2468	0.0057

Table 4: Kendall W values measured over three different datasets.

homogeneity of the configuration scenario might have a strong influence on the design of a configurator. If a scenario is homogeneous, i.e., algorithm configurations that work well on a subset of the instance set have a high chance of doing well on the other instances, the configurator does not need to evaluate algorithm configurations on many problem instances, in this way the configuration budget can be reserved for exploring the algorithm configuration space. On the other hand, if a scenario is heterogeneous, the configurator needs to evaluate as many instances as possible to make sure that the final best configurations work generally well on the whole instance set. We measure the homogeneity of the studied configuration benchmarks using the *Kendall concordance coefficient (W)*, as done in [20]. The closer W is to one, the stronger is the agreement; $W = 1$ indicates perfect homogeneity. We calculate the W values on three datasets:

- m_{real}: W value of the real performance of the 1000 random configurations used to train the surrogate model.
- $m_{prediction}$: W value of the predicted performance of the same 1000 random configurations.
- $mean_{1000}$ and std_{1000}: mean and standard deviation of the W values of 10 sets of the predicted performance of 1000 different randomly generated configurations.

All these values are reported in Table 4. The $mean_{1000}$ and std_{1000} values might give a better overview on the homogeneity of the surrogate benchmarks, given that they are not based on the configurations that were used to train the models. The benchmark ACOTSP is very homogeneous as its m_{real} value is nearly one, while the other ones are more heterogeneous, especially CPLEX-BIGMIX and SPEAR-SWV. All surrogate benchmarks are more homogeneous than their original ones, as $m_{prediction} > m_{real}$ in all cases. Except for ACOTSP, we have also that $mean_{1000} > m_{prediction}$. This indicates that the surrogate landscapes are somehow flattened with respect to the original search space, this is expected due to the trade-off of using prediction models instead of real target algorithm evaluations to obtain the performance values. The $mean_{1000}$ value can be used as a general indication for the whole surrogate benchmark search space since the std_{1000} values are relatively small. Additionally, the relative ordering of homogeneity between the original benchmarks is reproduced by the surrogates, which can be considered as an indication of the consistency between the surrogate and the real benchmarks.

4.4 Analysis on important algorithm parameters

Given a performance dataset of a configurable algorithm on a set of problem instances, it is possible to analyse which algorithm parameters impact the most on algorithm performance [10]. Since this information can partially represent the landscape of the corresponding algorithm configuration space, we do the analyses on the surrogate and the real performance data and compare them. Similar analyses would indicate consistency between the real and surrogate benchmarks. The toolbox *fANOVA (functional Analysis of Variance)* [10] is used for this task. fANOVA first fits a regression model based on a random forest on the provided performance dataset to predict the average performance of every algorithm configuration over the whole problem instance set. Afterwards, the variance of algorithm performance is decomposed into components associated with every subset of algorithm parameters. The ratio between the variance explained by each subset and the overall variance indicates the importance of the subset. We apply fANOVA for each benchmark to the original performance dataset of 1000 random configurations, and their corresponding predicted performance dataset. Due to the large amount of memory required with fANOVA on large dataset, we limit the maximum number of instances in this analysis to 200, i.e., the maximum size of fANOVA's input performance dataset is 200000 data points. Table 5 shows results of fANOVA on both original and prediction cases on some benchmarks. Results of the fANOVA analysis on all benchmarks can be found in Table 7 of the appendix A.1. Since the full lists are long, we only show effects with a percentage of contribution on the overall performance variance above 5%. This threshold is chosen arbitrarily. We can see that the list of the important main and pairwise interaction effects are the same in both cases. The percentage values are not exactly the same, but the amount of difference is relatively small. These claims hold for not only the three benchmarks shown in Table 5 but also for all benchmarks. Therefore, we can conclude that the surrogate and the real benchmarks are consistent according to the algorithm parameter importance aspect.

5 EXPERIMENTAL RESULTS

The analyses performed in Section 4 showed an acceptable level of consistency between the original and the surrogate benchmarks. We therefore now use irace to configure the parameters of irace on the configuration scenarios defined by these surrogate benchmarks. The highest level irace (the one that is used to tune the execution of irace as a configurator for target algorithms) we call from now on meta-irace. Meta-irace uses the default irace's setting in all the following experiments to configure irace. The training instances for meta-irace correspond to random number seeds that are used to execute target-irace. The execution of target-irace, given a specific irace parameter setting δ, is used to determine how well this configuration δ of target-irace is capable to configure the actual target algorithm (here CPLEX, SPEAR, or ACOTSP). Instead of executing the target algorithm on one of its instances, we use the predictions from the surrogate benchmark as the predicted performance. In turn, the performance of a candidate configuration of irace (defined by δ) in meta-irace is then evaluated as the mean performance of the final configuration obtained by target-irace (using configuration δ) on the training set of the actual target algorithm (as estimated by the surrogate benchmark). Table 1 gives the domain of the parameters configured by meta-irace. For an explanation of these parameters, we refer back to Section 3. The default settings of N^{iter} and N^{min} depend on the number of parameters of the target algorithm configured by irace; their default values are 8, 6 and 5 for the CPLEX, SPEAR and ACOTSP benchmarks respectively. The default statistical test of irace is the F-test (Friedman test). However,

Original performance dataset	Predicted performance dataset
CPLEX-RCW	CPLEX-RCW
Sum of fractions for main effects 45.53%	Sum of fractions for main effects 44.96%
Sum of fractions for pairwise interaction effects 21.56%	Sum of fractions for pairwise interaction effects 21.54%
38.23% due to main effect: preprocessing_reduce	37.87% due to main effect: preprocessing_reduce
8.74% due to interaction: preprocessing_reduce x mip_strategy_subalgorithm	8.46% due to interaction: preprocessing_reduce x mip_strategy_subalgorithm
SPEAR-SWV	SPEAR-SWV
Sum of fractions for main effects 59.95%	Sum of fractions for main effects 61.88%
Sum of fractions for pairwise interaction effects 17.91%	Sum of fractions for pairwise interaction effects 17.14%
50.59% due to main effect: sp-var-dec-heur	52.37% due to main effect: sp-var-dec-heur
7.72% due to main effect: sp-phase-dec-heur	7.94% due to main effect: sp-phase-dec-heur
6.54% due to interaction: sp-var-dec-heur x sp-phase-dec-heur	6.65% due to interaction: sp-var-dec-heur x sp-phase-dec-heur
ACOTSP	ACOTSP
Sum of fractions for main effects 75.77%	Sum of fractions for main effects 79.23%
Sum of fractions for pairwise interaction effects 13.24%	Sum of fractions for pairwise interaction effects 9.96%
69.11% due to main effect: localsearch	74.70% due to main effect: localsearch

Table 5: fANOVA results on both the original and the predicted performance dataset on three benchmarks. For the sake of brevity, only effects with percentage of contribution on performance variance larger than 5% are shown.

in our experiments, we set the default test to the Student t-test due to the fact that the latter one is more compatible with the performance measure to be optimized in the surrogate benchmarks. When comparing different irace parameter settings, the performance of a parameter setting (default or tuned) is evaluated as the performance of the best configurations found by 50 executions of irace using those parameter settings. We set the configuration budget as 10 000 evaluations for all CPLEX and SPEAR benchmarks, and 5 000 evaluations for ACOTSP. These budgets are "reasonable" according to the knowledge we have of these benchmarks. However, since the selected configuration budget could have an effect on the best irace settings, the results of this study might not apply for different budgets.

5.1 Meta-tuning with a single benchmark

We first perform the configuration of irace on each surrogate benchmark scenarios. The aim of these experiments is to determine, individually for each benchmark, if it is possible to find a better parameter setting for irace than the current default one. Meta-irace had a budget of 5000 target-irace executions. Each meta-irace execution returns the five best target-irace parameter settings found (*elites*). Figure 2 compares the performance of irace using the elite and default settings. The best elite setting significantly improves the default setting on all surrogate benchmarks, but the amount of improvement varies from benchmark to benchmark. A list of the best settings for all benchmarks and the corresponding default setting is shown in Table 6. Having a closer look at these settings, we see a number of common characteristics shared among all benchmarks: compared to the corresponding default settings, the values of parameters N^{min} and *confidence_level* are generally lower, while N_{iter}'s values are higher, in addition, the elitist strategy is now always disabled.

These settings suggest that a more intense search is beneficial for the configuration of these benchmarks. A higher number of iterations and a low confidence statistical test allows the probabilistic sampling model of irace to converge to a small area of the parameter search space. Moreover, a low number of surviving configurations (N^{min}) enables that only the very best configurations continue guiding the search. The elitist behaviour (default) is not selected by the configuration process; we believe the reason for this is that non-elitist strategy is more flexible, allowing to adapt strongly the search behaviour of irace through the parameters settings.

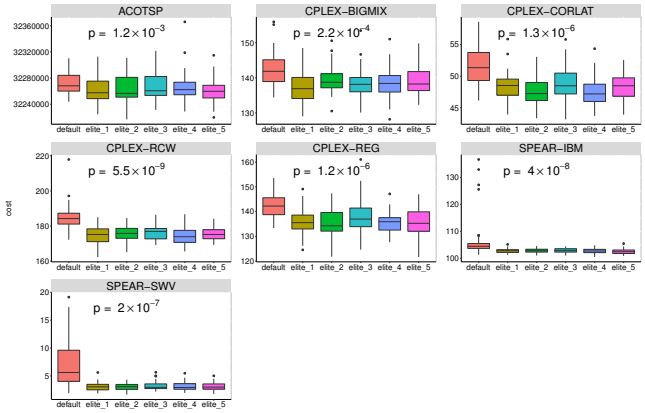


Figure 2: Performance of irace using the default setting and the five elite settings obtained by meta-irace on each surrogate benchmark. The pairwise Wilcoxon signed rank test at confidence level 99% indicates statistically significant differences in all cases (p-value is given in each plot) in favor of the best performing elite settings.

5.2 Meta-tuning with multiple benchmarks

In this section, we study the possibility of finding an irace parameter setting that obtains an improved perfomance simultaneously for multiple benchmarks. This is necessary, as target algorithm and benchmark set specific settings as in the previous section would correspond to an over-tuning of target-irace. We choose to perform the configuration of irace using the surrogate benchmarks that set CPLEX as the target algorithm. We select only these benchmarks to keep constant the number of parameters of the target algorithm. The budget for the meta-tuning is increased to 10000 target-irace evaluations. The final performance is evaluated, as previously described, by 50 executions of irace on each benchmark. Figure 3 shows the normalized test performance values of irace using the default setting and the five elite irace settings obtained by meta-irace for all four CPLEX benchmarks. Again, the improvement over irace with the default setting is statistically significant, as the corresponding Wilcoxon signed rank test's *p-value* is 5.5×10^{-25}.

As shown in Table 6, we get the same observations on the difference between the best and the default settings of irace as in the

	N^{iter}	N^{min}	*confidence_level*	*enable_elitist*	*elitist_instances*	*test_type*	μ	T^{first}	*enable_soft_restart*
ACOTSP	15	3	0.87	false		t-test	11	9	false
SPEAR-IBM	26	4	0.53	false		t-test-bonferroni	5	20	true
SPEAR-SWV	32	1	0.79	false		t-test-bonferroni	20	17	false
CPLEX-BIGMIX	33	2	0.74	false		t-test	9	8	true
CPLEX-CORLAT	14	5	0.81	false		t-test	7	5	false
CPLEX-RCW	35	1	0.82	false		t-test	3	5	false
CPLEX-REG	35	2	0.52	false		t-test-holm	4	6	true
CPLEX-all	37	1	0.65	false		t-test	5	5	true
default-ACO-TSP	5	5	0.95	true	1	t-test	5	5	true
default-SPEAR	6	6	0.95	true	1	t-test	5	5	true
default-CPLEX	8	8	0.95	true	1	t-test	5	5	true

Table 6: Best parameter settings for irace obtained from the meta-irace experiments and the corresponding default setting of irace for each target algorithm. The table contains results of both single-benchmark irace configurations and the multiple-benchmark irace configurations.

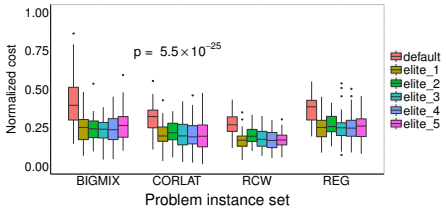


Figure 3: Normalized performance of irace using the default setting and the five elite settings obtained by meta-irace configuring irace on the four CPLEX benchmarks and pairwise Wilcoxon signed rank test p-value (confidence level 99%) of the performance of irace using the default setting and the best performing elite setting.

single-benchmark case, i.e., the elite settings have higher values for N^{iter}, lower values for N^{min} and *confidence_level*, and the elitist strategy is disabled.

The observations described above are applied for not only the best irace configurations, but also the elite ones. Details of those elite configurations are available in Table 8 of the appendix A.2.

6 ANALYSIS ON IRACE'S PARAMETERS

As shown in Section 5, configuring the parameters of irace produced significant improvements over the default parameter settings on all surrogate benchmarks. In this section, we study the contribution of irace parameters to those improvements. In particular, we investigate the following questions:

- Which parameters should be changed to obtain better performance?
- What characteristics have the best parameter settings? Do they match the ones observed in the settings that improved the default performance on the surrogate scenarios in Section 5?
- Which parameters do not have a significant effect on the performance?
- Do parameters have the same contribution to performance for all benchmarks? Are the best parameter settings scenario-dependent?

We study the contribution of single parameters and pairwise parameter interactions to the performance of irace using the fANOVA toolbox [10]. We also show the possibility of an important higher-order interaction which is not given by the fANOVA analysis. Finally, we study the last question further by showing that the set of "essential" parameters on the path between irace default configuration and the elite configurations, i.e., if the values of those parameters in the default configuration are set as in the elite configurations, the performance of the resulting configurations are not statistically different from the elites themselves, varies among different benchmarks.

6.1 Analysis of single parameters and pairwise interactions using fANOVA

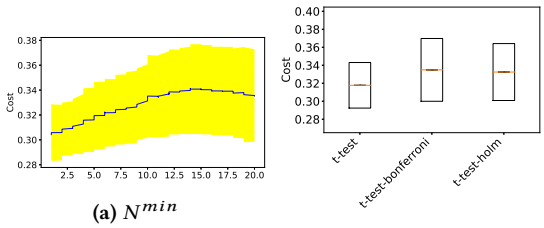
Each irace configuration experiment produces a dataset of the evaluated performance of different irace parameter settings. The *toolbox fANOVA* [10] can then be applied on that dataset to analyse the importance of each single parameter (i.e., single effect) and each parameter pair (i.e., pairwise interaction effect). For each effect, fANOVA provides two outputs: a value representing the importance of the effect, and a marginal plot that show the "quality" of the values (with a degree of uncertainty) of all parameters belonging to the effect. Consequenlty, this tool can be used to understand which parameters are important for performance and which values of these parameters are the best. For the analysis we use the fANOVA option that focuses only on the area of the configuration space where the average performance is not worse than the one of the default settings. As an example, we give a part of the output of fANOVA analysis on configuring irace with multiple CPLEX benchmarks:

- Sum of fractions for main effects 24.59%
- Sum of fractions for pairwise interaction effects 28.15%
- 10.35% due to main effect: N^{min}
- 3.87% due to main effect: *test_type*
- 3.57% due to interaction: N^{iter} x N^{min}

The single effect of parameter N^{min} explains the highest percentage of the variation of the performance (10.35%) and the good values for N^{min} in the marginal plot (Figure 4a) are not close to the default value (8), so we can conclude that N^{min}'s default value should be adjusted to obtain better performance. On the other hand, the best choice for parameter *test_type* in its marginal plot (Figure 4b) is the same as its default (t-test).

We apply fANOVA on all irace configuration experiments. Details of these results are shown in the appendix A.3. Here we summarize the major points that we have observed.

(i) The total percentage of variance explained by single and pairwise effects is in the range of 52% to 66%, which indicates that there can be room for other important effects of higher-order interaction. (ii) For the irace configuration experiments with the cost metric of



(a) N^{min}

(b) ***test_type***

Figure 4: Marginal plots of the two single effects N^{min} and *test_type* on the performance dataset of configuring irace with multiple CPLEX surrogate benchmarks.

running time, the two parameters N^{min} and N^{iter} are most often among the top contributing effects and the marginal plots confirm the observations in Section 5: N^{min} should be set lower than its default value, but N^{iter} higher than the default. For the irace configuration experiment with the cost metric of solution quality (ACOTSP), only N^{iter} seems to play an important role for improving irace settings. (iii) The observation obtained in Section 5 for parameter *confidence_level* is supported by the marginal plots in all experiments; however, the percentage values of the relevant effects are not always on the top list, which indicates that this parameter might not be as important as N^{min} and N^{iter}. (iv) The degree of contribution of irace parameters on the improvement over the default settings varies among different experiments, especially for the parameters that are not N^{iter} and N^{min}.

6.2 A higher-order interaction effect

The analysis performed by fANOVA does not explain the consistent selection of the non-elitist strategy reported in Section 5. This could evidence that the importance of this parameter lies in some higher-order interaction effect that the toolbox fANOVA doesn't analyse on. We examine this hypothesis by running the following experiment on each benchmark: we take the best settings found by configuring irace and enable the elitist strategy in them (with *elitist_instances* set as 1), then we compare the test performance of the original and the modified best settings to see whether enabling the elitist strategy causes significant deterioration on the performance or not. Results show that the hypothesis is confirmed on all experiments except SPEAR-SWV and ACOTSP:

- For all CPLEX experiments (including the multiple-benchmark) and for SPEAR-IBM, there is statistically significant worsening (with a confident level of 95%) on all elite configurations.
- For SPEAR-SWV and ACOTSP, there is worsening on the elite configurations, but it is not always statistically significant.

6.3 Essential parameters

In this part, we show that the list of the essential parameters actually varies among different benchmarks. We identify this list by sequentially setting a parameter in the elite configurations as its default value as long as no statistically significant worsening (with a confidence level of 95%) is observed on any elite configuration. This procedure is called ablation [7]. Below are the lists found for four of the benchmarks, which imply that the contribution of irace's parameters can have some dependency with the specific configuration scenario to some certain degree:

- ACOTSP: N^{iter}, N^{min} and *enable_elitist*
- CPLEX-REG: N^{iter}, N^{min} and *enable_elitist*
- CPLEX-BIGMIX: N^{iter}, N^{min}, *enable_elitist* and *confidence_level*
- SPEAR-SWV: N^{iter}, N^{min}, *enable_elitist* and T^{first}

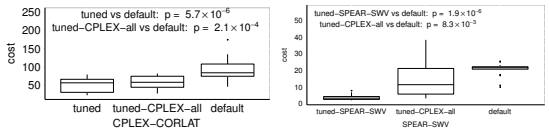
7 RESULTS ON REAL BENCHMARKS

In this section, we compare the performance of the default and the best irace settings found in Section 5 on a number of available real configuration benchmarks. We first examine the performance of the best irace settings obtained by meta-irace by testing them on the benchmarks for which they were configured on. For all irace executions, we use the same experimental settings (irace configuration budget and cut-off time) as in the irace configuration with surrogate models. Figure 5 shows the real test performance of 20 irace executions for each setting on the three benchmarks CPLEX-CORLAT, SPEAR-SWV and ACOTSP, whose problem instance sets are publicly available. In addition, the best setting from irace configuration with multiple CPLEX benchmarks, denoted as *tuned all*, is also tested on both CPLEX-CORLAT and SPEAR-SWV. The p-value given by the pairwise Wilcoxon signed rank tests between the default and the best configurations in each plot confirms our results with the surrogates on CPLEX-CORLAT and SPEAR-SWV. However, the result for ACOTSP shows that the tuned irace configuration performs statistically significantly worse than irace's default setting, contradicting the results on the surrogate benchmarks. Hence, despite the indication given in Section 4 that the surrogate benchmark could be representative for the real benchmark, there is apparently some discrepancy between surrogates and real benchmarks that led to this result.

We also did a number of tests in which the default and the best irace settings are run on real benchmarks that were not used to build the surrogates studied in the previous sections. In particular, we tested the tuned configurations of irace obtained from ACOTSP on a benchmark with Iterated Local Search algorithm (43 parameters) and solution quality as the cost metric, and CPLEX with multiple benchmarks on Regions100 [11], Regions200 [11] and the SAT-solver Lingeling [12] with 137 parameters and with running time as the cost metric. In most of these benchmarks we observe rather similar behavior between the tuned irace settings and the default settings (no significant difference) with the exception of the Regions100 benchmark, where the tuned irace settings improve statistically significantly over the default settings.

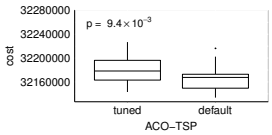
In general, the results on the real configuration benchmarks indicate that the configuration of irace parameters on the surrogate benchmark has helped to some extent, but not as much as the results on the surrogate benchmark alone would have suggested. Hence, the surrogate appears not to be a complete replacement of the real configuration benchmarks, which may be due to the smoothing behavior of the surrogate, the fact that the interesting part of the configuration space with high performing configurations may be too sparsely sampled in the available data, or that the inherent stochasticity in the configuration benchmarks is not well matched in the surrogates.

More details on the real configuration benchmarks and results are provided in the appendix A.4. The benchmarks can be downloaded from the supplementary website of this paper [**?**]



(a) CPLEX-CORLAT

(b) SPEAR-SWV



(c) ACOTSP

Figure 5: Real test performance of irace using default configuration and the best configuration obtained by configuring irace on the same benchmark's surrogate. The best configuration obtained from the experiment with multiple CPLEX benchmarks, namely *tuned-CPLEX-all*, is also tested.

8 CONCLUSION AND FUTURE WORK

In this paper, we have investigated the idea of using surrogate configuration benchmarks for studying the parameters of the automated algorithm configurator irace. Six benchmarks with the cost metric of running time [14] and one with solution quality were used. Three measures, namely the (i) prediction quality of the surrogate models, (ii) the homogeneity of the benchmarks w.r.t. configurations of the target algorithm, and (iii) the list of important algorithm parameters from fANOVA [10] showed a reasonable level of consistency between the surrogate and the real benchmarks. The two meta-tuning experiments using irace to configure irace on the surrogate benchmarks, on single benchmark sets and across multiple benchmark sets, indicate that statistically significant improvements over irace default settings are possible. Further analysis indicated the parameters that most affected the performance of irace, information that may be helpful in directing the further development of this configurator. In particular, these results indicate that other than the current default setting may lead to somewhat better performance. Tests of the tuned irace settings on real configuration benchmark , however, led to mixed conclusions and could not fully confirm the performance advantages of the irace settings tuned on the surrogate benchmarks. Supplementary data of this work is available at [**?**].

There are a number of directions in which this work can be extended. A first is to extend the study of the potential differences in the surrogate benchmarks and the real configuration benchmarks that lead to the overly optimised tuned settings of irace. Conjectures we are following are to examine the role of continuous parameters, increasing the sampling of high-performance configurations to obtain better quality surrogates in the relevant parts of the configuration space, or examining the smoothing effect the predictions seem to have. While trying to further study such effects on the available surrogates, we are currently collecting and building more surrogate benchmarks, with the goal of having a library of surrogate configuration benchmarks with a variety of characteristics and being more representative of different configuration scenarios. Such a library would be useful not only for irace, but also for studying the parameters of other automated configurators. Furthermore, it would be also useful to study the dependency of the best configurations of irace or other configurators on the characteristics (e.g. number of parameters, homogeneity of benchmark, configuration target, level of stochasticity etc.) of the configuration tasks. A better understanding of this relationship would also be useful to better identify the particular advantages of the various configurators that are currently available. A useful step would certainly also be a automated selection of the appropriate configurator in dependence of the characteristics of the configuration scenario.

REFERENCES

[1] C. Ansótegui, Y. Malitsky, H. Samulowitz, M. Sellmann, and K. Tierney. Model-based genetic algorithms for algorithm configuration. *IJCAI-15*, pp. 733–739. IJCAI/AAAI Press, Menlo Park, CA (2015).

[2] C. Ansótegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In: *CP 2009*, LNCS, vol. 5732, pp. 142–157. Springer (2009).

[3] D. Babić and F. Hutter. Spear theorem prover. In *SAT'08: Proceedings of the SAT 2008 Race* (2008).

[4] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. The sequential parameter optimization toolbox. In: *Experimental Methods for the Analysis of Optimization Algorithms*, pp. 337–360. Springer (2010).

[5] A. Biedenkapp, M. Lindauer, K. Eggensperger, F. Hutter, C. Fawcett, and H. H. Hoos. Efficient parameter importance analysis via ablation with surrogates. In *Proceedings of the AAAI conference*, to appear. (2017)

[6] N. Dang, L. Pérez Cáceres, T. Stützle and P. de Causmaecker. Hoos. Supplementary material: Configuring irace using surrogate configuration benchmarks. http://iridia.ulb.ac.be/supp/IridiaSupp2017-05. (2017)

[7] K. Eggensperger, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Efficient benchmarking of hyperparameter optimizers via surrogates. In: *AAAI*, pp. 1114–1120. AAAI Press (2015).

[8] C. Fawcett and H. H. Hoos. Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, 22(4):431–458 (2016).

[9] H. H. Hoos. Automated algorithm configuration and parameter tuning. In: *Autonomous Search*, pp. 37–71. Springer (2012).

[10] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In: *LION 5*, LNCS, vol. 6683, pp. 507–523. Springer (2011).

[11] F. Hutter, H. H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proc. of the 31th Int. Conf. on Machine Learning*, vol. 32, pp. 754–762 (2014).

[12] 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 (2009).

[13] F. Hutter, M. T. Lindauer, A. Balint, S. Bayless, H. H. Hoos, and K. Leyton-Brown. The configurable SAT solver challenge (CSSC). *Artificial Intelligence*, 243:1–25 (2017).

[14] F. Hutter, M. López-Ibáñez, C. Fawcett, M. T. Lindauer, H. H. Hoos, K. Leyton-Brown, and T. Stützle. AClib: a benchmark library for algorithm configuration. In:*LION 8*, LNCS, vol. 8426, pp. 36–40. Springer (2014).

[15] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, 206:79–111 (2014).

[16] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492 (1998).

[17] S. Koziel, D. E. Ciaurri, and L. Leifsson. Surrogate-based methods. *Computational Optimization, Methods and Algorithms, Studies in Computational Intelligence*, vol. 356, pp. 33–59. Springer (2011).

[18] 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*, pp. 43–58 (2016).

[19] M. López-Ibáñez, L. Pérez Cáceres, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The irace package: User guide. Tech. Rep. TR/IRIDIA/2016-004, IRIDIA, Université Libre de Bruxelles, Belgium (2016).

[20] O. Maron and A. W. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Research*, 11(1–5):193–225 (1997).

[21] L. Pérez Cáceres, M. López-Ibáñez, and T. Stützle. An analysis of parameters of irace. In: *EvoCOP 2014*, LNCS, vol. 8600, pp. 37–48. Springer (2014).

[22] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. SATzilla: portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606 (2008)

[23] E. Zarpas. Benchmarking SAT solvers for bounded model checking. In:*Int. Conf. on Theory and Applications of Satisfiability Testing*, vol. 3569, pp. 340–354 (2005).

A APPENDIX

A.1 fANOVA analysis on all benchmarks

Original performance dataset	Predicted performance dataset
CPLEX-BIGMIX Sum of fractions for main effects 39.75% Sum of fractions for pairwise interaction effects 18.76% 14.72% due to main effect: mip_strategy_subalgorithm 9.73% due to main effect: mip_strategy_variableselect 6.56% due to main effect: mip_limits_cutsfactor 5.57% due to main effect: preprocessing_reduce	CPLEX-BIGMIX Sum of fractions for main effects 42.77% Sum of fractions for pairwise interaction effects 17.31% 17.98% due to main effect: mip_strategy_subalgorithm 9.47% due to main effect: mip_strategy_variableselect 7.61% due to main effect: mip_limits_cutsfactor
CPLEX-REG Sum of fractions for main effects 32.73% Sum of fractions for pairwise interaction effects 21.70% 19.12% due to main effect: mip_strategy_subalgorithm 8.10% due to main effect: mip_strategy_variableselect 5.07% due to interaction: mip_strategy_variableselect x mip_strategy_subalgorithm	CPLEX-REG Sum of fractions for main effects 34.17% Sum of fractions for pairwise interaction effects 22.30% 19.61% due to main effect: mip_strategy_subalgorithm 8.55% due to main effect: mip_strategy_variableselect 5.72% due to interaction: mip_strategy_variableselect x mip_strategy_subalgorithm
CPLEX-RCW Sum of fractions for main effects 45.53% Sum of fractions for pairwise interaction effects 21.56% 38.23% due to main effect: preprocessing_reduce 8.74% due to interaction: preprocessing_reduce x mip_strategy_subalgorithm	CPLEX-RCW Sum of fractions for main effects 44.96% Sum of fractions for pairwise interaction effects 21.54% 37.87% due to main effect: preprocessing_reduce 8.46% due to interaction: preprocessing_reduce x mip_strategy_subalgorithm
CPLEX-CORLAT Sum of fractions for main effects 27.68% Sum of fractions for pairwise interaction effects 21.03% 9.45% due to main effect: mip_strategy_variableselect 7.08% due to main effect: mip_strategy_subalgorithm 5.30% due to main effect: mip_cuts_covers	CPLEX-CORLAT Sum of fractions for main effects 30.21% Sum of fractions for pairwise interaction effects 20.55% 10.36% due to main effect: mip_strategy_variableselect 7.85% due to main effect: mip_strategy_subalgorithm 5.54% due to main effect: mip_cuts_covers
SPEAR-IBM Sum of fractions for main effects 77.32% Sum of fractions for pairwise interaction effects 8.49% 75.53% due to main effect: sp-var-dec-heur	SPEAR-IBM Sum of fractions for main effects 78.32% Sum of fractions for pairwise interaction effects 8.23% 76.73% due to main effect: sp-var-dec-heur
SPEAR-SWV Sum of fractions for main effects 59.95% Sum of fractions for pairwise interaction effects 17.91% 50.59% due to main effect: sp-var-dec-heur 7.72% due to main effect: sp-phase-dec-heur 6.54% due to interaction: sp-var-dec-heur x sp-phase-dec-heur	SPEAR-SWV Sum of fractions for main effects 61.88% Sum of fractions for pairwise interaction effects 17.14% 52.37% due to main effect: sp-var-dec-heur 7.94% due to main effect: sp-phase-dec-heur 6.65% due to interaction: sp-var-dec-heur x sp-phase-dec-heur
ACOTSP Sum of fractions for main effects 75.77% Sum of fractions for pairwise interaction effects 13.24% 69.11% due to main effect: localsearch	ACOTSP Sum of fractions for main effects 79.23% Sum of fractions for pairwise interaction effects 9.96% 74.70% due to main effect: localsearch

Table 7: `fANOVA` results on both the original and the predicted performance dataset of every benchmark. For the sake of brevity, only effects with percentage of contribution on performance variance larger than 5% are shown.

A.2 Elite configurations obtained from all experiments with meta-irace and the surrogate benchmarks

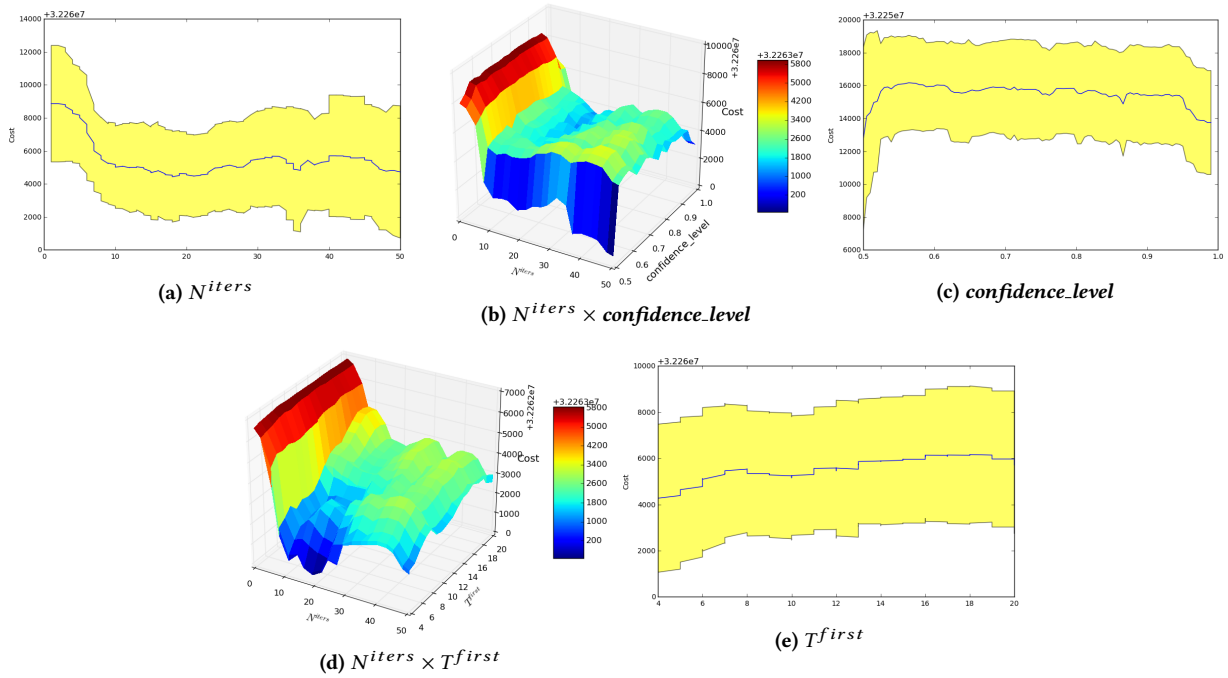
	N^{iter}	N^{min}	*confidence_level*	*enable_elitist*	*elitist_instances*	*test_type*	μ	T^{first}	*enable_soft_restart*
ACO-TSP	15	3	0.87	false		t-test	11	9	false
	15	4	0.9	false		t-test	13	8	false
	11	1	0.9	false		t-test	10	8	false
	15	3	0.86	false		t-test	11	9	false
	13	4	0.91	false		t-test	13	9	false
default-ACO-TSP	5	5	0.95	true	1	t-test	5	5	true
SPEAR-IBM	26	4	0.53	false		t-test-bonferroni	5	20	true
	26	4	0.53	false		t-test-bonferroni	4	20	true
	24	2	0.76	false		t-test	12	17	false
	24	3	0.8	false		t-test	11	18	false
	26	3	0.76	false		t-test	9	20	false
SPEAR-SWV	32	1	0.79	false		t-test-bonferroni	20	17	false
	35	2	0.79	false		t-test-bonferroni	18	19	false
	38	1	0.78	false		t-test-bonferroni	18	19	false
	40	1	0.75	false		t-test-bonferroni	19	19	false
	32	1	0.79	false		t-test-bonferroni	20	17	false
default-SPEAR	6	6	0.95	true	1	t-test	5	5	true
CPLEX-BIGMIX	33	2	0.74	false		t-test	9	8	true
	42	2	0.65	false		t-test	6	8	true
	33	2	0.75	false		t-test	8	8	true
	42	2	0.65	false		t-test	6	8	true
	40	1	0.68	false		t-test	8	7	true
CPLEX-CORLAT	14	5	0.81	false		t-test	7	5	false
	13	1	0.69	false		t-test	6	7	false
	13	5	0.79	false		t-test	6	5	false
	16	5	0.73	false		t-test	5	6	false
	17	3	0.85	false		t-test	4	5	false
CPLEX-RCW	35	1	0.82	false		t-test	3	5	false
	20	1	0.75	false		t-test	3	4	true
	20	1	0.73	false		t-test	3	4	true
	31	1	0.76	false		t-test	2	5	false
	21	1	0.75	false		t-test	3	4	true
CPLEX-REG	35	2	0.52	false		t-test-holm	4	6	true
	29	1	0.55	false		t-test-holm	4	5	true
	37	1	0.51	false		t-test-holm	4	5	false
	29	1	0.55	false		t-test-holm	3	5	true
	12	1	0.77	false		t-test	7	5	true
CPLEX-all	37	1	0.65	false		t-test	5	5	true
	37	1	0.66	false		t-test	5	5	true
	32	2	0.79	false		t-test	3	5	true
	25	2	0.83	false		t-test	7	6	false
	33	2	0.79	false		t-test	3	4	true
default-CPLEX	8	8	0.95	true	1	t-test	5	5	true

Table 8: Elite configurations for irace obtained from the meta-tuning experiments and the corresponding default configuration of irace for each target algorithm. The table contains results of both the single-benchmark meta-tunings and the multiple-benchmark meta-tuning

A.3 fANOVA analysis on irace's parameters

ACOTSP	SPEAR-SWV
Sum of fractions for main effects 24.47%	Sum of fractions for main effects 30.38%
Sum of fractions for pairwise interaction effects 34.04%	Sum of fractions for pairwise interaction effects 34.10%
9.16% due to main effect: N^{iters}	8.91% due to main effect: N^{iters}
5.07% due to interaction: N^{iters} x *confidence_level*	7.83% due to main effect: N^{min}
4.96% due to main effect: *confidence_level*	4.23% due to interaction: *elitist_instances* x *enable_elitist*
4.17% due to interaction: N^{iters} x T^{first}	4.23% due to main effect: *elitist_instances*
3.73% due to main effect: T^{first}	3.58% due to interaction: N^{iters} x N^{min}
SPEAR-IBM	CPLEX-all
Sum of fractions for main effects 24.33%	Sum of fractions for main effects 24.59%
Sum of fractions for pairwise interaction effects 28.95%	Sum of fractions for pairwise interaction effects 28.15%
10.63% due to main effect: N^{min}	10.35% due to main effect: N^{min}
7.31% due to main effect: N^{iters}	3.87% due to main effect: *test_type*
6.04% due to interaction: N^{iters} x N^{min}	3.57% due to interaction: N^{iters} x N^{min}
	3.39% due to interaction: N^{min} x T^{first}
	3.27% due to main effect: N^{iters}
CPLEX-BIGMIX	CPLEX-REG
Sum of fractions for main effects 37.76%	Sum of fractions for main effects 24.10%
Sum of fractions for pairwise interaction effects 29.87%	Sum of fractions for pairwise interaction effects 38.28%
20.74% due to main effect: N^{min}	6.14% due to main effect: T^{first}
4.68% due to main effect: *enable_elitist*	6.08% due to interaction: N^{iters} x T^{first}
4.61% due to interaction: N^{iters} x N^{min}	6.01% due to main effect: N^{min}
4.57% due to main effect: N^{iters}	5.51% due to interaction: T^{first} x *confidence_level*
3.28% due to interaction: N^{min} x *confidence_level*	4.98% due to main effect: *confidence_level*
	4.81% due to interaction: N^{min} x T^{first}
	3.94% due to main effect: N^{iters}
	3.22% due to interaction: N^{min} x *confidence_level*
CPLEX-CORLAT	CPLEX-RCW
Sum of fractions for main effects 23.76%	Sum of fractions for main effects 35.44%
Sum of fractions for pairwise interaction effects 31.47%	Sum of fractions for pairwise interaction effects 31.33%
7.88% due to main effect: N^{min}	11.29% due to main effect: N^{min}
4.58% due to main effect: T^{first}	9.64% due to main effect: *test_type*
3.84% due to interaction: N^{min} x T^{first}	4.14% due to interaction: *test_type* x N^{min}
3.59% due to main effect: *confidence_level*	3.77% due to main effect: N^{iters}
	3.17% due to main effect: μ

Table 9: Single and pairwise interaction effects obtained from fANOVA analysis on all meta-tuning experiments with surrogate benchmarks. The analysis is on the irace parameter configuration space where the average performance is not worse than irace's default configuration's. For the sake of brevity, only effects with a contribution value on the percentage of variance explained not less than 3% are listed. The analysis is on the irace configuration space where the average performance is not worse than irace default configuration.



(a) N^{iters}

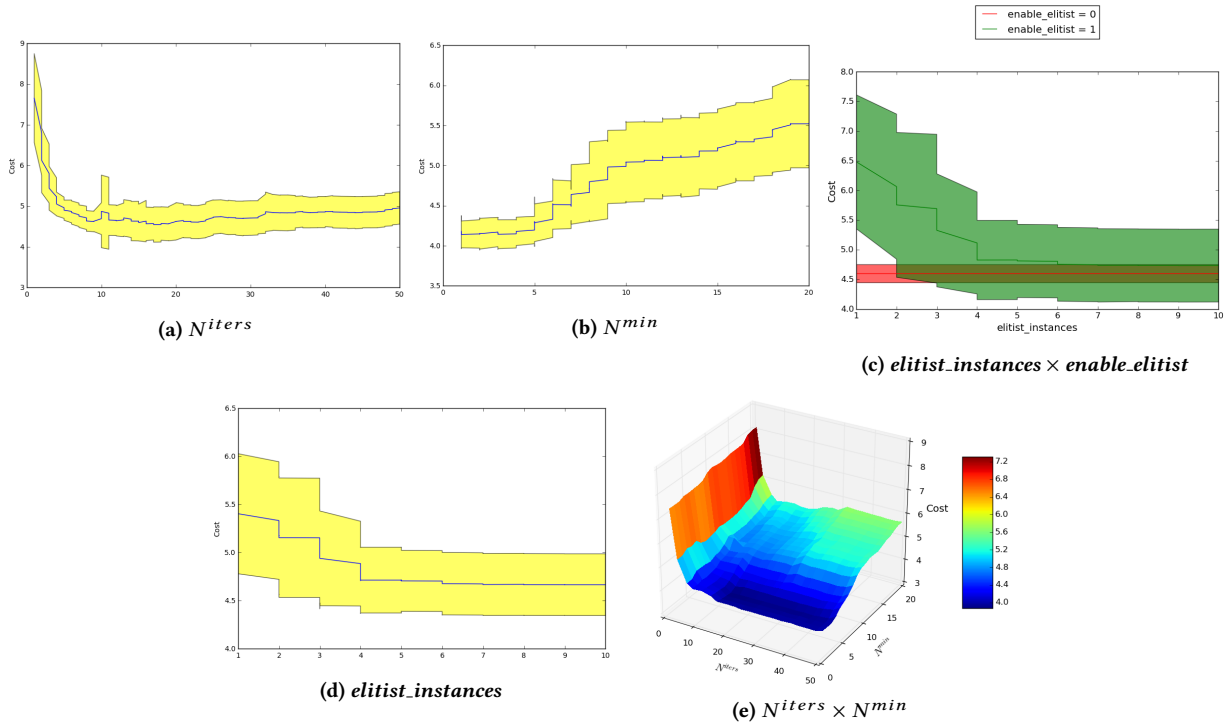
(b) $N^{iters} \times$ *confidence_level*

(c) *confidence_level*

(d) $N^{iters} \times T^{first}$

(e) T^{first}

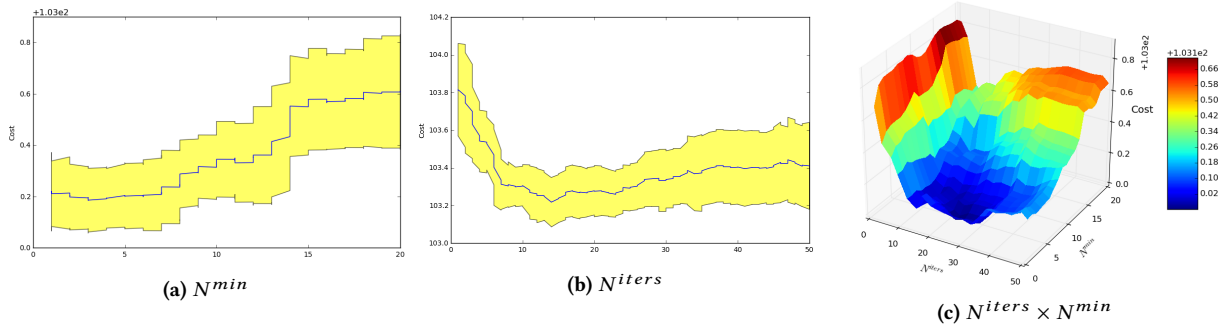
Figure 6: Marginal plots of single effects and pairwise interaction effects of the fANOVA analysis on the meta-tuning with meta-irace on ACOTSP. Only effects with a contribution value on the percentage of variance explained not less than 3% are shown. The analysis is on the irace configuration space where the average performance is not worse than irace default configuration.



(a) N^{iters} (b) N^{min} (c) *elitist_instances* × *enable_elitist*

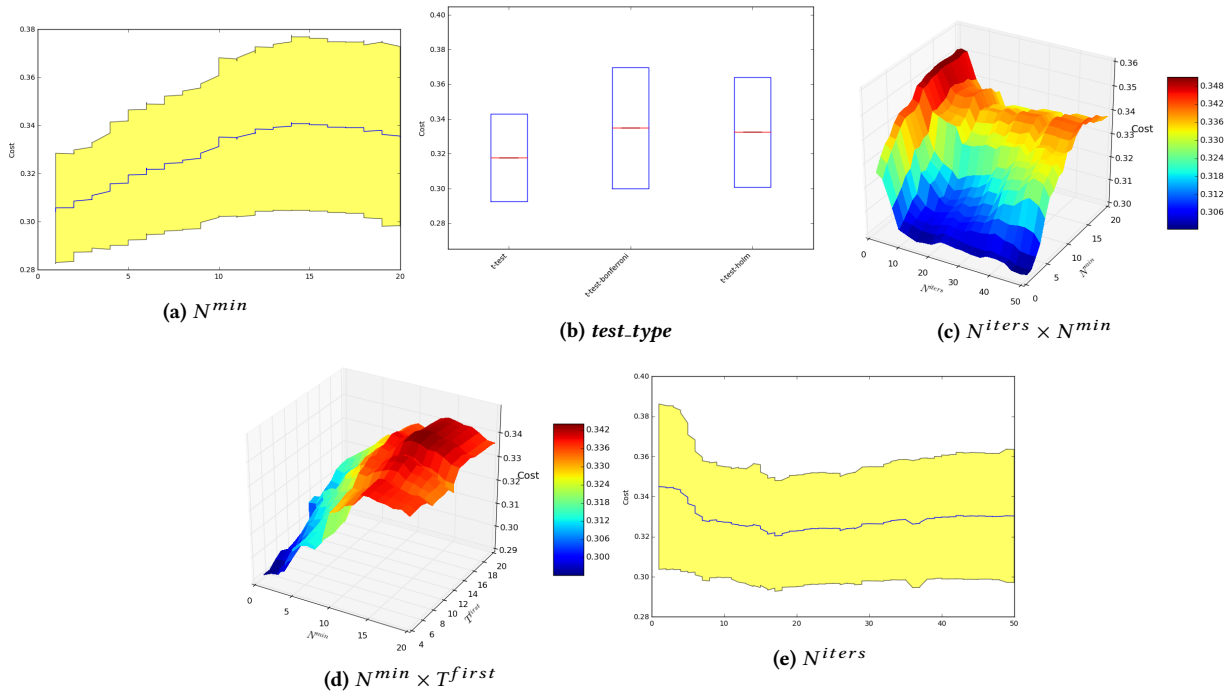
(d) *elitist_instances* (e) $N^{iters} \times N^{min}$

Figure 7: Marginal plots of single effects and pairwise interaction effects of the fANOVA analysis on the meta-tuning with meta-irace on SPEAR-SWV. Only effects with a contribution value on the percentage of variance explained not less than 3% are shown. The analysis is on the irace configuration space where the average performance is not worse than irace default configuration.



(a) N^{min} (b) N^{iters} (c) $N^{iters} \times N^{min}$

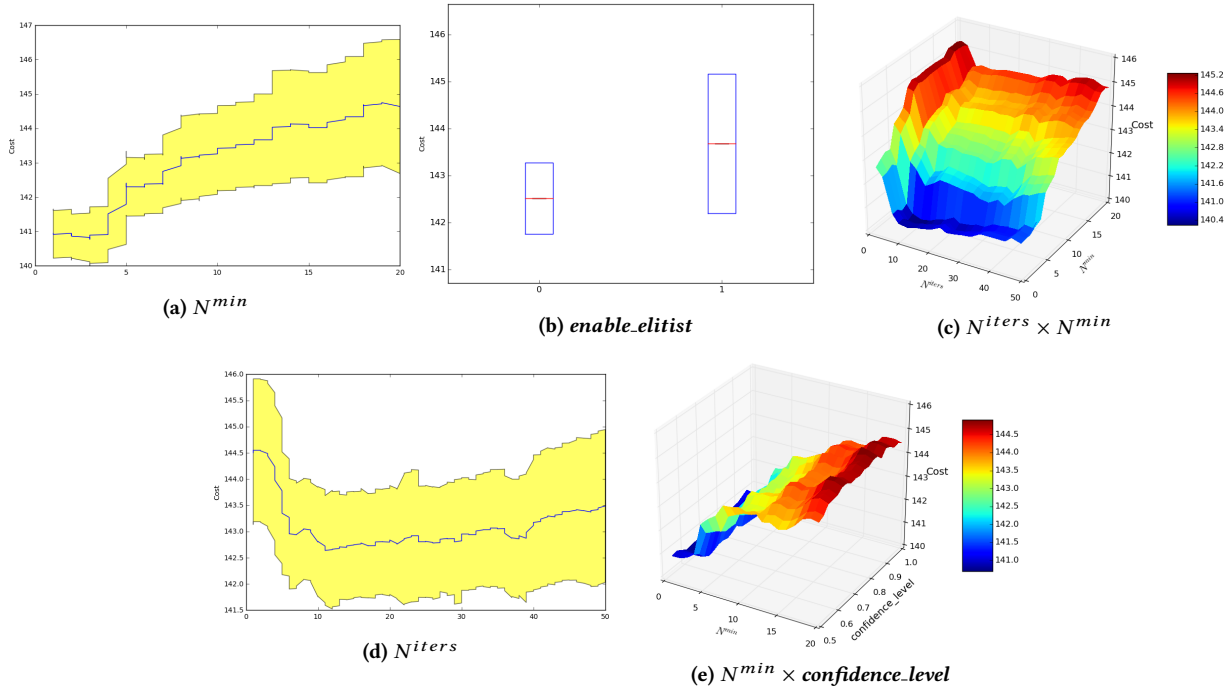
Figure 8: Marginal plots of single effects and pairwise interaction effects of the fANOVA analysis on the meta-tuning with meta-irace on SPEAR-IBM. Only effects with a contribution value on the percentage of variance explained not less than 3% are shown. The analysis is on the irace configuration space where the average performance is not worse than irace default configuration.



(a) N^{min} (b) *test_type* (c) $N^{iters} \times N^{min}$

(d) $N^{min} \times T^{first}$ (e) N^{iters}

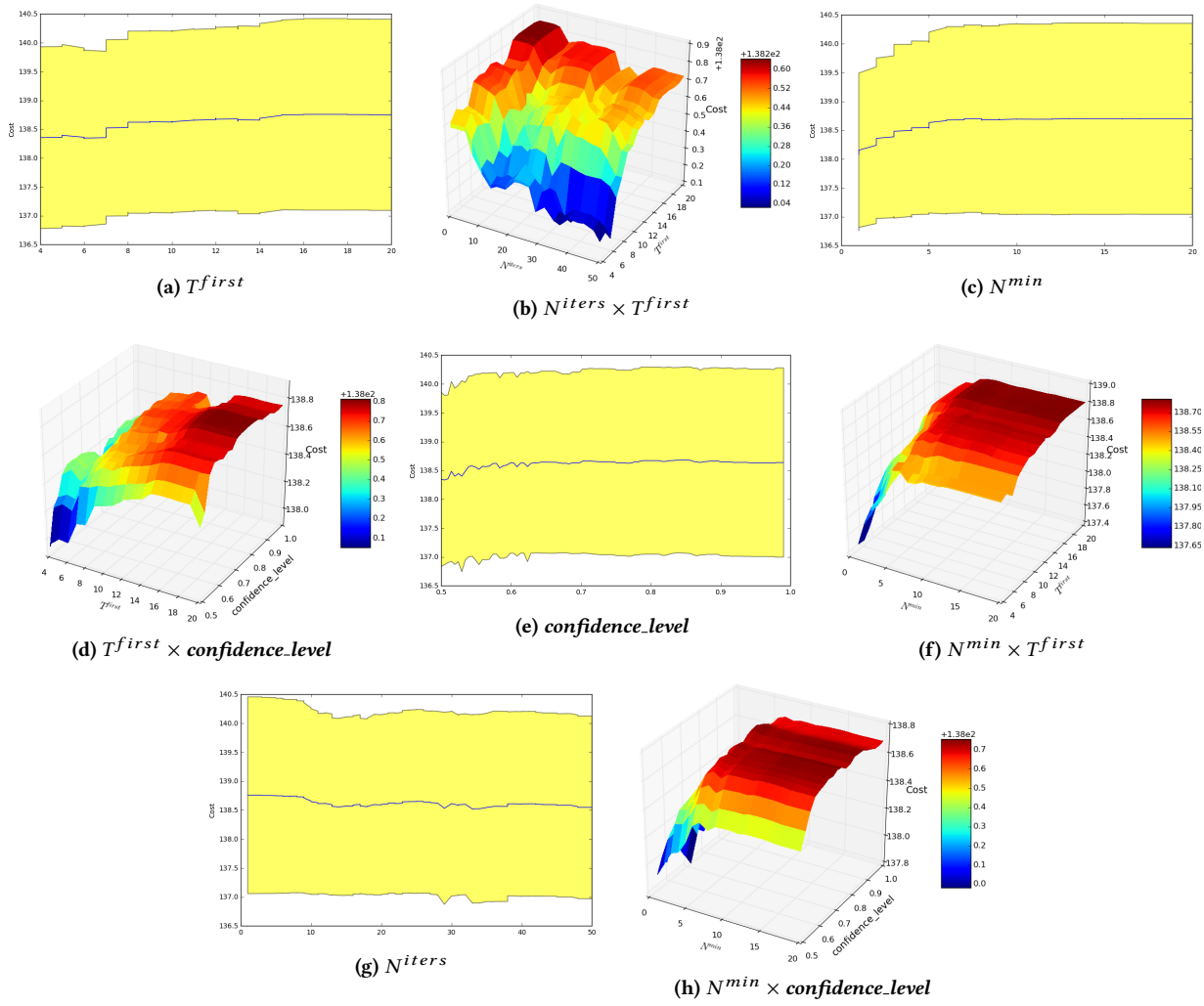
Figure 9: Marginal plots of single effects and pairwise interaction effects of the fANOVA analysis on the meta-tuning with meta-irace on CPLEX-all. Only effects with a contribution value on the percentage of variance explained not less than 3% are shown. The analysis is on the irace configuration space where the average performance is not worse than irace default configuration.



(a) N^{min} (b) *enable_elitist* (c) $N^{iters} \times N^{min}$

(d) N^{iters} (e) $N^{min} \times$ *confidence_level*

Figure 10: Marginal plots of single effects and pairwise interaction effects of the fANOVA analysis on the meta-tuning with meta-irace on CPLEX-BIGMIX. Only effects with a contribution value on the percentage of variance explained not less than 3% are shown. The analysis is on the irace configuration space where the average performance is not worse than irace default configuration.



(a) T^{first}

(b) $N^{iters} \times T^{first}$

(c) N^{min}

(d) $T^{first} \times$ ***confidence_level***

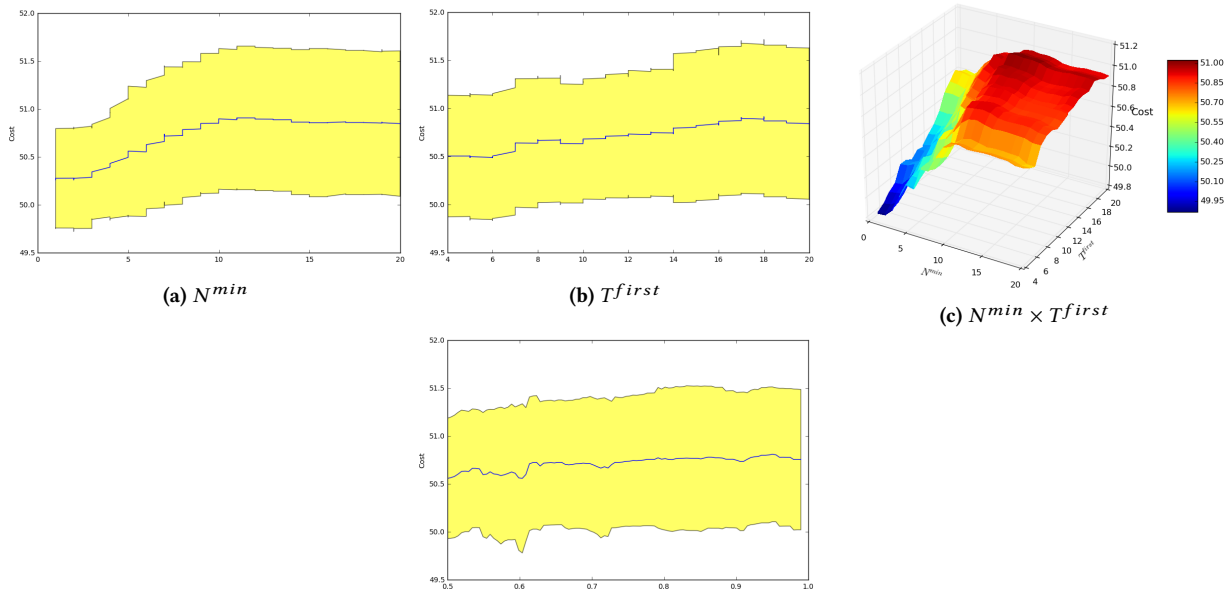
(e) ***confidence_level***

(f) $N^{min} \times T^{first}$

(g) N^{iters}

(h) $N^{min} \times$ ***confidence_level***

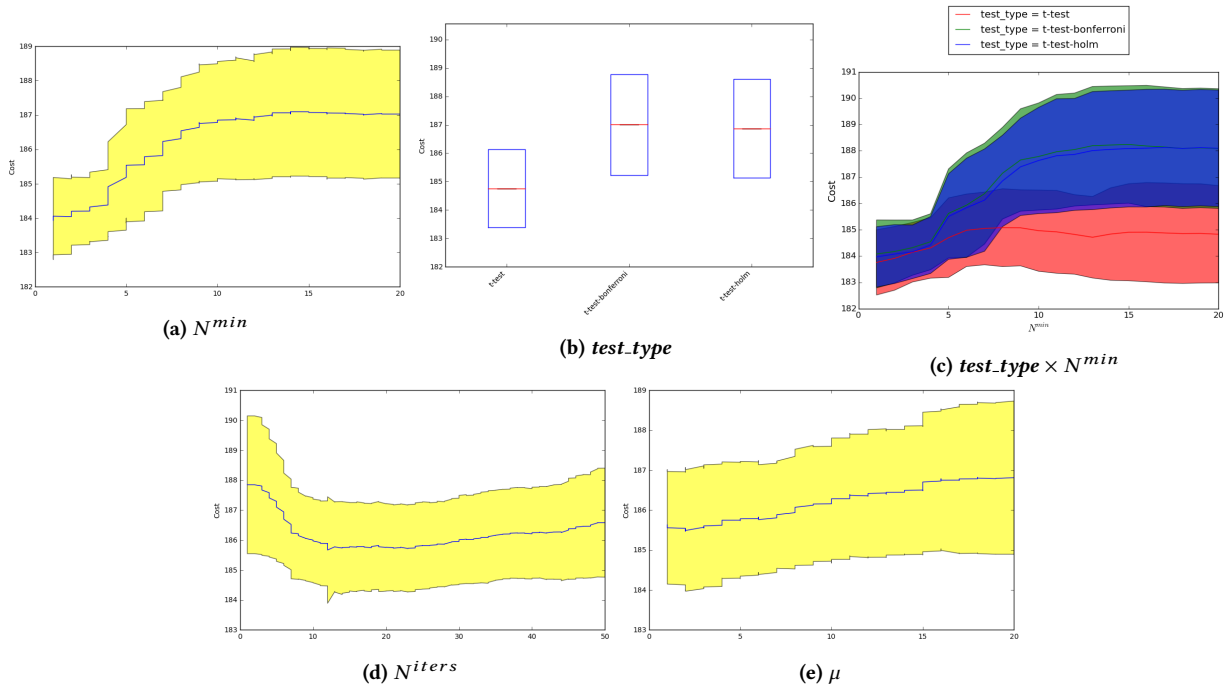
Figure 11: Marginal plots of single effects and pairwise interaction effects of the fANOVA analysis on the meta-tuning with meta-irace on CPLEX-REG. Only effects with a contribution value on the percentage of variance explained not less than 3% are shown. The analysis is on the irace configuration space where the average performance is not worse than irace default configuration.



(a) N^{min} **(b)** T^{first} **(c)** $N^{min} \times T^{first}$

(d) ***confidence_level***

Figure 12: Marginal plots of single effects and pairwise interaction effects of the fANOVA analysis on the meta-tuning with meta-irace on CPLEX-CORLAT. Only effects with a contribution value on the percentage of variance explained not less than 3% are shown. The analysis is on the irace configuration space where the average performance is not worse than irace default configuration.



(a) N^{min} (b) *test_type* (c) *test_type* $\times N^{min}$

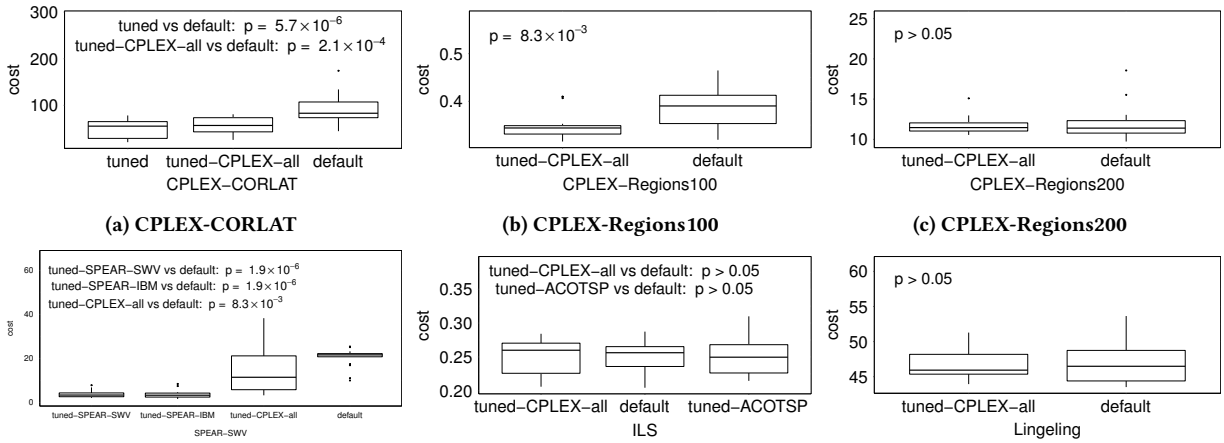
(d) N^{iters} (e) μ

Figure 13: Marginal plots of single effects and pairwise interaction effects of the fANOVA analysis on the meta-tuning with meta-irace on CPLEX-RCW. Only effects with a contribution value on the percentage of variance explained not less than 3% are shown. The analysis is on the irace configuration space where the average performance is not worse than irace default configuration.

A.4 Results on real benchmarks

Algorithm	Parameters	Instance sets	#Instances	Metric
CPLEX	76 parameters (int: 6, real: 4, categorical: 66, conditional: 4)	Regions100 Regions200 CORLAT	1000 1000 1000	Running time
SPEAR	26 parameters (int: 4, real: 12, categorical: 10, condition: 9)	SWV	604	Running time
ACOTSP	11 parameters (int: 4, real: 4, categorical: 3, conditional: 5)	Travelling Salesman Problem	50	Solution quality
ILS	43 parameters (int: 14, real: 8, categorical: 21, conditional: 40)	Permutation flowshop	30	Solution quality
Lingeling	137 parameters (int: 0, real: 0, categorical: 137, conditional: 0)	SAT	299	Running time

Table 10: All the real benchmarks used in our experiments for studying the best irace's configurations obtained from a number of meta-irace experiments with the surrogate benchmarks. In particular, we tested the best irace's configurations returned by meta-irace on the single surrogate model of CPLEX-CORLAT, SPEAR-IBM, SPEAR-SWV, ACOTSP and on the multiple surrogate models of CPLEX



(a) CPLEX-CORLAT **(b) CPLEX-Regions100** **(c) CPLEX-Regions200**

(d) SPEAR-SWV **(e) ILS** **(f) Lingeling**

Figure 14: Test performance of irace using the default configuration and the best configurations obtained by meta-irace with the single surrogate model of CPLEX-CORLAT, SPEAR-IBM, SPEAR-SWV, ACOTSP and on the multiple surrogate models of CPLEX. The test performance is measured on the real configuration benchmarks described in Table 10