A combined approach for analysing heuristic algorithms

DOI: 10.1007/s10732-018-9388-7
Handle: http://hdl.handle.net/1942/26682

This is a post-peer-review, pre-copyedit version of an article published in Journal of Heuristics. The final authenticated version is available online at: http://dx.doi.org/10.1007/s10732-018-9388-7

A Combined Approach for Analysing Heuristic Algorithms

Jeroen Corstjens[1], Nguyen Thi Thanh Dang[2], Benoît Depaire[1], An Caris[1], and Patrick De Causmaecker[2]

[1]UHasselt, Research Group Logistics, Agoralaan, 3590 Diepenbeek, Belgium
[1]jeroen.corstjens@uhasselt.be
[2]KU Leuven, CODeS, imec-ITEC, Etienne Sabbelaan 53 (7659), 8500 Kortrijk, Belgium

August 11, 2018

Abstract

When developing optimisation algorithms, the focus often lies on obtaining an algorithm that is able to outperform other existing algorithms for some performance measure. It is not common practice to question the reasons for possible performance differences observed. These types of questions relate to evaluating the impact of the various heuristic parameters and often remain unanswered. In this paper, the focus is on gaining insight in the behaviour of a heuristic algorithm by investigating how the various elements operating within the algorithm correlate with performance, obtaining indications of which combinations work well and which do not, and how all these effects are influenced by the specific problem instance the algorithm is solving. We consider two approaches for analysing algorithm parameters and components — functional analysis of variance and multilevel regression analysis — and study the benefits of using both approaches jointly. We present the results of a combined methodology that is able to provide more insights than when the two approaches are used separately. The illustrative case studies in this paper analyse a Large Neighbourhood Search algorithm applied to the Vehicle Routing Problem with Time Windows and an Iterated Local Search algorithm for the Unrelated Parallel Machine Scheduling Problem with Sequence-dependent Setup Times.

Keywords: Functional Analysis of Variance fANOVA Multilevel Regression Algorithm Performance Vehicle Routing Problem with Time Windows Large Neighbourhood Search Iterated Local Search Unrelated Parallel Machine Scheduling Problem

1 Introduction

Experimentation on heuristic algorithms commonly entails the computational testing of an algorithm on some benchmark problem set and comparing results against those of known algorithms. The objective is to be better than

the competing algorithms for some performance measure (e.g., solution quality or running time). Such an approach for evaluating algorithms does not explain, however, 'why' one method achieves better results than other ones [17]. Which algorithm elements (i.e., parameter or components) contribute more or less to a good performing algorithm? Are there specific combinations of elements that work well or not? Or is the observed superior performance due to a more efficient implementation of the algorithm? These kind of questions often remain unanswered when following a competitive evaluation methodology. Nevertheless, such insights are necessary to truly understand algorithm behaviour [20, 34].

The choice of a suitable configuration for a heuristic algorithm is a task that in the past often entailed a trial-and-error approach and relied on researchers' experience [10]. Automated algorithm configurators, such as ParamILS [21], irace [26], Sequential Model-based Algorithm Configuration (SMAC) [18] and GGA [1], replace the manual effort with the computational power of machines and provide (a set of) well-performing parameter setting(s)[1]. These methods have proven to result in better-performing heuristic algorithms [32, 35]. However, they often do not provide any further information about, for example, which algorithm elements contribute most to performance. Recently, methodologies have been proposed for evaluating heuristic algorithms that go a step further and aim for a more profound understanding of the heuristic elements' impact on performance [2, 7, 9, 13, 19, 20, 30]. Similarly, the influence of problem instance characteristics has been investigated to identify the strengths and weaknesses of heuristic algorithms across large and diverse sets of problem instances [38]. Instance characteristics are employed in empirical hardness models [25] to predict how much time a given algorithm (e.g., CPLEX) requires to solve a given problem instance. In this research, we consider two model-based methodologies [9, 20] and investigate for a particular application scenario whether both approaches deliver consistent insights. More importantly, we look into possible opportunities for a complementary use of both methodologies. Our focus therefore lies on answering the following questions. How can both methodologies be formulated in a combined methodology and what is the added value of jointly applying both approaches? And are the insights obtained from the analysis results of both approaches consistent or are there any differences observed?

The investigated methodologies are functional analysis of variance (fANOVA) [20] and multilevel regression [9]. fANOVA quantifies the relative importance of the algorithm elements and their interactions according to the amount of variance in the performance data they explain, giving an indication of which effects are most important to performance. The multilevel regression methodology explicitly separates the performance impact of algorithm elements and problem instances. It is focused on quantifying the relationship between algorithm elements and performance and how this relationship is moderated by the characteristics of a specific problem instance.

Each methodology has its own advantages. fANOVA does not rely on statistical assumptions, such as normality and homoscedasticity, that are required

[1]We interpret a parameter setting as a set of values and included operators.

by multilevel regression models. It is also computationally cheaper than the multilevel regression model approach for our given case study. The multilevel regression model, on the other hand, offers a more detailed analysis of the algorithm, since it can investigate algorithm element effects for a specific setting of the other elements and a specific problem instance. Moreover, the interpretation of effects in fANOVA is carried out through variance percentages and visual inspection of plots. The latter might be difficult for interaction effects. The interpretation of the regression analysis is based on quantified effects, and plots are merely used to support and visualise interpretation. The regression analysis also provides a statistical significance test to indicate whether there really is a link between the values chosen for a specific algorithm parameter and the obtained performance, or whether any observed relationships are likely due to chance.

In this paper, we combine both methodologies by relying on the importance analysis provided by fANOVA to formulate a proper regression model for the multilevel methodology. This prevents an overly complex regression model with many variable (interactions) that contribute little to performance. This regression model can then be used for a more detailed analysis of the important effects and for confirmatory analyses with hypotheses testing. Both approaches and their combination are assessed by means of two case studies. The first one tests a number of configurations for a Large Neighbourhood Search (LNS) algorithm on a number of problem instances for the Vehicle Routing Problem with Time Windows (VRPTW). The second considers an Iterated Local Search (ILS) method for the unrelated parallel machine scheduling problem with sequence-dependent setup times (UPMSP).

The paper is organized as follows. The methodologies of fANOVA and multilevel regression analysis are introduced in section 2, along with the proposed combination of the two approaches. The two case studies and analysis results are presented in section 3 and 4. Finally, conclusions and future work are given in section 5.

2 fANOVA and Multilevel Regression Models

2.1 fANOVA

The *fANOVA* method proposed in [20] is an approach for analysing the importance of algorithm elements on performance using a random forest prediction model and the functional analysis of variance [16]. In this paper, we use the implementation provided at `https://github.com/frank-hutter/fanova` [2]. The approach studies the contribution of every single element and every element interaction on the performance of the algorithm. In particular, given a data set of performance values of different algorithm parameter settings on a number of problem instances, fANOVA first builds a random forest-based prediction model

[2] A newer implementation is recently introduced at `https://github.com/automl/fanova`. The two versions give similar analysis results. The reason why we use this old version is because it runs much faster in our experience, probably due to the different underlying choices of programming languages used in each version.

to predict the average performance of every algorithm parameter setting over the entire problem instance space. Afterwards, functional analysis of variance [16] is applied on the prediction model to decompose the overall algorithm performance variance into additive components, each one corresponds to a subset of the algorithm elements. The ratio between the variance associated with each component and the overall performance variance is used as an indicator of the importance of the corresponding algorithm parameter subset. For example, the following shows a part of the output of a fANOVA on the Large Neighbourhood Search algorithm used in our study in Section 3 for a Vehicle Routing Problem with Time Windows problem instance set:

```
Sum of fractions for main effects 75.10%
Sum of fractions for pairwise interaction effects 6.26%
72.62% due to main effect: repair
1.72% due to main effect: destroy
1.60% due to interaction: repair x destroy
```

The interpretation of the first two lines is that 75.10% of the algorithm performance variance can be explained by single elements, and 6.26% by the interactions of every pair of algorithm elements. The remaining 18.64% is explained by higher-order ($\geq$3) interactions and error inherent in the model. The third line indicates that the *repair* component is the most important element, as the component itself can explain a huge part (72.62%) of the overall algorithm performance variance. The other component *destroy* and their pairwise interaction *repair* and *destroy* are less important. The details of the algorithm and those elements will be further described in section 3.

In addition to the value indicating the importance of each algorithm element subset, fANOVA also provides some insights on which regions are good and bad (with a degree of uncertainty) for each element inside the subset through a marginal plot. Given a specific value for each algorithm element in the subset, the corresponding marginal prediction value is the average performance value of the algorithm over the entire configuration space associated with all elements not belonging to the subset. A marginal plot shows the mean and the variance of the marginal prediction values given by the random forest's individual trees. Figure 1 shows the marginal plot of *repair*. This categorical element has a domain of three values: *Greedy*, *Regret2* and *GreedyRegret2*, each of which is associated with a boxplot in Figure 1. The boxplot for *Greedy*, for example, shows the mean and the variance of the average algorithm performance values over the entire problem instance space of all algorithm parameter settings having *repair* = *Greedy*. The plot implies that *Regret2* is the best choice for the element *repair*.

Although fANOVA generally focuses on analysing the importance of algorithm elements, it can also be used to study the interaction between elements and problem instance features — as we do in our case study — by adding those features into the prediction model of fANOVA. The features are treated exactly in the same way as the algorithm elements. It must be noted that such an integration can only be valid if the features are independent. In other words, instance features might be correlated, but their values in the problem instance set under study must be arbitrarily chosen. In our case study, instances are gen-

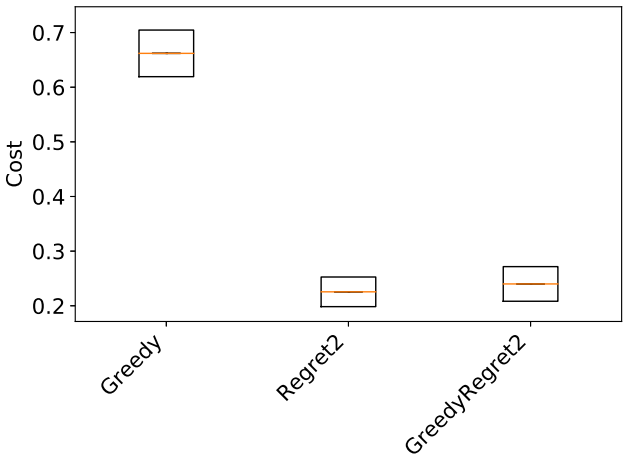


Figure 1: fANOVA's marginal plot for the main effect of parameter *repair*.

erated by randomly sampling values from all features' domains, which satisfies this independency requirement.

2.2 Multilevel Regression

The multilevel methodology proposed by Corstjens et al. [9] is an approach for analysing the relationships between algorithm elements, problem instance characteristics and algorithm performance using a multilevel regression (MLR) model. The approach studies how the performance measure will change when modifying an element from one value to another and how this change is influenced by the problem instance the algorithm is solving. Unlike fANOVA, regression models do not focus on the variance in the data that is explained by each of the investigated variables, but on estimating the variable coefficients [28]. More precisely, regression is applied to describe the relationship between a response variable and variables that explain the response, the explanatory variables. This relationship is formulated mathematically in a regression model which describes how the response value will change when an explanatory variable changes with some value [29]. For every calculated coefficient estimate a confidence interval is provided to indicate whether the estimated performance change by some parameter value is significantly different from zero or not. If not, the observed impact is likely due to chance.

Furthermore, the multilevel aspect of the methodology enables to efficiently study how effects vary by group by relying on multilevel models[11]. Within the context of heuristic experimentation it is possible to identify a multilevel structure when experimenting with different parameter settings on a single problem instance of some combinatorial optimisation problem. Any observed differences in performance can then be attributed to the algorithm elements and not the problem instance. To expose the influence of the problem instance, such a multilevel structure is then determined for multiple problem instances. It can then be analysed how the performance impact of the various elements changes when

considering different problem instances. It enables answering questions like "Does a heuristic operator work equally well on instances with a few customers and instances with many customers? When service time is long on average, is it better to lower the cooling rate within simulated annealing or increase it? ..." [9].

Given a data set of performance values of different algorithm parameter settings on different problem instances, a multilevel regression model is formulated to predict the average expected performance for a particular parameter setting and problem instance. Note that the interpretation of predictions between fANOVA and MLR differs. Evaluating a specific parameter by fixing it at different values, fANOVA predicts an average performance for every value taking into account all values of all other algorithm parameters over the entire instance space. MLR, on the other hand, predicts for each value the average performance given a fixed setting of the other parameters and considering a specific problem instance. The regression model therefore enables a more detailed analysis of parameter effects.

The multilevel regression model is an extension of a classical regression model in which the coefficients have their own probability model, being the second, higher level. It has accompanying parameters which are the predictors at this second level [14]. A possible formulation of a multilevel regression model — using the same example as in section 2.1 — is given in equations (1) to (3) where three algorithm parameters are considered (equation (1)) and three problem instance characteristics operating at the problem, i.e., group, level (equations(2) and (3)).

$$Y_i = \alpha_{j[i]} + \beta_{1j[i]} Greedy_i + \beta_{2j[i]} Regret2_i + \beta_{3j[i]} Random_i + \epsilon_i \quad (1)$$

$$\alpha_j = \mu_0^{\alpha} + \mu_1^{\alpha} customers_j + \mu_2^{\alpha} demand_j + \mu_3^{\alpha} servicetime_j + \eta_j^{\alpha} \quad (2)$$

$$\beta_{kj} = \mu_0^{\beta_k} + \mu_1^{\beta_k} customers_j + \mu_2^{\beta_k} demand_j + \mu_3^{\beta_k} servicetime_j + \eta_j^{\beta_k} \quad (3)$$

where

$i \in I$	scenario, a combination of a specific problem instance with a specific parameter setting
$j \in J$	problem instance
$j[i]$	index variable to code problem instance membership ($j[i] = j$), e.g., $j[90] = 5$ means the 90th scenario solves problem instance 5
Y_i	objective function value of scenario i
$Greedy_i$	algorithm element in scenario i
$customers_j$	problem instance characteristic in problem instance j
α_j	varying regression intercept, representing the solution quality given problem instance j when all algorithm elements are set equal to 0
β_{kj}	represents the varying effect of algorithm parameter k on Y given scenario i and problem instance j

μ_0 represents a mean problem effect

$\mu_{1,2,3}$ represent the effect of the problem-level predictors on the varying algorithm element effects

η_j error at the problem instance level and is assumed to be $\sim N(0,\sigma^2)$

ϵ_i error at the parameter setting level and is assumed to be $\sim N(0,\sigma_e^2)$

Equation (1) represents the lowest level where the objective function value for scenario i is observed, with a scenario defined as the combination of a specific problem instance with a specific parameter setting. The objective function value observed is hypothesised to depend on the values that are set for the various heuristic parameters, as indicated by the variables $Greedy_i$, $Regret2_i$ and $Random_i$, and with the β coefficients representing the performance impact of each parameter. Equations (2) and (3) represent the problem level and define how the effects in equation (1) (i.e., the β's) are moderated by the problem instance characteristics, represented by the variables $customers_j$, $demand_j$ and $servicetime_j$. With this model we can learn the impact a single algorithm parameter has on performance and how it is influenced by the problem instance characteristics.

In the next subsection we explore the complementary use of fANOVA and multilevel regression.

2.3 A Combined Methodology

We propose a methodology that combines the use of fANOVA and multilevel regression analysis. The idea is to use the ranking of effects fANOVA provides to formulate a multilevel regression model that includes only the most important effects. The search for a suitable regression model that includes all relevant variables can often be a cumbersome task [14]. The more variables and variable interactions are included, the more complex the model becomes and the more arduous it is to interpret the estimated effects. The challenge thus lies in deciding which explanatory variables and interactions to include in the model. fANOVA can provide assistance with this issue. Since the analysis gives a ranking on the importance of effects, a regression model could then be formulated based on this ranking in order to prevent an overly complex model with many variables. The regression analysis can then more easily focus on these important effects. A regression model that is less complex in terms of fewer variables and variable interactions included also implies time savings when fitting the model to the data.

The regression analysis facilitates a more detailed analysis since it provides effect estimates for a particular parameter setting and problem instance, while the fANOVA analysis estimates marginal performance for a particular parameter value by averaging over all other parameters and problem instance characteristics. Furthermore, the multilevel regression adds contribution on the importance analysis by calculating confidence intervals for each of the effects. This indicates which effects are actually statistically significant and which are likely due to chance. Finally, since regression models assume specific functional forms — linear or nonlinear (exponential, logarithmic, ...) —, it has the ability

to extrapolate results, unlike the random forest prediction model fANOVA uses which has an upper and lower bound for predicted values. Nonetheless, caution is advised when extrapolating results to regions in the configuration or problem instance space where there is no data to support them as the assumed trend does not necessarily hold for out-of-sample ranges [14].

In the next section, a case study is performed that explores how parameters relate to performance, and how problem instance characteristics possibly influence performance impact of parameters. We compare the findings of both fANOVA and multilevel regression analysis and aim to show their consistency. This will lead to the conclusion that a fANOVA is suited to serve as an exploratory analysis, exposing correlations between input factors (parameters and problem instance characteristics) and performance. However, the next step is looking to explain the patterns that are observed in consecutive experiments with hypothesis testing and theory formulation. The latter entails a confirmatory analysis that cannot be performed with fANOVA, but for which we rely on parametric regression models [22].

3 Case study 1: a large neighbourhood search algorithm for the vehicle routing problem with time windows

3.1 Description

In a first case study, we solve instances of the vehicle routing problem with time windows using a large neighbourhood search (LNS) algorithm. Our implementation of this algorithm is a simplification of the well-known Adaptive Large Neighbourhood Search (ALNS) metaheuristic developed by Pisinger and Ropke [33] that is able to solve multiple variants of the vehicle routing problem, among which the VRPTW. The algorithm iteratively destroys and repairs the current solution, each time randomly selecting a destroy and repair operator from a set of operators. The more an operator has contributed to finding a better solution, the greater the probability it will be chosen in future iterations. This process is repeated until some stopping criterion is met. This algorithm was chosen because of its popularity and effectives in VRP literature. The multitude of parameters it contains also makes it a suitable research subject for parameter analysis. However, a simplification is performed to reduce the number of parameters to investigate as it is usually better to start small when planning experiments [24]. Furthermore, we currently focus on establishing the methodology to evaluate heuristic methods rather than concentrating our attention on the many parameters ALNS has. Therefore, a less elaborate version of the heuristic method is preferred. The LNS algorithm in this paper does not adjust the probabilities for selecting repair and destroy operators every iteration based on their performance, but keeps them fixed and equal throughout the search process. We also consider fewer operators, more specifically three destroy operators — random, worst and related removal — and two repair operators — greedy and regret-2. The pseudocode is provided in Algorithm 1.

The LNS algorithm is run on a data set consisting of 4000 different combinations of problem instances and parameter settings[3]. Benchmarks, such as the Solomon [39] instances, were not chosen due to concerns of overfitting and lack of information on the probability distributions for the problem instance characteristics. This implies that any results found are not necessarily generalisable to instances not observed in the sample [4]. Hence, a heuristic algorithm performing well on a benchmark problem set might not perform well on other instances [3]. The data set was generated according to a two-phase sampling scheme as applied in [9]. First, 200 instances for the vehicle routing problem with time windows (VRPTW) were generated by drawing random values for the problem instance characteristics listed in Table 1. For each of the generated problem instances 20 parameter setting variants were defined again by drawing random values for each of the algorithm parameters listed in Table 2. The algorithm was run for each of the 4000 scenarios and returned a total cost measure indicating the total distance travelled by all vehicles to provide service to all customers in the problem.

Two independent data sets of 4000 observations were created. One for fANOVA and one for the multilevel regression analysis. The motivation is to prevent overfitting analysis findings to a single data set. Searching for a model that is the best fit for a single data set might risk fitting noise in the data — patterns present in the sample but not in the population — and might result in a model which performs poorly on other data points from the same population. A fitted model should be able to make accurate predictions for new data points instead of only the data points used to learn the model [12]. Furthermore, a second data set also allows to gain more confidence on the effects that appear relevant in the first data set and to detect possible false positives. The latter implies that a variable (interaction) might show to have a contribution to performance in the sample data, while it does not in the population. Using a second sample reduces the risk of having false positives [37]. For these reasons, the multilevel regression analysis is performed using new sample data.

3.2 Analysis results

First, fANOVA is applied on the given algorithm performance data set. Then, a multilevel regression model is formulated based on the importance analysis provided by fANOVA. As will be discussed, the conclusions of both approaches are consistent. However, not all effects taken from fANOVA are statistically significant according to the multilevel regression model.

[3]Increasing sample size will increase precision of the estimates, meaning their confidence intervals become narrower. Effects that are already significant will only become more significant. Whether or not an increased sample size will contribute much to the analysis is difficult to judge. As sample size increases, even the smallest effects become significant, but that does not make them important [41]. In our case, a larger sample size did not alter analysis conclusions other than adding more precision. It did require substantially more time to fit the regression models, so we assessed the current sample size of 4000 scenarios to sufficiently represent the major variations in performance and to be practical in terms of time to fit the regression model.

Algorithm 1 Large Neighbourhood Search

Input: Problem instance j, Parameter setting θ
Output: Best found solution x^{best}
Initialization: initial solution x constructed by regret-2 heuristic
Stage 1: Vehicle Minimisation
repeat
  Remove one route from x
  Schedule removed customers into remaining routes (as in Stage 2)
until 20% of maximum run time met
Stage 2: Minimisation of total distance covered
repeat
  select destroy and repair methods $d \in \Omega^-$ and $r \in \Omega^+$ at random
  $x^{candidate} = r(d(x))$
  if $x^{candidate}$ is accepted **then** $x = x^{candidate}$ **end if**
  if $c(x^{temporary}) < c(x^{best})$) **then** $x^{best} = x^{candidate}$ **end if**
until maximum run time met

Table 1: Characteristics of the VRPTW instances

Characteristic	**Type**	**Value ranges**
Customers	Integer	U[25, 400]
Demand per customer	Integer	U[10,50]
Service time per customer	Integer	TRIA(min,max) min∼U[10,30] max∼U[30,50]
Time window width per customer	Integer	TRIA[min,max] min∼U[20,50] max∼U[50,80]
maximum running time	Integer	TRIA(60,1800)

Only the characteristics used in the analysis are listed. A full list of all characteristics is given in Table 7 in the Appendix.

Table 2: Parameters of the Large Neighbourhood Search algorithm

Parameter	**Type**	**Value ranges**
random seed	Integer	U[1, 1000000]
determinism parameter	Integer	U[1, 100]
noise parameter	Continuous	U[0, 1]
cooling rate	Continuous	U[0.01,0.99]
start temperature control parameter	Continuous	U[0.01,1]
destroy operators	Categorical	Random, Worst, Related, RandomWorst, RandomRelated, WorstRelated, RandomWorstRelated
repair operators	Categorical	Greedy, Regret2, GreedyRegret2

3.2.1 fANOVA analysis results

First, the cost is normalised on an instance-basis since the range of the cost values returned by the algorithm can vary among different instances.

$$p_j^c = \frac{(f_j^c - \min_{c' \in C_j}, f_j^{c'})}{(\max_{c' \in C_j}, f_j^{c'} - \min_{c' \in C_j}, f_j^{c'})} \tag{4}$$

where f_j^c and p_j^c are the original and normalised cost values of parameter setting c on problem instance j, and C_j is the set of all parameter settings that have been run on instance j.

The output generated by fANOVA, for the entire problem instance set, is as

follows.

```
Sum of fractions for main effects 60.56%
Sum of fractions for pairwise interaction effects 16.54%
53.03% due to main effect: repair
4.73% due to interaction: repair x customers
4.36% due to interaction: repair x destroy
3.52% due to main effect: customers
3.05% due to interaction: destroy
2.96% due to interaction: destroy x customers
... (remaining effects: < 1%)
```

Since we want to focus on important effects, only the ones with contribution percentage values higher than 1% are listed. At this threshold 71.65% (of the 77.4% covered by fANOVA) of the variance in performance is explained. A lower cut-off value would add little and only increase regression model complexity. For example, a cut-off point at 0.5% only increases the overall explained variance to 72.19%. On the other hand, a cut-off at, say 5%, would only explain about 50% of the performance variance and result in an uninformative regression model that only included the repair operators.

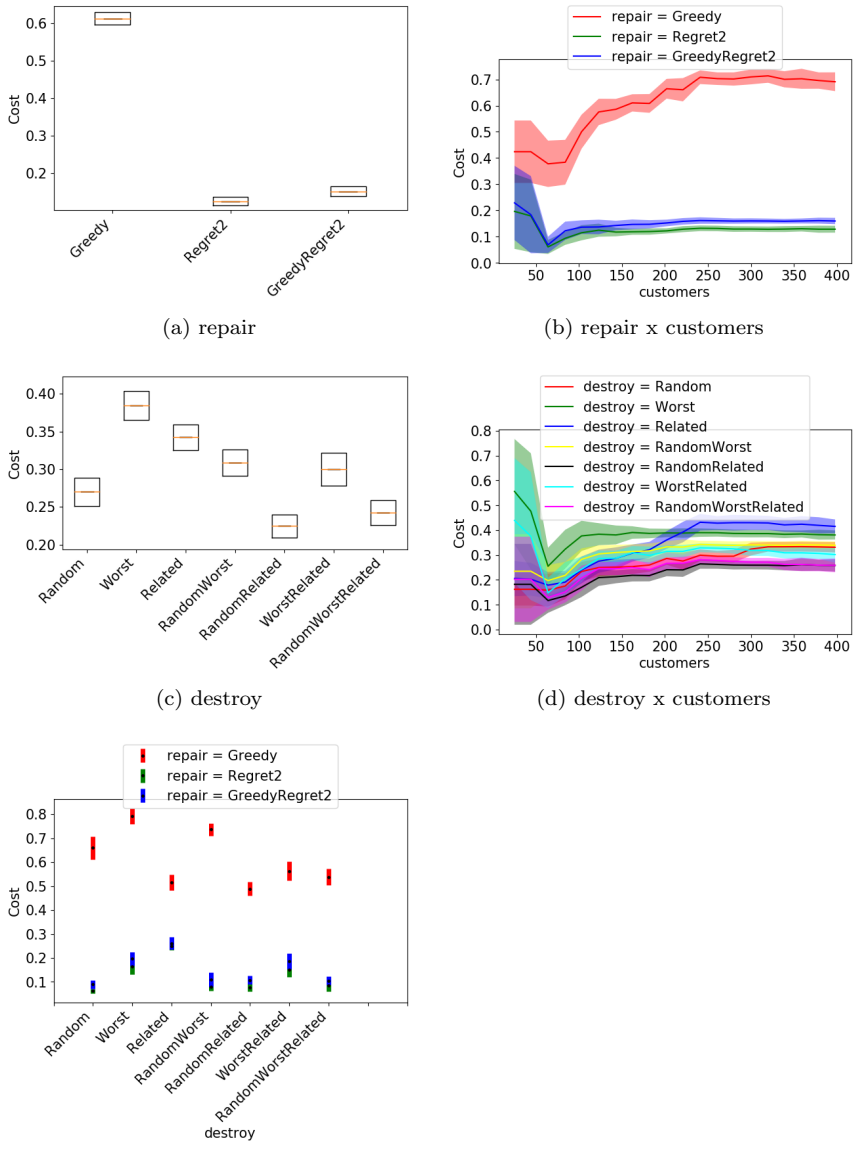
The marginal plot for each effect is given in Figure 2. Since we are only interested in the algorithm parameters and their interactions with the problem instance features, the main effect *customers* is omitted.

The single parameter *repair* explains a huge part (53.03%) of the total algorithm performance variance, indicating that this parameter plays the most important role in the performance of the algorithm. Figure 2a shows that *Greedy* is clearly the worst choice for the parameter *repair*. The algorithm achieves the best overall performance with *Regret2* as the only repair operator, although *GreedyRegret2* comes quite close. How the impact of the chosen repair operator(s) changes given different problem instance sizes is explained in Figure 2b. We can see that the disadvantage of using the repair operator *Greedy* becomes more pronounced as the number of customers increases, especially when the number of customers is larger or equal to 50. The performance difference between the two repair operators *Regret2* and *GreedyRegret2* only starts to become visible when the number of customers reaches 200.

The second categorical algorithm parameter, *destroy*, has much less importance than *repair*. For this parameter, the choice of values, sorted in increasing order of marginal normalized cost values — i.e., from good to bad performance —, is as follows: *RandomRelated*, *RandomWorstRelated*, *Random*, *WorstRelated*, *RandomWorst*, *Related*, *Worst*. The influence of different problem instance sizes on the impact of the chosen destroy operator(s) is depicted in Figure 2d, but this marginal plot is difficult to interpret visually.

The final marginal plot (Figure 2e) shows the interaction between the two categorical parameters and indicates consistency with the main effect observations: *Greedy* is always the worst choice, despite its combination with any destroy operator; and the choice of *Regret2* generally offers better performance than *GreedyRegret2*, although not very clear. Moreover, among all combinations

of *repair* and *destroy* operators, using *Regret2* as a *repair* operator combined with the destroy operator *Random* is predicted to perform best.



(a) repair

(b) repair x customers

(c) destroy

(d) destroy x customers

(e) repair x destroy

Figure 2: Marginal plots of main and pairwise interaction effects of fANOVA.

3.2.2 Multilevel Regression analysis results

Based on the fANOVA output, a multilevel regression model is fitted. An important aspect is that the statistical assumptions of independence, normality and homoscedasticity typically underlying regression models are fulfilled. Anal-

ysis of the residuals for a linear model revealed the presence of heteroscedasticy (i.e., increasing error variance). It is common to remedy this issue by applying a variance-stabilising transformation which results in a non-linear model[28]. We empirically found a suitable model after taking the reciprocal transformation[4] of the response variable and the cube root[5] of the centred problem instance characteristic *Customers*. Further details on variable transformations can be found in, for example, [28]. The resulting model in equations (5)-(7) describes a non-linear relationship between the performance measure and the algorithm elements explaining it. The R script is provided in Appendix.

Note that, contrary to fANOVA, the regression model includes a binary or dummy variable for each possible combination of repair operators and each combination of destroy operators for which the impact on performance can be studied. So instead of one repair and one destroy variable, the regression model has respectively three and seven variables. Perfect multicollinearity prevents the inclusion of all repair and destroy dummies, and therefore, one repair and one destroy operator configuration are chosen as a baseline. In both cases, it is the configuration including all repair or destroy operators. So the estimates for *Greedy* and *Regret2* represent the change in total cost when switching from using both repair operators (*GreedyRegret2*) to using only one of both. The estimates for *Random*, *RandomRelated*, and so on represent the change in total cost when switching from using all three destroy operators (*RandomWorstRelated*) to a configuration with less destroy operators.

$$\begin{aligned}\frac{1}{Y_i} =&\alpha_{j[i]} + \beta_{1j[i]}Greedy_i + \beta_{2j[i]}Regret2_i + \beta_{3j[i]}Random_i + \cdots + \\ &\beta_{8j[i]}RandomRelated_i + \beta_{9j[i]}Greedy_i \times Random_i + \cdots + \\ &\beta_{20j[i]}Regret2_i \times RandomRelated_i + \epsilon_i\end{aligned} \tag{5}$$

$$\alpha_j = \mu_0^{\alpha} + \mu_1^{\alpha}Customers_j^{\frac{1}{3}} + \eta_j^{\alpha} \tag{6}$$

$$\beta_{zj} = \mu_0^{\beta_z} + \mu_1^{\beta_z}Customers_j^{\frac{1}{3}} + \eta_j^{\beta_z} \tag{7}$$

All operator effects are modelled as varying effects depending on the problem instance characteristic *Customers*, as indicated by the output of fANOVA. Table 3 lists all significant effects. An effect estimate is significantly different from zero if its 95% confidence interval [l-95% CI, u-95% CI] does not include zero. The complete regression table is given in Table 8 in Appendix. In the following paragraphs we show how to interpret the significant effects.

Figure 3 shows the predicted objective function values for all repair and destroy operator configurations, all other variables fixed at their average value. Panel (a) displays the effect of switching to *Greedy* while panel (b) shows the impact of switching to *Regret2*. These predictions show that using *Regret2* as

[4]When the observations are all positive continuous values, the logarithmic transformation is typically applied [14]. However, the residual plot of the log-transformed values still shows increasing error variance, but not for the inverse values.

[5]Since the problem instance characteristic *Customers* is a centred variable, it has both positive and negative values. This excludes the logarithmic and square root transformations since they would delete the negative values. The cube root transformation has the advantage of being able to deal with negative values and is therefore chosen.

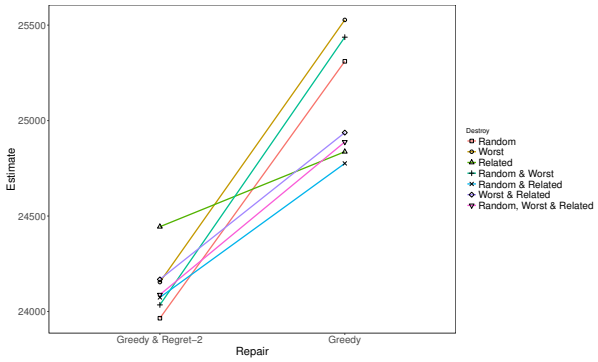
Table 3: Regression table of significant effects[a]

Variable	Estimate	Est.Error	l-95% CI	u-95% CI
Intercept[b,c]	4,151.65	128.11	3,899.79	4,398.53
Greedy	−133.46	5.48	−144.15	−122.90
Regret2	16.98	3.66	9.78	24.18
Random	21.28	3.47	14.27	28.06
Worst	−11.32	3.69	−18.62	−3.98
Related	−60.46	4.70	−69.83	−51.33
RandomWorst	9.20	3.66	1.91	16.30
WorstRelated	−13.92	3.56	−20.97	−6.96
Customers$^{\frac{1}{3}}$	−453.86	29.99	−513.62	−396.08
Greedy × Random	−88.45	7.90	−103.98	−72.62
Greedy × Worst	−89.67	8.87	−107.18	−72.35
Greedy × Related	68.31	6.76	55.07	81.35
Greedy × RandomWorst	−95.71	7.61	−110.70	−80.76
Greedy × RandomRelated	15.19	5.74	3.96	26.44
Greedy × Customers$^{\frac{1}{3}}$	−16.51	1.23	−18.89	−14.11
Regret2 × Customers$^{\frac{1}{3}}$	2.91	0.81	1.35	4.49
Random × Customers$^{\frac{1}{3}}$	3.84	0.77	2.35	5.35
Related × Customers$^{\frac{1}{3}}$	−9.73	1.05	−11.80	−7.70
RandomWorst × Customers$^{\frac{1}{3}}$	1.59	0.82	0.02	3.21
Greedy × Random × Customers$^{\frac{1}{3}}$	−15.83	1.76	−19.25	−12.36
Greedy × Related × Customers$^{\frac{1}{3}}$	10.56	1.54	7.50	13.53
Greedy × RandomWorst × Customers$^{\frac{1}{3}}$	−11.61	1.68	−14.87	−8.36
Greedy × WorstRelated × Customers$^{\frac{1}{3}}$	3.13	1.45	0.25	5.94

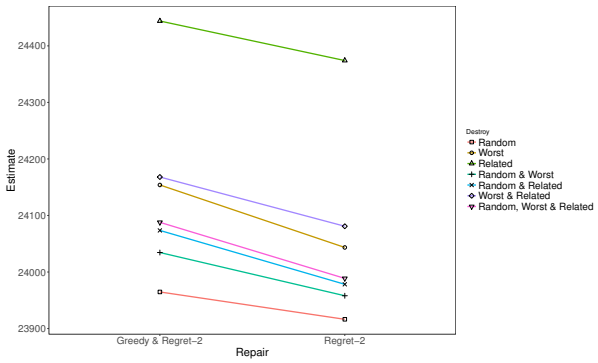
[a] Since the reciprocal transformation of the response variable returned very small values, causing difficulties in the sampling procedure of the *brms* package, all transformed (response) values were multiplied by 100 000 000.
[b] The effects of Greedy & Regret-2 and Random, Worst & Related, the reference levels for the repair and destroy operator dummies, are accounted for in the Intercept.
[c] The Intercept value is backtransformed to the original scale through division by 100 000 000 and taking the inverse of the resulting value.

the sole repair operator is expected to give the best performance results for all destroy operators it is combined with, while relying only on greedy repair is expected to give the worst results for all destroy operator combinations. It is also observed that the relative performance of the destroy operators per individual repair operator differs. The way a solution is destroyed has an impact on how good *Greedy* or *Regret2* is at repairing this solution. *Greedy* seems to have more difficulty in repairing a solution from which customers were removed randomly while *Regret2* is better able to cope with such a situation. *Regret2*, however, appears to find it more difficult to repair a solution from which related customers were removed — with relatedness interpreted in terms of distance as in Pisinger and Ropke [33]. These insights, confirming the analysis of Corstjens et al. [9], spark a new research challenge to discover why certain operator combinations perform (relatively) different.

The previous results are valid for an average instance having 211 customers. We can investigate how these observations are affected when considering instances of different sizes. This is the aforementioned group effect — or problem instance effect in our case — on the performance impact of the operators. The marginal effects are obtained by taking the first derivative of the regression equation. For example, the marginal impact on performance for *Greedy* combined with all destroy operators is derived as follows.



(a)



(b)

Figure 3: Predicted total cost for an average problem instance (211 customers).

$$Y = (\alpha_{j[i]} + \beta_{1j[i]} Greedy_i + \cdots + \beta_{20j[i]} Regret2_i \times RandomRelated_i)^{-1} \quad (8)$$

$$\frac{\partial Y}{\partial Greedy} = -\frac{\beta_{1j[i]}}{(\alpha_{j[i]} + \beta_{1j[i]} Greedy_i)^2} \quad (9)$$

$$\frac{\partial Y}{\partial Greedy} = -\frac{\mu_0^{\beta_1} + \mu_1^{\beta_1} Customers_j^{\frac{1}{3}}}{(\mu_0^{\alpha} + \mu_1^{\alpha} Customers_j^{\frac{1}{3}} + (\mu_0^{\beta_1} + \mu_1^{\beta_1} Customers_j^{\frac{1}{3}}) Greedy_i)^2} \quad (10)$$

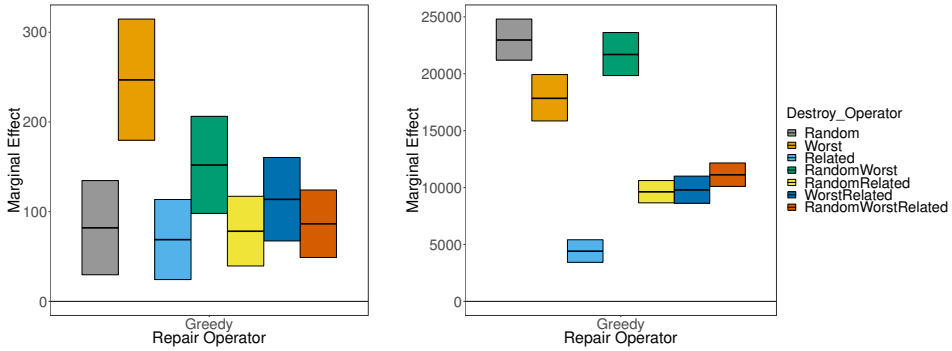
$$\frac{\partial Y}{\partial Greedy} = -\frac{-133.46 - 16.51 Customers_j^{\frac{1}{3}}}{(4151.65 - 453.86 Customers_j^{\frac{1}{3}} - 133.46 - 16.51 Customers_j^{\frac{1}{3}})^2} \quad (11)$$

The more customers, the more negative the impact becomes. Including, for example, the significant interaction effect of *Greedy* with *Random*, the estimated impact is expressed as in equations (12) and (13).

$$\frac{\partial Y}{\partial Greedy} = -\frac{\mu_0^{\beta_1} + \mu_1^{\beta_1} Customers_j^{\frac{1}{3}} + (\mu_0^{\beta_2} + \mu_1^{\beta_2} Customers_j^{\frac{1}{3}})}{(\mu_0^{\alpha} + \mu_1^{\alpha} Customers_j^{\frac{1}{3}} + (\mu_0^{\beta_1} + \mu_1^{\beta_1} Customers_j^{\frac{1}{3}}) + (\mu_0^{\beta_2} + \mu_1^{\beta_2} Customers_j^{\frac{1}{3}}))^2} \quad (12)$$

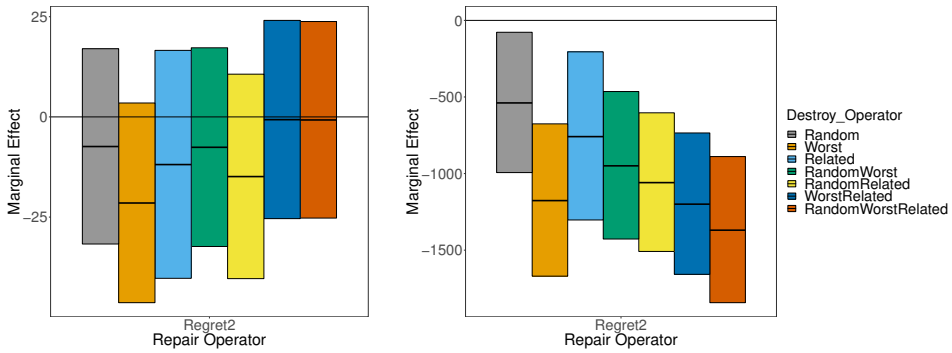
$$\frac{\partial Y}{\partial Greedy} = -\frac{-133.46 - 16.51 Customers_j^{\frac{1}{3}} - 88.45 - 15.83 Customers_j^{\frac{1}{3}}}{(4151.65 - (133.46 + 88.45) - (453.86 + 16.51 + 15.83) Customers_j^{\frac{1}{3}})^2} \quad (13)$$

The problem instance size influence on the interaction effect is interpreted as follows: The combination greedy and random tends to perform increasingly worse than the combination of greedy with all destroy operators as more and more customers have to be served. Figures 4 and 5 show the marginal effect of *Greedy* and *Regret2* for the smallest (a) and largest (b) instance size for all destroy operator combinations. Similarly, the influence of *Customers* on the effect of switching from *RandomWorstRelated* to any other (set of) destroy operator(s) can be investigated. Figures 6 and 7 show the marginal effects of the destroy operators for a problem instance with (a) 25 and (b) 400 customers.



(a) 25 customers (b) 400 customers

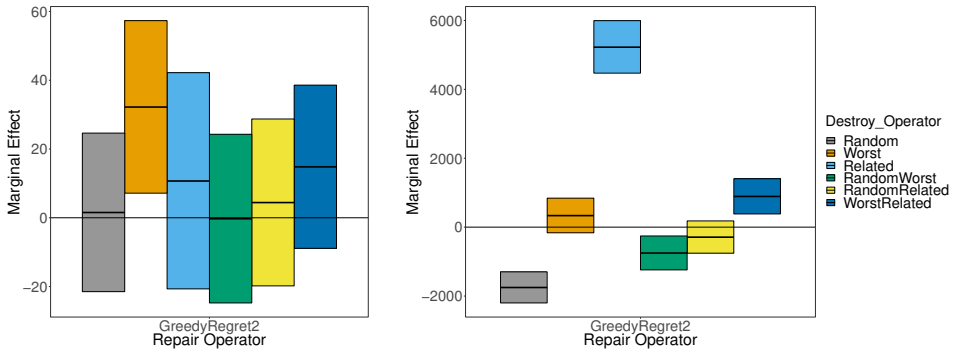
Figure 4: Marginal effect of *Greedy* for (a) 25 and (b) 400 customers.



(a) 25 customers (b) 400 customers

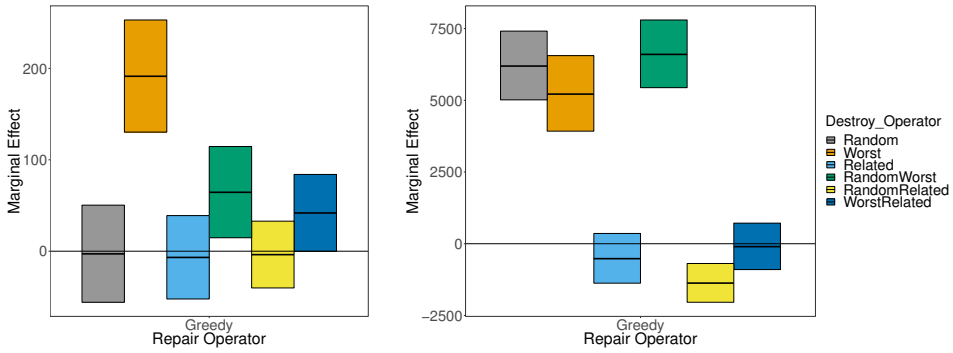
Figure 5: Marginal effect of *Regret-2* for (a) 25 and (b) 400 customers.

Summarising the observations, regression results suggest to avoid relying only on greedy repair, as this is expected to give the worst results in all considered conditions. Concerning the sole use of regret-2, different conclusions are drawn for smaller and larger instances, due to the significant influence of the number of customers an instance has to serve. On the smallest instances, no



(a) 25 customers (b) 400 customers

Figure 6: Marginal effect of destroy operators (with *GreedyRegret2* or *Regret2*) for (a) 25 and (b) 400 customers.



(a) 25 customers (b) 400 customers

Figure 7: Marginal effect of destroy operators (with *Greedy*) for (a) 25 and (b) 400 customers.

significant improvement is observed over using both repair operators. Furthermore, most of the performance differences between the various destroy operator configurations cannot be distinguished from each other as well, meaning the choice of destroy operators is irrelevant for these problem instance sizes. Only the combination with worst removal is indicated to perform significantly worse than the scenario with all destroy operators. On the larger instances, performance differences are more pronounced. The threshold problem instance size at which it becomes beneficial to use regret-2 alone — i.e., where the effect of *Regret2* turns significantly positive because the 95% confidence interval no longer includes zero — is around 186 customers. Regret-2 performs significantly better for all destroy operator combinations, but the combination with related removal is not recommended, since it has the smallest expected performance improvement. Random removal is the preferred combination.

The analysis also led to the observation that random removal and related removal perform relatively different given the repair operator they are combined

with. Random removal is predicted to perform better than related removal when using the regret-2 repair operator, but the opposite is observed when using the greedy repair operator. It shows that new questions might come out of an (exploratory) analysis like the one performed in this paper. In [8] a detailed analysis of the destroy and repair process is performed to explain why random performs better than related removal given the use of regret-2.

3.2.3 Discussion

Comparing the regression analysis to the findings of fANOVA, conclusions are consistent. On the larger problem instances, both approaches find that using regret-2 alone combined with random removal is expected to perform best. For the smaller problem instances, there is consistency in that there is no clear performance difference between using either regret-2 alone or together with greedy. Concerning the destroy operators, the regression analysis cannot identify any (combination of) destroy operator(s) as the preferred one to use since their performances cannot be distinguished from each other. This is also observed in the marginal plot *Destroy* x *Customers* provided by fANOVA, as there is no clear difference in performance between destroy operators on different problem sizes. Therefore, results are again consistent in both analyses.

Furthermore, the observations on the problem instance size influence in both analysis are deducted from different effects. In the fANOVA results, the 2-way interaction between operator(s) and *Customers* is considered an important effect. The multilevel regression analysis, however, also analyses the 3-way interactions between repair and destroy operators and *Customers*. The observed consistencies do make the regression analysis more robust, since these findings are confirmed by a methodology (fANOVA) which does not rely on the statistical assumptions of independence, normality and homoscedasticity of the error terms.

In addition, the regression model facilitates a more detailed analysis, since it provides effect estimates for a particular parameter setting and problem instance, while fANOVA estimates marginal performance for a particular parameter value by averaging over all other parameters and problem instance characteristics. The regression results are able to identify for each combination of repair and destroy operators an instance size interval for which a significant difference in performance is expected. For example, combined with both repair operators *Random* is expected to outperform *Worst* for 134 customers or more, *Related* for 125 customers or more, *RandomWorst* for 214 customers or more, *WorstRelated* for 166 customers or more, *RandomRelated* for 209 customers or more, and *RandomWorstRelated* for 173 customers or more.

Finally, the formulated regression model would have been different if fANOVA had not been performed in advance. We would have fitted a multilevel regression model including all algorithm parameters and components and all problem-level predictors, since there would have been no prior knowledge on which elements have an important or significant impact on performance. Furthermore, parameter interactions would have been included as well. In short, this more extensive

model would have had to estimate substantially more effects than the simpler model based on fANOVA. A possible extensive model was fitted (see Table 9 in Appendix) to illustrate our case. What we observed was that the extensive model is much more time-consuming while not necessarily bringing more information into the analysis:

- First of all, the time required to fit the model is almost twice as long for the extended model (about 44 hours) compared to the simple model (about 22 hours plus 3.5 hours for the fANOVA analysis). Then, comparing the significant effects of both models, it is observed that the majority of effects are significant in both models. One effect appears to be no longer significant in the larger model. This can be because of collinearity issues arising when including more and more variables in a regression model, which may cause variance estimates of coefficients to be inflated, thereby possibly incorrectly showing non-significance of effects. This appears to be the case for the mentioned effect when calculating its variance inflation factor (VIF)[15]. Furthermore, no large value changes are observed when comparing the significant effect estimates from both models. This shows that the simple model does not lack any important variable which might bias the effect estimates, an issue known in statistics as "omitted variable bias" [40].

- Studying the predictive influence of every problem instance characteristic in the extended model, it can be concluded that the problem instance size is the most influential problem instance characteristic, as changing this factor leads to large changes in the total cost values. The other problem instance characteristics show influence as well for particular operator combinations, but the performance change they bring about is of much smaller magnitude than is the case for varying problem instance size values. Therefore, fANOVA understandably denoted the problem instance size as the most important problem characteristic. Furthermore, the predictions for varying problem instance sizes are almost the same in both models, so the larger model does not provide additional information that alters these predictions. In conclusion, we believe the regression model based on fANOVA is a sufficiently detailed model that provides insight into the effects related to the largest shifts in algorithm performance.

4 Case study 2: an iterated local search algorithm for the unrelated parallel machine scheduling problem

4.1 Description

A second case study is found in [36] who solve instances of the unrelated parallel machine scheduling problem with sequence-dependent setup times (UPMSP) using four stochastic local search methods, namely iterated local search [27], simulated annealing [23], late acceptance hill climbing [5] and step counting hill climbing [6]. The problem entails scheduling a set of jobs on a set of machines

Algorithm 2 Iterated Local Search

Input: Initial solution x, p_0, p_{max}, rna_{max}, $iter_{max}$
Output: Best found solution x^{best}
$x^{best} \leftarrow$ x $\leftarrow$ RNA(x,rna_{max})
$i \leftarrow 0$
$p \leftarrow p_0$
$p_{max} \leftarrow p_0 \times p_{max}$
while time limit is not reached **do**
  for j $\leftarrow$ 1 to p **do**
    x $\leftarrow$ generate neighbour x' using a random neighbourhood k, $x' \in N_k(x)$
  end for
  $x \leftarrow$ RNA(x,rna_{max})
  if $f(x) < f(x^{best})$ **then**
    $x^{best} \leftarrow x$
    i $\leftarrow 0$
    $p \leftarrow p_0$
  else
    $x \leftarrow x^{best}$
    $i \leftarrow i + 1$
  end if
  if $i \leq iter_{max}$ **then**
    i $\leftarrow 0$
    $p \leftarrow p + p_0$
    if $p > p_{max}$ **then** $p \leftarrow p_0$ **end if**
  end if
end while

such that the completion time of all jobs (i.e., the makespan) is minimised. Each job is characterised by a processing time per machine, with the setup time between two jobs being both sequence and machine-dependent. It is a highly relevant variant of the machine scheduling problem for production lines with heterogeneous machines [36].

Among the four local search methods implemented, the iterated local search and the simulated annealing algorithms have the most parameters. Since simulated annealing is already incorporated in LNS, we will analyse the iterated local search algorithm. The general idea of the algorithm is as follows. Starting from an initial solution, perturbations are applied, followed by a descent method with a random non-ascendant (RNA) method. The RNA only accepts improvements or different solutions with the same makespan. Four neighbourhood structures are considered to explore the search space. Each neighbourhood reschedules one or multiple jobs according to some logic. Two parameters — *policy* and *main machine* — determine whether this search is diversified or intensified. The pseudocode is provided in Algorithm 2 (taken from [36]) and the algorithm parameters and neighbourhoods are listed in Table 4. Detailed information on the algorithm is provided in [36]. We allow the same time limit as in [36], using the PassMark software benchmark [31] to account for different hardware configurations.

In [42], there are 1640 benchmark instances generated based on 164 combinations of problem characteristic values (10 instances per combination). Since we run multiple parameter settings per instance, each combination of problem characteristic values is already replicated a number of times in the experimental design. Therefore, we limit the instance set to 164 instances by taking one random instance from each combination. Collecting performance data on all 1640 instances would be not computationally realistic. The problem instance charac-

Table 4: Parameters of the Iterated Local Search algorithm

Parameter	Type	Value ranges
p_0	Integer	U[1,100]
p_{max}	Integer	U[1, 10]
rnd_{max}	Integer	U[100000,10000000]
$iter_{max}$	Integer	U[1,10000]
policy	Categorical	U[regular,intensification]
main machine	Categorical	U[random,makespan]
neighbourhoods	Categorical	Shift, Switch, TaskMove, Swap ShiftSwitch, ShiftTaskMove, ShiftSwap, SwitchTaskMove, SwitchSwap, TaskMoveSwap ShiftSwitchTaskMove, ShiftSwitchSwap, ShiftTaskMoveSwap, SwitchTaskMoveSwap, ShiftSwitchTaskMoveSwap

teristics are listed in Table 5. Similar to the VRPTW case study, two data sets were created for both analyses, each containing 82 instances. For each problem instance 40 parameter settings were generated, resulting in 3280 observations per data set.

Table 5: Characteristics of the UPSMP instances

Characteristic	Type	Values
Jobs	Categorical	[6,8,10,12,50,100,150,200,250]
Machines	Categorical	[2,3,4,5,10,15,20,25,30]
Setup time	Integer	U[1, s] with s ∈ [9,49,99,124]
Processing time	Integer	U[1, 99]

4.2 Analysis results

4.2.1 fANOVA analysis results

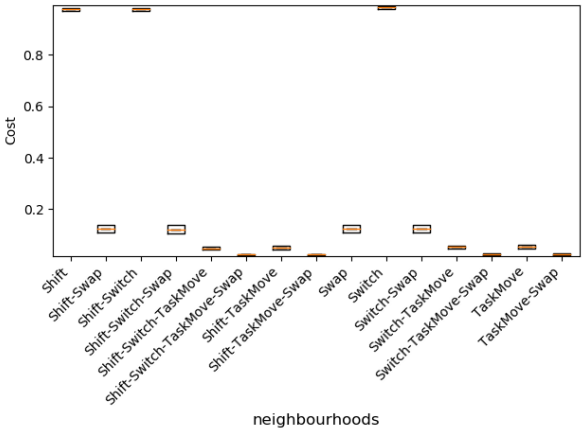
The output generated by fANOVA, using the same normalisation of the performance measure as in the VRPTW case study, is as follows.

```
Sum of fractions for main effects 97.35%
Sum of fractions for pairwise interaction effects 1.58%
96.86% due to main effect: neighbourhoods
0.50% due to interaction: neighbourhoods x jobs
0.45% due to interaction: neighbourhoods x machines
... (remaining effects: < 0.4%)
```

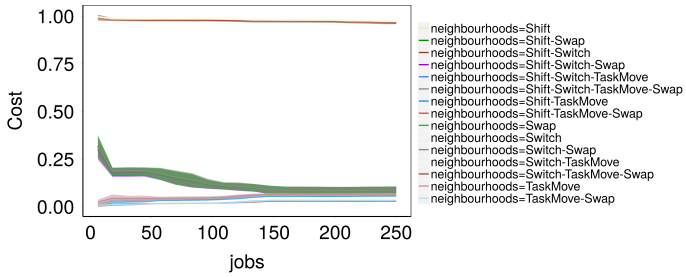
The single parameter *neighbourhoods* explains almost all performance variance in ILS and, therefore, the choice of neighbourhood is of key importance. All other effects have a contribution well below 1%. The marginal plot for *neighbourhoods* is shown in Figure 8a. *Shift*, *Switch* and *ShiftSwitch* are clearly the worst neighbourhood choices. The combinations that include both *TaskMove* and *Swap* achieve the best makespan results.

4.2.2 Multilevel Regression analysis results

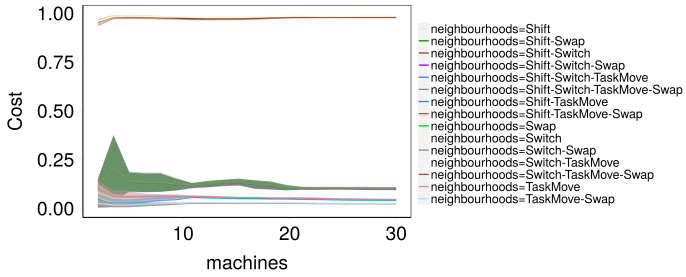
fANOVA clearly indicates the choice of neighbourhood(s) to represent almost all variance in algorithm performance. Consequently, a multilevel regression



(a) neighbourhoods



(b) neighbourhoods x jobs



(c) neighbourhoods x machines

Figure 8: Marginal plots of main and pairwise interaction effects of fANOVA.

with only the neighbourhood variables could be fitted. Yet, one of our interests is analysing problem instance influence on parameter or component effects. Therefore, we would like to include some problem instance characteristics in the regression model. fANOVA lists the interactions of number of jobs and machines with the neighbourhoods as the the second and third most important effect, explaining respectively 0.50% and 0.45% of the variance in performance. Their marginal plots are shown in Figure 8b and 8c, although it is not easy to intepret results visually due to the large number of neighbourhoods and the overlap between their average performance plotted. We include these two characteristics in

the multilevel regression. The log-linear regression model formulated in equations (14)-(16) accounts for 97.81% of the performance variance according to fANOVA. Table 6 lists the significant effects and the complete regression table (Table 10) can be found in Appendix.

$$\begin{aligned}\log Y_i =& \alpha_{j[i]} + \beta_{1j[i]} Shift_i + \beta_{2j[i]} Switch_i + \beta_{3j[i]} TaskMove_i \\ &+ \cdots + \beta_{8j[i]} SwitchTaskMove_i + \beta_{9j[i]} SwitchSwap_i \\ &+ \cdots + \beta_{14j[i]} SwitchTaskMoveSwap_i + \epsilon_i \end{aligned} \tag{14}$$

$$\alpha_j = \mu_0^{\alpha} + \mu_1^{\alpha} Jobs_j + \mu_2^{\alpha} Machines_j + \eta_j^{\alpha} \tag{15}$$

$$\beta_{zj} = \mu_0^{\beta_z} + \mu_1^{\beta_z} Jobs_j + \mu_2^{\beta_z} Machines_j + \eta_j^{\beta_z} \tag{16}$$

Table 6: Regression table of significant effects[a]

Variable	Estimate	Est.Error	l-95% CI	u-95% CI
Intercept[b]	4.51	0.05	4.41	4.60
Shift	1.63	0.03	1.57	1.70
Switch	1.64	0.03	1.57	1.71
TaskMove	0.13	0.02	0.09	0.16
Swap	0.44	0.03	0.38	0.49
ShiftSwitch	1.63	0.03	1.57	1.70
ShiftTaskMove	0.08	0.02	0.05	0.11
ShiftSwap	0.43	0.03	0.37	0.48
SwitchTaskMove	0.07	0.01	0.05	0.10
SwitchSwap	0.44	0.03	0.38	0.49
TaskMoveSwap	0.03	0.01	0.01	0.06
ShiftSwitchTaskMove	0.08	0.01	0.05	0.11
ShiftSwitchSwap	0.43	0.03	0.38	0.48
ShiftTaskMoveSwap	0.03	0.01	0.01	0.05
Jobs	0.01	0.001	0.01	0.01
Machines	−0.07	0.01	−0.08	−0.06
Shift × Machines	0.06	0.004	0.06	0.07
Switch × Machines	0.06	0.004	0.05	0.07
TaskMove × Machines	0.005	0.002	0.0002	0.01
Swap × Jobs	−0.003	0.0004	−0.004	−0.002
Swap × Machines	0.02	0.004	0.02	0.03
ShiftSwitch × Machines	0.06	0.005	0.06	0.07
ShiftTaskMove × Jobs	0.001	0.0003	0.0001	0.001
ShiftSwap × Jobs	−0.003	0.0004	−0.004	−0.002
ShiftSwap × Machines	0.02	0.004	0.02	0.03
SwitchTaskMove × Jobs	0.001	0.0002	0.0004	0.001
SwitchSwap × Jobs	−0.003	0.0004	−0.004	−0.002
SwitchSwap × Machines	0.02	0.004	0.02	0.03
ShiftSwitchSwap × Jobs	−0.003	0.0004	−0.004	−0.002
ShiftSwitchSwap × Machines	0.02	0.003	0.02	0.03
ShiftTaskMoveSwap × Machines	0.004	0.001	0.001	0.01

[a] The effect of Shift, Switch, Task Move & Swap, the reference level for the neighbourhood dummies, is accounted for in the Intercept.
[b] The Intercept value is backtransformed to the original scale taking the exponent $e^{4.51} = 90.92$.

The results for an average problem instance (i.e., 88 jobs to schedule over 14 machines) indicate that a configuration using all four neighbourhoods — the baseline neighbourhood configuration — is never significantly outperformed by a scenario with less neighbourhoods. The configuration *SwitchTaskMoveSwap* is the only one that does not perform significantly worse. All configurations without *TaskMove* perform substantially worse. Furthermore, consistent with the fANOVA results, using *Shift* or *Switch* alone or together should be avoided as they lead to the largest expected deterioration in total makespan compared to *ShiftSwitchTaskMoveSwap*. The configuration *Swap*, *ShiftSwap*, *SwitchSwap* and *ShiftSwitchSwap* perform similarly, additionally indicating that the neighbourhoods shift and switch contribute little to nothing. The marginal effects in Figure 9 shows these observations. The neighbourhoods *TaskMove* and *Swap* seem to reinforce each other. These two neighbourhoods reschedule jobs from

one machine to another one, while shift and switch make moves in a job sequence for a single machine. The contribution of the latter to minimising makespan appears to be minor. This raises the question under what conditions rescheduling within a single machine has a worthwhile contribution since the two best performing configurations *SwitchTaskMoveSwap* and *ShiftSwitchTaskMoveSwap* have at least one neighbourhood operating within one machine.

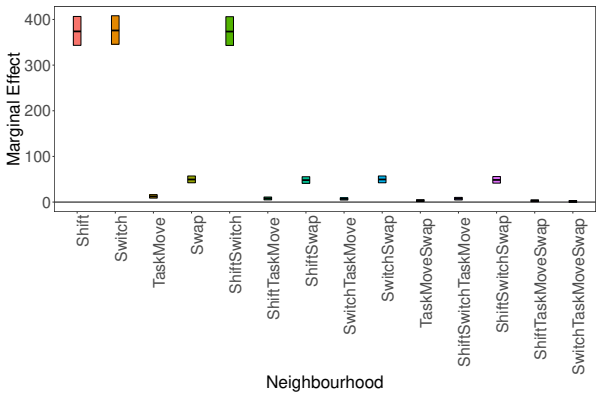
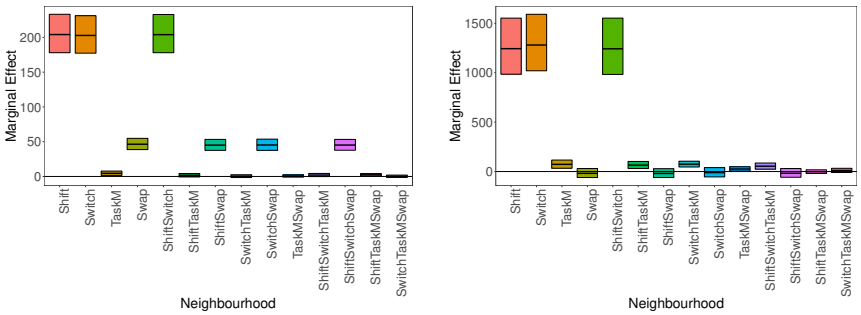


Figure 9: Marginal effect neighbourhoods for an average problem instance.

Investigating the problem instance influence, it is observed that neighbourhood combinations including *Swap* but excluding *TaskMove* are positively influenced by the number of jobs to schedule, ceteris paribus. On problem instances with 200 jobs or more, the performance of these neighbourhood combinations can no longer be distinguished from the performance of *ShiftSwitchTaskMoveSwap*. So, as more and more jobs have to be scheduled, *Swap* surpasses *TaskMove* and becomes the most important neighbourhood structure. In this case, using *Swap* alone would suffice to obtain the best performance results. Extrapolating this positive influence, *Swap* is expected to significantly outperform *ShiftSwitchTaskMoveSwap* for instances with 290 jobs or more.

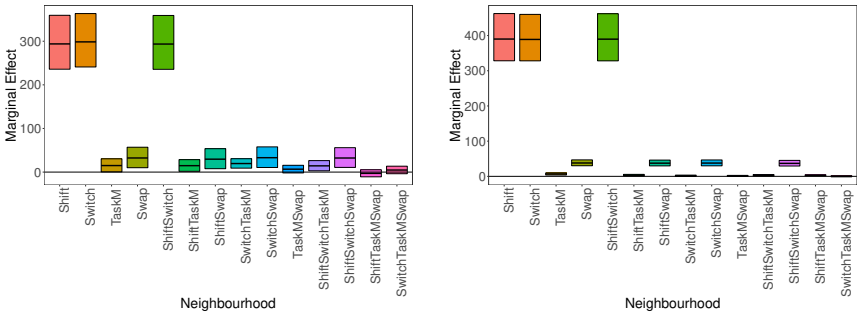
Finally, the number of machines available to schedule jobs on negatively influences the performance of nine neighbourhood combinations. When there are only 2 machines available, the performance of most combinations cannot be distinguished — except *Shift*, *Switch* and *ShiftSwitch* that remain the worst choices overall. When an increasing number of machines are available for scheduling jobs, differences become more clear.

In short, improvements to makespan are primarily achieved by rescheduling jobs across multiple machines rather than single machines. When scheduling less than 200 jobs, moving a random job to a different machine (i.e., task move neighbourhood) is most beneficial to performance, while swapping two random jobs between two machines (i.e. swap neighbourhood) is most beneficial when having to schedule 200 jobs or more.



(a) 6 jobs (b) 250 jobs

Figure 10: Marginal effect of neighbourhoods for (a) 6 and (b) 250 jobs.



(a) 2 machines (b) 30 machines

Figure 11: Marginal effect of neighbourhoods for (a) 2 and (b) 30 machines.

4.2.3 Discussion

Both fANOVA and MLR agree on the findings for an average problem instance. The positive influence of the number of jobs on the combinations with *Swap*, but without *TaskMove* is also observed in both analyses. MLR, however, indicates for job sizes ≤ 200 that there no longer is a significant difference with *ShiftSwitchTaskMoveSwap*, while this is not clear in Figure 8b, which suggests the latter still performs better on average. Regarding the influence of machines, both analyses show it has a minor influence.

Without the fANOVA, a regression model that includes all algorithm elements and problem instance characteristics would have been fitted. Finding a stable extended model proved to be very difficult, as we still got convergence warnings after fitting such an extended model for 113 hours. In comparison, the fANOVA required 35 minutes to finish and the regression model based on fANOVA fitted in 5.5 hours without any issue. Once again, it shows the efficiency gain of the combined methodology. Although the convergence warnings prohibit us to interpret the regression estimates for the extended model and are therefore not reported, we do know that the VIFs for many terms are a lot higher. For example, the variable *Shift* has a VIF of 1.74 in the simple model,

while it is 5.28 in the extended model, implying there is a lot more collinearity and thus instability in the estimates of the extended model.

5 Conclusion

In this paper, we have presented the complementary use of two approaches for analysing performance of heuristic algorithms with multiple parameters and components: fANOVA [19] and multilevel regression (MLR) [9]. The analysis results provided by fANOVA are useful when formulating a proper regression model in MLR since it leads to a more concise regression model with fewer input variables. MLR, on the other hand, provides a more detailed analysis of the effects of algorithm elements and enables confirmatory analyses to be performed. The two methodologies are applied on different data sets drawn from the same algorithm parameter and problem characteristic distributions, thereby avoiding an "overfitting" of the findings. Experimental results of two case studies performed on a vehicle routing and a variant of a machine scheduling problem have shown that the MLR can help to give additional insights on the analysis results provided by fANOVA.

We believe that the line of research introduced in this paper makes a useful contribution towards the engineering cycle of developing optimization algorithms. fANOVA is available as a Python package provided by the authors of the method, and the multilevel regression needs to be implemented manually at the moment. Future work includes making the combination of the two approaches more ready to use, i.e., the addition and the interpretation of the multilevel regression into fANOVA should be automated.

Acknowledgement

This work is funded by COMEX (Project P7/36), a BELSPO/IAP Programme. The computational resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government department EWI. The authors would like to thank Túlio Toffolo for providing us the data for the second case study.

References

[1] Carlos Ansótegui, Meinolf Sellmann, and Kevin Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In *International Conference on Principles and Practice of Constraint Programming*, pages 142–157. Springer, 2009.

[2] Thomas Bartz-Beielstein, Konstantinos E Parsopoulos, and Michael N Vrahatis. Design and analysis of optimization algorithms using computational statistics. *Applied Numerical Analysis & Computational Mathematics*, 1(2):413–433, 2004.

[3] Dimitris J. Bertsimas and David Simchi-Levi. A new generation of vehicle routing research: Robust Algorithms, Addressing Uncertainty. *Operations Research*, 44(2):286, April 1996.

[4] Mauro Birattari. *Tuning Metaheuristics*, volume 197 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, 2009.

[5] Edmund K Burke and Yuri Bykov. A late acceptance strategy in hill-climbing for exam timetabling problems. In *PATAT 2008 Conference, Montreal, Canada*, 2008.

[6] Yuri Bykov and Sanja Petrovic. An initial study of a novel step counting hill climbing heuristic applied to timetabling problems. In *Proceedings of 6th Multidisciplinary International Scheduling Conference (MISTA 2013)*, 2013.

[7] Marco Chiarandini and Yuri Goegebeur. Mixed models for the analysis of optimization algorithms. *Experimental Methods for the Analysis of Optimization Algorithms*, 1:225, 2010.

[8] Jeroen Corstjens, An Caris, and Benoît Depaire. Explaining heuristic performance differences for vehicle routing problems with time windows. In *Learning and Intelligent Optimization*, Lecture Notes in Computer Science, page in press. Springer Berlin Heidelberg, 2018.

[9] Jeroen Corstjens, Benoît Depaire, An Caris, and Kenneth Sörensen. A multilevel evaluation method for heuristics with an application to the vrptw. *Manuscript submitted for publication*, 2017.

[10] Steven P. Coy, Bruce L. Golden, George C. Runger, and Edward A. Wasil. Using Experimental Design to Find Effective Parameter Settings for Heuristics. *Journal of Heuristics*, 7(1):77–97, January 2001.

[11] Jan De Leeuw, Erik Meijer, and Harvey Goldstein. *Handbook of multilevel analysis*. Springer, 2008.

[12] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.

[13] Chris Fawcett and Holger H Hoos. Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, pages 1–28, 2015.

[14] Andrew Gelman and Jennifer Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, December 2006.

[15] Joseph F Hair, Rolph E Anderson, Barry J Babin, and Wiiliam C Black. *Multivariate data analysis: A global perspective*, volume 7. Pearson Upper Saddle River, NJ, 2010.

[16] Giles Hooker. Generalized functional anova diagnostics for high-dimensional functions of dependent variables. *Journal of Computational and Graphical Statistics*, 2012.

[17] John N Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1(1):33–42, September 1995.

[18] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, number 6683 in Lecture Notes in Computer Science, pages 507–523. Springer Berlin Heidelberg, 2011.

[19] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Identifying key algorithm parameters and instance features using forward selection. In *International Conference on Learning and Intelligent Optimization*, pages 364–381. Springer, 2013.

[20] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *International Conference on Machine Learning*, pages 754–762, 2014.

[21] Frank Hutter, Holger H Hoos, Kevin Leyton-Brown, and Thomas Stützle. Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36(1):267–306, 2009.

[22] Zachary Jones and Fridolin Linder. Exploratory data analysis using random forests. In *Prepared for the 73rd annual MPSA conference*, 2015.

[23] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[24] John Lawson and John Erjavec. *Basic Experimental Strategies and Data Analysis for Science and Engineering*. CRC Press, 2016.

[25] Kevin Leyton-Brown, Eugene Nudelman, and Yoav Shoham. Empirical hardness models: Methodology and a case study on combinatorial auctions. *Journal of the ACM (JACM)*, 56(4):22, 2009.

[26] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[27] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer, 2003.

[28] D.C. Montgomery. *Design and Analysis of Experiments, 8th Edition*. John Wiley & Sons, Incorporated, 2012.

[29] David S Moore, George P McCabe, and Bruce A Craig. *Introduction to the Practice of Statistics*. W. H. Freeman, 6th edition, December 2007.

[30] Volker Nannen and Agoston E Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In *International Joint Conference on Artificial Intelligence*, volume 7, pages 975–980, 2007.

[31] PassMark software. Cpu benchmarks. `https://www.cpubenchmark.net/`, 2018.

[32] Paola Pellegrini and Mauro Birattari. The relevance of tuning the parameters of metaheuristics. A case study: The vehicle routing problem with stochastic demand. Technical Report TR/IRIDIA/2006-008, IRIDIA, Universit Libre de Bruxelles, Brussels, Belgium, 2006.

[33] David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, August 2007.

[34] Ronald L. Rardin and Reha Uzsoy. Experimental Evaluation of Heuristic Optimization Algorithms: A Tutorial. *Journal of Heuristics*, 7(3):261–304, May 2001.

[35] Jussi Rasku, Nysret Musliu, and Tommi Kärkkäinen. Automating the parameter selection in vrp: An off-line parameter tuning tool comparison. In *Modeling, Simulation and Optimization for Science and Technology*, pages 191–209. Springer, 2014.

[36] Haroldo G Santos, Túlio AM Toffolo, Cristiano LTF Silva, and Greet Vanden Berghe. Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem. *International Transactions in Operational Research*, 2016.

[37] Joseph P. Simmons, Leif D. Nelson, and Uri Simonsohn. False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological Science*, 22(11):1359–1366, 2011.

[38] Kate Smith-Miles and Simon Bowly. Generating new test instances by evolving in instance space. *Computers & Operations Research*, 63:102–113, November 2015.

[39] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

[40] James Stock and Mark W. Watson. *Introduction to Econometrics*. Prentice Hall, New York, 2011.

[41] Gail M Sullivan and Richard Feinn. Using effect size — or why the p value is not enough. *Journal of graduate medical education*, 4(3):279–282, 2012.

[42] Eva Vallada and Rubén Ruiz. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 211(3):612–622, 2011.

Appendix

```r
library(brms)

Data <- read.table ("./Results_9000_scenarios_test_data.csv", header=T, sep=",")

#variable transformations
Data$c.cust_numb <- scale(Data$customer_number, center = TRUE, scale = FALSE)
Data$c.cust_numb_cbrt <- sign(Data$c.cust_numb)*abs(Data$c.cust_numb)**(1/3)
total_cost_inverse <- (1/Data$total_cost)*100000000

M1 <- brm(total_cost_inverse ~ Greedy + Regret2 + Random + Worst + Related + RandomWorst + WorstRelated +
                          RandomRelated + Greedy:Random + Greedy:Worst + Greedy:Related + Greedy:RandomWorst +
      Greedy:WorstRelated + Greedy:RandomRelated +
                          Regret2:Random + Regret2:Worst + Regret2:Related + Regret2:RandomWorst + Regret2:
                                WorstRelated + Regret2:RandomRelated +
                          c.cust_numb_cbrt +
                          Greedy:c.cust_numb_cbrt +
                          Regret2:c.cust_numb_cbrt +
                          Random:c.cust_numb_cbrt +
                          Worst:c.cust_numb_cbrt +
                          Related:c.cust_numb_cbrt +
                          RandomWorst:c.cust_numb_cbrt +
                          WorstRelated:c.cust_numb_cbrt +
                          RandomRelated:c.cust_numb_cbrt +
                          Greedy:Random:c.cust_numb_cbrt +
                          Greedy:Worst:c.cust_numb_cbrt +
                          Greedy:Related:c.cust_numb_cbrt +
                          Greedy:RandomWorst:c.cust_numb_cbrt +
                          Greedy:WorstRelated:c.cust_numb_cbrt +
                          Greedy:RandomRelated:c.cust_numb_cbrt +
                          Regret2:Random:c.cust_numb_cbrt +
                          Regret2:Worst:c.cust_numb_cbrt +
                          Regret2:Related:c.cust_numb_cbrt +
                          Regret2:RandomWorst:c.cust_numb_cbrt +
                          Regret2:WorstRelated:c.cust_numb_cbrt +
                          Regret2:RandomRelated:c.cust_numb_cbrt +
                         (1 + Greedy + Regret2 + Random + Worst + Related + RandomWorst + WorstRelated +
                          RandomRelated + Greedy:Random + Greedy:Worst + Greedy:Related + Greedy:RandomWorst
                                 + Greedy:WorstRelated + Greedy:RandomRelated +
                          Regret2:Random + Regret2:Worst + Regret2:Related + Regret2:RandomWorst + Regret2:
                                WorstRelated + Regret2:RandomRelated|problem), data= Data,
                       control = list(max_treedepth = 15), chains = 4,warmup = 1000,iter = 4000, cores=4)
```

Table 7: Problem Instance Characteristics

Characteristic	**Type**	**Value ranges**
number of customers	Integer	U[25, 400]
vehicle capacity	Integer	150
x/y-coordinates	Integer	U[0,500]
customer demand	Integer	U[10,50]
Service time	Integer	TRIA(min,max)
		min∼U[10,30]
		max∼U[30,50]
time window depot	Integer	Start = 0; End = 900
time window customer		
- time window centre	Integer	U[0 + travel time, 900 - travel time - service time]
- time window width	Integer	TRIA[min,max]
		min∼U[20,50]
		max∼U[50,80]
- start		Centre - 0.5*width
- end		Centre + 0.5*width
Maximum running time	Integer	TRIA(60,1800)

Table 8: Regression table VRPTW-LNS

Variable	Estimate	Est.Error	l-95% CI	u-95% CI
Intercept	4,151.65	128.11	3,899.79	4,398.53
Greedy	−133.46	5.48	−144.15	−122.90
Regret2	16.98	3.66	9.78	24.18
Random	21.28	3.47	14.27	28.06
Worst	−11.32	3.69	−18.62	−3.98
Related	−60.46	4.70	−69.83	−51.33
RandomWorst	9.20	3.66	1.91	16.30
WorstRelated	−13.92	3.56	−20.97	−6.96
RandomRelated	2.50	3.56	−4.46	9.55
Customers$^{\frac{1}{3}}$	−453.86	29.99	−513.62	−396.08
Greedy × Random	−88.45	7.90	−103.98	−72.62
Greedy × Worst	−89.67	8.87	−107.18	−72.35
Greedy × Related	68.31	6.76	55.07	81.35
Greedy × RandomWorst	−95.71	7.61	−110.70	−80.76
Greedy × WorstRelated	5.65	6.39	−6.70	18.42
Greedy × RandomRelated	15.19	5.74	3.96	26.44
Regret2 × Random	−8.77	5.06	−18.68	1.43
Regret2 × Worst	2.32	5.22	−7.96	12.43
Regret2 × Related	−5.04	5.54	−15.91	5.86
Regret2 × RandomWorst	−3.68	5.14	−13.65	6.61
Regret2 × WorstRelated	−2.14	5.01	−11.93	7.85
Regret2 × RandomRelated	−0.65	5.14	−10.71	9.39
Greedy × Customers$^{\frac{1}{3}}$	−16.51	1.23	−18.89	−14.11
Regret2 × Customers$^{\frac{1}{3}}$	2.91	0.81	1.35	4.49
Random × Customers$^{\frac{1}{3}}$	3.84	0.77	2.35	5.35
Worst × Customers$^{\frac{1}{3}}$	0.58	0.82	−1.03	2.20
Related × Customers$^{\frac{1}{3}}$	−9.73	1.05	−11.80	−7.70
RandomWorst × Customers$^{\frac{1}{3}}$	1.59	0.82	0.02	3.21
WorstRelated × Customers$^{\frac{1}{3}}$	−1.26	0.80	−2.81	0.34
RandomRelated × Customers$^{\frac{1}{3}}$	0.79	0.78	−0.73	2.31
Greedy × Random × Customers$^{\frac{1}{3}}$	−15.83	1.76	−19.25	−12.36
Greedy × Worst × Customers$^{\frac{1}{3}}$	−3.20	1.96	−7.03	0.60
Greedy × Related × Customers$^{\frac{1}{3}}$	10.56	1.54	7.50	13.53
Greedy × RandomWorst × Customers$^{\frac{1}{3}}$	−11.61	1.68	−14.87	−8.36
Greedy × WorstRelated × Customers$^{\frac{1}{3}}$	3.13	1.45	0.25	5.94
Greedy × RandomRelated × Customers$^{\frac{1}{3}}$	2.01	1.27	−0.49	4.48
Regret2 × Random × Customers$^{\frac{1}{3}}$	−2.06	1.14	−4.31	0.16
Regret2 × Worst × Customers$^{\frac{1}{3}}$	−1.25	1.16	−3.51	1.03
Regret2 × Related × Customers$^{\frac{1}{3}}$	−1.77	1.25	−4.19	0.66
Regret2 × RandomWorst × Customers$^{\frac{1}{3}}$	−1.19	1.15	−3.45	1.02
Regret2 × WorstRelated × Customers$^{\frac{1}{3}}$	−0.37	1.13	−2.56	1.84
Regret2 × RandomRelated × Customers$^{\frac{1}{3}}$	−1.24	1.14	−3.48	0.95

[a] The effects of Regret-2 & Greedy and Random, Worst & Related, the reference levels for the repair and destroy operator dummies, are accounted for in the Intercept.

Table 9: Regression table large model VRPTW-LNS

Variable	Estimate	Est.Error	l-95% CI	u-95% CI
Intercept	4,155.43	127.41	3,901.56	4,402.79
Greedy	−134.86	5.28	−145.22	−124.59
Regret2	17.19	3.62	9.93	24.22
Random	20.76	3.46	13.92	27.58
Worst	−11.20	3.63	−18.36	−4.13
Related	−60.37	4.40	−68.99	−51.79
RandomWorst	9.94	3.67	2.74	17.01
WorstRelated	−13.64	3.56	−20.64	−6.73
RandomRelated	2.67	3.49	−4.17	9.56
Cooling_rate	2.14	2.24	−2.23	6.59
Start_temp_ctrl_param	−2.45	2.11	−6.59	1.75
Noise_param	−10.51	3.41	−17.08	−3.79
Determinism_param	0.10	0.05	−0.003	0.21
Customers$^{\frac{1}{3}}$	−460.22	29.13	−516.48	−403.65
Avg_demand	311.17	134.11	54.63	578.72
Avg_service_time	−52.64	30.16	−112.13	6.94
Avg_time_window_width	43.59	20.86	1.25	83.80
Runtime	27.54	23.19	−19.01	72.27
Greedy × Random	−84.60	7.79	−100.09	−69.34
Greedy × Worst	−86.67	8.43	−103.29	−70.35
Greedy × Related	69.73	6.78	56.50	82.87
Greedy × RandomWorst	−92.14	7.44	−106.74	−77.80
Greedy × WorstRelated	6.41	6.22	−5.56	18.75
Greedy × RandomRelated	17.86	5.66	6.87	29.00
Regret2 × Random	−9.39	5.08	−19.30	0.69
Regret2 × Worst	−0.41	5.19	−10.64	9.78
Regret2 × Related	−7.11	5.42	−17.59	3.52
Regret2 × RandomWorst	−3.29	5.18	−13.52	6.97
Regret2 × WorstRelated	−2.87	5.00	−12.66	7.21
Regret2 × RandomRelated	−1.50	5.09	−11.45	8.60
Random × Determinism_param	−0.06	0.08	−0.21	0.09
Worst × Determinism_param	−0.20	0.08	−0.36	−0.04
Related × Determinism_param	−0.48	0.08	−0.64	−0.32
RandomWorst × Determinism_param	−0.06	0.08	−0.22	0.09
WorstRelated × Determinism_param	−0.10	0.07	−0.25	0.05
RandomRelated × Determinism_param	−0.03	0.07	−0.18	0.11
Greedy × Noise_param	−22.22	5.20	−32.55	−12.00
Regret2 × Noise_param	−3.13	4.71	−12.28	6.05
Greedy × Customers$^{\frac{1}{3}}$	−16.40	1.18	−18.72	−14.08
Greedy × Avg_demand	3.41	5.29	−6.98	13.76
Greedy × Avg_service_time	1.91	1.28	−0.59	4.40
Greedy × Avg_time_window_width	−2.30	0.82	−3.89	−0.69
Greedy × Runtime	0.41	0.99	−1.55	2.33
Regret2 × Customers$^{\frac{1}{3}}$	3.00	0.81	1.43	4.57
Regret2 × Avg_demand	−0.59	3.86	−8.36	6.87
Regret2 × Avg_service_time	−0.86	0.89	−2.60	0.87
Regret2 × Avg_time_window_width	0.40	0.56	−0.70	1.49
Regret2 × Runtime	−0.02	0.67	−1.35	1.26
Random × Customers$^{\frac{1}{3}}$	3.78	0.76	2.27	5.25
Random × Avg_demand	−1.20	3.42	−7.96	5.40
Random × Avg_service_time	0.29	0.85	−1.38	1.98
Random × Avg_time_window_width	−0.07	0.55	−1.15	0.99
Random × Runtime	−0.28	0.63	−1.53	0.96
Worst × Customers$^{\frac{1}{3}}$	0.46	0.83	−1.20	2.10
Worst × Avg_demand	−0.65	3.33	−7.29	5.86
Worst × Avg_service_time	−0.24	0.90	−1.99	1.54
Worst × Avg_time_window_width	−0.45	0.57	−1.56	0.66
Worst × Runtime	0.49	0.72	−0.95	1.90
Related × Customers$^{\frac{1}{3}}$	−9.83	1.01	−11.82	−7.85
Related × Avg_demand	−0.24	4.27	−8.67	8.15
Related × Avg_service_time	0.48	1.07	−1.64	2.59
Related × Avg_time_window_width	−2.43	0.72	−3.84	−1.02
Related × Runtime	2.64	0.82	1.02	4.21
RandomWorst × Customers$^{\frac{1}{3}}$	1.53	0.82	−0.07	3.11
RandomWorst × Avg_demand	0.62	3.89	−7.05	8.29
RandomWorst × Avg_service_time	−0.77	0.93	−2.58	1.05
RandomWorst × Avg_time_window_width	0.36	0.57	−0.75	1.47
RandomWorst × Runtime	0.02	0.68	−1.32	1.33
WorstRelated × Customers$^{\frac{1}{3}}$	−1.24	0.79	−2.80	0.33
WorstRelated × Avg_demand	3.96	3.25	−2.38	10.29
WorstRelated × Avg_service_time	0.55	0.86	−1.17	2.23
WorstRelated × Avg_time_window_width	−0.29	0.53	−1.29	0.74
WorstRelated × Runtime	0.70	0.69	−0.64	2.06
RandomRelated × Customers$^{\frac{1}{3}}$	0.79	0.79	−0.77	2.33
RandomRelated × Avg_demand	−0.30	3.48	−7.10	6.43
RandomRelated × Avg_service_time	−0.26	0.85	−1.92	1.42
RandomRelated × Avg_time_window_width	−0.56	0.55	−1.63	0.51
RandomRelated × Runtime	1.00	0.67	−0.33	2.31
Cooling_rate × Customers$^{\frac{1}{3}}$	−0.70	0.50	−1.68	0.27
Cooling_rate × Avg_demand	−0.18	2.37	−4.80	4.53
Cooling_rate × Avg_service_time	−0.29	0.56	−1.37	0.81
Cooling_rate × Avg_time_window_width	−0.72	0.36	−1.42	−0.02
Cooling_rate × Runtime	−0.83	0.42	−1.64	−0.01
Start_temp_ctrl_param × Customers$^{\frac{1}{3}}$	−0.54	0.48	−1.48	0.39
Start_temp_ctrl_param × Avg_demand	0.02	2.23	−4.32	4.39
Start_temp_ctrl_param × Avg_service_time	1.03	0.52	−0.003	2.03

Start_temp_ctrl_param × Avg_time_window_width	−0.19	0.35	−0.86	0.49
Start_temp_ctrl_param × Runtime	−0.17	0.40	−0.96	0.61
Determinism_param × Customers$^{\frac{1}{3}}$	−0.0003	0.01	−0.01	0.01
Determinism_param × Avg_demand	0.02	0.02	−0.03	0.06
Determinism_param × Avg_service_time	−0.001	0.01	−0.01	0.01
Determinism_param × Avg_time_window_width	0.001	0.004	−0.01	0.01
Determinism_param × Runtime	−0.01	0.004	−0.01	0.002
Noise_param × Customers$^{\frac{1}{3}}$	−1.20	0.50	−2.18	−0.20
Noise_param × Avg_demand	0.27	2.32	−4.28	4.83
Noise_param × Avg_service_time	−0.71	0.54	−1.78	0.36
Noise_param × Avg_time_window_width	−0.62	0.35	−1.31	0.08
Noise_param × Runtime	0.89	0.41	0.08	1.68
Greedy × Random × Customers$^{\frac{1}{3}}$	−15.25	1.72	−18.55	−11.86
Greedy × Random × Avg_demand	−2.10	8.59	−18.77	15.04
Greedy × Random × Avg_service_time	0.61	1.93	−3.18	4.44
Greedy × Random × Avg_time_window_width	−1.30	1.19	−3.64	1.06
Greedy × Random × Runtime	3.43	1.46	0.58	6.34
Greedy × Worst × Customers$^{\frac{1}{3}}$	−2.62	1.95	−6.39	1.23
Greedy × Worst × Avg_demand	−1.55	8.70	−18.66	15.49
Greedy × Worst × Avg_service_time	1.76	2.01	−2.20	5.68
Greedy × Worst × Avg_time_window_width	−1.45	1.35	−4.10	1.20
Greedy × Worst × Runtime	3.00	1.65	−0.28	6.24
Greedy × Related × Customers$^{\frac{1}{3}}$	10.38	1.54	7.35	13.43
Greedy × Related × Avg_demand	10.60	7.09	−3.15	24.54
Greedy × Related × Avg_service_time	−1.34	1.64	−4.57	1.90
Greedy × Related × Avg_time_window_width	2.79	1.12	0.59	4.96
Greedy × Related × Runtime	−1.05	1.32	−3.61	1.57
Greedy × RandomWorst × Customers$^{\frac{1}{3}}$	−10.93	1.69	−14.26	−7.62
Greedy × RandomWorst × Avg_demand	0.80	7.70	−14.09	15.89
Greedy × RandomWorst × Avg_service_time	3.09	1.81	−0.49	6.61
Greedy × RandomWorst × Avg_time_window_width	−1.52	1.14	−3.76	0.74
Greedy × RandomWorst × Runtime	2.51	1.37	−0.15	5.19
Greedy × WorstRelated × Customers$^{\frac{1}{3}}$	2.92	1.42	0.09	5.67
Greedy × WorstRelated × Avg_demand	6.12	6.62	−6.90	19.12
Greedy × WorstRelated × Avg_service_time	−1.04	1.54	−4.10	2.03
Greedy × WorstRelated × Avg_time_window_width	0.49	0.98	−1.43	2.41
Greedy × WorstRelated × Runtime	0.50	1.24	−1.94	2.91
Greedy × RandomRelated × Customers$^{\frac{1}{3}}$	1.86	1.25	−0.53	4.35
Greedy × RandomRelated × Avg_demand	−0.65	5.67	−11.78	10.50
Greedy × RandomRelated × Avg_service_time	0.32	1.34	−2.30	2.95
Greedy × RandomRelated × Avg_time_window_width	0.65	0.86	−1.02	2.34
Greedy × RandomRelated × Runtime	−1.33	1.08	−3.42	0.78
Regret2 × Random × Customers$^{\frac{1}{3}}$	−2.14	1.12	−4.31	0.10
Regret2 × Random × Avg_demand	1.06	5.56	−9.82	12.02
Regret2 × Random × Avg_service_time	0.57	1.24	−1.84	3.00
Regret2 × Random × Avg_time_window_width	−0.03	0.79	−1.59	1.54
Regret2 × Random × Runtime	0.62	0.93	−1.20	2.42
Regret2 × Worst × Customers$^{\frac{1}{3}}$	−1.80	1.17	−4.12	0.54
Regret2 × Worst × Avg_demand	2.41	5.23	−7.67	12.74
Regret2 × Worst × Avg_service_time	0.54	1.27	−1.92	3.03
Regret2 × Worst × Avg_time_window_width	−0.05	0.85	−1.72	1.62
Regret2 × Worst × Runtime	−0.59	0.97	−2.47	1.33
Regret2 × Related × Customers$^{\frac{1}{3}}$	−1.64	1.22	−4.02	0.77
Regret2 × Related × Avg_demand	2.88	5.72	−8.37	14.13
Regret2 × Related × Avg_service_time	−0.06	1.37	−2.75	2.64
Regret2 × Related × Avg_time_window_width	0.35	0.86	−1.35	2.03
Regret2 × Related × Runtime	−1.23	0.99	−3.18	0.70
Regret2 × RandomWorst × Customers$^{\frac{1}{3}}$	−1.22	1.15	−3.48	1.05
Regret2 × RandomWorst × Avg_demand	−1.90	5.75	−12.99	9.55
Regret2 × RandomWorst × Avg_service_time	1.72	1.27	−0.77	4.26
Regret2 × RandomWorst × Avg_time_window_width	−0.40	0.79	−1.93	1.20
Regret2 × RandomWorst × Runtime	−0.39	0.93	−2.19	1.45
Regret2 × WorstRelated × Customers$^{\frac{1}{3}}$	−0.79	1.12	−3.00	1.42
Regret2 × WorstRelated × Avg_demand	2.40	5.22	−7.70	12.67
Regret2 × WorstRelated × Avg_service_time	−0.34	1.23	−2.73	2.10
Regret2 × WorstRelated × Avg_time_window_width	−0.03	0.78	−1.57	1.50
Regret2 × WorstRelated × Runtime	−0.53	0.92	−2.34	1.28
Regret2 × RandomRelated × Customers$^{\frac{1}{3}}$	−1.13	1.13	−3.31	1.09
Regret2 × RandomRelated × Avg_demand	−0.50	5.51	−11.25	10.30
Regret2 × RandomRelated × Avg_service_time	0.63	1.23	−1.76	3.06
Regret2 × RandomRelated × Avg_time_window_width	0.55	0.82	−1.05	2.16
Regret2 × RandomRelated × Runtime	−0.75	0.95	−2.62	1.16

Table 10: Regression table UPMSP-ILS

Variable	Estimate	Est.Error	l-95% CI	u-95% CI
Intercept	4.51	0.05	4.41	4.60
Shift	1.63	0.03	1.57	1.70
Switch	1.64	0.03	1.57	1.71
TaskMove	0.13	0.02	0.09	0.16
Swap	0.44	0.03	0.38	0.49
ShiftSwitch	1.63	0.03	1.57	1.70
ShiftTaskMove	0.08	0.02	0.05	0.11
ShiftSwap	0.43	0.03	0.37	0.48
SwitchTaskMove	0.07	0.01	0.05	0.10
SwitchSwap	0.44	0.03	0.38	0.49
TaskMoveSwap	0.03	0.01	0.01	0.06
ShiftSwitchTaskMove	0.08	0.01	0.05	0.11
ShiftSwitchSwap	0.43	0.03	0.38	0.48
ShiftTaskMoveSwap	0.03	0.01	0.01	0.05
SwitchTaskMoveSwap	0.01	0.01	−0.01	0.04
Jobs	0.01	0.001	0.01	0.01
Machines	−0.07	0.01	−0.08	−0.06
Shift × Jobs	−0.0005	0.001	−0.002	0.001
Shift × Machines	0.06	0.004	0.06	0.07
Switch × Jobs	−0.0003	0.001	−0.001	0.001
Switch × Machines	0.06	0.004	0.05	0.07
TaskMove × Jobs	0.0004	0.0003	−0.0001	0.001
TaskMove × Machines	0.005	0.002	0.0002	0.01
Swap × Jobs	−0.003	0.0004	−0.004	−0.002
Swap × Machines	0.02	0.004	0.02	0.03
ShiftSwitch × Jobs	−0.0005	0.001	−0.002	0.001
ShiftSwitch × Machines	0.06	0.005	0.06	0.07
ShiftTaskMove × Jobs	0.001	0.0003	0.0001	0.001
ShiftTaskMove × Machines	0.001	0.002	−0.003	0.01
ShiftSwap × Jobs	−0.003	0.0004	−0.004	−0.002
ShiftSwap × Machines	0.02	0.004	0.02	0.03
SwitchTaskMove × Jobs	0.001	0.0002	0.0004	0.001
SwitchTaskMove × Machines	−0.001	0.002	−0.005	0.002
SwitchSwap × Jobs	−0.003	0.0004	−0.004	−0.002
SwitchSwap × Machines	0.02	0.004	0.02	0.03
TaskMoveSwap × Jobs	0.0002	0.0002	−0.0001	0.001
TaskMoveSwap × Machines	0.0003	0.001	−0.003	0.003
ShiftSwitchTaskMove × Jobs	0.0004	0.0002	−0.0000	0.001
ShiftSwitchTaskMove × Machines	0.001	0.002	−0.003	0.005
ShiftSwitchSwap × Jobs	−0.003	0.0004	−0.004	−0.002
ShiftSwitchSwap × Machines	0.02	0.003	0.02	0.03
ShiftTaskMoveSwap × Jobs	−0.0002	0.0002	−0.001	0.0001
ShiftTaskMoveSwap × Machines	0.004	0.001	0.001	0.01
SwitchTaskMoveSwap × Jobs	0.0001	0.0002	−0.0002	0.0005
SwitchTaskMoveSwap × Machines	−0.001	0.001	−0.004	0.002

[a] The effect of Shift, Switch, Task Move & Swap, the reference level for the neighbourhood dummies, is accounted for in the Intercept.