Parameter analysis of variable neighborhood search applied to multiprocessor scheduling with communication delays*

Tatjana Jakšić-Krüger[0000−0001−6766−4811],
Tatjana Davidović[0000−0001−9561−5339],
Vladisav Jelisavčić

Mathematical Institute of the Serbian Academy of Science and Arts,
Kneza Mihaila 36, 11000 Belgrade, Serbia
{tatjana,tanjad,vladisav}@mi.sanu.ac.rs

Abstract. When dealing with hard, real-life optimization problems, metaheuristics methods are considered a very powerful tool. If designed properly, they can provide high-quality solutions in reasonable running times. The adequate implementation assumes, among other things, finding the best combination of parameters, i.e., identifying their values that correspond to a good performance on most of the test instances. The machine learning methods have become standard practice for parameter tuning and experimental evaluation of metaheuristic algorithms. Multitude of methods have become simple to utilize due to development of R library. Our goal is to contribute to the developing methodology of experimental analysis of metaheuristic algorithms.We are specially interested in Variable neighborhood search (VNS), a very popular metaheuristic for more than 20 years with many successful applications. Its basic form has a small number of parameters, however, each particular implementation can involve a problem-dependent set of parameters. This makes parameter tuning a challenging task. We apply the analysis and tuning of parameters to permutation-based VNS designed to schedule communicating tasks to multiprocessor systems of arbitrary topology. Our goal is to further examine possible interactions between the considered parameters and their influence on the performance of the resulting VNS algorithm. We apply the sophisticated approach to the tuning of VNS parameters that relies on statistical methods and machine learning. The obtained results are presented and discussed in this study.

Keywords: Stochastic algorithms, experimental evaluation, statistical methods, parameter control.

* This research was supported by Serbian Ministry of Education, Science and Technological Development through Mathematical Institute SANU, Agreement No. 451-03-9/2021-14/200029 and by the Science Fund of Republic of Serbia, under the project "Advanced Artificial Intelligence Techniques for Analysis and Design of System Components Based on Trustworthy BlockChain Technology".

1 Introduction

Instead of developing a novel approach to some selected optimization problem, our aim is to present a systematic view into the current state of the algorithm design. Ideally, we are able to detect to which degree can different parts of algorithm change the performance, to identify interactions between algorithm's parameters, and to propose suitable statistical methods for this kind of analysis. Most often we deal with the choice between the time and the quality of a solution. When the optimality of solution is not the imperative in the considered optimization problem, we are happy with the generation of an sub-optimal solution in a short amount of time. Obviously, it is not possible to list all the possible scenarios one might encounter, and thus, our aim is to propose tactics for algorithm design that performs well in the most frequently occurring situations.

There are numerous real-life optimization problems that belong to the combinatorial optimization class. The main characteristic of these problems is that the solution space is finite or at most countably infinite [18]. Solution quality depends on the value of a given objective function that needs to be optimized (minimized or maximized). Usually, combinatorial optimization problems are easy to formulate but difficult for solving because the number of solutions grows exponentially with the problem size. The application of exact methods is impractical and using metaheuristics seems more appropriate [19]. The main characteristic of metaheuristics is that they cannot guarantee the optimality of the generated solutions, however, in practice they provide high-quality solutions very fast.

The intention of our study is to inspect the effectiveness of the well known Variable neighborhood search (VNS) algorithm by the means of machine learning tools. VNS is known as the metaheuristic with small number of parameters, the most important one being k_{max} [13, 17]. However, in practice an implementation of the VNS algorithm may depend on more parameters, which might be distinguished as the model- and problem-specific. In order to design efficient VNS implementation for the considered problem, we need, among other things, to find the best combination of parameters, i.e., to identify their values that correspond to a good performance on most of the test instances. As a case study, we considered permutation-based VNS implementation for Multiprocessor scheduling problem with communication delays (MSPCD) proposed in [8]. Due to the problem complexity, numerous parameters were introduced requiring adequate analysis of their influence to the algorithm's performance, as well as of the interaction between them. We believe that the gathered knowledge might be used not only to predict the performance of the VNS algorithm depending on the structure of test instances, but also as the recipe for the design and analysis of future VNS implementations.

The remainder of this paper is organized as follows. We start with motivation and related work in algorithm experiments in the next section. Description of the MSPCD problem and the corresponding permutation-based VNS implementation are given in Section 3. Section 4 presents the set of parameters and the experiments that we conducted in order to identify their most appropriate values. Section 5 concludes the paper.

2 Motivation and related work

Often in practice an *problem-oriented* heuristic algorithm suffers from limitations such as getting stuck in local optimum, fast convergence to and plateauing at the local optimum, influence of the initial configuration, etc. Metaheuristic methods have been developed as general methods that enable avoiding some of these shortcomings by balancing between the intensification and diversification (exploration and exploitation) of the search [19]. We are particulary interested in the VNS algorithm [13, 17], single-solution method that belongs to the class of metaheuristics with the core engine supported by local search strategy. VNS is a popular method, it has been applied to many optimization problems which is why we are interested in contributing to its proper implementation design.

By examining extremely reach literature on the application of VNS, we have noticed lack of systematic approaches to algorithm design. To the best of our knowledge, the parameter tuning of VNS has always been performed by hand, i.e., by preliminary comparison and evaluation on a sub-set of (more or less) representative test instances. Often in practice, not only parameter values are determined ad-hoc, but also the set of parameters that should be analysed. This may result in missing some important parameters or some of the promising values. In addition, it becomes unsustainable for increasing number of algorithm's parameters. Therefore, we argue that manual analysis of stochastic algorithms with dynamic design for modular or numerical parts is not objective. The automatic tools of parameter analysis are necessary for various reasons: to help with the authors' bias, provide visualization and detect patterns that normally are not possible with human eye.

The algorithm design of metaheuristic methods has been subject in many papers in the literature [1, 4–6, 11, 14–16]. We are in particular inspired by work of Mc'Geoch [16] which distinguishes between algorithm design, algorithm tuning, and code tuning. Very often, to produce general and precise results, the algorithmic experiments should take place on a scale between abstraction and instantiation. In our case, the experimental goals are based on algorithm and code tuning which take place in instantiation space. Our performance indicators are selected so that we can analyse different design choices, from parameter tuning to the modular parts of the VNS algorithm. Moreover, these indicators are matched to the investigated parameters reported in the literature [8]. As a performance measurements we are using solution's quality, computational effort (platform-independent) and CPU time (platform-dependent).

Having all that in mind, we chose to present our experimental analysis on VNS implementation for MSPCD, proposed in [8], for all the aforementioned reasons, as well as for the significance of the problem itself. This particular VNS implementation allows to study multiple various (numerical and categorical) parameters. MSPCD represents an attractive combinatorial optimization problem due to its importance in modern applications not only in computer science, but in numerous other fields, such as team building and scheduling, organization of production lines, cutting and packing. Although concentrated on the particular VNS implementation as a case study, we believe that our approach can con-

tribute to the broader methodology that deals with experimental analysis of any metaheuristic method.

3 Description of MSPCD problem and VNS implementation

Implementation details of VNS for MSPCD are presented in [8] where the authors have recognized several VNS parameters. Our goal is to continue this work with regard to the parametric analysis of the VNS. Here, we provide the short description of the problem, followed by the review of the VNS algorithm implementation.

3.1 Problem description

The Multiprocessor Scheduling Problem with Communication Delays (MSPCD) can be defined as follows: given n tasks (modules or jobs) have to be scheduled on a multiprocessor system with m identical processors connected in an arbitrary way specified by the distance matrix $D_{m\times m}$. This means that we need to decide where and when each task will be executed in order that the total execution time is minimized. For each task, given are its processing time (duration) p_i, as well as the list of its successors and the corresponding communication delays (the amount of intermediate results required to complete task i) if these tasks are to be executed on different processors. Knowing the successors of each task, it is easy to reconstruct the corresponding list of predecessors and to complete the information about precedence constraints between tasks defining the order of task execution. More precisely, a task cannot start its execution unless all of its predecessors are completed and all relevant data (defined by the communication delays) are transferred. Formal definition of MSPCD and the corresponding mathematical formulation are presented in [9].

3.2 Variable Neighborhood Search algorithm and its parameters

VNS consists of three main steps: shaking (SH), local search (LS) and neighborhood change (NC). The role of SH step is diversification, i.e., to prevent search being trapped in a local optimum, while LS step has to ensure the improvement of the current solution by visiting its neighbors (intensification) [13, 17]. The main advantages of VNS are its simplicity and a small number of parameters. Basic variant of VNS [17] has a single parameter k_{max} maximal number of neighborhoods considered, i.e., number of different neighborhood types and/or maximal distance with respect to one neighborhood type. Recent implementations of VNS [13] may consider some additional parameters, however, sometimes even k_{max} may be selected in such a way to be dependent on the problem input data making VNS a parameterless method.

The general steps of VNS may be found in [13], together with various modifications. VNS is known as "a descent, *first improvement method* with randomization" [12] (pg. 3981) and here we refer to this variant as first-improvement

VNS (FI-VNS). The pseudo-code of FI-VNS is given in Alg. 1. It is an adaptation of the corresponding algorithm from [12] (pg. 3979, Algorithm 7) by the inclusion of new parameters that we consider in the implementation of VNS for MSPCD. More details about the performed changes is given in Section 4. As

Algorithm 1: Pseudo code of the FI-VNS algorithm

```
   Input: problem input data, k_max, k_step, k_min, p_plateau, MAX_step
 1 Initialization: x_bsf = Init(), STOP = FALSE;
 2 dstep = 0;
 3 repeat
 4     Apply k_min strategy;
 5     k = k_min;
 6     repeat
 7         x' = Sh(x_bsf,k);
 8         x'' = LS(x');
 9         k = k + k_step;
10         dstep++;
11         if (f(x'') < f(x_bsf)) then
12             x_bsf=x''; /* Move */
13             k = k_min;
14         else if (f(x'') == f(x_bsf)) then
15             prob = rand[0, 1];
16             if prob ≤ p_plateau then
17                 x_bsf=x'';
18         if dstep ≤ MAX_step then
19             STOP = TRUE;
20     until ((k > k_max) || STOP);
21 until STOP;
```

shown, a FI-VNS iteration starts from an initial solution x_{inic} and runs its step Sh, LS, and NC until the best-so-far solution (the current approximation of the best solution x_{bsf}) has been improved or until VNS explores k_{max} (the maximal number) of neighbourhoods around the x_{bsf} solution. The best-so-far solution x_{bsf} represents a global knowledge exchanged between the VNS iterations. In particular, the initial x_{bsf} solution is generated by applying Largest Processing Time first (LPT) constructive heuristic and improving it by the local search in the initialization phase (see Alg. 3, function Init). In the main VNS loop, SH tries to move the search far from the current best solution. Then, once LS is finished, the newly found solution x'' is compared against the x_{bsf} solution (Alg. 1, line 9). If the improvement is made, the VNS iteration is reset to $k = 1$ and SH starts from the newly discovered x_{bsf} solution. Otherwise, the value for k increases and a VNS iteration terminates if k reaches k_{max} or if the stopping criterion is satisfied. In both cases, the neighbourhood counter k is reinitialized to k_{min}. As a consequence, the execution time might vary greatly from one itera-

tion to another, which is our source of inspiration to utilize the *best improvement* BI-VNS strategy (see Alg. 2).

The second variant of the VNS algorithm is founded on the best-improvement concept, described in [12] (pg. 3981, Algorithm 11), presented here as BI-VNS (Alg. 2). For the purpose of demonstrating differences between the two VNS variants, in the corresponding pseudo-codes we describe how global knowledge is being utilized.

Algorithm 2: Pseudo code of the BI-VNS algorithm.

Input: problem input data, k_{max}, k_{step}, k_{min}, $p_{plateau}$, MAX_step

```
Initialization: x_bsf = Init() STOP = FALSE;
dstep = 0;
repeat
    Apply k_min strategy;
    k = k_min;
    x_min = x_bsf; /* current best */
    repeat
        x' = Sh(x_bsf,k);
        x'' = LS(x');
        k = k + k_step;
        dstep++;
        if (f(x'') < f(x_min)) then
            x_min = x''; /* Move */
        else if (f(x'') == f(x_min)) then
            prob = rand[0, 1];
            if prob ≤ p_plateau then
                x_min = x'';
        if (dstep ≤ MAX_step) then
            STOP = TRUE;
    until (k > k_max) || STOP;
    if (f(x_min) ≤ f(x_bsf)) then
        x_bsf=x_min;
until STOP;
```

Unlike FI-VNS, BI-VNS algorithm does not restart the neighborhood counter after each improvement of x_{bsf}. Therefore, in Alg. 2 we need an auxiliary variable x_{min} keeping the current best solution to be used in updating x_{bsf} when k reaches k_{max}. Working load between successive BI-VNS iterations is more balanced than in FI-VNS due to consistency in the number of explored neighbourhoods. To be able to compare performance of FI-VNS and BI-VNS, we utilize *dstep* in Alg. 1 and Alg. 2 to count the number of discrete steps (i.e., the number of SH and LS executions). The counter is controlled by MAX_step which we appoint as the stopping criterion.

Algorithm 3: PHASES WITHIN THE VNS ITERATION.

```
1  Function Init:
2      x = LPT(); /* list scheduling heuristic to generate initial solution */
3      x' ← LS(x);
4      if (f(x') < f(x)) then
5          x = x';
6      return x ;
7  Function SH(x, N, k):
8      /* Generate feasible solution x' from k^th neighborhood of x */
9      for (i = 1; i <= k; i++) do
10         x' ∈ N(x); /* at random */
11         x = x'
12     return x ;
13 Function LS(x', N, FI, forward):
14     /* Apply a local search method on x' depending on input parameters*/
15     repeat
16         Let N(x) = x_1, ..., x_p;
17         if (¬forward) then
18             Reverse(N(x));
19         i ← 0;
20         x' ← x;
21         repeat
22             i ← i + 1;
23             if (f(x_i) < f(x)) then
24                 x ← x_i ;
25                 if FI then
26                     Break;
27         until i = p;
28     until f(x') ≤ f(x);
29     return x';
```

There are several research goals of the parameter analysis we want to accomplish. We formulate them through the following research questions: (1) Which VNS parameter is the most influential? (2) How each parameter individually influences the performance? (3) Are there interactions between VNS parameters?

4 Empirical study of the VNS parameters

Besides parameters related to the definition of VNS, each particular implementation can involve a problem-dependent set of parameters. These kind of parameters are for example solution quality measurement parameter (e.g., objective function value, fitness or utility), number and types of used neighborhoods, or improvement strategy. In order to obtain efficient algorithm, the one that pro-

vides high-quality solutions for the majority of tested instances, the appropriate values of all parameters have to be identified. Finding the best possible combination of parameter values is usually an optimization problem itself, parameter values represent decision variables, while the objective function is the same as in the considered combinatorial optimization problem. Therefore, this part of the metaheuristic design deserves special attention and it is a main subject of our paper.

4.1 Previous study

The authors in [8] describe some of the steps required by the suggestions of the experimental algorithmics [16]. For example, choice of the data structure in which the solution is hold and the study of modular parts of the VNS algorithm: (1) different types of neighborhoods (Swap-1, Swap-2, Swap-3 and IntCh); (2) heuristics for initial solution generation; (3) task scheduling rule; (4) search direction (forward-backward); (5) FI- and BI- improvement strategies. Tuning of the VNS algorithm's parameters (maximal number of neighborhoods, neighborhood definitions, i.e. combinations and restrictions, stopping criterion) has been conducted manually. Their results have shown that ES is the most efficient scheduling rule when starting from feasible permutation based on critical path distance. However, the authors have not provide concrete results. Results in Tables 7 and 8 from [8] indicate high interaction between parameters, due to "chaotic behavior of the scheduling process".

Several VNS parameters which we categorize as numerical and categorical variables have been recognized in [8]. The categorical variables are: shaking rules, neighbourhood combinations, restricted neighbourhoods, the search direction and the search strategy. The numerical parameters are k_{max}, k_{step}, $p_{plateau}$, while the influence of k_{min} was not considered as its value was always 1. The experimental study was conducted gradually, where for each categorical variable an independent study is performed with fixed configuration of other parameters. In [8], maximal CPU time (t_{max}) is utilised as stopping criterion.

4.2 Current study

A stochastic nature of meta-heuristics requires executions of the algorithm several times in order to correctly estimate the response value. A response value commonly refers to the quality of a solution (sometimes referred to as usefulness or utility), or the computational effort (e.g., running time, number of function evaluations, number of iterations). A thorough review of different performance measures is provided in [3, pg. 110] and in [2, pg. 40]. If the idea is to estimate an effort to reach a solution of a predetermined quality, the performance measure describes what is known as *efficiency* of the algorithm.

The differences between [8] and our study can be summarized as follows. Here we consider only the CP+ES heuristic to generate initial solution and we utilize only one type of the neighbourhood (Swap-1). On the other hand, we analyse two VNS search strategies, FI-VNS and BI-VNS, we introduce new parameter k_{min}

and provide new strategies to assign values to k_{min} and k_{step}. Our experimental setup is precised in Sec. 4.4.

4.3 Problem instances and performance estimator

We utilize the same set of problem instances as in [8], i.e., the random test instances with known optimal solution. The set of 5 problem instances with $n = 50$ and different values of precedence relation density ρ are utilized within the preliminary study of stopping criterion (see Sec. 4.5). For the purpose of parameter analysis, we restrict our study to one problem instance-ogra50_50. We argue that in our case we should employ optimal value as the target solution and base the performance on the *successful runs* [3]. In particular, the most often employed measure, the average (median) objective function value, might not be suitable for our analysis. The problem is that the two quite distant solutions may produce an optimum with the same probability in the upcoming step. Thus, from the view of the convergence properties of the VNS algorithm, we are less certain about what defines the quality of the solution. We believe that in this situation it is suitable to provide a *target value* and count the number of steps required to find the solution which quality reaches that value. To obtain a statistically meaningful representative of the algorithm's execution, we repeat runs for 100 different seeds and count the occurrences of target solution (optimal in our case). This is our performance estimator (opt_{count}).

4.4 The parameters we want to investigate

We analyse the parameters of VNS listed in Table 1. There are several *basic* VNS parameters i.e., k_{max}, k_{step}, k_{min} and $p_{plateau}$. These parameters have been recognized in the literature as an integral part of the VNS algorithms. In addition, for this particular implementation, we consider several categorical parameters that have important impact to the overall performance, such as `VNS-search-strategy`, `FI-LS` and `forward`. Different values of `VNS-search-strategy` produce two quite different algorithms as presented in Section 3.2, while changing values for the other two parameters generates only new variants of the same algorithm. The two categorical parameters within gray cells in Table 1, local search (`FI-LS`) and neighborhood search direction (`forward`) are problem-specific (i.e., closely connected to MPSCD). In particular, we want to compare two type of local search: LS using the first improvement search strategy (`FI-LS=1`) and the best improvement local search (`FI-LS=0`). According to [8] FI-VNS with `FI-LS=1` performs faster and better than `FI-LS=0`. However, when we appoint MAX_step as the stopping criterion the better performance is exhibited when `FI-LS=0` for both FI-VNS and BI-VNS algorithms. The categorical parameter `forward` is also problem-specific, however it is reported in [8] that its influence is not significant.

Categorical parameter `KMIN` is introduced in order to denote different strategies of calculating k_{min}. Namely, our goal is to compare the variants of VNS that visit all neighborhoods in the `SH` step with the one that always skips the same neighborhoods in the case when $k_{step} > 1$. The first strategy appoints

Table 1. VNS parameters.

VNS parameters	Numeric	k_{max}	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
		k_{step}	$[1, \lceil \frac{x}{2} \rceil - 1] \cup \{k_{max} - 1\}$
		$p_{plateau}$	$\{0, 0.1, 0.5, 0.9, 1\}$
	Categorical	VNS-search-strategy	`BI-VNS`
			`FI-VNS`
		`KMIN`	`kmin`: $k_{min} = 1$
			`kminrand`: $k_{min} = rand[1, k_{step}]$
		`FI-LS`	0
			1
		`forward`	0
			1

$k_{min} = 1$, i.e., the search always starts from the smallest neighbourhood. We denote this strategy as `kmin`. The second strategy is to change k_{min} as a function of k_{step}. When denoted as `kminrand` it specifies that k_{min} is chosen from $[1, k_{step}]$ uniformly at random.

In practice, the values of specific parameters are determined with regard to the dimension of the problem and/or other problem related characteristics. Here, we focus on the smaller values for k_{max} as provided in Table 1. The values for k_{step} have been specified in the preliminary study as integers from interval $[1, \lceil \frac{x}{2} \rceil - 1]$. For example if $k_{max} = 4$ than the possible values for k_{step} are $\{1, 2, 3\}$, while for $k_{max} = 10$, $k_{step} \in \{1, 2, 3, 4, 9\}$.

4.5 Setting the stopping criterion value

In our preliminary study we try to answer a question if there exists such a value for MAX_step beyond which the average solution's quality does not improve in some practical sense. The goal is to observe changes of the the objective function value and the runtime at different levels of stopping criterion of the corresponding algorithm. In the first phase of the preliminary experiments we have set MAX_step to 500. This corresponds to about 60 seconds of CPU runtime for problem instance ogra50_10. We generate convergence plots, i.e., the best objective function values for different *dstep*. The plots corresponding to the ogra50_50 instance are presented in Fig. 1. The goal is to find lower value of MAX_step that allows multiple and fair comparisons between different algorithm's configurations.

From Fig. 1, with help of descriptive statistics we may conclude that for problem instance ogra50_50, $MAX\_step = 200$ is suitable to differentiate VNS parameter's configurations. More precisely, to find the appropriate value for *dstep* for which we can conduct meaningful empirical study of the VNS's configuration, we observe the curves in Fig. 1. The figure shows the propagation of the average,

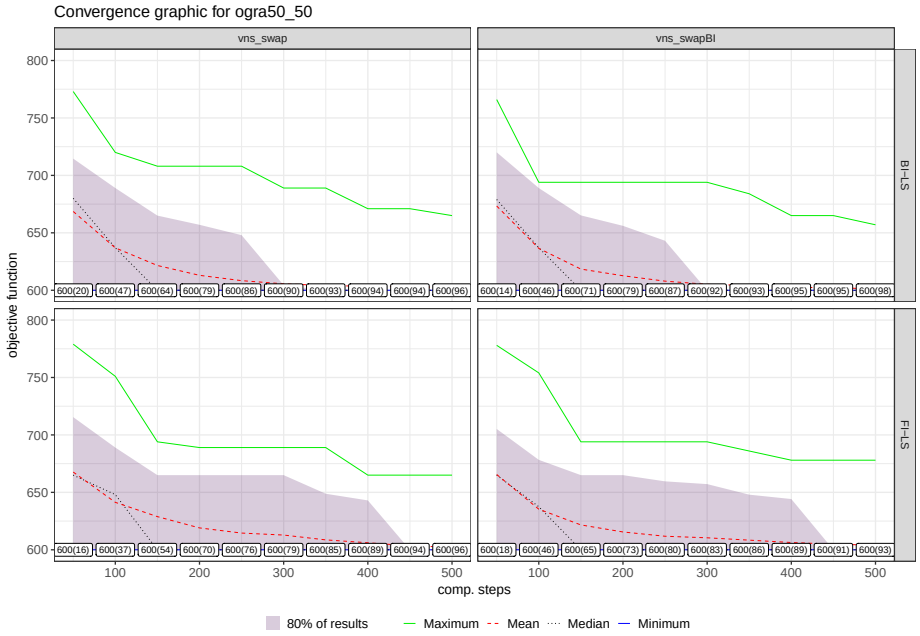

Fig. 1. Convergence plots of four VNS algorithms for problem instance ogra50_50.

minimal and maximal solutions' quality. We are able to detect the start of the stagnation phase, i.e., the value for *dstep* for which the algorithm reaches stagnation. When we observe red doted line, that signifies mean objective function value over 100 repetitions, suitable value for *dstep* seems to be 200. We prefer to observe mean values (over median) due to the fact that mean value is more sensitive to the span of objective function values.

4.6 Methodology

Our strategy may be divided into two stages: 1. variable impact, and 2. modeling. The first stage of our study pertains to answer our first research question of the most influential parameters. The modeling stage concerns choice of the right model to fit our data. The goal is to better explain relationships between VNS parameters and the opt_{count} which then provide the best view on the nature of their influence. It is the necessary step of the experimental study in order to measure size of the effect of VNS parameters on the estimated performance [3]. We may start with presuming a linear model (which is often the practice) and visualize the residuals of the model. This is a good practice to check assumptions for the linear model which is linearity, normality and homoscedasticity of the residuals. We utilize R software tools.

4.7 Results

We separate our results for the two VNS search strategies. Both studies start with measuring importance of each considered VNS parameter by employing random forest method and linear regression model on all parameters. The visualization of the linear model via *ggplot2* and *flexplot* packages reveal the existence of the non-linear terms or interactions.

The variable importance results for FI-VNS are presented in Table 2. The

Table 2. Size of effect for FI-VNS parameters

rank	VNS parameter	RF	Semi-Partial R^2	Standardized β coefficient	p-value
1	k_{max}	174.4	0.332	0.57	$< 10^{-16}$
2	`forward`	155.5	0.266	-0.52	$< 10^{-16}$
3	`FI-LS`	28.5	0.056	-0.24	$< 10^{-16}$
4	$p_{plateau}$	16.5	0.017	0.13	$< 10^{-16}$
5	k_{step}	4.2	0.0	0.01	0.44
6	`KMIN`	1.2	0.003	0.05	$< 10^{-5}$

random forest was conducted via *cforest* and *varImp* functions (*party_1.3-8*) for *seed*(1010). The results are reported under RF column. With multivariate linear *lm* function[1] we calculate Semi-Partial R^2. By applying *lm.beta* (package *lm.beta_1.5-1*) on the result of *lm*, we determined standardized β coefficient. The linear regression model of these 6 parameters has identified a significant regression equation with the outcome $F(6, 2473) = 851$, $p < 10^{-16}$, adjusted $R^2 = 0.67$. From Table 2 we can conclude that k_{step} does not show practical or statistical significant effect on the performance. In principal, we could eliminate it from the further modeling. `KMIN` shows statistical importance, but due to small effect size it shows negligible practical significant influence on the estimated performance. Thus, we may also remove it from the further modeling and examine other possible relationships between the remaining parameters and the opt_{count}.

After extracting the three parameters (k_{step},$p_{plateau}$,`KMIN`) the regression model is defined with the following significant regression equation: $opt_{count} = 5k_{max} - 0.31k_{max}^2 - 39.37forward + 3.74kmax \cdot forward - 6.5FI\_LS$ with the outcome $F(5, 1234) = 1124$, $p < 10^{-16}$, adjusted $R^2 = 0.82$. The model shows nonlinear relationship and interactions, specifically among k_{max} and `forward` parameters. This means that for the optimal performance we need to take care about the values of both parameters. On the other hand, influence of `FI_LS` is independent from others and based on the negative coefficient we know that `FI_LS=0` produces optimal solutions more often than `FI_LS=1`. With regard to k_{max} we observe the concave relationship (due to negative coefficient) and an

[1] lm() (package *stats4* R version 4.0.4.

uphill slope with positive coefficient 4.67. This indicates that as k_{max} increases, the opt_{count} is higher, however, the performance will reach a stagnation point after which it might degrade. The final linear model is visualized in Fig. 3.

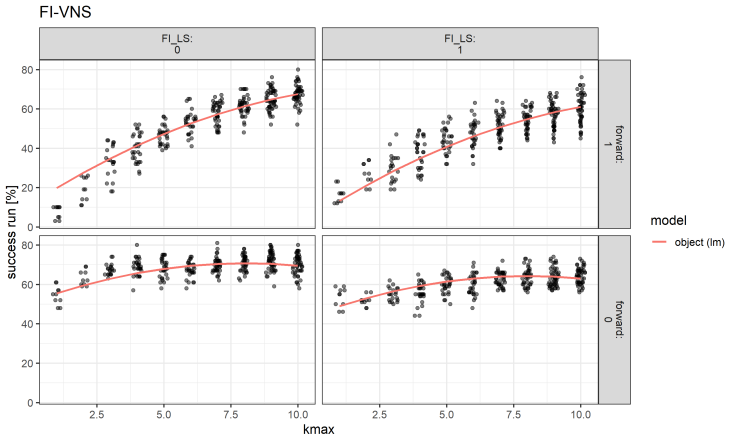

Fig. 2. Regression model for FI-VNS and its corresponding main parameters.

The variable importance results for BI-VNS are presented in Table 3. The linear regression model of these 6 parameters has identified a significant regression equation with the outcome $F(6, 1233 = 649.9)$, $p < 10^{-16}$, adjusted $R^2 = 0.76$. From the table we conclude that last three parameters may not be included

Table 3. Size of effect for FI-VNS parameters

rank	VNS parameter	RF	Semi-Partial R^2	Standardized β coefficient	p-value
1	`forward`	301.63	0.47	-0.69	$< 10^{-16}$
2	k_{max}	189.75	0.24	0.51	$< 10^{-16}$
3	`FI-LS`	41.55	0.05	-0.22	$< 10^{-16}$
4	$p_{plateau}$	-0.37	0.0	-0.003	0.8
5	k_{step}	16.42	0.01	-0.06	$< 10^{-5}$
6	`KMIN`	2.8	0.0	-0.003	0.8

in our regression model of the performance analysis. Therefore, we are able to produce final significant regression model for BI-VNS defined in R as follows: $opt_{count} = 4.67k_{max} - 0.3k^2_{max} - 47.71forward + 3.9kmax \cdot forward - 6.84FI\_LS$ with the outcome $F(5, 1234) = 1698$, $p < 10^{-16}$, adjusted $R^2 = 0.87$. The final linear model is visualized in Fig. 3.

Discussion about individual parameters `KMIN` parameter has shown low statistical impact on the overall performance. However, it is still recommended to conduct graphical analysis due to the nature of the relationship between k_{min} and k_{step}. We produce plots that we provide on the following web-page. As shown, there are interactions between `KMIN` and k_{step}, which means that the success of a `KMIN` strategy depends on the values of k_{step} and k_{max} consequently. In addition, we observe the same for interactions between `KMIN` and k_{max}. Consequently, for the optimal performance it is necessary to conduct tuning via some software package like iRace, ParamILS, SPO etc [4].

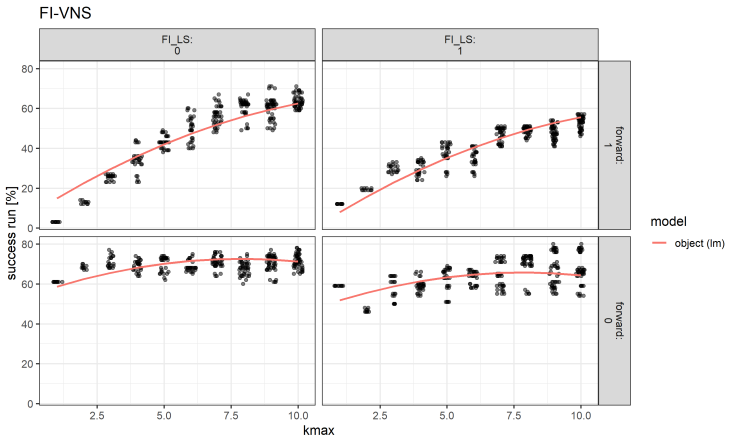

Fig. 3. Regression model for BI-VNS and its corresponding main parameters.

5 Conclusion

In this paper we apply the multivariate regression model and random forests to experimental analysis of parameters used in permutation-based VNS designed to schedule communicating tasks to multiprocessor systems of arbitrary topology. Our goal is to promote the application of automated experimental evaluation in order to conduct objective performance analysis of stochastic algorithms. We are able to identify the most influential and nonlinear relationships between the algorithm's parameters. As the future work we plan to extend the list of values for some of the parameters and apply the presented methodology to larger problem instances.

References

1. Barr, R.S., Golden, B.L., Kelly, J.P., Resende, M.G.C., Stewart Jr., W.R.: Designing and reporting on computational experiments with heuristic methods. J. Heuristics **1**(1), 9–32 (1995)

2. Barrero, D.F.: Reliability of performance measures in tree-based Genetic Programming: A study on Kozas computational effort. Ph.D. thesis, University of Alcalá (2011)
3. Bartz-Beielstein, T.: Experimental Research in Evolutionary Computation; The New Experimentalism. Natural Computing Series, Springer (2006)
4. Bartz-Beielstein, T., Doerr, C., Bossek, J., Chandrasekaran, S., Eftimov, T., Fischbach, A., Kerschke, P., López-Ibánez, M., Malan, K.M., Moore, J.H., et al.: Benchmarking in optimization: Best practice and open issues (2020), arXiv, cs.NE, 2007.03488
5. Beiranvand, V., Hare, W., Lucet, Y.: Best practices for comparing optimization algorithms. Optim. Eng. **18**(4), 815–848 (2017)
6. Czarn, A., MacNish, C., Vijayan, K., Turlach, B., Gupta, R.: Statistical exploratory analysis of genetic algorithms. Evol. Comput, IEEE Transactions on **8**(4), 405–421 (2004)
7. Davidović, T., Crainic, T.G.: Benchmark-problem instances for static scheduling of task graphs with communication delays on homogeneous multiprocessor systems. Comput. Oper. Res. **33**(8), 2155–2177 (2006)
8. Davidović, T., Hansen, P., Mladenović, N.: Permutation-based genetic, tabu, and variable neighborhood search heuristics for multiprocessor scheduling with communication delays. Asia-Pac. J. Oper. **22**(03), 297–326 (2005)
9. Davidović, T., Liberti, L., Maculan, N., Mladenović, N.: Towards the optimal solution of the multiprocessor sheduling problem with communication delays. In: Proceedings of the 3rd Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2007). pp. 128–135. Paris, France (2007)
10. Davidović, T., Liberti, L., Maculan, N., Mladenović, N.: Towards the optimal solution of the multiprocessor sheduling problem with communication delays. In: Proceedings of the 3rd Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2007). pp. 128–135. Paris, France (2007)
11. Eiben, A.E., Smit, S.K.: Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm Evol. Comput **1**(1), 19–31 (2011)
12. Floudas, C.A., Pardalos, P.M. (eds.): Encyclopedia of Optimization. Springer, 2nd edn. (2009)
13. Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S.: Variable neighborhood search: basics and variants. EURO Journal on Computational Optimization **5**(3), 423–454 (2017)
14. Hooker, J.N.: Needed: An empirical science of algorithms. Oper. Res. **42**(2), 201–212 (1994)
15. Kendall, G., Bai, R., Błazewicz, J., De Causmaecker, P., Gendreau, M., John, R., Li, J., McCollum, B., Pesch, E., Qu, R., et al.: Good laboratory practice for optimization research. Journal of OR Society **67**(4), 676–689 (2016)
16. McGeoch, C.C.: A guide to experimental algorithmics. Cambridge University Press (2012)
17. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput Oper Res **24**(11), 1097–1100 (1997)
18. Papadimitriou, C.H., Steiglitz, K.: Combinatorial optimization: algorithms and complexity. Courier Dover Publication, New York (1998)
19. Talbi, E.G.: Metaheuristics: from design to implementation, vol. 74. John Wiley & Sons (2009)