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, metaheuristic methods are considered a very powerful tool. If designed properly, they can provide high-quality solutions in reasonable running times. 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 analysis and performance assessment a challenging task. Contribution of this work is twofold: we develop a new variant of the VNS algorithm for the considered optimization problem and simplify the methodology for experimental analysis of metaheuristic algorithms. We conclude three stages of the parameter analysis: parameter definition, deciding the most influential parameters and analysis of their relationship. The analysis contributes to the design of VNS as a search problem in the space of its parameters. We apply the sophisticated approach that equally relies on visual as well as on the statistical and machine learning methods that have become standard practice for parameter tuning and experimental evaluation of metaheuristic algorithms. The obtained results are presented and discussed in this study.

Keywords: Stochastic algorithms, Experimental evaluation, Statistical methods, Parameter control, Parameter tuning.

1 Introduction

In addition to developing a novel approach to some selected optimization problem, our aim is to also present one view into the current state of the algorithm

* 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".

design. Ideally, we are able to detect to which degree various algorithm's parts can change the performance, to identify their interactions and to propose suitable statistical methods for this kind of analysis. When the optimality of a solution is not the imperative in the considered optimization problem, we are happy with the generation of sub-optimal solutions provided in a short amount of time. In addition, we deal frequently with the choice between the time and the quality of a solution. In practice, a *problem-oriented* heuristic algorithm often 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]. 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 examine the algorithm's design choices of a metaheuritic method. We are particulary interested in the VNS algorithm [12, 16], single-solution method that belongs to the class of metaheuristics with the core engine supported by local search procedure. It is known as the metaheuristic with small number of parameters, the most important one being k_{max} (Sec. 3.2, [12, 16]). However, an implementation of the VNS algorithm in practice often depends on more parameters, distinguished as the model- and problem-specific [3]. In order to design an efficient VNS algorithm for the considered problem, we first need to recognize all the model-specific parts of the algorithm that may constitute possible parameters. Next, the following research questions should be addressed: (1) Which VNS parameter is the most influential? (2) How each parameter individually influences the performance? (3) Are there interactions between VNS parameters? Finally, we need to find the best combination of parameters, i.e., to identify configurations that correspond to preferable algorithm's performance of the VNS algorithm on one or more test instances.

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

2 Motivation and Related Work

The algorithm design of metaheuristic methods has been the subject of many papers in the literature [1, 4–6, 9, 13–15]. We are particularly inspired by the work of Mc'Geoch [15] 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 the numerical parameters to the modular parts of the VNS algorithm. Moreover, these indicators are matched to the investigated parameters reported in the literature [7]. As performance measurements we are using solution's quality, computational effort (platform-independent) and CPU time (platform-dependent).

Our analysis is the simplification of the statistical Design of Experiments strategy shown in [3], and it is divided into three parts: (1) variable impact, (2) modeling, and (3) tuning. The first part pertains to find the most influential parameters. In the second step we apply statistical and visualization tools to investigate possible interactions between the VNS parameters. The third part is related to the tuning proces which means identifying configurations that correspond to preferable algorithm's performance. However, we do not conduct the third step in this paper and focus only on the first two stages of the parameter analysis. We start with presuming a linear model (which is often the practice) and visualize the residuals of the model. We utilize R software tools [18].

We consider Multiprocessor scheduling problem with communication delays (MSPCD) [17]. Permutation-based VNS implementatio for MSPCD is proposed in [7] where the authors recognized several VNS parameters. We continue the parametric analysis of the VNS, by making the distinction between different types of parameters. We are inspired by the fact that we might be able to predict the performance of the VNS algorithm, assuming that the structure of test instances is the same throughout the benchmark set. We also hope that our results can contribute in develpoing the general approach to design and analysis of future VNS implementations.

The VNS algorithm is a popular method that 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 a lack of systematic approaches to algorithm design. As we could notice, the parameter tuning of VNS has always been performed by hand, i.e., by preliminary comparison and evaluation on some predefined combinations of parameters on a sub-set of (more or less) representative test instances. Often in practice, not only parameter values are determined adhoc, 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, this approach 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.

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 contribute

to the more wide-spread utilization of the existing experimental methodology for metaheuristics.

3 Description of MSPCD and VNS Implementation

In this section we provide a short description of MSPCD, followed by the presentation of the FI-VNS and the new best improvement VNS algorithm (BI-VNS).

3.1 Problem Description

MSPCD can be defined as follows: given are n tasks (modules or jobs) that have to be scheduled on a multiprocessor system with m identical processors connected in an arbitrary way specified by the distance matrix $D_{m\times m}$. For each task, given are its processing time (duration) p_i, as well as the list of its successors and the corresponding communication costs C_{ij} (the amount of intermediate results required to complete task j) if tasks i and its successor j 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 costs) are transferred between the corresponding processors. The time required for data transfer (communication delay) depends not only on the communication costs, but also on the distance between processors executing the corresponding tasks, as well as on the hardware-related parameter defining the ratio between computation and communication speeds. Therefore, we need to decide where and when each task should be executed in order to minimize the total execution time. Formal definition of MSPCD and the corresponding mathematical formulation are presented in [8].

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) [12, 16]. The main advantages of VNS are its simplicity and a small number of parameters. Basic variant of VNS [16] 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 [12] 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 [12], together with various modifications. VNS is known as "a descent, *first improvement method* with randomization" [11] (pg. 3981) and here we refer to this variant as the 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 [11] (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 are given in Section 4. As

Algorithm 1: Pseudo code of the FI-VNS algorithm

```
   Input: problem data, N, FI, forward, k_max, k_step, k_min, p_plateau, MAX_step
 1 Initialization: x_bsf = Init(N, FI, forward), STOP = FALSE;
 2 dstep = 0;
 3 repeat
 4 |   Apply k_min strategy;
 5 |   k = k_min;
 6 |   repeat
 7 |   |   x' = Sh(x_bsf, N, k);
 8 |   |   x'' = LS(x', N, FI, forward);
 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 the CP+ES constructive heuristic. It utilizes critical path (CP) based permutation scheduled by Earliest start (ES) rule and improved 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_{min} 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 when the stopping criterion is satisfied. As a consequence of FI-VNS algorithm structure, the execution time might vary

greatly from one iteration to another, which is our source of inspiration to utilize the *best improvement* BI-VNS strategy (see Alg. 2) described in [11] (pg. 3981, Algorithm 11). For the purpose of demonstrating differences between the two

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

```
Input: problem data, N, FI, forward, k_max, k_step, k_min, p_plateau, MAX_step
Initialization: x_bsf = Init(N, FI, forward), STOP = FALSE;
dstep = 0;
repeat
    Apply k_min strategy;
    k = k_min;
    x_min = x_bsf; /* current best */
    repeat
        x' = Sh(x_bsf, N, k);
        x'' = LS(x', N, FI, forward);
        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;
```

VNS variants, in the corresponding pseudo-codes we use colors to describe how global knowledge is being utilized.

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. Counting computational steps allows us more reliable experiments with regard to the reproducibility.

Algorithm 3: PHASES WITHIN THE VNS ITERATION.

```
1  Function Init(N, FI, forward):
2      x = CP+ES(); /* constructive heuristic to generate initial solution */
3      x' ← LS(x, N, FI, forward);
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';
```

4 Empirical Study of the VNS Parameters

Besides the parameters related to the given definition of VNS in Section 3.2, 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, an improvement strategy or some other operator. To obtain an efficient algorithm that provides high-quality solutions for the majority of tested instances, the appropriate values of its parameters need to be identified. Finding the best possible parameters configuration is an optimization problem itself, parameter values represent decision variables, while the objective function corresponds to the considered optimization problem. Therefore, this part of the

metaheuristic design deserves special attention and it is the main subject of our paper.

Our work extends the analysis conducted in [7]. We utilize the same benchmark set and in this section we revisit the results of the previous study in order to point out the differences with the new results.

4.1 Previous Study

VNS parameters from [7] are classified as numerical and categorical variables in the following way. The categorical variables are operators: shaking rules, neighbourhood combinations, restricted neighbourhoods, the search direction and the search strategy. The numerical parameters are k_{max}, k_{step}, $p_{plateau}$. It is presumed that k_{min} equals 1. Authors have conducted the experimental study gradually, i.e. an independent study is performed for each categorical variable with the fixed configuration of other parameters. The maximal CPU time (t_{max}) is utilised as the stopping criterion.

We notice that [7] follow several steps of the experimental algorithmics as suggested in [15]. For example, choice of the data structure in which the solution is held as well as the study of modular parts of the VNS algorithm are conducted for the following operators: (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- local search procedure. Additionally, tuning of the VNS parameters (maximal number of neighborhoods, neighborhood definitions, i.e. combinations and restrictions, stopping criterion) has been conducted manually. With regard to the numeric parameters in [7], the results in Tables 7 and 8 indicate high interaction between $k_{max}, k_{step}, p_{plateau}$. Therefore, the authors conclude that the scheduling process exhibits a chaotic behaviour. In the new study we address the issue of the non-linear relationship between the numerical parameters with the help of the regression model.

4.2 New Study

Differences between the previous and the new study can be summarized as follows. We implement the best improvement VNS algorithm (BI-VNS) and reintroduce the parameter k_{min}. Next, we propose new values for numerical parameters k_{max}, k_{step} and $p_{plateau}$ that were not previously studied. The four parameters (k_{min}, k_{max}, $k_{step}, p_{plateau}$) are recognized in the literature as an integral part of the VNS algorithms and are known as the *basic* VNS parameters. Several categorical parameters are introduced that, we believe, have an important impact on the performance: `VNS-strategy` (`VNSs`), `LS-type`, `KMINS` and `direction` (`dir`). Categorical parameter `VNSs` allows us to revisit the study of FI-VNS from [7] and compare it against the new BI-VNS. It also allows us to examine dependencies that the VNS algorithm may have with the basic VNS parameters. To control for two types of local search techniques we utilize the categorical parameter `LS-type`. Auxiliary categorical parameter `KMINS` denotes four different

Table 1. VNS parameters.

VNS parameters	Numerical	k_{max}	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
		k_{step} (if $k_{max} \leq 4$)	$[1, k_{max} - 1]$
		k_{step} (if $k_{max} > 4$)	$[1, \lceil \frac{k_{max}}{2} \rceil - 1] \cup \{k_{max} - 1\}$
		$p_{plateau}$	$\{0, 0.1, 0.5, 0.9, 1\}$
	Categorical	`VNS-strategy`	BI-VNS (FI-VNS = 0) FI-VNS (FI-VNS = 1)
		`KMINS`	`kmin1`: $k_{min} = 1$ `kminrand`: $k_{min} = rand[1, k_{step}]$ `kr2`,`kr3`: mixed
		`LS-type`	BI-LS (FI-LS = 0) FI-LS (FI-LS = 1)
		`direction`	forward = 0 forward = 1

strategies for calculating values of k_{min}. The `direction` parameter is introduced in the previous study as an indicator of the neighbourhood search direction (see Fig.4 in [7]). The overall list of VNS parameters is shown in Table 1 (numerical parameters are defined with italic and categorical with teletype font). To generate an initial solution we consider CP+ES heuristic and only one type of the neighbourhood (Swap-1). This particular algorithm's design choice is based on the report from [7].

As presented in Section 3.2, values of `VNSs` produce two algorithms FI-VNS and BI-VNS, while other categorical parameters define the so-called algorithm variants. Gray cells in Table 1 mark problem-specific (i.e., closely connected to MPSCD) parameters: `LS-type` and `dir`. We compared two types of local search: the first improvement search strategy (`FI-LS=1`) and the best improvement local search (`FI-LS=0`) denoted as BI-LS. According to [7] the FI-VNS algorithm with `FI-LS` performs faster and better than with `BI-LS`. However, when we appoint MAX_step as the stopping criterion we find that `BI-LS` exhibits better performance within both FI- and BI-VNS algorithms (see Section 4.4). Parameter `dir` is also problem-specific, however it was reported in [7] as not significant. We show that `dir` is the second most important considered VNS parameter that greatly influences algorithm's performance w.r.t. d_{step}.

With `KMINS` our goal is to compare two main ideas of the neighbourhood search: (1) visit all the neighborhoods of the current best solution ($k_{step} = 1$), (2) skip neighborhoods ($k_{step} > 1$). We should note that the implementations of these two ideas differ w.r.t. the two main VNS search strategies and each implementation yields a different algorithm variant. Foremost, the search terminates if the better solution is found (for FI-VNS) or all k_{max} neighborhoods have been visited (for BI-VNS). In addition, in BI-VNS the variable k_{min} is defined only once, while in FI-VNS we must reinitialize k_{min} at line 13. We denote strategy

(1) as `kmin1` strategy, which appoints $k_{min} = 1$ at the line 4 of both algorithms, as well as at the line 13 of Alg. 1. The second strategy, denoted as `kminrand`, is to change k_{min} as a function of k_{step}. In particular, in initialization and reinitialization phases of FI-VNS the value of k_{min} becomes a random number from $[1, k_{step}]$. There are two more `KMINS` strategies (`kr2`, `kr3`) that are possible within the FI-VNS algorithm. Algorithm variant defined by `kr2` initializes variable k_{min} as a random number from $[1, k_{step}]$. However, at line 13 it resets k to 1. The variant `kr3` defines $k_{min} = 1$ at line 4, and at line 13 it determines k as a random number from $[1, k_{step}]$.

Values for k_{max} are often in practice determined w.r.t. the dimension of the problem and/or other problem-related characteristics. We focused on $k_{max} \leq 10$ in order to examine the values that were not utilized in the previous study. The values for k_{step} are specified as integers from the set $[1, \lceil \frac{x}{2} \rceil - 1] \cup \{x - 1\}$, if $x \geq 4$ where $x = k_{max}$. Moreover, if $k_{max} = 1 \vee k_{max} = 2$ then $k_{step} = 1$. If $k_{max} = 4$ then the possible values for k_{step} are $\{1, 2, 3\}$, while for $k_{max} = 10$, $k_{step} \in \{1, 2, 3, 4, 9\}$.

4.3 Experimental Methodology

For a deeper understanding of the algorithm operators and numerical parameters we employ visual, statistical and machine learning tools to help us decide about the most influential parameters. The next step is the study of the interconnection of parameters and their main effects. Typical data analysis via machine learning tools may be performed over many different variables, however, visual analysis is possible for only five or at most six variables at a time. There are other methodological settings that we need to be aware of in order to procure data that helps us gain a better understanding of VNS operators.

The stochastic nature of meta-heuristics requires repeating algorithm's runs several times in order to properly estimate the *usefulness* or *utility* of the algorithm's performance. The measure of performance may refer to the solution of an optimization problem and/or computational effort such as running time, number of function evaluations, number of iterations, etc. Moreover, the definition of performance measure (PM) depends on the goal and conditions of the research. For example, PM under limited amount of computer resources is often based on central tendency estimators: the average objective function value from n independent runs, or the median [3]. We are especially interested in the *efficiency* of the algorithm, thus, we focus on determining the effort required to reach a solution of a predetermined quality (target value). Our goal is to understand under which parameter configurations and computational restrictions is possible to obtain the highest probability of producing an optimum. We argue that in our case we should employ optimal value as the target solution and base the performance on the objective function's value (i.e. solution's quality). Consequently, the goodness of performance can be related to the number of successful runs within the limited budget. A thorough review of different performance measures is provided in [3, pg. 108] and in [2, pg. 40].

To obtain a statistically meaningful representative of the algorithm's performance, we repeat executions n_{run} number of times. Another important aspect of the experimental methodology is the benchmark set. The problem benchmark set consists of random test instances with known optimal solution (details about problem's instance are given in [7]). We are particularly interested in problems with 50 tasks as it provides opportunity for VNS to find the optimal solution. Moreover, in order to address the goals of our study we employ one problem instance ogra50_50 to be scheduled on the hypercube of $m = 4$ processors. The performance estimator represents the number of the occurrences of the target solution within n_{run} repetitions, which we denote as opt_{count}. In particular, $opt_{count} = \frac{\#succ}{n_{run}} \cdot 100$, where $\#succ$ denotes the number of successful runs. Finally, we set $n_{run} = 100$, therefore we employ 100 different seeds.

Essential part of the experimental study of randomized algorithms is choosing the appropriate type and value of the stopping criterion (SC). The goal is to avoid *floor* effect as the solution's quality reaches stagnation. To avoid comparisons where all variants exhibit the stagnation, we should find a suitable value for MAX_step beyond which the algorithm's performance does not improve in some practical sense. In Fig.1 convergence plots of two VNS algorithms are presented, BI-VNS:FI-LS (BI-VNS algorithm with FI-LS search) and FI-VNS:BI-LS (FI-VNS algorithm with BI-LS search). Graphics show the propagation of the average, median, minimal and maximal solutions' quality. Between different d_{step} we are able to compare descriptive statistics, thus providing details about the convergence properties. The small boxes show the best result (brackets hold the occurrence for the n_{run} repetitions). For SC analysis we first set $MAX\_step = 500$, which was large enough for the algorithms to approach the pick of their performance. We concluded that $MAX\_step = 200$ is suitable for our parameter analysis due to the fact that the algorithm variants exhibit significant differences (before reaching stagnation phase). In addition, the variance significantly decreases compared to smaller MAX_step.

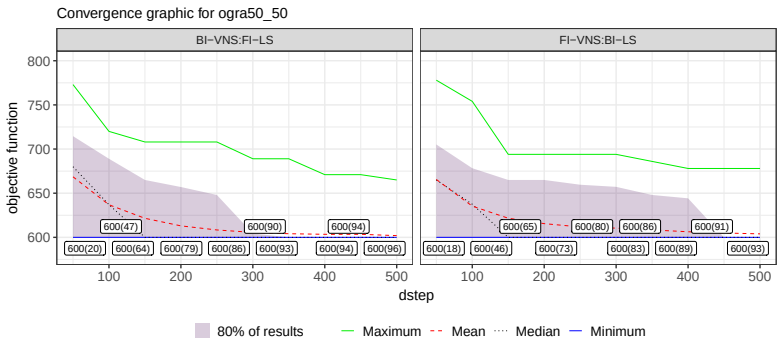

Fig. 1. Convergence plots of two VNS variants for: $p_{plato} = 0, k_{step} = 1, \texttt{forward} = 1$.

4.4 Results

With the help of graphical tools from flexplot [10], the first impressions of the parameter importance is achieved. Histograms in Fig. 2, divided among parameters `dir`, `VNSs` and `LS-type`, show the shape and the spread of the data distribution. Most distinguished distributions are between levels of `dir`. The backward direction search ($forward = 0$) produced bell-shape distribution with smaller spread compared to $foward = 1$ which has long left tail. This indicates that the VNS variant with forward search direction is much more sensitive to values of other VNS parameters. For better understanding of the reason behind such distributions, we need to conduct the parameter importance analysis followed by the regression model of their interconnections

Estimate of the significance of each VNS parameter was determined by random forest and linear regression model (Table 2) [18]. The random forest from package *party_1.3-8* was conducted via R function *cforest* and the values in column RF of Table 2 were generated by function *varimp*() from the same package. We set *seed*(1010). The third and the fourth columns correspond to the Semi-Partial R^2 and the standardized β coefficients determined via the multivariate linear regression *lm* function[1] [10]. In particular, Semi-Partial R^2 values are extracted from the report generated with *estimates*[2] on the result of *lm*(). The β coefficients represent normalized regression coefficients generated by R function *lm.beta*() (package *lm.beta_1.5-1*). Simple linear regression model of seven VNS parameters has identified a significant regression equation with the outcome $F(7, 3712) = 1161$, $p < 10^{-16}$, adjusted $R^2 = 0.69$.

In Table 2 the VNS parameters are ranked based on four indicators: random forest, Semi-Partial R^2, β coefficients and p-value. The higher the value of the

[1] lm(), package *stats4*, R version 4.0.4, https://www.R-project.org/
[2] package *flexplot*.

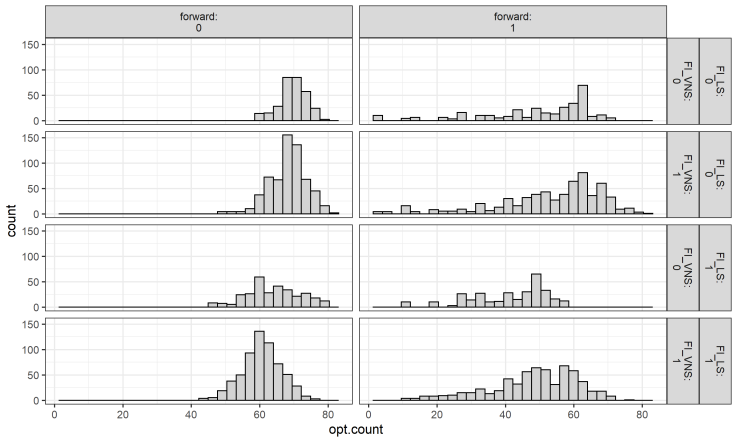

Fig. 2. Histogram of complete data for all VNS parameters.

Table 2. Size of the effect of VNS parameters

rank	VNS parameter	Random forest	Semi-Partial R^2	Standardized β coefficient	p-value
1	`direction`	202.93	0.333	-0.577	$< 10^{-16}$
2	k_{max}	174.70	0.291	0.547	$< 10^{-16}$
3	`LS-type`	33.01	0.053	-0.229	$< 10^{-16}$
4	`VNS-strategy`	14.46	0.002	0.056	$< 10^{-8}$
5	$p_{plateau}$	10.15	0.007	0.082	$< 10^{-16}$
6	k_{step}	9.99	0.000	-0.018	0.07
7	`KMINS`	1.48	0.000	-0.012	0.24

first three indicators, larger is the importance of the parameter. The ranking also indicates the significance of proper algorithm design in order to achieve the highest performance. For example, the most influential operator is `dir`, followed by `LS-type`. `VNSs` has not shown to be as important as was initially expected. Numerical parameter $p_{plateau}$ showed small influence, however, the random forest and linear regression coefficients do not agree on the ranking. Based on the p-values, parameters k_{step} and `KMINS` do not show statistically significant main effect on the performance.

After the parameter selection phase (screening), we are able to address the question of interactions. We do this by combining the first four ranked parameters into the model. We have produced the following significant regression equation: $opt_{count} = -55.56\texttt{dir} + 4.53k_{max} - 12.14\texttt{LS-type} - 2.36\texttt{VNSs} - 0.31k_{max}^2 + 5kmax\cdot\texttt{dir} + 16.9\texttt{dir}\cdot\texttt{LS-type} + 7.73\texttt{VNSs}\cdot\texttt{dir} + 0.77kmax\cdot\texttt{LS-type} - 2.36kmax\cdot\texttt{dir}\cdot\texttt{LS-type}$, with the outcome $F(10, 3709) = 2069$, $p < 10^{-16}$, adjusted $R^2 = 0.85$. The regression model is visualized in Fig. 3. The variability of data around the model in Fig. 3 is caused by the numerical VNS parameters (each point corresponds to the unique parameter configuration).

The most significant interaction term is between k_{max} and `dir`, thus, their configuration is intertwined. In case of the $forward = 1$ the influence of k_{max} is pronounced following the concave curve due to the negative coefficient of the second order term (k_{max}^2). This indicates that opt_{count} increases with k_{max}. In the case of $forward = 0$, the parameter k_{max} has less influence, however, the growth trend is detected. Fig. 3 shows some nonlinear behavior that is not explained by the model. We suspect that the non-linearity originates from interactions between the numerical VNS parameters. The second most significant nonlinear term is between `LS-type`, `dir` and k_{max}. Thus, to achieve the highest performance we need to simultaneously configure these three parameters. However, `VNSs` should also be considered due to the significant interaction with `dir`.

We can distinguish the effects that parameters impose independently from each other. For example, the model and Fig. 3 indicate that the negative coefficient in front of `LS-type` suggests that BI_LS (red curve) produces optimal solutions more often than FI_LS (blue curve). $forward = 0$ indicates better performance over all categorical data and to achieve $opt_{count} \geq 80$, it is enough to employ smaller values for k_{max}. Furthermore, the plots for $forward = 0$ indicate

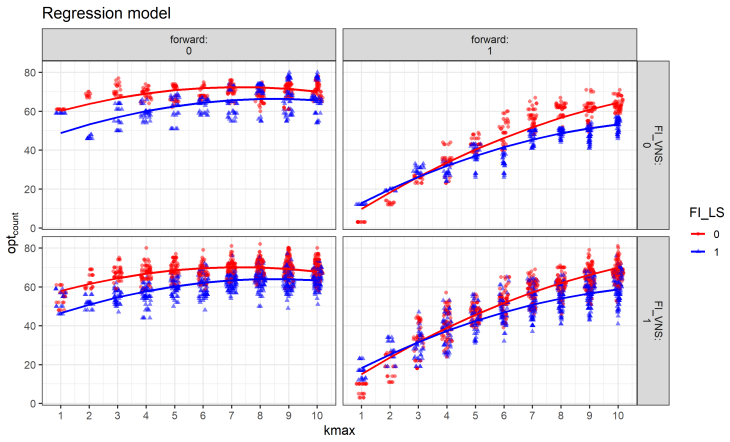

Fig. 3. Regression model of the VNS parameters (*visualize*() [10]).

that the stagnation phase is reached for larger k_{max} (which is the region that promises larger opt_{count}). Finally, FI-VNS and BI-VNS have demonstrated similar performance, however, we suspect that the differences increase for $k_{max} > 10$. To test this hypothesis, we generated additional data and graphics disclosed at `https://www.mi.sanu.ac.rs/~tatjana/conf/VNS-param-analysis/`. The plots show that BI-VNS demonstrated much better performance when compared to FI-VNS if combined with $forward = 0$ and FI-LS. In particular, it generated $opt_{count} > 90$ for $kmax \geq 20$.

Regardless of the small statistical influence reported in Tab. 1, the numerical parameters show practical impact (see Fig. 3). Data and plots may be found at the mentioned link. The results show that the parameter $p_{plateau}$ does not influence the performance in case of BI-VNS, however, it should be properly configured in case of FI-VNS. Parameter k_{step} is closely interconnected with VNS categorical parameters which means that the success of VNS variant depends on the values of k_{step} and k_{max}. Parameter `KMIN` has not shown an overall statistically significant influence, however, strategies `kminrand` and `kr2` exhibited noticeable impact on the results of FI-VNS. Due to their high nonlinear relationship, to achieve the best results it is necessary to conduct automatic tuning of k_{step},$p_{plateau}$,k_{min} via a software package such as iRace, ParamILS, SPO, etc [4].

5 Conclusion

This paper applies a multivariate regression model, random forests and visualization tools to study the effects of the VNS parameters on one problem instance. The BI-VNS algorithm is developed and compared against FI-VNS. We promote the application of methodological experimental evaluation to metaheuristic algorithms that aids in their objective performance analysis. Three phases of

the parameter analysis were followed: (a) recognition of algorithm's parameters; (b) detection of the most influential parameters; (c) data analysis of parameters' main and interaction effects. We were able to identify the most influential nonlinear relationships between the algorithm's parameters. In this paper we focused on the study of categorical parameters, and pointed out the main observations about the numerical parameters. The future work is to generalize the presented methodology to work with the larger optimization problem set.

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 Koza's computational effort. Ph.D. thesis (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.
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. IEEE Trans. Evol. Comput. **8**(4), 405–421 (2004)
7. 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)
8. Davidović, T., Liberti, L., Maculan, N., Mladenović, N.: Towards the optimal solution of the multiprocessor sheduling problem with communication delays. In: Proceedings of MISTA 2007. pp. 128–135. Paris, France (2007)
9. Eiben, A.E., Smit, S.K.: Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm Evol. Comput. **1**(1), 19–31 (2011)
10. Fife, D.: Flexplot: Graphically-based data analysis. Psychological Methods (2021)
11. Floudas, C.A., Pardalos, P.M: Encyclopedia of Optimization. Springer (2009)
12. Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S.: Variable neighborhood search: basics and variants. EURO J. Comput. Optim. **5**(3), 423–454 (2017)
13. Hooker, J.N.: Needed: An empirical science of algorithms. Oper. Res. **42**(2), 201–212 (1994)
14. 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. J. Oper. Res. Soc. **67**(4), 676–689 (2016)
15. McGeoch, C.C.: A guide to experimental algorithmics. CUP (2012)
16. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput Oper Res **24**(11), 1097–1100 (1997)
17. Papadimitriou, C.H., Yannakakis, M.: Towards an architecture-independent analysis of parallel algorithms. SIAM journal on computing **19**(2), 322–328 (1990)
18. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Austria (2021), `https://www.R-project.org/`
19. Talbi, E.G.: Metaheuristics: from design to implementation, vol. 74. (2009)