Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ORIGINAL ARTICLE

An adaptive large neighborhood search algorithm for robust rig routing†

Igor Kulachenko | Polina Kononova

[1]Sobolev Institute of Mathematics, Siberian Branch of the Russian Academy of Sciences, Novosibirsk, Russia

Correspondence
Igor Kulachenko.
Email: ink@math.nsc.ru
Polina Kononova.
Email: pkononova@math.nsc.ru

Abstract

In this research, we discuss the real-world Drilling Rig Routing Problem (DRRP) in an uncertain environment. There is a set of customers that are drilling sites requiring work on drilling wells. For each customer, we know the number of wells that need to be drilled within a given time interval, and it is allowed to partition the set of wells so that several drilling rigs perform the required work. The work on a well is unsplittable and must be performed by only one rig. A rig does not have to return to the depot it started from after it completes its work plan. This problem is close to an open uncapacitated multi-depot split delivery vehicle routing problem with time windows. The main difference with traditional SDVRP is that it is the service time that is split, not the demand, and there is only a limited number of studies on such problems.

Yet, in the real world, unforeseen circumstances can affect drilling time, and that, if disregarded, can lead to a disruption of the work plan. Thus, in this problem, we maximize possible deviations of drilling times from expected values when there is still a feasible solution to perform all well-drilling requests in time. At the same time, total traveling costs must be no more than a given threshold. It is a so-called threshold robustness approach.

This research proposes a MILP model for the DRRP with uncertainties and an Adaptive Large Neighborhood Search (ALNS) algorithm to solve it. The destruction operators used in the algorithm are working either at the route, customer, or visit level. The reconstruction operators consider the possibility of changing the work partition. Computational results for the algorithm and Gurobi solver show the dominance of the ALNS scheme for the medium-size instances. Experiments also show the factors that allow one to increase the robustness of a solution.

KEYWORDS:
logistics, threshold robustness, uncapacitated vehicles, split delivery, time windows, metaheuristics

1 | INTRODUCTION

Vehicle Routing Problems (VRP) are well-known NP-hard combinatorial optimization problems with a large number of various real-world applications Golden, Raghavan, and Wasil (2008); Toth and Vigo (2014).

The described problem is similar to the Split Delivery Vehicle Routing Problem with Time Windows (SDVRPTW), where a fleet of capacitated homogeneous vehicles has to serve a set of customers requiring service within a specific time interval. But contrary to the classical VRP, a customer can be visited by several vehicles Archetti and Speranza (2012); Ho and Haugland (2004). But usually, SDVRPs consider split deliveries concerning

†Funded by.

customer demand for goods, not service time. The fact that vehicles may not return to their initial location classifies the DRRP as Open VRP Li, Golden, and Wasil (2007). In Yakici and Karasakal (2013), an Uncapacitated Open SDVRP with splittable service demand was studied. In this problem, the maximum service completion time was minimized, and time windows were not considered. %%%%%% Add refs about threshold robustness.

The rest of this paper is structured as follows. We first introduce the mathematical model for the problem in Sect.2. Algorithm details are described in Sect.3. Computational results are discussed in Sect.4. The last Sect.5 concludes the paper.

2 | PROBLEM FORMULATION

We considered the drilling rig routing problem earlier in Kulachenko and Kononova (2021). In this problem, one has a set of drilling sites (hereinafter, sites or customers) and a fleet of uncapacitated drilling rigs (hereinafter, rigs or vehicles). The sites consist of a predetermined number of wells planned for drilling, and a rig travels from one site to another, performing drilling until it drills all assigned wells. The initial location for each rig is known, and it does not have to return to the initial location after it completes all the assigned work. Each customer has a time window that is a period of time during which all work on the site has to be started and completed. That means each customer has the earliest time when a rig can start drilling and the latest time when all wells have to be already drilled. To speed up the work on the site, one can make use of several drilling rigs. The entire set of wells of the site is to be distributed among the drilling rigs serving this site. The work of different drilling rigs on the same site does not depend on each other. A rig that has started work on a well completes it to the end. The objective is to determine a set of rig routes (including the numbers of assigned wells) to perform all well-drilling requests in time to minimize the total traveling distance. Further, we will consider only such instances for which it is known that the fleet of rigs can complete all the work in time.

Let the schedule for the rigs be already known, in which all the work is completed in time. This paper considers the problem of studying the solution robustness to uncertain initial data. In the real world, well drilling may take longer than initially thought. Such an increase in time can lead to going beyond the time window not only at the original site but also at other sites where further work will take place. So, we consider the drilling times to be uncertain and want solutions to be robust to these uncertainties. For this purpose, we find a solution that would be feasible even in the case when all the drilling times are multiplied by some real-value number. Consequently, what we want is to maximize this number to improve the robustness of a solution, but the total traveling cost should not exceed a certain threshold. As a threshold, we use increased minimum total traveling time found by the algorithms from our previous paper Kulachenko and Kononova (2021). It is called the threshold robustness approach in the literature, and this approach has not been applied to vehicle routing problems before [refs].

2.1 | Non-linear Model

Let K be the set of vehicles, each of which is initially located at v_k, that is its corresponding depot. Let us define complete oriented graph $G = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ is a set of vertices consisting of the set of depots $\mathcal{V}^{\mathrm{DEP}} = \{v_1, v_2, \ldots, v_{|K|}\}$ and the set of customers $\mathcal{V}^{\mathrm{CST}} = \{v_{|K|+1}, v_{|K|+2}, \ldots, v_{|K|+n}\}$, and $\mathcal{A}$ is a set of arcs. For convenience, we additionally define the set $\mathcal{V}_k = \mathcal{V}^{\mathrm{CST}} \cup \{v_k\}$ that is the set of vertices that can be visited by vehicle k. For each customer $i \in \mathcal{V}^{\mathrm{CST}}$, we know the number of wells w_i that is need to be drilled. All driiling work for the customer i has to be done in the time interval $[e_i, l_i]$. The work at the site i cannot be started before e_i, even if the vehicle has already arrived. The planning horizon is referred as $H = \max_{i \in \mathcal{V}^{\mathrm{CST}}} l_i$. Drilling any well at the site i performed by vehicle k requires d_{ik} days. For $i \in \mathcal{V}^{\mathrm{DEP}}$, we assume $d_{ik} = 0$.

For each arc $(i, j) \in \mathcal{A}$ the traveling time t_{ijk} from i to j for vehicle k is known. Some customers can be serviced only by certain vehicles, for example, due to geographical distance from roads. If vehicle k cannot travel from i to j, then we assume $t_{ijk} = H$. Since at the end of the planning horizon, the vehicles do not have to return to the depot but can remain at the last visited site, then we set the traveling times from all the sites to all depots to zero $t_{i,v_k,k} = 0$, $i \in \mathcal{V}^{\mathrm{CST}}$, $k \in K$. It allows us to consider a model with circular routes. We denote by T the value of the total traveling time that cannot be exceeded.

We introduce the following binary decision variables:

$$x_{ijk} = \begin{cases} 1, & \text{if vehicle } k \in K \text{ travels from vertix } i \text{ to vertix } j,\ j \in \mathcal{V}^k, \\ 0, & \text{otherwise,} \end{cases}$$

$$y_{ik} = \begin{cases} 1, & \text{if vehicle } k \in K \text{ visits the vertex } i \in \mathcal{V}^k, \\ 0, & \text{otherwise.} \end{cases}$$

The decision variables s_{ik}, $i \in \mathcal{V}_k$, $k \in K$ represent the time when vehicle k starts the work at site $i \in \mathcal{V}^{\mathrm{CST}}$. We assume $s_{ik} = 0$, if $i \in \mathcal{V}^{\mathrm{DEP}}$.

The decision variables $w_{ik} \in \mathbb{Z}_{\geq 0}$, $i \in \mathcal{V}_k$, $k \in K$ represent the total number of wells drilled by rig k at site i, or 0 in the case of $i = v_k$.

Let us move on to the description of variables and parameters associated with uncertainties. Imagine that now drilling a well at site i preformed by rig k requires not d_{ik}, but $(1+\varepsilon_{ik})d_{ik}$ days, where $\varepsilon_{ik} \geq 0$ are additional variables, $i \in \mathcal{V}^{\mathrm{CST}}$, $k \in K$, which set the maximum increase in drilling time when servicing site i performed by k.

Let $\alpha_i > 0$ be the weight coefficient for customer i, which regulates the importance of guaranteed robustness for the customer. Thus, using $\alpha_i < 1$, one can simulate a situation in which the acceptable increase in drilling time at site i is to be greater than on the j object with $\alpha_j = 1$. And for customers with $\alpha_i >> 1$, an increase in drilling time has little effect on the objective function.

It is necessary to find the maximum acceptable increase in drilling time ε_{ik} at each site so that all the work will be performed in time. The variable $\varepsilon_{\min} \geq 0$ corresponds to the minimum product of α_i and ε_{ik} for $i \in \mathcal{V}^{\mathrm{CST}}$, $k \in K$.

We formulate the problem as a problem of mixed integer quadratic programming:

$$\max \varepsilon_{\min} \tag{1}$$

$$\sum_{k\in K}\sum_{i\in\mathcal{V}_k}\sum_{j\in\mathcal{V}_k} t_{ijk}x_{ijk} \leq T, \tag{2}$$

$$\varepsilon_{\min} \leq \alpha_i\varepsilon_{ik}, \;\; i \in \mathcal{V}^{\mathrm{CST}}, k \in K, \tag{3}$$

$$y_{v_k k} = 1, \;\; k \in K, \tag{4}$$

$$s_{v_k,k} = 0, \;\; w_{v_k,k} = 0, \;\; k \in K, \tag{5}$$

$$\sum\nolimits_{j\in\mathcal{V}_k} x_{ijk} = \sum\nolimits_{j\in\mathcal{V}_k} x_{jik} = y_{ik}, \;\; i \in \mathcal{V}_k, k \in K, \tag{6}$$

$$\sum\nolimits_{k\in K} w_{ik} = w_i, \;\; i \in \mathcal{V}^{\mathrm{CST}}, \tag{7}$$

$$w_{ik} \geq y_{ik}, \;\; i \in \mathcal{V}^{\mathrm{CST}}, k \in K, \tag{8}$$

$$w_{ik} \leq w_i y_{ik}, \;\; i \in \mathcal{V}^{\mathrm{CST}}, k \in K, \tag{9}$$

$$s_{ik} \geq e_i, \;\; i \in \mathcal{V}^{\mathrm{CST}}, k \in K, \tag{10}$$

$$s_{ik} + (1+\varepsilon_{ik})d_{ik}w_{ik} \leq l_i, \;\; i \in \mathcal{V}^{\mathrm{CST}}, k \in K, \tag{11}$$

$$s_{ik} + (1+\varepsilon_{ik})d_{ik}w_{ik} + t_{ijk} - s_{jk} \leq \Delta_{ijk}(1-x_{ijk}), \;\; i \in \mathcal{V}_k, j \in \mathcal{V}^{\mathrm{CST}}, k \in K, \tag{12}$$

$$x_{ijk},\; y_{ik} \in \{0,1\},\; \varepsilon_{ik}, \varepsilon_{\min} \in [0, M_\varepsilon],\; s_{ik}, w_{ik} \in \mathbb{Z}_{\geq 0}, \;\; i,j \in \mathcal{V}_k, k \in K. \tag{13}$$

The objective function (1) maximizes the robustness for all the sites when taking into account the weight coefficients α_i. Constraint (2) makes sure that the threshold on the total traveling time will not be exceeded. Inequalities (3) set value for $\varepsilon_{\min}$. Inequalities (4)–(5) set the distribution of vehicles by their depots and prohibit them from visiting other ones. Constraints (6) set circular routes from the depot. Inequalities (7) ensure that all the wells will be drilled at the site. Auxiliary inequalities (8) show that if a vehicle visits the site, it drills at least one well there. And inequalities (9) forbid vehicles not visiting the site to drill wells there. Inequalities (10)–(11) guarantee that all the drilling work will be done in correspondence with time windows Inequalities (12), where $\Delta_{ijk} = \max\{l_i + t_{ijk} - e_j, 0\}$ are constants, set the service starting time and prohibit a vehicle from visiting one site several time. Constraints (13) define the variable types, M_ε are constants that limit the scope of variables ε_{ik}, $\varepsilon_{\min}$.

2.2 | Linearization

The constraints (11)–(12) are non-linear, but they can be linearized by introduction of additional variables: binary variables w_{ikj} and continuous ones ξ_{ikj} and ξ_{ik}. The latter are the equivalent of the products $\varepsilon_{ik}w_{ikj}$ and $\varepsilon_{ik}w_{ik}$ respectively.

$$w_{ik} = \sum\nolimits_{j=1}^{w_i} j w_{ikj}, \;\; i \in \mathcal{V}^{\mathrm{CST}}, k \in K, \tag{14}$$

$$\sum\nolimits_{j=1}^{w_i} w_{ikj} \leq 1, \;\; i \in \mathcal{V}^{\mathrm{CST}}, k \in K, \tag{15}$$

$$\xi_{ik} = \sum\nolimits_{j=1}^{w_i} \xi_{ikj}, \;\; i \in \mathcal{V}^{\mathrm{CST}}, k \in K, \tag{16}$$

$$j\varepsilon_{ik} - \xi_{ikj} \leq M_\varepsilon w_i(1-w_{ikj}), \quad i \in \mathcal{V}^{\mathrm{CST}}, k \in K, j \in \{1,\ldots,w_i\}, \tag{17}$$

$$\xi_{ikj} \leq M_\varepsilon w_i w_{ikj}, \quad i \in \mathcal{V}^{\mathrm{CST}}, k \in K, j \in \{1,\ldots,w_i\}, \tag{18}$$

$$\xi_{ikj} \leq j\varepsilon_{ik}, \quad i \in \mathcal{V}^{\mathrm{CST}}, k \in K, j \in \{1,\ldots,w_i\}, \tag{19}$$

$$s_{ik} + d_{ik}(w_{ik} + \xi_{ik}) \leq l_i, \;\; i \in \mathcal{V}^{\mathrm{CST}}, k \in K, \tag{20}$$

$$s_{ik} + d_{ik}(w_{ik} + \xi_{ik}) + t_{ijk} - s_{jk} \leq \Delta_{ijk}(1-x_{ijk}), \;\; i \in \mathcal{V}_k, j \in \mathcal{V}^{\mathrm{CST}}, k \in K, \tag{21}$$

$$x_{ijk},\; y_{ik} \in \{0,1\}, \varepsilon_{ik}, \varepsilon_{\min} \in [0, M_\varepsilon],\; s_{ik}, w_{ik} \in \mathbb{Z}_{\geq 0}, \;\; (i,j) \in \mathcal{V}_k^2, k \in K, \tag{22}$$

$$w_{ikj} \in \{0,1\},\; \xi_{ik}, \xi_{ikj} \in [0, M_\varepsilon w_i], \;\; i \in \mathcal{V}_k, k \in K, j \in \{1,\ldots,w_i\}. \tag{23}$$

Now, (1)–(10), (14)–(23) is the mixed-integer linear programming model for the problem under consideration.

3 | ALGORITHM

We use an Adaptive Large Neighborhood Search scheme to solve the problem.

3.1 | Solution Representation

Solution σ of the problem consists of individual vehicle routes $\sigma = \{\sigma_1, \dots, \sigma_{|\mathsf{K}|}\}$. The route of vehicle k can be represented as the sequence $\sigma_\mathsf{k} = (\sigma_0^\mathsf{k}, \sigma_1^\mathsf{k}, \dots, \sigma_{\mathsf{u}_\mathsf{k}}^\mathsf{k})$, where $\sigma_\mathsf{u}^\mathsf{k}$ is the pair $(\mathsf{i}_\mathsf{u}^\mathsf{k}, \mathsf{w}_{\mathsf{uk}})$, $\mathsf{i}_\mathsf{u}^\mathsf{k}$ is the u^{th} visited site, and w_{uk} is the number of wells that vehicle k drills at that site. Each route begins from a depot $\mathsf{i}_0^\mathsf{k} = \mathsf{v}_\mathsf{k}$, and the includes only customer visits: $\mathsf{i}_\mathsf{u}^\mathsf{k} \in \mathcal{V}^{\mathrm{CST}}$ for $\mathsf{u} > 0$.

We refer to solution σ as *feasible*, if given $\varepsilon_{\mathsf{ik}} = 0$, all the work can be performed in time according to vehicle routes provided.

3.2 | Objective Function Calculation

Objective function (1) landscape has large plateaus with non-smooth transitions between them. Such a landscape makes it difficult for local search methods to work. Many different solutions have the same value of the objective function. In order to introduce additional diversity, we will track the possible acceptable increase in drilling time separately for each of the sites and separately for each of the vehicle routes.

We introduce new notation and add additional terms to the objective function. Let

$$\varepsilon_i^{\mathrm{cst}} = \min\{\alpha_i \varepsilon_{ik} | (i, w_{uk}) \in \sigma_k, w_{uk} > 0\}$$

be the maximum allowable increase in drilling time at the site i for each vehicle operating at this site. We denote by $\varepsilon_\mathsf{k}^{\mathrm{veh}}$ the maximum acceptable increase in drilling time during the operation of the vehicle k at all the sites it visits,

$$\varepsilon_k^{\mathrm{veh}} = \min\{\alpha_i \varepsilon_{ik} | (i, w_{uk}) \in \sigma_k, w_{uk} > 0\}.$$

We assume that $\varepsilon_\mathsf{k}^{\mathrm{veh}} = 0$ for those vehicles that do not perform any work.

We add to the objective function a linear convolution of $\varepsilon_{\min}$, the average possible increase in time for the sites and for the routes with coefficients $\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3$. The coefficients are set so that the main goal is to maximize $\varepsilon_{\min}$. New terms make it possible to make the landscape of the objective function more diverse, and their maximization does not conflict with the primary goal.

A popular technique in local search algorithms is to leave the domain of feasible solutions in an attempt to find a better feasible domain. Let us assume that we consider solutions σ that are feasible with respect to time windows, but the total traveling time for them may exceed the specified threshold T. The excess of the threshold is introduced into the objective function with penalty λ. We get a new objective function $\mathsf{f}_\mathsf{T}(\sigma)$, which we will maximize.

$$\begin{aligned} \mathsf{f}_\mathsf{T}(\sigma) = & \mathsf{C}_1 \varepsilon_{\min} + \frac{\mathsf{C}_2}{|\mathsf{K}|} \sum_{\mathsf{k} \in \mathsf{K}} \varepsilon_\mathsf{k}^{\mathrm{veh}} + \frac{\mathsf{C}_3}{|\mathcal{V}^{\mathrm{CST}}|} \sum_{\mathsf{i} \in \mathcal{V}^{\mathrm{CST}}} \varepsilon_\mathsf{i}^{\mathrm{cst}} + \\ & \lambda \cdot \max\left\{0, \sum_{\mathsf{k} \in \mathsf{K}} \sum_{\mathsf{i} \in \mathcal{V}_\mathsf{k}} \sum_{\mathsf{j} \in \mathcal{V}_\mathsf{k}} \mathsf{t}_{\mathsf{ijk}} \mathsf{x}_{\mathsf{ijk}} - \mathsf{T}\right\} \end{aligned} \tag{24}$$

Calculations in local search are carried out with respect to the objective function $\mathsf{f}_\mathsf{T}(\sigma)$, while in tables for the section of computational experiments, only $\varepsilon_{\min}$ values assuming the T threshold is not exceeded.

Let us describe the procedure for calculating the objective function $\mathsf{f}_\mathsf{T}(\sigma)$. Let σ be some admissible solution. First, $\varepsilon_\mathsf{k}^{\mathrm{veh}}$ is calculated independently for each vehicle k. Then, $\varepsilon_\mathsf{i}^{\mathrm{cst}}$ and $\varepsilon_{\min}$ are determined from them, after that the value of $\mathsf{f}_\mathsf{T}(\sigma)$ is calculated.

Let $\sigma_\mathsf{k} = (\sigma_0^\mathsf{k}, \sigma_1^\mathsf{k}, \dots, \sigma_{\mathsf{k}_\mathsf{n}}^\mathsf{k})$ is the route of vehicle k consisting of k_n objects. We renumber all customers in the order of visits by vehicle k. In what follows, we refer to the i-th visited customer of the vehicle k in the schedule σ_k as customer i. At the i-th site, vehicle k performs work on w_{ik} wells.

Let us construct a *early* schedule for the vehicle k. Let denote by $\mathsf{e}_\mathsf{i}^\mathsf{k}$ the shortest start time of work at the i-th site, it cannot be less than the beginning of the time window $[\mathsf{e}_\mathsf{i}, \mathsf{l}_\mathsf{i}]$ and earlier than the arrival time of the vehicle k after the work is completed at the previous site:

$$e_0^k = 0, w_{0k} = 0, d_{0k} = 0,$$

$$e_i^k = \max\left\{e_i, e_{i-1}^k + d_{i-1,k} w_{i-1,k} + t_{i-1,ik}\right\}, i \in \{1, \dots, k_n\}.$$

Let us construct a *late* schedule for the vehicle k. Let denote by $\mathsf{l}_\mathsf{i}^\mathsf{k}$ the latest starting time of work at the i-th site, so that work at each sites is completed in time. This value depends on the end time of the $[\mathsf{e}_\mathsf{i}, \mathsf{l}_\mathsf{i}]$ time window and work at the next site, so it is built from the end:

$$l_{k_n}^k = l_{k_n} - d_{k_n,k} w_{k_n,k},$$

$$l_{i-1}^k = \min\left\{l_{i-1}, l_i^k - t_{i-1,i,k}\right\} - d_{i-1,k}w_{i-1,k}, i \in \{1,\ldots,k_n\}.$$

Since σ is a feasible solution, it is true that $e_i^k \leq l_i^k$ for each i.

We will calculate the maximum $\varepsilon_k^{\text{veh}}$ for the entire route. The upper bound on the value of $\varepsilon_k^{\text{veh}}$ is:

$$\text{UB}_k = \min_{i\in\{1,\ldots,k_n\}}\left\{\alpha_i \frac{l_i^k - e_i^k}{d_{ik}w_{ik}}\right\}. \tag{25}$$

We apply the dichotomy method by calculating the value of $\tilde{\varepsilon}$. Initially

$$\tilde{\varepsilon} = \frac{UB_k}{2}.$$

At each step, we examine if the schedule is feasible for the value $\varepsilon_{ik} = \frac{\tilde{\varepsilon}}{\alpha_i}$. Recall that the schedule for vehicle k is feasible if all the work is completed in time for the time spent at the site $(1+\varepsilon_{ik})d_{ik}w_{ik}$. Depending on whether the schedule is feasible, the left or right border is shifted, dividing the interval in half. The process stops when there is at most one ε value between the current upper and lower bounds such that the schedule for vehicle k is feasible and $\frac{\varepsilon}{\alpha_i}d_{ik}w_{ik} \in \mathbb{Z}$ for at least one $i \in \{1,\ldots,k_n\}$.

As a result, the required value of $\varepsilon_k^{\text{veh}}$ can be calculated as

$$\varepsilon_k^{\text{veh}} = \min_{i\in\{1,\ldots,k_n\}}\{\varepsilon : \frac{\varepsilon}{\alpha_i}d_{ik}w_{ik} \in \mathbb{Z},\ \varepsilon \geq \tilde{\varepsilon}\}.$$

3.3 | Algorithm

An Adaptive Large Neighborhood Search (ALNS) algorithm uses the ruin-and-recreate principle when we first remove some parts from the current solution and then try to insert removed parts in a better way. In the Large Neighborhood Search (LNS) algorithm, the so-called destroy and reconstruction operators are used for this purpose. Unlike LNS, the adaptive version of the algorithm uses several destruction and reconstruction operators, depending on their successful application in the previous steps. The algorithm 1 presents the pseudocode for the ALNS scheme used in this work. The algorithm takes as an input the initial solution σ. The greedy heuristic from Kulachenko and Kononova (2021) is used to build the initial solution. At each iteration of the algorithm, the destruction and reconstruction operators are probabilistically selected based on their previous success (step 6), after which they are applied in turn (step 7). If the acceptance criterion is met for the obtained solution, then we transit to this solution (step 9) and, if necessary, update the best-found solution (step 11). After performing I successive iterations of choosing and applying operators, the procedure for updating their weights is performed. The procedure takes into account how successful the application of these operators was during the given segment (step 15). The algorithm continues its work until the stop criterion (step 4) is met.

Algorithm 1 Adaptive Large Neighborhood Search

```
1:  function ALNS(σ)
2:      σ* ← σ
3:      Initialize weights
4:      while the stopping criterion is not met do
5:          for I iterations do
6:              probabilistically select a destruction operator and a reconstruction operator based on their current weights
7:              apply the destruction and reconstruction operators to σ to obtain σ'
8:              if σ' satisfies the acceptance criterion then
9:                  σ ← σ'
10:                 if f(σ*) < f(σ') then
11:                     σ* ← σ'
12:                 end if
13:             end if
14:         end for
15:         Adjust weights
16:     end while
17:     return σ*
18: end function
```

Now, let us now define the destruction and reconstruction operators for a feasible solution σ. In this section, individual site visits from different routes will be considered. If for the application of the operator it is important from which route the visit is taken, then we denote this visit by (i, w_{uk}, k). This notation means that the vehicle k visits the u-th in order site i and drills w_{uk} wells at it, and the work at the site by vehicle k starts at time s_{uk}. If for the application of the operator, it does not matter which route the visit belonged to earlier, then we denote by (i, w) the visit for which it is necessary to arrive at the site i and drill w wells.

3.4 | Destruction operators

Several levels are considered for the application of destruction operators: removal of individual customer visits, removal of all visits for individual customers, removal of all customer visits for separate routes. These operators are described in the subsections below.

3.4.1 | Removal of individual customer visits

These operators remove n_1 individual customer visits from the current solution. In our experiments, we generate a random number from the range $0.05n_{\text{visits}}$ to $0.3n_{\text{visits}}$ at each iteration and pass this value as argument n_1 for the operator. Value of n_{visits} can be different at different iterations, and it is equal to the total number of customer visits throughout the current solution. We may remove all visits corresponding to a customer, as well as we may remove only a part of these visits.

Random visit removal. With this approach, visits for removal are selected randomly and independently of each other throughout the removal process.

Related visit removal. The mechanism of this operator is described in the algorithm 2. We start with the removal of a randomly selected visit from the schedule. Then, the selection of the visit to remove later is based on its proximity to the already removed visits. The proximity measure between two visits (i, w_{uk}, k) and $(j, w_{u'k'}, k')$ is based on the geographic distance between customers, and the difference between service start times in the schedule before application of the destruction operator $|s_{u'k'} - s_{uk}| + \alpha_v t_{ij}$. The contribution of these two quantities is controlled by the α_v parameter. For a geographic distance that does not depend on the choice of vehicle, we introduce a new notation $t_{ij} = \min_{k \in K} t_{ijk}$. The d parameter controls the tendency to choose closer visits to those that were already removed. For large values of the d parameter, we tend to remove visits closest to those removed in previous iterations. Lower values of this parameter allow some level of randomness in selecting visits for removal.

Algorithm 2 Related visit removal

Parameters: α_v, d
function RELATED_VISIT_REMOVAL(σ, n_1)
  Randomly select customer visit (j, w_{uk}, k), $j \in \mathcal{V}^{CST}$, and remove it from the current solution σ
  $L \leftarrow \{(j, w_{uk}, k)\}$
  while $|L| < n_1$ **do**
    Randomly select customer visit $(i, w_{u'k'}, k') \in L$
    for each visit (j, w_{uk}, k), $j \in \mathcal{V}^{CST}$ from σ **do**
      $z_{uk} \leftarrow |s_{u'k'} - s_{uk}| + \alpha_v t_{ij}$
    end for
    Sort the z_{uk}'s in non decreasing order and put them in list B
    Choose a random number $x \in (0, 1)$
    $pos \leftarrow \lceil |B| \cdot x^d \rceil$
    Select visit $(j^*, w_{u^*k^*}, k^*) \in B$ associated with the z_{uk} value at position pos in B, and remove it from the solution σ
    $L \leftarrow L \cup \{(j^*, w_{u^*k}, k^*)\}$
  end while
  return L
end function

3.4.2 | Removal of individual customers

These operators remove all visits of some n_2 customers from the current solution. In our experiments, we generate a random number from the range $0.05|\mathcal{V}^{CST}|$ to $0.3|\mathcal{V}^{CST}|$ at each iteration and pass this value as argument n_2 for the operator.

Random removal of customers. In this approach, customers are selected randomly and independently of each other throughout the removal process.

Related customer removal. This operator is very similar to the one described in the 2 algorithm, except that instead of individual visits, customers are now selected and all visits are removed, in all routes they belong to. For two customers i and j we define the proximity measure z_{ij} as $\max\left(0, e_j - l_i, e_i - l_j\right) + \alpha_c t_{ij}$.

3.4.3 | Removal of individual routes

These operators remove all visits for some n_3 routes from the current solution. In our experiments, we generate a random number from the range $0.05|K|$ to $0.3|K|$ at each iteration and pass this value as argument n_2 for the operator.

Random removal of routes. In this approach, the routes for removal are chosen randomly and independently of each other throughout the removal process.

Related route removal. This operator is similar to the one described in the algorithm 2, but routes are selected for removal. Let's define *center of gravity* of route σ_k as the average location of all customers from the route. For any two routes σ_k and $\sigma_{k'}$, we define two different proximity measures. In the GC measure, the value $z_{kk'}$ is defined as the traveling time from the center of gravity of one route to the center of another. For the measure MinD, the value $z_{kk'}$ is calculated as the minimum of the distances between customers i_u^k and $i_u^{k'}$ from the routes σ_k and $\sigma_{k'}$ respectively.

3.5 | Reconstruction operators

After applying the destruction operators and removing some visits from the current solution, it is necessary to reconstruct the solution to perform all the necessary work. At the same time, the distribution of work between different vehicles at one site can either remain the same as it was before applying the operators or change. Thus, two types of solution reconstruction operators are considered. In the first one, the operator looks for the best suitable place to insert a removed visit (i, w). The second operator can divide visit (i, w) into several and find a suitable place for each of them. In doing so, the distribution of work between different vehicles at one site may change, but we still will perform all the requested drilling work.

It should be noted that in the process of applying the above-described reconstruction operators, solutions can be obtained with multiple visits to one customer in one route. The ability to return to an already visited customer sometimes allows one to get more robust solutions.

To reduce the computation time and employ some diversification in the local search, we evaluate only 50% of possible insertion places for all the following operators.

3.5.1 | Greedy heuristic without the work repartition

For this approach, all visits have the same work distribution as it was before applying the operators. The pseudocode describing the mechanism of the operator is given in the algorithm 3. Let L be a list of visits (i, w) that is not inserted yet, and σ be a partial solution containing routes with other visits. At each step of the algorithm, a subset of visits B from L is randomly selected, and then for each of them the cost of insertion into each of the σ_k routes is calculated. Let the solution σ^{ku} be obtained from the solution σ by inserting the visit (i, w) into the route σ_k after visiting $(j, w_{uk}) \in \sigma_k$. Then the insertion cost is defined as

$$c(\sigma, \sigma^{ku}) = (f_T(\sigma) - f_T(\sigma^{ku})) + \alpha_{ins}(T(\sigma^{ku}) - T(\sigma)),$$

where $T(\sigma)$ is the total traveling time of all vehicles in the solution σ. At the step 12, only one visit with the lowest insertion cost is inserted into the solution σ in the corresponding best place. Note that to insert a visit (i, w) into the route of some vehicle $k \in K$, we consider only places that fit in the time window $[e_i, l_i]$ and preserve the feasibility of the solution. The procedure ends when all the visits from the list L have been inserted into the solution σ. If there is no feasible solution, the original solution is returned.

3.5.2 | Regret-based heuristic without the work repartition

The algorithm for this heuristic is similar to that described in the previous subsection. The difference is in the visit insertion cost function. Let the cost of the best insertion be $c(\sigma, \sigma^{ku})$ for a visit (i, w), and $c(\sigma, \sigma^{k'u'})$ for the second best place, $k \neq k'$. Let us introduce the function $regret_{iw} = c(\sigma, \sigma^{k'u'}) - c(\sigma, \sigma^{ku})$ for customer visit (i, w). Then visit (i^*, w^*) with the largest value of $regret_{i^*w^*}$ is selected for the insertion. The idea behind this approach is to insert those visits that, when inserted in later steps, contribute the most to the added cost of the route.

In the regret-based operator, we can also use the generalized regret measure that is calculated as follows:

$$gen\_regret_{iw} = \sum_{k' \in K} c(\sigma, \sigma^{k'u'}) - c(\sigma, \sigma^{ku}),$$

Algorithm 3 Least-cost visit insertion

Parameters: α_{ins}, b
procedure LEAST-COST_INSERTION(σ, L)
  while L $\neq \emptyset$ **do**
    Randomly select $\min(|\mathsf{L}|, \mathsf{b})$ visits B $\subseteq$ L
    for visit $(\mathsf{i}, \mathsf{w}) \in \mathsf{B}$ **do**
      for k $\in$ K and suitable visits $(\mathsf{j}, \mathsf{w}_{\mathsf{uk}}) \in \sigma_{\mathsf{k}}$ **do**
        $\sigma' \leftarrow \sigma$
        Insert visit (i, w) into solution σ' into route of vehicle k after visiting its u-th customer
        Calculate the insertion cost $\mathsf{c}(\sigma, \sigma')$
      end for
    end for
    Apply the insertion of visit $(\mathsf{i}^*, \mathsf{w}^*) \in \mathsf{B}$ into route σ_{k^*}, which correspond to the minimal insertion cost
    if there are two consecutive customer visits i^* **then**
      Merge two consecutive visits into one
    end if
    L $\leftarrow$ L $\setminus \{(\mathsf{i}^*, \mathsf{w}^*)\}$
  end while
end procedure

where cost $\mathsf{c}(\sigma, \sigma^{\mathsf{k}'\mathsf{u}'})$ corresponds to the best insertion place for vehicle $\mathsf{k}' \in \mathsf{K}$. If customer i cannot be served by vehicle k′, then the cost function $\mathsf{c}(\sigma, \sigma^{\mathsf{k}'\mathsf{u}'})$ is assumed to be equal to 1000.

3.5.3 | Random repartition greedy heuristic

This heuristic is the same as the one described in the section 3.5.1 aside from the repartition procedure for list L performed before insertion of the customer visits. The repartion procedure for customer i is applied for the set of all uninserted visits for this customer. This procedure is applied with probability $\mathsf{r}_{\text{shake}}$ for each customer, and its idea is the following. First, with probability $\mathsf{r}^{\text{split}}_{\text{shake}}$, the new fictional customer visit $(\mathsf{i}, 0)$ is added to list L. Then, we change the partition of the drilling work for the uninserted visits of customer i. While doing that the number of wells served cannot change by more than k for each visit. In our experiments,unless otherwise stated, we use the following values for the parameters: $\mathsf{r}_{\text{shake}} = 0.5$, $\mathsf{r}^{\text{split}}_{\text{shake}} = 0.1$, $\mathsf{k} = 3$.

3.5.4 | Greedy heuristic with the work repartition

This approach allows one to change the work partition for customer visits. One visit is divided into two added to different routes. The pseudo-code of the operator is given in the algorithm 4. Let (i, w) be a random visit from the list L, containing w wells at site i. Let us define a set of vehicle $\mathsf{K}^{\mathsf{i}} \in \mathsf{K}$ that we consider for performing drilling at the site i. The set K^{i} contains all the vehicles that performed work at the site i before the application of destruction operators. In addition, no more than three vehicles are to be added to it, which could visit the customer i with a minimal detour when inserted into places suitable for time windows in the partial solution σ. The $\mathsf{p} - 1$ variant of dividing the set of wells of this visit into two subsets is considered:

$$\left\{\left(\lceil w/p \rceil, w - \lceil w/p \rceil\right), \left(\lceil 2w/p \rceil, w - \lceil 2w/p \rceil\right), \dots\right\},$$

where p is an algorithm parameter. For each partition, all suitable places for a pair of vehicles from the set K^{i} are checked and the best insertion is chosen.

3.6 | Acceptance criterion

The acceptance criterion (step 9 of the algorithm 1) is the one used in simulated annealing Kirkpatrick, Gelatt, and Vecchi (1983). That is, the new solution σ' is accepted instead of the current solution σ if $\mathsf{f}_{\mathrm{T}}(\sigma') \geq \mathsf{f}_{\mathrm{T}}(\sigma)$, otherwise the decision σ' is accepted with probability

$$\exp \frac{f_{\mathrm{T}}(\sigma) - f_{\mathrm{T}}(\sigma')}{\tau + \tau_{\text{offset}}},$$

Algorithm 4 Least-cost visit insertion with the work repartition

Parameters: p, α_{ins}
procedure REPARTITION_INSERTION(σ, L)
  while $L \neq \emptyset$ **do**
    Randomly select customer visit $(i, w) \in L$
    for $q \in \{0, 1, \ldots, p\}$ **do**
      $w' \leftarrow \lceil qw/p \rceil$
      for a pair of suitable visits $(j_1, w_{u_1 k_1}), (j_2, w_{u_2 k_2}) \in \sigma, k_1, k_2 \in K^i$ **do**
        $\sigma' \leftarrow \sigma$
        Insert visit (i, w') into solution σ' into route of vehicle k_1 after visiting its u_1-th customer
        Insert visit $(i, w - w')$ into solution σ' into route of vehicle k_2 after visiting its u_2-th customer
        Calculate the insertion cost $c(\sigma, \sigma')$
      end for
    end for
    Apply the best found pair of insertion
    if there are two consecutive customer visits i **then**
      Merge two consecutive visits into one
    end if
    $L \leftarrow L \setminus \{(i, w)\}$
  end while
end procedure

where τ is the current temperature value. The temperature is lowered at the 15 step of the algorithm 1 according to the formula $\tau = \alpha\tau$, where $\alpha \in (0, 1)$ is the parameter responsible for the rate of temperature decrease. The additional term $\tau_{\text{offset}} > 0$ is responsible for the offset of the temperature value τ, in order to avoid equating the temperature to zero. In our experiments, the value $\tau_{\text{offset}} = 10^{-8}$ is used. Every $I^{\text{sa}}_{\text{restart}}$ iterations without improving the value of best found $\varepsilon_{\min}$, the temperature is reset to the initial value. To employ intensification in the local search we also decrease the ranges for the values of n_1, n_2, n_3 by 3 times after $I^{\text{sa}}_{\text{restart}}/2$ iterations without improving the value of best found $\varepsilon_{\min}$. We stop this decreasing as soon as temperature is reset.

3.7 | Adaptive mechanism

In total, we use seven destruction and five reconstruction operators. At each iteration of the ALNS algorithm, the destruction and reconstruction operators are selected and applied. For the choice not to be random but to take into account the previous success of the applied operators, a weight is introduced for each of them, which changes from time to time (step 15 of the algorithm 1). In our experiments, we use the adaptive mechanism described in [Azi 2014].

4 | COMPUTATIONAL EXPERIMENTS

The described VNS algorithm is implemented in `C++` with `MSVC++ 14.16` compiler using standard release options. Version `9.5.0` of `Gurobi` solver is used for comparison purposes. The instances previously generated by us in Kulachenko and Kononova (2021) were used to test our algorithm. These instances had to be modified to be used. First, we need to relax time windows to allow some freedom. In our experiments, we use multiplier 2.5 for the instances with 150 customers and 1.5 for all the others. Next, to get some estimates of what the threshold T is equal to, we need to calculate the (sub)optimal value of the traveling time for the problem without uncertainties, i.e., with $\varepsilon_{ik} = 0, \forall i \in \mathcal{V}_k, k \in K$. This value is referred to as T^* hereinafter. When it is not specifically stated what the threshold T is, then we assume that it is equal to $1.5T^*$. We consider the instance sets of three different dimensions: 20 customers and 3 vehicles, 50 customers and 6 vehicles, 150 customers and 12 (or 18, more details in the section 4.3) vehicles. The instances with 50 and 150 customers correspond to instance sets S1, S2, S3 from Kulachenko and Kononova (2021). The small-sized instances with 20 customers were additionally generated in a similar manner to have the possibility to compare the results of the algorithm with optimal values.

TABLE 1 The results for instances with 20 customers.

	Gurobi		3 sec		10 sec		30 sec		60 sec		100 sec	
#	BF	UB	avg	best	avg	best	avg	best	avg	best	avg	best
S1.1	0.65	0.65	**0.65**	100								
S1.2	1.2	1.2	1.1	29	1.1	40	1.2	63	1.2	76	1.2	90
S1.3	0.05	0.05	**0.05**	100								
S1.4	0.9	1.2	0.85	28	0.88	69	0.88	73	0.89	82	0.89	88
S1.5	0.75	1	0.67	9	0.71	35	0.73	48	0.73	58	0.74	75
S2.1	0.1	0.1	**0.1**	100								
S2.2	0.42	0.42	0.41	77	0.41	86	**0.42**	100				
S2.3	0.6	0.6	**0.6**	100								
S2.4	0.1	0.1	**0.1**	100								
S2.5	1	1	0.94	50	0.98	70	0.99	91	1	96	**1**	100
S3.1	0.6	0.6	**0.6**	100								
S3.2	0.1	0.1	**0.1**	100								
S3.3	1.2	1.2	**1.2**	100								
S3.4	0.47	0.47	**0.47**	100								
S3.5	0.33	0.33	**0.33**	100								

4.1 | Small-sized instances

All experiments in this subsection are conducted on a computer with an `AMD Ryzen 5 2600` 3.9 GHz processor and 32 GB of RAM running under `Microsoft Windows 10` (64-bit).

First, we consider small-sized instances for which the optimal values can be obtained. Table 1 shows the computational results for these instances. Gurobi was limited to 1 hour running, although it is worth noting that Gurobi required no more than a minute for almost every instance. BF column corresponds to the best-found solution, and UB column corresponds to the best bound found. The remaining columns correspond to the implemented ALNS algorithm. It was run 100 times per instance. Columns 'avg' correspond to the average result for 100 runs, and columns 'best' correspond to the number of times the algorithm has found a solution with the BF column objective value. The average values are shown in bold if it is the least time limit configuration when the algorithm found only best-value solutions. The results are omitted if they are the same as those found in the previous time limit configuration. The table reflects the ability of the algorithm implemented to find optimal-value solutions in a reasonable time.

To demonstrate how the threshold level influences the maximum level of robustness, we can get experiments on small-sized instances with different T values are carried out. Figure 1 illustrates how the value of the objective function depends on T. Bigger-sized instances have a similar dependency. Table 2 demonstrates how frequently we achieve a solution with the best found objective value with threshold level being different. We perform 100 runs for each instance, and the time limit is 30 seconds. The results equal to 100 are omitted. Columns 'min' and 'avg' show to the minimum and average value for the results in the corresponding row. Row 'avg' shows the average value for the corresponding T value.

4.2 | Medium-sized instances

All experiments in this subsection are conducted on a computer with an `Intel Core i7-9700` 3.0 GHz processor and 16 GB of RAM running under `Microsoft Windows 10` (64-bit).

For medium-sized instances, Gurobi is able to find optimal solutions in a reasonable time only for a few instances. And finding a good-quality solution often requires much more time for it. Table 3 shows the results for these instances with runtime limit being different both for the ALNS and Gurobi. Since the solutions found by Gurobi can be quite different on these instances depending on what seed is taken, we decide to run it five times for each instance and time limit (except 5 hours), and the table includes the corresponding results. The ALNS algorithm was run 10 times

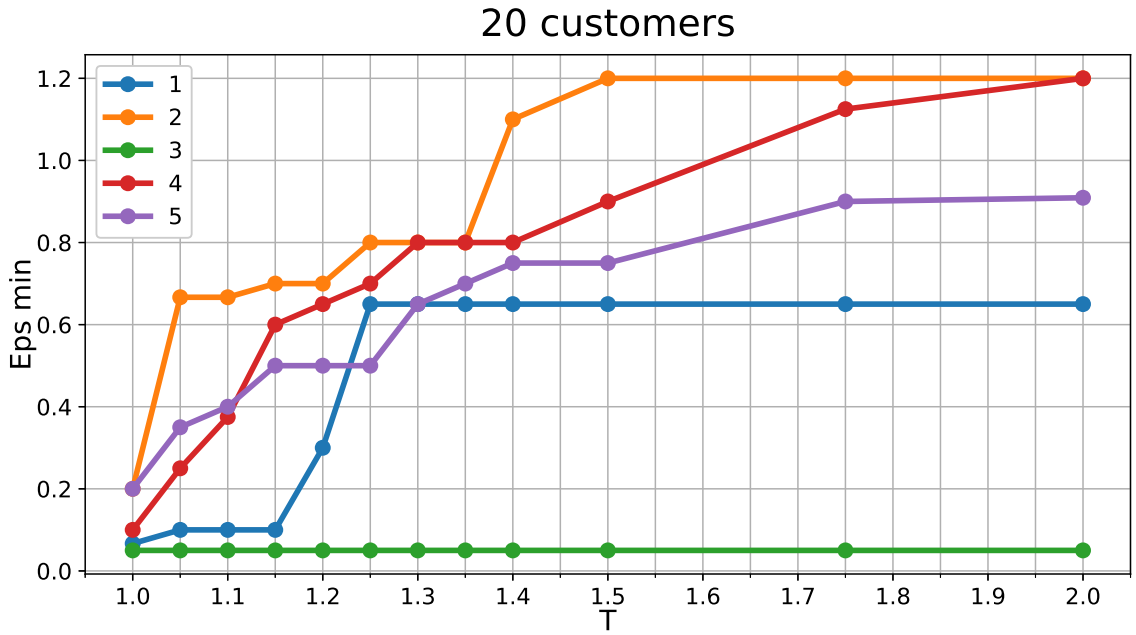

FIGURE 1 Different T.

TABLE 2 The results for instances with 20 customers.

T / #	1T*	1.05T*	1.1T*	1.15T*	1.2T*	1.25T*	1.3T*	1.35T*	1.4T*	1.5T*	1.75T*	2T*	min	avg
S1.1		98											98	99
S1.2		96		75					86	63			63	93
S1.3														100
S1.4		51	61	92		89	75	98	99	73		17	17	79
S1.5	38			99			56	39	15	48	29	6	6	60
S2.1	24	98											24	93
S2.2					58	39	40	82	88				39	83
S2.3	68	97											68	97
S2.4														100
S2.5		92	93	98	11	58	98	45	24	91			11	75
S3.1	64	68	73	97									64	91
S3.2	78												78	98
S3.3			1			99	91						1	90
S3.4	73	97											73	97
S3.5	68	59											59	93
avg	80.87	90.40	88.53	97.40	91.27	92.33	90.67	90.93	87.47	91.67	95.27	88.20	53.40	90.42

for each instance and time limit. The results of the algorithm are shown in bold if they are not worse than the ones achieved by Gurobi in 5 hours. The results are underlined if they are proved to be optimal.

TABLE 3 The results for instances with 50 customers.

	ALNS 1 min		ALNS 5 min		ALNS 10 min		Gurobi 10 min		ALNS 20 min		Gurobi 30 min		Gurobi 1 hour		Gurobi 5 hours	
#	avg	best	avg	best	avg	best	avg	best	avg	best	avg	best	avg	best	LB	UB
S1.1	0.38	**0.4**	**0.42**	**0.5**	**0.43**	**0.5**	0.26	0.3	**0.44**	**0.5**	0.34	0.38	0.3	0.3	0.4	0.72
S1.2	0.11	0.2	0.16	**0.25**			0.1	0.1	0.18	**0.25**	0.16	0.2	0.2	0.2	0.25	0.51
S1.3	0.13	0.2	0.19	**0.38**	0.21	**0.38**	0.1	0.2	0.22	**0.38**	0.1	0.1			0.25	0.6
S1.4	**0.1**	**0.1**	**0.1**	**0.1**	**0.12**	**0.33**	0.08	0.1	**0.19**	**0.35**	0.1	0.2	0.1	0.1	0.1	0.79
S1.5	**0.1**	**0.1**	**0.1**	**0.1**			0.11	0.2	**0.12**	**0.25**	0.12	0.2	0.2	0.2	0.1	0.9
S2.1	0.07	**0.25**	0.19	**0.25**	**0.22**	**0.25**	0	0			0.06	0.12	0.2	0.2	0.2	0.82
S2.2	0.25	0.25	0.25	0.25			0.42	0.5	0.28	**0.5**	0.35	0.5	0.5	0.5	0.5	0.5
S2.3	**0.02**	**0.12**	**0.02**	**0.12**			0	0	**0.04**	**0.17**					0	0.33
S2.4	0.1	**0.1**	**0.1**	**0.1**			0.1	0.1							0.1	0.1
S2.5	0.21	**0.5**	0.36	**0.5**	0.43	**0.5**	0.26	0.35	0.49	**0.5**	0.21	0.33	0.4	0.4	0.5	1.1
S3.1	0.17	**0.5**	0.32	**0.5**	0.44	**0.5**	0.19	0.45			0.4	0.5	0.5	0.5	0.5	0.6
S3.2	0.18	**0.5**	0.38	**0.5**	0.48	**0.5**	0.05	0.13	**0.5**	**0.5**	0.26	0.4	0.2	0.2	0.5	0.6
S3.3	0.16	0.2	0.2	**0.22**			0.22	0.22			0.21	0.22	0.2	0.2	0.22	0.22
S3.4	0.31	**0.4**	**0.39**	**0.4**	**0.39**	**0.4**	0.21	0.4			0.37	0.4	0.4	0.4	0.38	0.4
S3.5	0.52	0.6	0.52	0.6			0.52	0.6	0.54	0.6	0.54	0.6	0.5	0.5	0.67	0.82

TABLE 4 The results for instances with 150 customers.

	12 vehicles						18 vehicles					
	ALNS 5 min		ALNS 1 hour		Gurobi 10 hours		ALNS 5 min		ALNS 1 hour		Gurobi 10 hours	
#	avg	best	avg	best	init	LB	avg	best	avg	best	init	LB
S3.1	0.3	0.4	0.83	1.2	0.1	0.2	0.66	0.8	1	1.5	0.1	0.2
S3.2	0.16	0.2	0.62	0.85	0.1	0.15	0.41	0.67	0.84	0.85	0.1	0.1
S3.3	0.18	0.4	0.22	0.4	0.1	0.11	0.29	0.5	0.74	0.8	0.1	0.2
S3.4	0.37	0.7	0.51	1	0.1	0.15	0.6	0.8	0.88	1	0.1	0.2
S3.5	0.25	0.5	0.48	0.8	0.1	0.1	0.55	0.67	0.96	1.2	0.1	0.2

4.3 | Large-sized instances

All experiments in this and the next subsection are conducted on a computer with an `Intel Core i7-8700` 3.2 GHz processor and 32 GB of RAM running under `Microsoft Windows 10` (64-bit).

4.4 | Importance of the operators used

5 | CONCLUSION

This research proposes a MILP model for the DRRP with uncertainties and a method based on Adaptive Large Neighborhood Search (ALNS). The destruction operators used in the algorithm are working either at the route, customer, or visit level, and the reconstruction operators take into account the possibility of changing the work partition. Computational results for the algorithm and Gurobi solver show the dominance of the ALNS scheme for the medium-size instances.

It is planned to examine the influence of alpha coefficients on the robustness level in the future. We also plan to consider possible returns of rigs to customers and investigate what impact it has on the problem.

ACKNOWLEDGMENTS

This research is supported by the Russian Science Foundation under Grant 21-41-09017.

Financial disclosure

None reported.

Conflict of interest

The authors declare no potential conflict of interests.

References

Aloise, D. J., Aloise, D., Rocha, C., Ribeiro, C. C., Filho, R., & Moura, L. (2006). Scheduling workover rigs for onshore oil production. *Discrete Applied Mathematics*, *154*(5), 695–702. IV ALIO/EURO Workshop on Applied Combinatorial Optimization.

Archetti, C., & Speranza, M. G. (2012). Vehicle routing problems with split deliveries. *International Transactions in Operational Research*, *19*(1–2), 3–22.

Archetti, C., & Speranza, M. G. (2014). A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, *2*, 223-246.

Braekers, K., Ramaekers, K., & Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, *99*, 300–313.

TABLE 5 The results for instances with 50 customers.

	ref		wo shake		wo grepart		wo both repart		only visit level		only cust level		only route level		wo regret2		wo regret gen		wo both regrets		wo greedy ins	
#	avg	best	avg	best	avg	best	avg	best	avg	best	avg	best	avg	best	avg	best	avg	best	avg	best	avg	best
S1.1	0.44	0.5	0.42	0.5	0.35	0.4	0.1	0.15	0.42	0.5	**0.45**	0.5	0.38	0.5	**0.47**	0.5	0.4	0.5	**0.45**	0.5	**0.46**	0.5
S1.2	0.1	0.1	0.1	0.1	0.09	0.1	0	0	**0.12**	**0.25**	**0.16**	**0.25**	0.1	0.1	**0.12**	**0.25**	0.1	0.1	**0.13**	**0.25**	**0.13**	**0.25**
S1.3	0.16	0.3	0.15	**0.36**	0.11	0.2	0.05	0.05	0.14	0.3	0.16	0.25	0.11	0.2	**0.18**	**0.38**	**0.16**	**0.38**	**0.19**	**0.4**	**0.2**	**0.38**
S2.1	0.01	0.07	0.01	0.07	0.01	0.07	0	0	0.01	0.07	**0.03**	0.07	0.01	0.07	**0.02**	**0.1**	**0.02**	**0.1**	**0.03**	**0.1**	**0.03**	0.07
S2.2	0.25	0.25	0.25	0.25	0.25	0.25	0	0	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
S2.3	0	0	**0**	**0.04**	0	0	0	0	**0**	**0.04**	**0.01**	**0.07**	0	0	0	0	0	0	**0.01**	**0.1**	0	0
S3.1	0.47	0.5	0.45	0.5	0.4	0.5	0.13	0.5	**0.49**	0.5	0.4	0.5	0.41	0.5	0.42	0.5	**0.5**	0.5	**0.5**	0.5	0.45	0.5
S3.2	0.53	0.6	**0.54**	0.6	0.5	0.5	0.07	0.07	**0.53**	0.6	**0.53**	0.6	0.51	0.6	**0.54**	0.6	0.51	0.6	0.51	0.6	0.52	0.6
S3.3	0.12	0.2	**0.18**	0.2	**0.18**	0.2	0	0	**0.12**	**0.22**	**0.21**	**0.22**	0.11	0.2	**0.2**	0.2	**0.14**	0.2	**0.2**	**0.22**	**0.16**	**0.22**

TABLE 6 The results for instances with 50 customers.

	ref		wo regrets n greedy		only grepart		wo route level		only GC	
#	avg	best	avg	best	avg	best	avg	best	avg	best
S1.1	0.44	0.5	**0.46**	0.5	0	0	0.41	0.5	**0.44**	0.5
S1.2	0.1	0.1	**0.14**	**0.25**	0	0	0.1	0.1	**0.12**	**0.25**
S1.3	0.16	0.3	**0.26**	**0.4**	0	0	0.14	0.3	0.14	0.3
S2.1	0.01	0.07	**0.03**	0.07	0	0	**0.03**	0.07	0	0.07
S2.2	0.25	0.25	0.25	0.25	0	0	0.25	0.25	0.25	0.25
S2.3	0	0	0	0	0	0	0	0	**0.01**	**0.12**
S3.1	0.47	0.5	0.45	0.5	0	0	**0.5**	0.5	**0.5**	0.5
S3.2	0.53	0.6	**0.53**	0.6	0	0.5	**0.53**	0.6	0.52	0.6
S3.3	0.12	0.2	**0.18**	**0.22**	0	0	**0.14**	**0.22**	0.12	0.2

Bräysy, O., & Gendreau, M. (2005). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transp. Sci.*, *39*(1), 104-118.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of np-completeness*. San Francisco: Freeman.

Gendreau, M., & Tarantilis, C. D. (2010). Solving large-scale vehicle routing problems with time windows: The state-of-the-art. In *Cirrelt-2010-04*. Montreal.

Golden, B. L., Raghavan, S., & Wasil, E. A. (2008). *The vehicle routing problem: latest advances and new challenges*. Springer US.

Hansen, P., Mladenović, N., Todosijević, R., & Hanafi, S. (2017). Variable neighborhood search: basics and variants. *EURO J. Comput. Optim.*, *5*(3), 423-454.

Hemmelmayr, V. C., Doerner, K. F., Hartl, R. F., & Vigo, D. (2014). Models and algorithms for the integrated planning of bin allocation and vehicle routing in solid waste management. *Transportation Science*, *48*, 103-120.

Ho, S. C., & Haugland, D. (2004). A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. *Comput. Oper. Res.*, *31*(12), 1947–1964.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In C. Coello (Ed.), *Learning and intelligent optimization* (pp. 507–523). Berlin, Heidelberg: Springer.

Kernighan, B. W., & Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, *49*(2), 291-307.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.

Kulachenko, I. N., & Kononova, P. A. (2020a). A hybrid local search algorithm for consistent periodic vehicle routing problem. *Diskretn. Anal. Issled. Oper.*, *27*(2), 43–64.

Kulachenko, I. N., & Kononova, P. A. (2020b). A hybrid local search algorithm for consistent periodic vehicle routing problem. *J. Appl. Industr. Math.*, *14*(2), 339–351.

Kulachenko, I. N., & Kononova, P. A. (2020c). A matheuristic for the drilling rig routing problem. In A. Kononov, M. Khachay, V. Kalyagin, & P. Pardalos (Eds.), *Mathematical optimization theory and operations research* (pp. 343–358). Cham: Springer International Publishing.

Kulachenko, I. N., & Kononova, P. A. (2021). A hybrid algorithm for the drilling rig routing problem. *J. Appl. Industr. Math.*, *15*(2), 261–276.

Lambert, V., Laporte, G., & Louveaux, F. (1993). Designing collection routes through bank branches. *Computers & Operations Research*, *20*(7), 783–791.

Li, F., Golden, B., & Wasil, E. (2007). The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Comput. Oper. Res.*, *34*(10), 2918–2930.

Maniezzo, V., Stützle, T., & Voss, S. (2009). *Matheuristics: Hybridizing metaheuristics and mathematical programming*. Springer.

Mladenovic, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*, 1097–1100.

Nagata, Y., Bräysy, O., & Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, *37*(4), 724–737.

Pecin, D., Contardo, C., Desaulniers, G., & Uchoa, E. (2017). New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS Journal on Computing*, *29*(3), 489–502.

Ribeiro, G., Desaulniers, G., Desrosiers, J., Vidal, T., & Vieira, B. (2014). Efficient heuristics for the workover rig routing problem with a heterogeneous fleet and a finite horizon. *Journal of Heuristics*, *20*, 677–708.

Salhi, S., Imran, A., & Wassan, N. A. (2014). The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighborhood search implementation. *Computers & Operations Research*, *52*, 315 - 325. Recent advances in Variable neighborhood search.

Solomon, M. M. (1985). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, *35*, 254-265.

Talbi, E.-G. (2009). *Metaheuristics: From design to implementation*. Wiley Publ.

Talbi, E.-G. (2013). *Hybrid metaheuristics*. Berlin, Germany: Springer.

Toth, P., & Vigo, D. (2014). *Vehicle routing: Problems, methods, and applications, second edition*. USA: Society for Industrial and Applied Mathematics.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, *40*(1), 475–489.

Yakici, E., & Karasakal, O. (2013). A min–max vehicle routing problem with split delivery and heterogeneous demand. *Optimization Letters*, *7*, 1611-1625.

AUTHOR BIOGRAPHY

This is sample author biography text this is sample author biography text.