European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/eor

Invited Review

A review and ranking of operators in adaptive large neighborhood search for vehicle routing problems

Stefan Voigt

Catholic University of Eichstätt-Ingolstadt, Ingolstadt School of Management, Auf der Schanz 49, 85049 Ingolstadt, Germany

A R T I C L E I N F O

Keywords:
Metaheuristics
Routing
Adaptive large neighborhood search
Operators

A B S T R A C T

This article systematically reviews the literature on adaptive large neighborhood search (ALNS) to gain insights into the operators used for vehicle routing problems (VRPs) and their effectiveness. The ALNS has been successfully applied to a variety of optimization problems, particularly variants of the VRP. The ALNS gradually improves an initial solution by modifying it using removal and insertion operators. However, relying solely on adaptive operator selection is not advisable. Instead, authors often conduct experiments to identify operators that improve the solution quality or remove detrimental ones. This process is mostly cumbersome due to the wide variety of operators, further complicated by inconsistent nomenclature. The objectives of this review are threefold: First, to classify ALNS operators using a unified terminology; second, to analyze their performance; and third, to present guidelines for the development and analysis of ALNS algorithms in the future based on the outcomes of the performance evaluation. In this review, we conduct a network meta-analysis of 211 articles published between 2006 and 2023 that have applied ALNS algorithms in the context of VRPs. We employ incomplete pairwise comparison matrices, similar to rankings used in sports, to rank the operators. We identify 57 distinct removal and 42 insertion operators, and the analysis ranks them based on their effectiveness. Sequence-based removal operators, which remove sequences of customers in the current solution, are found to be the most effective. The best-performing insertion operators are those that exhibit foresight, such as regret insertion operators. Finally, guidelines and possible future research directions are discussed.

1. Introduction

The adaptive large neighborhood search (ALNS) is a metaheuristic proposed by Ropke and Pisinger (2006a) which extends the large neighborhood search (LNS) of Shaw (1998). The LNS iteratively destroys and repairs a solution by applying one destroy and subsequently a repair operator. The ALNS in contrast, allows multiple problem-specific operators that are chosen depending on their performance during the search. The ALNS has been applied to a wide variety of optimization problems, with a particular emphasis on vehicle routing problems (VRPs) (e.g. Hemmelmayr et al., 2012; Masson et al., 2013; Ropke & Pisinger, 2006a). It has also been applied to scheduling problems (e.g. Liu et al., 2017; Wen et al., 2016), and to more exotic applications, such as service deployment problems in a cloud computing context (Gullhav et al., 2017). We however, focus in this review on ALNSs for VRPs (see Appendix A for a short description and notation of basic VRP variants).

The performance of the ALNS is dependent on the choice of destroy and repair operators. This is underscored by authors who dedicate substantial efforts to the development and rigorous testing of these operators when implementing an ALNS for a particular problem. This process can be tedious as many operators have been proposed in the literature. Some of these operators are tailored to a specific problem, while others can be applied to VRPs in general and also be applicable to the problem the authors try to solve with their ALNS. It is not recommended to solely rely on the adaptive selection of operators during the search, as the performance might be poor if a wrong set of operators is chosen to start with. In order to select a well-performing set of operators, it is necessary to implement and test multiple operators. The performance of a specific operator is typically evaluated by comparing the results obtained when solving a set of benchmark instances with and without the operator. The process of selecting operators from the literature is further complicated by inconsistent nomenclature. Some operators have the same name, but are implemented differently, or have different names but are in fact identical. Additionally, many operators can be randomized, which increases the number of possible configurations.

This study employs a network meta-analysis to systematically review operators. A network meta-analysis is a statistical approach that summarizes research and compares multiple treatments used in various studies. Our study is inspired by the works of Mara et al. (2022) and Turkeš et al. (2021). Mara et al. (2022) reviewed research on the ALNS in general and highlighted the need for a rigorous analysis of operators. Turkeš et al. (2021) conducted the first meta-analysis in OR

E-mail address: stefan.voigt@ku.de.

https://doi.org/10.1016/j.ejor.2024.05.033
Received 22 June 2023; Accepted 16 May 2024
Available online 18 May 2024

on the adaptiveness of the ALNS. In contrast to their work, our study goes further by comparing not just two (i.e., ALNS versus non-adaptive LNS) but several treatments (i.e., several operators).

This review contributes to the literature as follows.

1. We classify existing operators and propose a consistent nomenclature, which helps to clarify the terminology used in the literature and facilitates the comparison of different operators.
2. We conduct a rigorous analysis of the performance of different operators. This analysis provides insights into the relative strengths and weaknesses of different operators, which can help researchers to make informed decisions when selecting operators for their specific problem.
3. We identify key design principles of well-performing ALNS implementations, which can serve as guidelines for designing future ALNS implementations.
4. We discuss guidelines and further avenues for research not only for the ALNS but for metaheuristics in general.

The remainder of this paper is organized as follows: Section 2 describes the ALNS framework and implementation variants found in the literature. Section 3 offers a comprehensive and detailed description of the research methodology employed in this study, which can be characterized as a network meta-analysis. We classify and rank removal and insertion operators through the network meta-analysis in Section 4 and Section 5, respectively. Section 6 elaborates on guidelines for implementing, evaluating, and selecting operators. Section 7 discusses research opportunities. Finally, Section 8 summarizes the findings and concludes the work.

2. Adaptive large neigborhood search

This section briefly describes the framework of the ALNS and highlights implementation variants found in the literature. Algorithm 1 depicts the general framework.

```
Input  : Starting solution $s$                          // Starting solution
Output: Best solution $s^*$
1  $s^* \leftarrow s$
2  while stop condition is not met do                    // Stop condition
3      Remove(), Insert() ← ChooseOperators()            // Selection of operators
4      $(s^{new}, C) \leftarrow$ Remove($s, q$)          // Removal
5      $s^{new} \leftarrow$ Insert($s^{new}, C'$)        // Insertion
6      $s^{new} \leftarrow$ LocalSearch($s^{new}$)       // Optional local search
7      if $f(s^{new}) < f(s^*)$ then                     // Acceptance criteria
8          $s^* \leftarrow s^{new}, s \leftarrow s^{new}$
9      else if $accept(f(s^{new}))$ then
10         $s \leftarrow s^{new}$
11     end
12     UpdateParameters()                                // Parameter update
13 end
```

Algorithm 1: Adaptive large neighborhood search framework

The ALNS starts with an initial solution s, which may simply be obtained by applying repair operators or by any means of a construction heuristics. The solution s may be required to be feasible or can also be allowed to be infeasible with respect to constraints using a generalized cost function. A generalized cost function calculates the objective as the sum of costs and a penalty depending on the degree of violations. The ALNS proceeds until a stop condition is met. For instance, the ALNS may stop after a predefined runtime, after a number of iterations without improvement, or after reaching a final temperature in case of simulated annealing.

At the beginning of each iteration, destroy and repair operators are chosen with a roulette wheel selection. Usually, only one destroy/repair operator is chosen during one iteration. The only exception is the *combined removal operator* of Voigt and Kuhn (2021), where several operators may be used during one iteration. The operators may be selected independently (i.e., there are two roulette wheels, one for destroy and one for repair operators) or pair-wise (i.e., there is only one roulette wheel, selecting a destroy–repair pair). As the focus is on ALNSs for VRPs, we will further use the more specific terms *removal* and *insertion* instead of the more general terms *destroy* and *repair*. The terms *removal* and *insertion* are particular appropriate in the context of VRPs as the solution is improved by removing and inserting customers, requests, or nodes. The design of operators is a crucial aspect of each ALNS, as it determines the solution quality. Section 4 discusses removal operators and Section 5 insertion operators in detail.

After the adaptive selection of operators, a set of q customers C is removed from s using the chosen removal operator, resulting in an incomplete solution s^{new}. The chosen insertion operator places some or all of the removed customers (customer bank, C') back into the solution s^{new}.

Some authors enhance the ALNS by applying local search methods such as 2-opt, after the removed customers have been reinserted. The local search may be executed either each time, if the objective value of s^{new}, i.e., $f(s^{new})$ is strictly lower, or within a certain range of the previous best solution $f(s^*)$.

The solutions are updated if the objective value $f(s^{new})$ is lower in case of a minimization problem. Worse solutions may be accepted depending on the acceptance criteria used (e.g., simulated annealing, record-to-record, hill climbing).

Lastly, the parameters are updated, either after each iteration or after a certain number of iterations, i.e., after a search segment. The parameter update mainly concerns the weights and probabilities for selecting operators. Please, see the meta-analysis of Turkeš et al. (2021) which discusses the adaptive weight adjustment and its benefit in more detail.

In summary, ALNS implementations vary by characteristics given in Table 1. Please note that this is not an exhaustive list, but rather highlights some of the more commonly encountered concepts.

Despite the various design decisions that must be made, a significant effort should be invested into the development of operators. In the following section, the research methodology to review and rank operators via a network meta-analysis is described.

3. Research methodology: Network meta-analysis for reviewing and ranking operators

Our objective is to conduct a comprehensive review and classification of operators utilized in ALNS for solving VRPs, ultimately identifying the most successful ones. Traditional literature reviews that solely rely on classifying operators and examining their frequency of usage can be misleading. This is because the most frequently used operators may not necessarily be the best choice in terms of performance. To address this, a network meta-analysis approach is employed, which offers a more robust evaluation method.

A meta-analysis involves a systematic review of research, supplemented by statistical methods, to summarize the outcomes of included studies (Moher et al., 2009). Typically, meta-analyses compare two treatment alternatives, for example the first meta-analysis in the field of OR by Turkeš et al. (2021) compares the ALNS with and without adaptive operator selection. However, in our review, more than two treatments are compared as ALNSs generally use several operators. Therefore, our review can be classified as a *network* meta-analysis (Hutton et al., 2015). This approach allows to go beyond conventional literature reviews and provides a more rigorous assessment of the performance of operators. In the following, Section 3.1 describes the identification and review of literature, Section 3.2 details the ranking of operators based on their performance, and Section 3.3 discusses potential biases and limitations.

Table 1
Components of ALNS and their variants.

Component	Variants
Search space	Only feasible solutions Infeasible solutions penalized by a generalized cost function
Starting solution (line 1 of Alg. 1)	Insertion operator Construction heuristics
Stop condition (line 2)	Runtime Number of iterations Number of iterations without improvement Temperature in simulated annealing
Selection of operators (line 3)	Independent selection Pair-wise selection
Operators (lines 4 and 5)	Number and type of removal operators Number and type of insertion operators
Local search (line 6)	None When criteria is met All solutions
Acceptance criteria (lines 7–10)	Simulated annealing Record-to-record Hill climbing
Parameter update (line 12)	After every iteration After a search segment

3.1. Identification and review of studies

We identify relevant studies by searching for publications with *adaptive large neighborhood search* in the title on Google Scholar following the procedure of Turkeš et al. (2021). In addition, we search for articles with *adaptive large neighborhood search* or *ALNS* as keywords in SCOPUS following the procedure of Mara et al. (2022). The search is restricted to contributions in English and published in a peer-reviewed academic journal, i.e., proceedings, theses, etc. are excluded. Furthermore, records that do not cover VRPs are excluded. Then, we review the 211 studies identified (see Online Appendix for the list) in a chronological order and classify operators used within these studies.

3.2. Ranking

We create a ranking of operators based on their performance in identified studies. First, we briefly describe how the performance of operators is usually examined *within* studies. The two most common methods are the *frequency-based* and *ablation-based* performance analysis. Second, we describe our ranking method based on pairwise comparisons *across* a set of studies.

3.2.1. Commonly used performance-analysis within studies

Frequency-based performance analysis. The *frequency-based* performance analysis shows the number of times the operators have been applied when solving a set of test instances. The more successful an operator, the more frequently it is applied during one run. Pairwise comparisons of the operators can be derived from the frequencies given. Table 2 shows an example with three operators and the resulting pairwise comparisons in Table 3. $w_{i,j} \in \{0,1\}$ indicates the outcome of the pairwise comparison of operator i against operator j. The variable $w_{i,j}$ equals 1 if operator i wins against operator j, it equals 0 if i loses and is undefined if there is a draw. The most frequently used operator (Operator 3) wins against the other operators denoted by two 1s in Table 3.

Ablation-based performance analysis. The *ablation-based* performance analysis assesses the performance of the operator by comparing the results when the operator is present compared to when the operator is excluded from the ALNS. In this case, pairwise comparisons of the operators are derived by their relative impact on solution quality. Operator i is considered to win the pairwise comparison against operator j if the gap increases more when operator i is excluded compared to the case where operator j is excluded.

Table 2
Frequency-based performance.

Operator	Frequency
Operator 1	4000
Operator 2	6000
Operator 3	10000

Table 3
Pairwise comparison: wins of operator i against j.

i \ j	Operator 1	Operator 2	Operator 3
Operator 1	–	0	0
Operator 2	1	–	0
Operator 3	1	1	–

3.2.2. Ranking of operators across studies based on incomplete comparison matrices

The methods used to evaluate the performance of operators within individual studies are not suitable for ranking across multiple studies. This is because it would be impractical to implement and test all identified operators on numerous problems and benchmark instances. Therefore, a network meta-analysis approach is employed to rank the operators across studies.

Network meta-analyses in the medical field typically use statistical analysis based on effect sizes with corresponding standard errors to derive a ranking between treatments. However, we refrain from applying a sound statistical analysis for the following reasons. First, authors do not report standard errors on their experiments. Second, effect sizes of the *frequency-based* and *ablation-based* experiments are not comparable. In fact it is even questionable, if it is possible to compare effect sizes within one type of experiment. Consider for example two studies A and B, which execute their ALNSs for a different number of iterations and report the average performance when an operator is or is not present. A may execute 100,000 iterations, while B only executes 100 iterations. The differences in performance between operators (i.e., the effect size) may be smaller in A compared to B, as the probability of finding better solutions even with inefficient operators increases with longer runtimes. Consider another experiment where study C has only few operators and study D a large set. One expects that the effect size is smaller in D, as it is more likely that one of the operators in D may compensate for an excluded operator.

Instead of quantifying an effect size, we conduct pairwise comparisons for each operator in every study. We then use the pairwise comparisons of operators extracted from the studies to construct a pairwise comparison matrix and ultimately derive a ranking of the performance of operators. The ranking of operators based on an incomplete comparison matrix is not trivial and has some similarities to ranking teams or players in sports. For example Bozóki et al. (2016) compared the performance of tennis players over a period of 40 years, where some players never played against each other, while others played more than 30 times against each other. In the following, a brief overview is given of the method used to rank operators based on an incomplete comparison matrix. For a more detailed explanation see Bozóki et al. (2016, 2010).

Every article k in the set of articles K analyzed gives a value for $w^k_{i,j}$ as shown in Table 3 (of course only when the operators i and j are present in the respective article and their performance is analyzed).

Let $x_{i,j}$ represent the number of comparisons where operator i was better than operator j. Then, $x_{i,j} = \sum_{k\in K} w^k_{i,j}$. Let $y_{i,j}$ denote the number of comparisons where operator i was worse than operator j, and $z_{i,j}$ indicate the total number of comparisons (excluding draws) between operator i and j with $z_{i,j} = x_{i,j} + y_{i,j}$.

We construct the win-loss ratio $a_{i,j} = \frac{x_{i,j}}{y_{i,j}}$. If the ratio $a_{i,j} > 1$, then operator i is considered to be better than operator j. If operator i always wins against j (i.e., it never loses $y_{i,j} = 0$, $z_{i,j} > 0$), the ratio would be undefined, so we assume $a_{i,j} = x_{i,j} + 2$ (Bozóki et al., 2016). If there is no comparison between operator i and j ($z_{i,j} = 0$) the value is missing.

$$\mathbf{A} = \begin{pmatrix} 1 & a_{1,2} & a_{1,3} & \dots & a_{1,n} \\ \frac{1}{a_{1,2}} & 1 & a_{2,3} & \dots & a_{2,n} \\ \frac{1}{a_{1,3}} & \frac{1}{a_{2,3}} & 1 & \dots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{a_{1,n}} & \frac{1}{a_{2,n}} & \frac{1}{a_{3,n}} & \dots & 1 \end{pmatrix} \in \mathbb{R}^{n\times n}_{+} \tag{1}$$

The matrix $\mathbf{A}$ is called pairwise comparison matrix, if $a_{i,i} = 1$ and $a_{i,j} = \frac{1}{a_{j,i}}$ $\forall i,j \in 1,\dots,n$. $\mathbf{A}$ is then used to determine a weight vector $\mathbf{w} = (w_1, w_2, \dots, w_n)$, where the elements $a_{i,j}$ are estimated as $\frac{w_i}{w_j}$ subject to the constraint $\sum_{i=1}^{n} w_i = 1$. The weights w_i basically condense all the pairwise comparisons such that the ratio of weights $\frac{w_i}{w_j}$ approximates $a_{i,j}$ as well as possible. In other words, w reflects the ranking of operators, the higher the weight w_i the better the operator i.

We use the logarithmic least squares method to determine $\mathbf{w}$. The minimization problem as shown below has a unique solution and is solvable if certain conditions are fulfilled (the interested reader is referred to Bozóki et al., 2010, p. 328–330 for a full description of the method and an illustrative example).

$$\min \sum_{(i,j):a_{i,j}} \left[\log a_{i,j} - \log\left(\frac{w_i}{w_j}\right)\right]^2 \tag{2}$$

$$\sum_{i=1}^{n} w_i = 1 \tag{3}$$

$$w_i > 0 \quad \forall i \in 1,\dots,n \tag{4}$$

The objective (2) aims to minimize the sum of squared deviations of the actual win-loss ratio $a_{i,j}$ and the estimated weight ratio $\frac{w_i}{w_j}$. Constraint (3) make sure that the weights are standardized, i.e., the sum equals one. Constraints (4) define the weights to be larger zero.

Before applying the methodology just described for identifying, classifying and ranking removal operators in Section 4 and repeating the same process for insertion operators in Section 5 biases and limitations are discussed in the following subsection.

3.3. Biases and limitations

There are several sources of bias across studies (publication, search and selection bias) and bias within studies as also explained in detail in Turkeš et al. (2021). In the following, sources of bias are discussed and evaluated whether they pose an issue in our meta-analysis.

- **Publication bias** occurs because positive results are more likely to be published, leading to an overestimation of the effect size. This is not an issue for our type of meta-analysis because we are interested in the performance of operators from positive results, i.e., in the ranking of operators from well-performing ALNSs. Therefore, it is reasonable to restrict our search to publications in peer-reviewed academic journals, assuming that the quality of ALNSs is superior in journals.
- **Search bias** occurs when a faulty search leads to missing relevant studies. We search only for records with ALNS in the title or in the keywords and therefore may miss several relevant studies. This search procedure may still be reasonable as authors often include the name of the metaheuristic in the title or keyword if there is a significant methodological contribution. We believe that it is unlikely that the subset of studies with ALNS in the title or keyword shows a significant different ranking of operators as the whole set of studies using an ALNS (without ALNS explicitly in the title/keyword). Therefore, the risk that this search procedure introduces a significant bias on the ranking of operators is low.
- **Selection bias** arises when eligibility criteria are not clearly mentioned or poorly defined, leading to some studies that may favor a particular result being overly represented. We defined a clear set of eligibility criteria and include as many studies as possible by allowing two types of performance analyses.
- There may be **bias within the individual studies** when evaluating the performance of operators because authors often only show results for the "final" ALNS, i.e., the version that they propose to solve the problem under consideration. This means that the reader gets no information on other (less efficient) operators that were implemented and tested but did not make it into the final ALNS. Therefore, it is likely that there were more pairwise comparisons made. This could lead to an overestimation of the performance of less efficient operators, because we do not see the cases where these operators lost. For more efficient operators, this risk is lower, as it is more likely that these operators are included and their performance is analyzed. As we are more interested in the higher ranking operators, this bias should only slightly affect our results.

In addition to the sources of bias previously listed, there are some limitations to our meta-analysis that are outlined in the following.

- Our main assumption is that we can rank operators more or less independent of the problem under consideration (as long as there is a routing component). However, it is possible that the performance of operators depends to a larger extent on the problem. For example, an operator that does not work for one problem may be beneficial for another one. Nevertheless, we are confident that operators that are more efficient on average can be identified. However, it is still up to the researcher to also test lower-ranked operators if there is evidence (based on the problem) that the operator may work efficiently.
- The version of the ALNS implemented may have an effect on the performance of the operators.
- Similarly, the parameter settings for the operators effect their performance. For example, the number of customers removed may significantly influence the performance of an operator.

- Both types of performance analyses, i.e., the frequency-based and ablation-based analysis, have their own limitations. The frequency-based performance analysis is in some cases not stable, i.e., the frequencies of the usage of operators may vary significantly between runs as pointed out by Mara et al. (2022). In addition, both types of analysis neglect the trade-off between runtime and solution quality. For example, an insertion operator that evaluates all insertion positions of a customer in all routes is computationally more demanding compared to an insertion operator that evaluates only the insertion positions in one route randomly chosen, but it is likely that the unrestricted operator improves on the solution quality (and therefore is chosen more often).
- In our analysis, we assume that the results presented in the literature are the ground-truth. This assumption is of course highly questionable. The experiments in the literature are conducted on a subset of instances for a limited number of replications. Additionally, the differences in performances are mostly not large, as the ALNSs are usually still able to find good solutions even if one operator is excluded. Therefore, it is uncertain if the pairwise comparisons are true.
- There is interaction between operators that is not captured through frequency-based or ablation-based performance analyses. For example operator A may only work well in interaction with operator X, Y, Z because A aims at intensification and X, Y, Z at diversification.

4. Classification and ranking of removal operators

A removal operator removes a number q of customers from a given solution s. The goal is to either intensify the search in promising areas or to diversify the search and explore new regions of the search space.

Classification. Removal operators vary in how they determine the set of removed customers. We classify removal operators according to the amount of information they use to determine this set. Simple removal operators do not use information on current or previous solutions (Section 4.1). More advanced operators use information derived from the current solution, either removing customers (Section 4.2) or complete routes (Section 4.3). The most advanced operators use information from the current and previous solutions, taking the history of the search into account (Section 4.4).

Naming convention. We suggest using a consistent and informative naming convention. The convention consists of four parts, as follows. The naming convention indicates that it is a removal operator (1). Please note, that for the sake of brevity (1) is left out in this review as long as it is obvious we are talking about removal operators.

(1) Removal: (2) criteria - (3) restricted set of removal candidates - (4) seed

The components (2–4) provide clarity on three key aspects of the removal operator:

(2) What is the criteria used to determine which customers are removed?
(3) Is the set of customers that can be removed restricted in any way?
(4) Does the removal operator use a seed customer/route?

Only components (1) and (2) are mandatory, (3) and (4) should be left out if there is no restriction on the set of removal candidates, or no seed element, respectively. For example the *node neighborhood removal* first introduced by Demir et al. (2012) (see Section 4.1.3) is termed: (1) Removal: (2) All customers - (3) from rectangular area - (4) around static seed customer.

Implementation details. Other relevant implementation details are listed in the following.

- **Randomization vs. determinism**: Removal operators can either choose removal candidates in a random or deterministic manner. Randomization can be achieved by choosing a random customer from a (ordered) list of candidates (Ropke & Pisinger, 2006a) or by adding noise to a criteria used in the removal operator (Hemmelmayr et al., 2012).
- **Static vs. dynamic seed**: In a static version the seed customers/routes are not updated after the removal of one customer, while in a dynamic version, the seed customers are updated after the removal of one customer (Hemmelmayr et al., 2012). Obviously, this design decision has a significant impact on the runtime of the removal operator.
- **One-shot vs. updated measure**: A measure, such as the cost difference of a solution with and without a customer, can be calculated to decide which customer to remove. This measure can be updated after the removal of one customer or remain unchanged (one-shot measure).

Unfortunately, only few articles are specific about the details listed. If the article does not specify the characteristics, it is difficult to accurately re-implement the proposed ALNS. We hope that this review will encourage researchers to include more information about the details of their removal operators in future studies, to improve the replicability of their work.

In the following, the classes of removal operators are described. We also indicate, to the best of our knowledge, the author(s) who first used these removal operator in an ALNS and the original name. It is important to note that these removal operators may have been used in other metaheuristics before. Please refer to the Online Appendix for a more detailed description of the underlying function and concepts of individual removal operators identified in the literature.

4.1. Customer removal operators using no information on current or previous solutions

The removal operators in this category do not use any information from the current or previous solutions to determine the set of removed customers. Instead, they use information derived from the instance, such as location or demand of customers. Therefore, any necessary measures and sets can be calculated beforehand, during pre-processing.

4.1.1. Customer removal operators - no problem-specific information

At the extreme end, removal operators that do not use any problem-specific information are designed solely for the purpose of diversifying the search. These operators often remove customers randomly, as in the case of the *random customers* operator. The following operators belong to this class:

- Random customers (*random removal*, Ropke & Pisinger, 2006a)
- Random customers - from restricted set (*forbidden random*, Li et al., 2016)
- Customers with lowest removal counts (*tabu based removal*, Li et al., 2016)

4.1.2. Customer removal operators - a priori customer-specific information

Removal operators in this class use a priori available customer-specific data (i.e., data that is independent from a solution), such as the customer's demand or location within the delivery area. For instance, the *all customers - from multiple randomly selected zones* operator divides the delivery area into zones and removes all customers from a chosen zone. The following operators belong to this class:

- Customers with lowest demand Adulyasak et al. (*minimum delivery quantity selection*, 2014)

- All customers - from multiple randomly selected zones (*zone removal*, Demir et al., 2012)
- Random customers and their nearest neighbor (*pair removal*, Mancini, 2016)

4.1.3. Customer removal operators - a priori related to seed customer

The removal operators in this category aim to remove customers that are related to each other and as such can be easily reinserted to create a new (acceptable) solution. These operators work in two stages. In the first stage, a seed (also called pivot) customer is chosen, typically at random. This seed customer may either be fixed for one removal iteration (static seed customer) or updated, so that the most recently removed customer becomes the new seed customer (dynamic seed customer). In the second stage, customers are selected based on the relatedness to the seed customer. The relatedness depends on a priori information obtained from the instance, such as the distance between customers or their demand similarity. This information can be calculated during pre-processing.

Removal operators in this class can be distinguished by the way they determine the relatedness of customers. For example *a priori score related customers - to seed customer* first selects a seed customer and then calculates a relatedness score based on information known a priori, such as a weighted sum of demand and distance. The following operators belong to this class:

- All customers - from rectangular area - around static seed customer (*node neighborhood removal*, Demir et al., 2012)
- A priori distance related customers - to static seed customer (*related removal*, Hemmelmayr et al., 2012)
- A priori distance related customers - to seed customer (*simplified shaw removal*, Ropke & Pisinger, 2006b)
- Demand related customers - to seed customer (*demand-based removal*, Demir et al., 2012)
- Time window related customers - to seed customer (*time-based removal*, Demir et al., 2012)
- A priori score related customers - to seed customer (*similarity removal*, Lei et al., 2011)

4.2. Customer removal operators using information on the current solution

Unlike the previous removal operators, the information or sets needed for these operators can only be obtained once a solution exists.

4.2.1. Customer removal operators - random customers from restricted set

These removal operators randomly select customers to be removed but only from a restricted set of customers. That set is derived from the current solution and cannot be obtained beforehand. For example, *random customers - from random route* selects only customers that have been served by the same (randomly selected) route in the current solution. The following operators are identified that differ in the way they restrict the set of removal candidates:

- Random customers - from random route (*remove k nodes from the same route*, Braaten et al., 2017)
- Random customers - from route with largest distance costs (*visits in long routes removal*, Wang et al., 2020)
- Random customers - from route with smallest number of customers (*minimum route removal*, Majidi et al., 2017)

4.2.2. Customer removal operators - based on sequences

These removal operators consider the sequence of customers in the current solution when selecting customers to be removed. For example, *All customers - from randomly selected sequence within concatenated routes* concatenates all routes and then randomly selects a sequence of customers to remove. The following operators belong to this class:

- Last customer from every route (*last customer removal*, Lahyani et al., 2019)
- Random customers, their predecessors and successors (*predecessors and successors removal*, Sacramento et al., 2020)
- All customers - from randomly selected sequence within concatenated routes (*sequence removal*, Li et al., 2016)
- All customers - from adjacent sequences (*adjacent string removal*, Friedrich & Elbert, 2022)
- All customers - from one of two Kruskal clusters from randomly selected route (*cluster removal*, Ropke & Pisinger, 2006b)

4.2.3. Customer removal operators - a posteriori related to seed customer

Removal operators in this category calculate a relatedness measure based on information derived from the current solution. These operators work similarly to the *Customer removal operators - a priori related to seed customer* (Section 4.1.3), the only difference is that the relatedness now depends on information derived from the current solution (e.g., start time) and therefore cannot be calculated during pre-processing. The following operators belong to this class:

- Start time related customers - to seed customer (*time-oriented removal*, Pisinger & Ropke, 2007)
- A posteriori distance related customers - to seed customer (*shaw removal*, Coelho et al., 2012)
- A posteriori score related customers - to seed customer (*shaw removal*, Ropke & Pisinger, 2006a)

4.2.4. Customer removal operators - worst placed customers from unrestricted set

These operators determine which customer is placed in an unsuitable position by calculating a measure with and without the customer. We denote this measure by δ_j. The higher δ_j, the more likely that the customer j is removed. For example the *worst cost customers* calculates the difference in costs $\delta_j = f(s_1) - f(s_0)$, where s_1 represents the solution with customer j and s_0 the solution without customer j. Customers are selected in a decreasing order of δ_j. As mentioned earlier these removal operators can be implemented in a one-shot vs. updated manner. In the one-shot version, δ_j is not updated during a single iteration, while in the updated version δ_j is updated every time a customer has been removed. Unfortunately, only few authors are explicit about this fact, therefore, we refrain from categorizing the worst removal operators according to this property. The following operators belong to this class:

- Worst distance cost customers (*worst-distance removal*, Demir et al., 2012)
- Worst lateness customers (*worst-time removal*, Demir et al., 2012)
- Worst cost customers (*worst removal*, Ropke & Pisinger, 2006a)
- Worst corresponding average route cost customers (*neighborhood removal*, Demir et al., 2012)
- Worst forward time slack customers (*critical destroy*, Grimault et al., 2017)
- Worst feasibility customers (*infeasible removal*, Hellsten et al., 2020)
- Worst arc costs customer pairs (*worst pair removal*, Gunawan et al., 2021)

4.2.5. Customer removal operators - worst placed customers from restricted set

Building on the previous class, these removal operators restrict the set from which worst customers can be removed. For example *worst distance costs customers - from random routes* first selects a random route and then removes the customers with the worst distance costs. The following operators belong to this class:

- Worst distance costs customers - from random zones (*zone worst distance removal*, Alinaghian & Shokouhi, 2018)

- Worst distance costs customers - from random routes (*max distance of random route removal*, Majidi et al., 2017)
- Worst distance costs customers - from routes with highest distance costs (*n-job removal*, Erdem, 2022)
- Worst penalty customers - from random routes (*highest penalty customer removal*, Özarık et al., 2021)
- Worst costs customers - from tightest routes (*tightest route selection and worst-distance customer removal*, Dönmez et al., 2022)

4.2.6. *Customer removal operators - other*

Lastly, this category includes operators that cannot be classified in one of the previous classes. There is only one removal operator called *feasibly exchangeable near customer pairs - from two random routes* (*route neighborhood removal*, Emeç et al., 2016) using a rather sophisticated way of determining the set of removed customers.

4.3. *Route removal operators using information on the current solution*

Route removal operators remove all customers currently served by the same route. Obviously, these kind of removal operators use information on the current solution (i.e., the route the customer is currently be served from). Route removal operators can be implemented in a single route version, where only one route is removed per iteration, or in a multiple route version, where the operator removes routes repeatedly until at least q customers have been removed.

4.3.1. *Route removal operators - random route*

The *all customers - from random route* (single: *route removal heuristic*, multiple: *route random sweep*, Qu & Bard, 2012; Ribeiro & Laporte, 2012) is the only operator in this category. It selects routes randomly and removes all customers served via the selected route(s).

4.3.2. *Route removal operators - low utilized route*

In contrast to the previous category, these removal operators select routes that have a low utilization, and use different criteria to determine what constitutes a low utilization, such as the smallest number of customers served by a route. The following operators belong to this class:

- All customers - from route with smallest number of customers (single: *least customers route removal*, multiple: *greedy route removal*, Keskin & Çatay, 2016; Özarık et al., 2021)
- All customers - from route with lowest maximum demand Hof and Schneider (single: *route removal*, 2019)
- All customers - from route with lowest capacity utilization (single: *least used vehicle removal*, multiple: *worst-vehicle capacity utilization removal*, Ghiami et al., 2019; Grangier et al., 2016)
- All customers - from route with lowest distance costs (multiple: *choose shortest routes*, Real et al., 2021)

4.3.3. *Route removal operators - worst route*

In contrast to the previous class, these operators focus on the objective value by determining the worst route in terms of, for example, waiting time or costs, and remove all customers from this worst route. The following operators belong to this class:

- All customers - from route with worst waiting time (multiple: *ship with long waiting time removal*, Bakkehaug et al., 2016)
- All customers - from route with worst costs (single: *worst vehicle removal*, multiple: *worst route removal*, Hurkmans et al., 2021; Lahyani et al., 2019)
- All customers - from route with worst average costs (multiple: *removal route short-haul*, Soriano et al., 2018)

4.3.4. *Route removal operators - other*

Finally, other types of route removal operators can again not be classified into the previous categories. The following two removal operators belong to this class:

- All customers - from route according to problem specific criteria
- Route-related customers - to seed route (multiple: *related route removal*, Azi et al., 2014)

4.4. *Removal operators using information on current and previous solutions*

4.4.1. *Removal operators - historical graph*

Removal operators in this class construct a graph using information collected from previous solutions to determine if a customer is oddly placed in the current solution. The operator stores information from previous solutions in a complete, directed weighted graph. The nodes in this graph correspond to customer visits. The weight of an edge (i, j) represents different measures. For example in case of the *sum of historical edge weights: highest best known solution costs customers*, edge weights correspond to the costs of the best solution in which customer i is visited directly before j. The weights are updated each time a new solution is found. Customers are selected according to a score, which is calculated by summing the edge weights in the graph corresponding to the current solution. The following operators belong to this class, all starting with *Sum of historical edge weights: ...* :

- ... highest best known solution costs customers (*neighbor graph removal*, Ropke & Pisinger, 2006b)
- ... highest best known route costs customers (*route-cost-based related removal*, Dayarian et al., 2016)
- ... lowest count same vehicle customers (*request graph removal*, Ropke & Pisinger, 2006b)
- ... lowest count same predecessor and successor customers (*history removal*, Masson et al., 2013)

4.4.2. *Removal operators - historical worst*

Removal operators belonging to this class calculate a measure based on the current worst cost compared to a cost achieved in previous solutions. The following operators belong to this class:

- Worst historical distance costs customers (*historical knowledge node removal*, Demir et al., 2012)
- Worst historical costs customers (*worst cost removal*, Voigt et al., 2022b)
- Worst historical average costs customers (*historic cost removal*, Voigt et al., 2022b)

4.5. *Statistics and ranking of removal operators*

4.5.1. *General statistics*

The analysis included a total of 1266 removal operators, averaging to 6 removal operators per contribution. Out of these, 281 were specific to certain problems and not generally applicable to routing problems, and 5 could not be categorized due to inadequate descriptions. After analyzing the remaining 980 removal operators, we identified 57 distinct removal operators, which we classified into 4 main categories with 15 subcategories. The accompanying Fig. 1 depicts the classification scheme and the frequency of each operator or class of operators used in the 211 articles studied.

Table 4 shows the top ten most frequently used removal operators on an individual level. These operators constitute roughly 74% (725 out of 980) of all categorized removal operators, suggesting that the ALNS implementations often utilize the same removal operators as previous ones. The three most frequently used removal operators (*random customers*, *worst cost customers*, and *a posteriori score related customers - to seed customers*) are standard operators originally proposed by Ropke

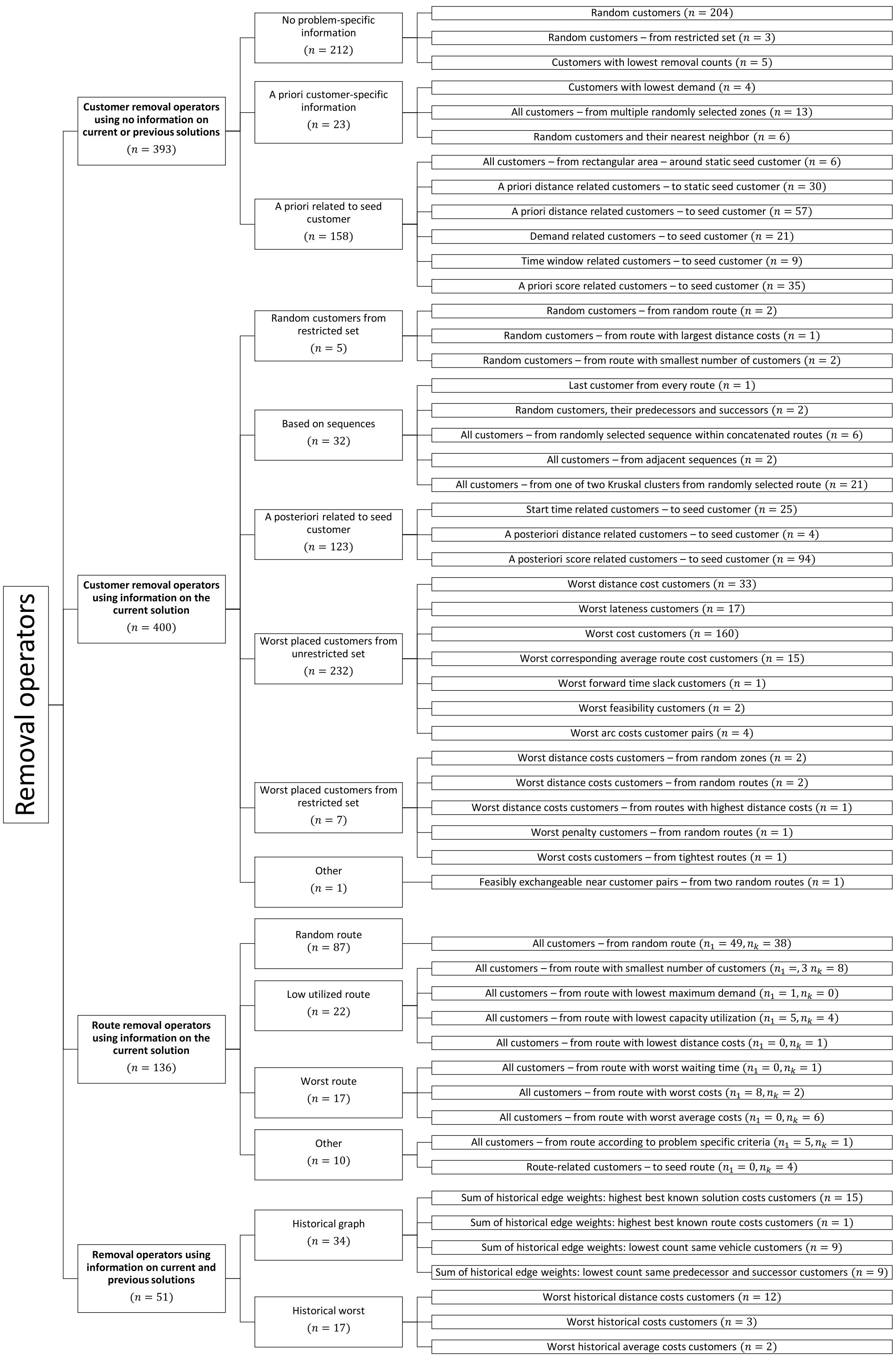
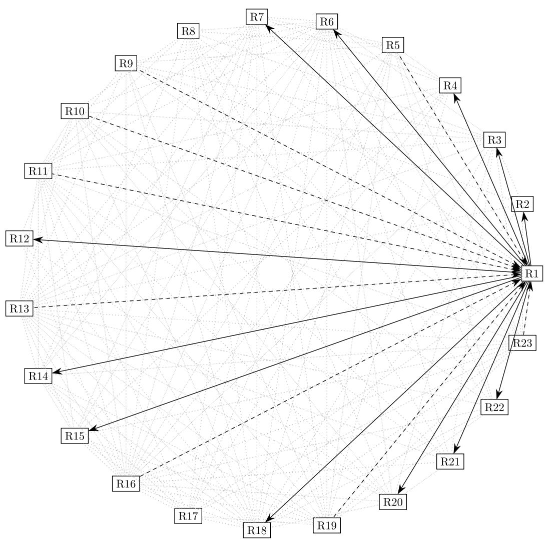
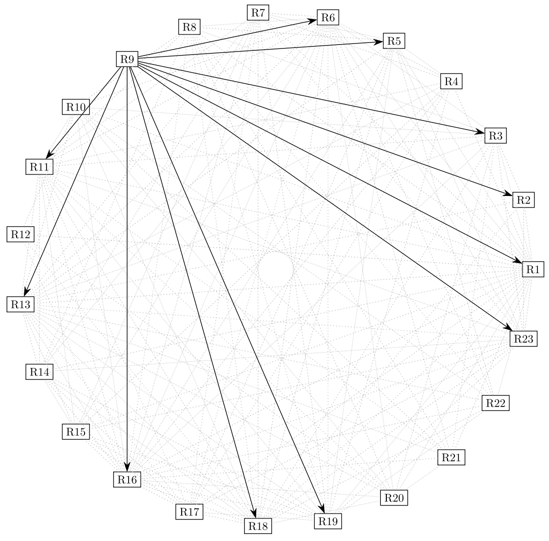

Removal operators
Customer removal operators using no information on current or previous solutions (n = 393)
No problem-specific information (n = 212)
Random customers (n = 204)
Random customers – from restricted set (n = 3)
Customers with lowest removal counts (n = 5)
A priori customer-specific information (n = 23)
Customers with lowest demand (n = 4)
All customers – from multiple randomly selected zones (n = 13)
Random customers and their nearest neighbor (n = 6)
A priori related to seed customer (n = 158)
All customers – from rectangular area – around static seed customer (n = 6)
A priori distance related customers – to static seed customer (n = 30)
A priori distance related customers – to seed customer (n = 57)
Demand related customers – to seed customer (n = 21)
Time window related customers – to seed customer (n = 9)
A priori score related customers – to seed customer (n = 35)
Customer removal operators using information on the current solution (n = 400)
Random customers from restricted set (n = 5)
Random customers – from random route (n = 2)
Random customers – from route with largest distance costs (n = 1)
Random customers – from route with smallest number of customers (n = 2)
Based on sequences (n = 32)
Last customer from every route (n = 1)
Random customers, their predecessors and successors (n = 2)
All customers – from randomly selected sequence within concatenated routes (n = 6)
All customers – from adjacent sequences (n = 2)
All customers – from one of two Kruskal clusters from randomly selected route (n = 21)
A posteriori related to seed customer (n = 123)
Start time related customers – to seed customer (n = 25)
A posteriori distance related customers – to seed customer (n = 4)
A posteriori score related customers – to seed customer (n = 94)
Worst placed customers from unrestricted set (n = 232)
Worst distance cost customers (n = 33)
Worst lateness customers (n = 17)
Worst cost customers (n = 160)
Worst corresponding average route cost customers (n = 15)
Worst forward time slack customers (n = 1)
Worst feasibility customers (n = 2)
Worst arc costs customer pairs (n = 4)
Worst placed customers from restricted set (n = 7)
Worst distance costs customers – from random zones (n = 2)
Worst distance costs customers – from random routes (n = 2)
Worst distance costs customers – from routes with highest distance costs (n = 1)
Worst penalty customers – from random routes (n = 1)
Worst costs customers – from tightest routes (n = 1)
Other (n = 1)
Feasibly exchangeable near customer pairs – from two random routes (n = 1)
Route removal operators using information on the current solution (n = 136)
Random route (n = 87)
All customers – from random route (n_1 = 49, n_k = 38)
Low utilized route (n = 22)
All customers – from route with smallest number of customers (n_1 =, 3 n_k = 8)
All customers – from route with lowest maximum demand (n_1 = 1, n_k = 0)
All customers – from route with lowest capacity utilization (n_1 = 5, n_k = 4)
All customers – from route with lowest distance costs (n_1 = 0, n_k = 1)
Worst route (n = 17)
All customers – from route with worst waiting time (n_1 = 0, n_k = 1)
All customers – from route with worst costs (n_1 = 8, n_k = 2)
All customers – from route with worst average costs (n_1 = 0, n_k = 6)
Other (n = 10)
All customers – from route according to problem specific criteria (n_1 = 5, n_k = 1)
Route-related customers – to seed route (n_1 = 0, n_k = 4)
Removal operators using information on current and previous solutions (n = 51)
Historical graph (n = 34)
Sum of historical edge weights: highest best known solution costs customers (n = 15)
Sum of historical edge weights: highest best known route costs customers (n = 1)
Sum of historical edge weights: lowest count same vehicle customers (n = 9)
Sum of historical edge weights: lowest count same predecessor and successor customers (n = 9)
Historical worst (n = 17)
Worst historical distance costs customers (n = 12)
Worst historical costs customers (n = 3)
Worst historical average costs customers (n = 2)

Fig. 1. Classification of removal operators.

and Pisinger (2006a). Hence, we refer to them as the *standard set* of removal operators. Furthermore, these three operators offer a reasonable mix between diversification (*random customers*), intensification (*worst customers*), and easy re-insertion (*a posteriori score related customers - to seed customers*). With regard to the four main categories, at least one removal operator from each of them appears in the top ten except for the *removal operators using information on current and previous solutions* category. The relatively low frequency of the latter category may have



(a) Most frequently used: *R1: Random customers*



(b) Most wins: *R9: All customers - from randomly selected sequence within concatenated routes*

Fig. 2. Graphs representing the incomplete pairwise comparison matrix for two removal operators.

Table 4
Top ten most frequently used removal operators - individual level.

Removal operator	Frequency
Random customers	204
Worst cost customers	160
A posteriori score related customers - to seed customer	94
A priori distance related customers - to seed customer	57
All customers - from random route (single)	49
All customers - from random route (multiple)	38
A priori score related customers - to seed customer	35
Worst distance cost customers	33
A priori distance related customers - to static seed customer	30
Start time related customers - to seed customer	25
Total	725

several reasons, such as being more demanding to implement or having poorer performance, which may have led them to be excluded from the final set of removal operators. In the following section, we rank the operators at the individual level to determine if there is discrepancy between the most frequently used and best-performing operators.

4.5.2. Ranking

Out of the 211 articles analyzed, 66 assessed the performance of their removal operators, allowing for the deduction of pairwise comparisons. We include only operators that were analyzed in at least three articles to ensure that the results are not heavily dependent on single studies. As a result, we are able to evaluate the performance of 23 operators. These 23 operators represent the majority with 89% (868 out of 980) of the removal operators used. The remaining 34 operators have not been analyzed in at least three articles and were therefore excluded from the ranking. We argue that it is unlikely for rarely used operators to demonstrate superior performance. However, this may not necessarily be true for recent operators, which could not have competed several times. Therefore, researchers should still check currently developed operators. Table 5 shows the outcome of the pairwise comparisons by displaying the number of wins $x_{i,j}$ and comparisons $z_{i,j}$ for each operator.

Fig. 2 visualizes the incomplete pairwise comparison matrix for two removal operators. Solid outgoing arcs represent when operator i wins against operator j ($x_{i,j} > y_{i,j}$), incoming dashed arcs represent losses ($x_{i,j} < y_{i,j}$) and light gray dotted lines without direction represent draws ($x_{i,j} = y_{i,j}$)

The graph in Fig. 2(a) represents the incomplete pairwise comparison matrix of *R1: Random customers*, which is the most frequently used. The graph shows that R1 wins against 12 operators (R2, R3, R4, R6, R7, R12, R14, R15, R18, R20, R21, R22). It loses against 8 operators (R5, R9, R10, R11, R13, R16, R19, R23), and draws against 2 operators (R8, R17). Based on these observations, we can expect R1 to rank in the upper midfield. Similarly, Fig. 2(b) shows that *R9: All customers - from randomly selected sequence within concatenated routes* which is the operator with the most wins, actually wins each comparison. Therefore, we can expect *R9* to have a top rank. However, note that this operator did not compete against every other operator.

Table 6 shows the results when deriving the weight vector $\mathbf{w}$ as described in Section 3.2 (the weights w_i are scaled by a factor of 100 for better readability). The higher the weight, the better the operator. The table also shows the number of comparisons for each operator $z_i = \sum_{j\in 1,\dots,n,j\neq i} z_{i,j}$ in column z_i, which provides an indication of the robustness of the results. Note that z_i is not equivalent to the frequency in Table 2 which shows how often an operator was used in the set of 211 articles examined. This helps to assess the robustness of results, as the robustness increases with the number of comparisons made.

As expected from the previous graph, *R9* ranks first. Notably, there is a slight discrepancy between the most used (Table 4) and the highest-ranking removal operators. The most frequently used removal operators (*standard set*) occupy ranks three to five: *R1: Random customers* ranks fifth, *R16: Worst cost customers* ranks fourth, and *R13: A posteriori score related customers - to seed customer* ranks third. The two sequence-based removal operators *R9* and *R10* occupy the first and second rank but are not even among the top 10 most frequently used operators. This suggests that adding one of these two operators to the *standard* set may improve its performance. It should be noted, however, that the number of comparisons for *R9* and *R10* are relatively low at 15 and 21, respectively, which may affect the robustness of the results. Nonetheless, both operators belong to the class of sequence-based removal operators, indicating that these type of operators may indeed perform better. Additionally, the recent metaheuristic SISR proposed by Christiaens and Vanden Berghe (2020) relies predominantly on a sequence-based removal operator and shows a good performance. Both facts support the idea that sequence-based operators could be beneficial. Despite this, the ranking confirms that the *standard* set remains a solid choice for ALNSs. It is worth noting that the difference in weights between ranks 5 to 8 are not large. The same is true for ranks 9 to 14, and so on, suggesting that there are classes of operators with similar performances.

Table 5
Comparing removal operators: Number of wins/total number of comparisons ($x_{i,j}/z_{i,j}$).

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20	R21	R22	R23
R1		2/2	3/4	3/3	4/10	10/18	5/7	6/12	0/3	0/3	4/9	2/2	8/26	5/8	3/4	18/46	3/6	9/16	3/10	3/3	3/3	5/5	2/5
R2	0/2		1/2						0/1				0/1					0/1					0/1
R3	1/4	1/2				0/1		1/1	0/1	0/1	2/2		0/1			1/2		1/2		0/1	1/1		0/1
R4	0/3				0/1	2/2	1/2				0/1		1/2		1/1	2/2	1/2		0/1				
R5	6/10			1/1			1/1	1/2	0/1				0/1			3/6	1/1	1/2	2/4			0/1	1/1
R6	8/18		1/1	0/2			2/6	1/3	0/1	0/2	4/8		2/9	3/5	1/3	2/14	2/4	3/6	1/2	2/3	1/2	1/1	2/3
R7	2/7			1/2	0/1	4/6					3/5		2/6	1/4	2/3	2/4	2/4	1/3	0/1				1/4
R8	6/12		0/1		1/2	2/3				0/1	0/1			0/1		5/12		2/3	2/3	0/1	1/1	1/1	
R9	3/3	1/1	1/1		1/1	1/1					1/1		2/3			1/1		1/1	1/1				1/1
R10	3/3		1/1			2/2		1/1			1/1	0/1	1/1			3/4		1/2		2/3	1/1	1/1	
R11	5/9		0/2	1/1		4/8	2/5	1/1	0/1	0/1			2/6	2/4	1/3	4/6	0/2	2/4	1/2	0/1	1/1		2/3
R12	0/2									1/1						2/3		0/1		0/1			
R13	18/26	1/1	1/1	1/2	1/1	7/9	4/6		1/3	0/1	4/6			4/6	3/4	10/21	3/4	3/9	1/3	1/1	1/1	2/3	3/4
R14	3/8					2/5	3/4	1/1			2/4		2/6		1/2	0/3	1/3	3/4	0/2				1/3
R15	1/4			0/1		2/3	1/3				2/3		1/4	1/2		0/2	0/2	0/2	0/1				1/1
R16	28/46		1/2	0/2	3/6	12/14	2/4	7/12	0/1	1/4	2/6	1/3	11/21	3/3	2/2		1/2	12/16	5/6	3/4	2/3	5/5	2/2
R17	3/6			1/2	0/1	2/4	2/4				2/2		1/4	2/3	2/2	1/2		0/1	0/2				1/2
R18	7/16	1/1	1/2		1/2	3/6	2/3	1/3	0/1	1/2	2/4	1/1	6/9	1/4	2/2	4/16	1/1			1/1		2/2	2/4
R19	7/10			1/1	2/4	1/2	1/1	1/3	0/1		1/2		2/3	2/2	1/1	1/6	2/2					0/1	1/1
R20	0/3		1/1			1/3		1/1		1/3	1/1	1/1	0/1			1/4		0/1			1/2		
R21	0/3		0/1			1/2		0/1		0/1	0/1		0/1			1/3				1/2			
R22	0/5				1/1	0/1		0/1		0/1			1/3			0/5		0/2	1/1				
R23	3/5	1/1	1/1		0/1	1/3	3/4		0/1		1/3		1/4	2/3	0/1	0/2	1/2	2/4	0/1				

R1: Random customers, R2: Random customers - from restricted set, R3: Customers with lowest removal counts, R4: Random customers and their nearest neighbor, R5: A priori distance related customers - to static seed customer, R6: A priori distance related customers - to seed customer, R7: Demand related customers - to seed customer, R8: A priori score related customers - to seed customer, R9: All customers - from randomly selected sequence within concatenated routes, R10: All customers - from one of two Kruskal clusters from randomly selected route, R11: Start time related customers - to seed customer, R12: A posteriori distance related customers - to seed customer, R13: A posteriori score related customers - to seed customer, R14: Worst distance cost customers, R15: Worst lateness customers, R16: Worst cost customers, R17: Worst corresponding average route cost customers, R18: All customers - from random route - single, R19: All customers - from random route - multiple, R20: Sum of historical edge weights: highest best known solution costs customers, R21: Sum of historical edge weights: lowest count same vehicle customers, R22: Sum of historical edge weights: lowest count same predecessor and successor customers, R23: Worst historical distance costs customers

Table 6
Ranking of removal operators.

#	Removal operator	w_i	z_i
1	R9: All customers - from randomly selected sequence within concatenated routes	12.48	15
2	R10: All customers - from one of two Kruskal clusters from randomly selected route	8.78	21
3	R13: A posteriori score related customers - to seed customer	6.41	112
4	R16: Worst cost customers	5.92	164
5	R1: Random customers	5.35	205
6	R19: All customers - from random route - multiple	5.28	40
7	R5: A priori distance related customers - to static seed customer	5.09	31
8	R18: All customers - from random route - single	4.95	80
9	R12: A posteriori distance related customers - to seed customer	3.82	8
10	R4: Random customers and their nearest neighbor	3.71	17
11	R17: Worst corresponding average route cost customers	3.66	35
12	R20: Sum of historical edge weights: highest best known solution costs customers	3.55	21
13	R8: A priori score related customers - to seed customer	3.53	42
14	R11: Start time related customers - to seed customer	3.50	60
15	R23: Worst historical distance costs customers	3.19	36
16	R14: Worst distance cost customers	3.17	45
17	R3: Customers with lowest removal counts	3.15	20
18	R6: A priori distance related customers - to seed customer	3.06	93
19	R7: Demand related customers - to seed customer	2.81	50
20	R22: Sum of historical edge weights: lowest count same predecessor and successor customers	2.35	20
21	R15: Worst lateness customers	2.33	28
22	R2: Random customers - from restricted set	2.01	8
23	R21: Sum of historical edge weights: lowest count same vehicle customers	1.89	15

Coming back to the relatively low frequency of the main category *removal operators using information on current and previous solutions*, the best operator from this class *R20* only achieves rank 12. We can deduce that the reason for the relatively low frequency lies not only in the rather higher complexity of implementing these operators but also in their weaker performance.

4.6. Main recommendation

Deducing from the analysis, it may be worthwhile to start with the *standard set* of operators which are the three most frequently used removal operators (*random customers*, *worst cost customers*, and *a posteriori score related customers - to seed customers*). These three operators offer a reasonable mix between diversification and intensification. This set can be enhanced with a sequence-based operator.

Then, operators should be added one by one from the next best-ranking class of operators, stopping when the results are no longer improving, or the performance seems sufficient. The researcher should not blindly walk through the list of operators but must ensure that the choice of operators balances intensification and diversification. Intensification can be achieved by removing customers that increase the cost the most, while diversification can be achieved by randomly removing customers or those that have rarely been removed in previous iterations. The level of intensification respectively diversification depends on the degree of randomization within the removal operator, as well as the update behavior (one-shot version vs. update version).

Irrespective of whether the removal operator is designed for intensification or diversification, it is preferable to remove customers that can be easily re-inserted. Otherwise, the insertion operators struggle at finding a solution, that has an acceptable objective value. If that is the case, the solution is unlikely to be accepted and the runtime spent during this removal and insertion phase will have been wasted. To conclude, a well-designed ALNS should include a balanced mix of diversifying and intensifying removal operators that preferably remove customers that can be easily reinserted.

5. Classification and ranking of insertion operators

An insertion operator is responsible for reinserting the customers that were previously removed back into the solution s. To be more precise, the insertion operator takes the next customer from the sorted or unsorted list of removed customers and inserts it into the solution based on a particular criterion, for example, the position that results in the lowest increase in costs. The goal is to create a complete solution (i.e., all customers are inserted) with an acceptable objective value.

Classification. The key decision of insertion operators is to determine in which position the previously removed customers are inserted. We classify insertion operators based on how they determine this position.

- **Random position**: Insert customers at a random position (Section 5.1).
- **Best cost position**: Select the position that results in the least increase in objective value (Section 5.2).
- **Best cost position with restricted options**: Evaluate just a subset of insertion positions (Section 5.3).
- **Best timing position**: Use time criteria instead of objective values (Section 5.4).

The second criteria used to classify insertion operators is the order in which the list of insertion candidates is sorted. The following sorting options are encountered in the literature:

- **Removal order**: The list remains unchanged, i.e., customers are re-inserted in the same order they were removed.
- **Random order**: The list is randomly shuffled.
- **Cost-based order**: The list is sorted according to a cost measure. Customers with low costs are inserted earlier.
- **Distance-based order**: Customers near previously inserted customers are inserted earlier.
- **Time-based order**: The list is sorted based on time measures, e.g., width of the time window.
- **Duration-based order**: The list is sorted according to a duration measure, with customers who cause minimal increases in route duration being inserted earlier.
- **Difficulty-based order**: Customers who are expected to be challenging to insert without significantly increasing costs are inserted earlier.
- **Regret-based order**: The list is sorted based on a regret measure. Preference is given to customers whose later insertions cause the highest additional cost increase.

Difficulty-based and regret-based operators attempt to overcome the myopic behavior of inserting customers with the lowest cost first by incorporating (expected) information about future insertion costs into their decision-making process. Difficulty-based operators use proxies to estimate the expected difficulty of insertions, for example customers with highest costs are inserted first. In contrast, regret-based operators calculate a regret value, which measures the cost difference between inserting the customer now and postponing the insertion. For example, the *customer with highest route regret - at best position* operator calculates the regret value as the difference between insertion costs in the best route and in the second (third, fourth, ...) route. A larger regret value indicates a greater urgency to insert the customer immediately.

Naming convention. We suggest using a naming convention that reflects firstly that it is an insertion operator and secondly how customers are inserted. Again, for the sake of brevity (1) is left out as long as it is obvious we are talking about insertion operators.

(1) Insertion: (2) order - (3) position - (4) noise - (5) restricted set of insertion positions

The components (2–5) describe four key aspects of the insertion operator:

(2) Does the operator sort the list of removed customers, and if so in what way?
(3) Into which position is the selected customer inserted?
(4) Is the chosen position deterministic or does it vary by applying noise?
(5) Is the set of insertion positions restricted?

Components (1–3) are mandatory. Components (4) and (5) should be left out if there is no noise, respectively no restriction on the set of available positions. For example the *regret insertion heuristics* first introduced by Ropke and Pisinger (2006b) (see Section 5.2) is termed: (1) Insertion: (2) Customer with highest route regret - (3) at best position. Note that no noise is applied and the set of insertion positions is not restricted, therefore (4) and (5) are omitted.

Implementation details. Similar to removal operators, insertion operators can be implemented using a *one-shot* or *updated measure* approach (as explained in Section 4). Again, authors are often not specific about this implementation detail. As a result, categories are not based on this characteristics. Nevertheless, we stress that this information is crucial for an accurate implementation.

In the following sections, the classes of insertion operators are described. A more detailed description of each individual insertion operator can be found in the Online Appendix.

5.1. Insertion operators - random position

In the literature, both types of *random position* operators are referred to as such, even though they can be distinguished based on the insertion order. These operators insert the customers at random positions. The primary purpose is obviously to diversify the search.

- In removal order - at random position (*random insertion*, Majidi et al., 2017)
- In random order - at random position (*random insertion*, Avci & Avci, 2019)

5.2. Insertion operators - best cost position

Insertion operators belonging to this class calculate the minimum insertion costs for customer j in solution s, denoted by $\delta_j = f(s_1) - f(s_0)$, where s_1 represents the solution with customer j and s_0 the solution without customer j. Customers are inserted at their respective minimum cost position, called the best position. The insertion costs are usually updated after an insertion. The *best cost position* operators insert only one customer in each iteration, resulting in changes to only one route (i.e., the route in which the customer can be inserted with minimum costs). In the next iteration, only the insertion costs on the changed route need to be recalculated. In contrast to the *random position* insertion operators, these operators intensify the search.

- **Removal order**
  - In removal order - at best position (*basic greedy insertion*, Ribeiro & Laporte, 2012)
  - In removal order - at best position - with noise (*sequential perturbed insertion*, Schiffer & Walther, 2018)
  - In removal order - at best position with route-frequency penalty (*diversification 1-by-1*, Li et al., 2016)

- **Random order**
  - In random order - at best position (*greedy insertion*, Lei et al., 2011)
  - In random order - at best position - with noise (*greedy insertion perturbation*, Contardo et al., 2012)
  - In random order - at second-best position (*second-best insertion*, Ghilas et al., 2016)
  - In random order - at second-best position - with noise (*second-best insertion with noise function*, Ghilas et al., 2016)
- **Cost-based order**
  - Customers with lowest cost first - at best position (*best insertion heuristics*, Ropke & Pisinger, 2006b)
  - Customers with lowest cost first - at best position - with noise (without dedicated name, Ropke & Pisinger, 2006a)
  - Customers with the n lowest cost first - at best position (*grasp insertion*, Goeke & Schneider, 2015)
- **Distance-based order**
  - Customers farthest to depot first - at best position (*farthest insertion*, Smith & Imeson, 2017)
  - Customers closest to depot first - at best position (*nearest insertion*, Smith & Imeson, 2017)
  - Customers closest to previously inserted customer first - at best position (*distance-related insertion*, Ortega et al., 2020)
- **Time-based order**
  - Customers with minimum earliest arrival time - at best position (*earliest delivery release date*, Grimault et al., 2017)
  - Customers with minimum latest arrival time - at best position (*time sorted greedy insertion*, Bakkehaug et al., 2016)
  - Customers sorted according to other time window criteria - at best position (e.g., Sacramento et al., 2020)
- **Difficulty-based order**
  - Customers with highest cost first - at best position (*auction-based insertion*, Koç, 2016)
  - Customers with highest average historic costs first - at best position (without a dedicated name, Voigt et al., 2022a)
  - Most constrained customers first - at best position (*most constrained first insertion*, Qu & Bard, 2012)
  - Customers with highest demand first - at best position (*demand and failure sorting insertion*, Lei et al., 2011)
- **Regret-based order**
  - Customer with highest route regret - at best position (*regret insertion heuristics*, Ropke & Pisinger, 2006b)
  - Customer with highest route regret - at best position - with noise (Ropke & Pisinger, 2006a)
  - Customer with highest position regret - at best position (*regret-k insertion heuristic*, Qu & Bard, 2012)
  - Customer with highest position regret - at best position - with noise (*regret insertion with noise function*, Demir et al., 2012)
  - Customer with highest position duration regret - at best position (*regret-k time window insertion*, Kancharla & Ramadurai, 2018)
  - Customer with lowest impact value - at best position (*non-myopic insertion*, Ticha et al., 2019)

5.3. *Insertion operators - best cost position - restricted options*

Insertion operators in this class are similar to the previous class of operators, but they operate on a restricted set of available insertion positions.

- **Removal order**
  - In removal order - at best position - restricted to route with minimal length (*balanced 1-by-1*, Li et al., 2016)
  - In (reversed) removal order - at best position - restricted to one best insertion position (*hybrid insertion operator*, Franceschetti et al., 2017)
- **Random order**
  - In random order - at best position - restricted to random route (*randomly insert*, Coelho et al., 2012)
- **Cost-based order**
  - Customers with lowest cost first - at best position - restricted to positions on new routes (*greedy insertion with new route openings*, Emeç et al., 2016)
  - Customers with lowest cost first - at best position - restricted to positions not on the previous route (*tabu 1-by-1*, Li et al., 2016)
  - Customers with lowest cost first - at best position - restricted to positions on the previous route (*local all-at-once*, Li et al., 2016)
  - Customers with lowest cost first - at best position - restricted to closest route (*closest insertion repair method*, Sacramento et al., 2019)
  - Customers with lowest cost first - at best position - restricted to loosest route (*loosest route selection and least-distance customer insertion*, Dönmez et al., 2022)
  - Customers with lowest cost first - at best position - restricted to position with highest time window gap (*fill the gap*, Braaten et al., 2017)
- **Regret-based order**
  - Customer with highest route regret - at best position - restricted to positions not on the previous route (*regret insertion with ban mechanism*, Jiang & Li, 2020)

5.4. *Insertion operators - best timing position*

In contrast to *Insertion operators - best cost position*, the following insertion operators consider timing aspects to identify suitable insertion positions.

- In random order - at position with least waiting time - restricted to routes in zone (*zone insertion*, Demir et al., 2012)
- Customers with lowest duration increase - at position with lowest duration increase (*time based insertion*, Keskin & Çatay, 2016)
- Customers with lowest duration increase or costs - at position with lowest duration increase or costs (*Solomon insertion*, Braaten et al., 2017)

5.5. *Statistics and ranking of insertion operators*

5.5.1. *General statistics*

We analyzed a total of 788 insertion operators, which corresponds to an average of 3.73 insertion operators per contribution. Of these, 176 are problem specific. Four could not be classified, as the description was not sufficiently detailed. Our analysis of the remaining 608 insertion operators resulted in 42 distinct insertion operators into 4 main

Table 7
Top ten most frequently used insertion operators - individual level.

Insertion operator	Frequency
Customers with lowest cost first - at best position	141
Customer with highest position regret - at best position	80
Customer with highest route regret - at best position	65
In removal order - at best position	45
Customers with lowest cost first - at best position - with noise	42
In random order - at best position	39
In removal order - at random position	21
Customer with highest position regret - at best position - with noise	20
Customer with highest route regret - at best position - with noise	19
In random order - at random position	14
Total	486

categories. Fig. 3 illustrates the classification and the frequency with which each operator or class of operators has been used in the 211 articles examined.

Table 7 shows the top ten most frequently used insertion operators on an individual level. These cover about 80% (486 out of 608) of the categorized insertion operators, showing an even lower variance compared to the removal operators, where the top ten accounted for 74%. The most frequently used insertion operator is *customers with lowest cost first - at best position*. The second and third rank are occupied by regret-based operators, namely *customer with highest position regret* and *customer with highest route regret*. Unlike removal operators, there is no apparent *standard* set. Instead, authors usually employ either the insertion operator based on *position regret* or the original version based on *route regret* (Ropke & Pisinger, 2006a).

5.5.2. Ranking

Out of 211 articles 60 evaluated the performance of insertion operators. Again, to be included in the ranking, insertion operators had to be evaluated in at least three articles. Fourteen insertion operators met this criterion, representing the majority (88%, or 535 out of 608) of used insertion operators. The remaining 28 insertion operators did not meet this criterion and were therefore excluded from the ranking. Table 8 displays the number of wins $x_{i,j}$ and number of comparisons $z_{i,j}$ for each operator included in the ranking.

Equivalent to the analysis of removal operators, graphs represent the incomplete pairwise comparison matrix for the most frequently used insertion operator I7 (Fig. 4(a)) and for I13 which has the most wins (Fig. 4(b)). *I7: Customers with lowest cost first - at best position* performs moderately well with eight wins (I1, I2, I3, I5, I6, I8, I9, I10: solid outgoing arcs), four losses against regret-based operators (I11–I14: incoming dashed arcs), and one tie with I4 (undirected light gray dotted line). Based on these observations, I7 is expected to rank in the upper midfield. *I13: Customer with highest position regret - at best position* wins every comparison except for a tie with *I5: In random order - at best position*. Only two direct comparisons are missing (I11 and I12).

Table 9 shows the results of deriving the weight vector **w**. Unsurprisingly, *I13: Customer with highest position regret - at best position* ranks first, being superior to other regret-based insertion operators, and in particular dominating *I11: Customer with highest route regret - at best position* which achieves rank 6. Note that the difference in weights is rather large with a sufficiently high number of comparisons (53 and 26).

Regarding insertion order, regret-based ordering methods appear to be more effective than random and removal ordering methods, with the exception of *I5: In random order - at best position*, which ranks third. This ranking of insertion orders is not surprising, as the comparisons do not take runtime into account, and the regret and cost-based ordering methods are more computationally demanding than random or removal order. Therefore, the third rank of *I5* is particularly noteworthy.

Regarding insertion position, *at best position* is the method of choice, other options are either infrequently used or show inferior performance, such as *at random position* (rank 10 and 13).

The most frequently used insertion operator, *I7: Customers with lowest cost first - at best position*, ranks fifth, which is in contrast to its frequent usage. Authors may consider using operator *I5* instead. However, note that the difference in weights is not substantial for ranks 3 to 6.

In general, the operator versions with noise perform worse than clean versions, except for *I12: Customer with highest route regret - at best position - with noise*, where the version with noise ranks higher. However, this insertion operator has only been analyzed in three articles with a total of eight comparisons, so further comparisons are necessary to make a definitive statement.

5.6. Main recommendation

To summarize, we suggest that the *standard* set of insertion operators should include two operators, namely *I13* and *I5*. *I13: Customer with highest position regret - at best position* is preferred against *I11: Customer with highest route regret - at best position* because of its higher rank. *I5: In random order - at best position* is preferred because of its simplicity compared to *I7: Customers with lowest cost first - at best position*. Again, as also discussed in the section on removal operators, this set of insertion operators also guarantees a balance of intensification (*I13*) and diversification (*I5*).

6. Guidelines for development and presentation of ALNS algorithms

The guidelines presented in this section aim to improve the effectiveness and transparency of ALNS research, with relevance not only to ALNSs but also to heuristic algorithms in general. We draw upon this extensive literature review and in addition refer to the valuable guide by Johnson (2002) for algorithm development and testing. Section 6.1 focuses on the evaluation and selection of operators within ALNS algorithms. We provide guidelines to ensure a structured approach to operator selection, taking runtime into account, and conducting performance assessments through ablation studies. These guidelines aim to support researchers in making informed decisions about which operators to include in their ALNS implementations. Section 6.2 shifts the focus to improving the presentation of ALNS research. Transparent and reproducible research findings are vital for advancing the field. To achieve this, we recommend including information on unsuccessful operators, presenting standard errors in results, providing details on the implementation of operators, and, whenever possible, publishing the ALNS code alongside the research article.

6.1. Evaluation and selection of operators

Follow a structured selection process. For future ALNS implementations, we suggest starting with the identified *standard* set of operators. For removal the set consists of four operators:

- *R1: Random customers*
- *R16: Worst cost customers*
- *R13: A posteriori score related customers - to seed customer*
- A sequence-based removal operator, e.g., *R10: All customers - from one of two Kruskal clusters from randomly selected route*.

For insertion the set consists of two operators:

- *I13: Customer with highest position regret - at best position*
- and *I5: In random order - at best position*

In the next step, the researcher may gradually add operators from the next best-performing group. The added operator may be kept if a performance measure (e.g., the gap or primal integral, see next paragraph) improves, otherwise the operator is removed and continued with the next one. This process should be stopped as soon as the overall performance seems sufficient.

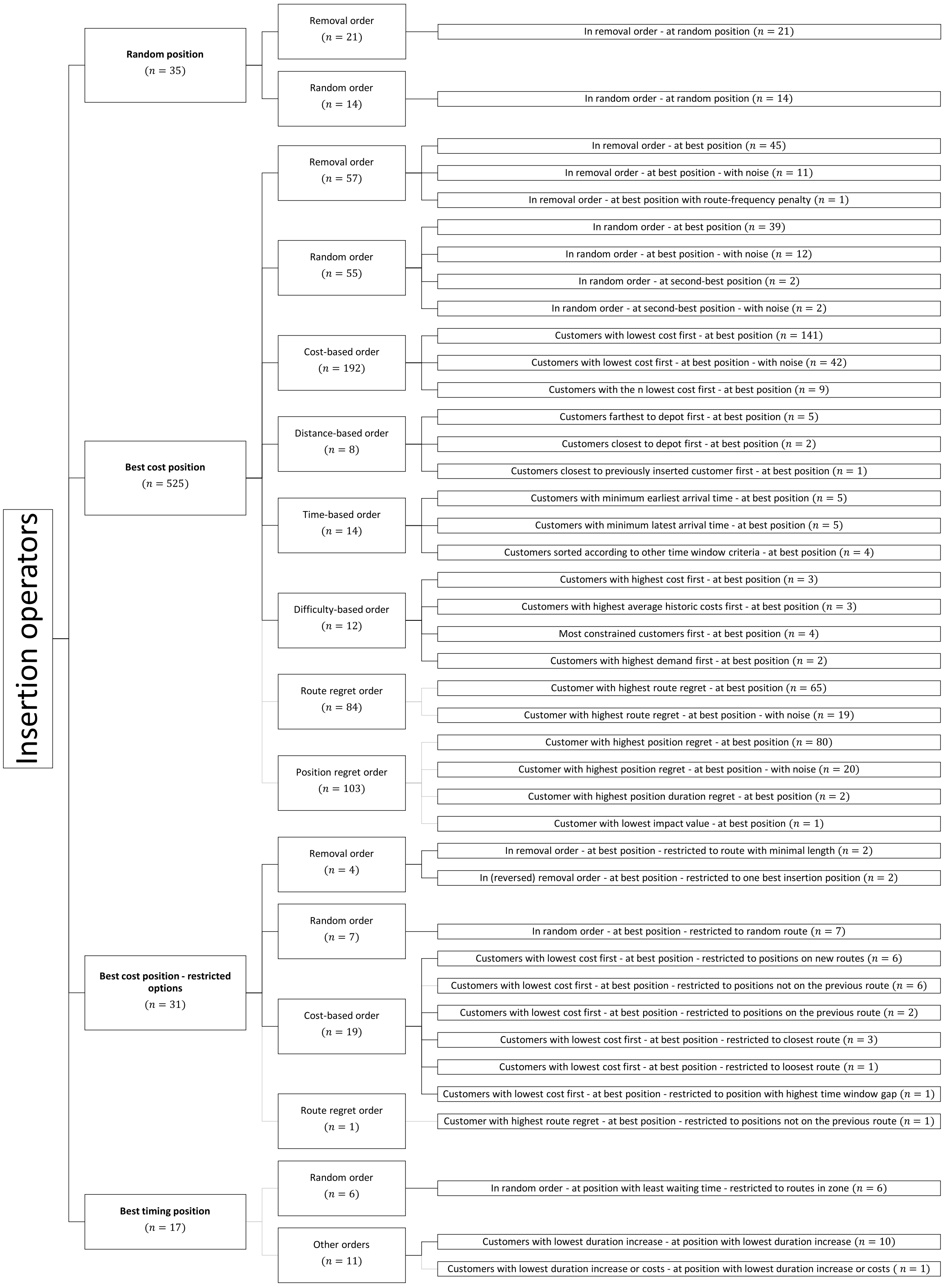

Insertion operators
Random position (n = 35)
Removal order (n = 21)
In removal order - at random position (n = 21)
Random order (n = 14)
In random order - at random position (n = 14)
Best cost position (n = 525)
Removal order (n = 57)
In removal order - at best position (n = 45)
In removal order - at best position - with noise (n = 11)
In removal order - at best position with route-frequency penalty (n = 1)
Random order (n = 55)
In random order - at best position (n = 39)
In random order - at best position - with noise (n = 12)
In random order - at second-best position (n = 2)
In random order - at second-best position - with noise (n = 2)
Cost-based order (n = 192)
Customers with lowest cost first - at best position (n = 141)
Customers with lowest cost first - at best position - with noise (n = 42)
Customers with the n lowest cost first - at best position (n = 9)
Distance-based order (n = 8)
Customers farthest to depot first - at best position (n = 5)
Customers closest to depot first - at best position (n = 2)
Customers closest to previously inserted customer first - at best position (n = 1)
Time-based order (n = 14)
Customers with minimum earliest arrival time - at best position (n = 5)
Customers with minimum latest arrival time - at best position (n = 5)
Customers sorted according to other time window criteria - at best position (n = 4)
Difficulty-based order (n = 12)
Customers with highest cost first - at best position (n = 3)
Customers with highest average historic costs first - at best position (n = 3)
Most constrained customers first - at best position (n = 4)
Customers with highest demand first - at best position (n = 2)
Route regret order (n = 84)
Customer with highest route regret - at best position (n = 65)
Customer with highest route regret - at best position - with noise (n = 19)
Position regret order (n = 103)
Customer with highest position regret - at best position (n = 80)
Customer with highest position regret - at best position - with noise (n = 20)
Customer with highest position duration regret - at best position (n = 2)
Customer with lowest impact value - at best position (n = 1)
Best cost position - restricted options (n = 31)
Removal order (n = 4)
In removal order - at best position - restricted to route with minimal length (n = 2)
In (reversed) removal order - at best position - restricted to one best insertion position (n = 2)
Random order (n = 7)
In random order - at best position - restricted to random route (n = 7)
Cost-based order (n = 19)
Customers with lowest cost first - at best position - restricted to positions on new routes (n = 6)
Customers with lowest cost first - at best position - restricted to positions not on the previous route (n = 6)
Customers with lowest cost first - at best position - restricted to positions on the previous route (n = 2)
Customers with lowest cost first - at best position - restricted to closest route (n = 3)
Customers with lowest cost first - at best position - restricted to loosest route (n = 1)
Customers with lowest cost first - at best position - restricted to position with highest time window gap (n = 1)
Route regret order (n = 1)
Customer with highest route regret - at best position - restricted to positions not on the previous route (n = 1)
Best timing position (n = 17)
Random order (n = 6)
In random order - at position with least waiting time - restricted to routes in zone (n = 6)
Other orders (n = 11)
Customers with lowest duration increase - at position with lowest duration increase (n = 10)
Customers with lowest duration increase or costs - at position with lowest duration increase or costs (n = 1)

Fig. 3. Classification of insertion operators.

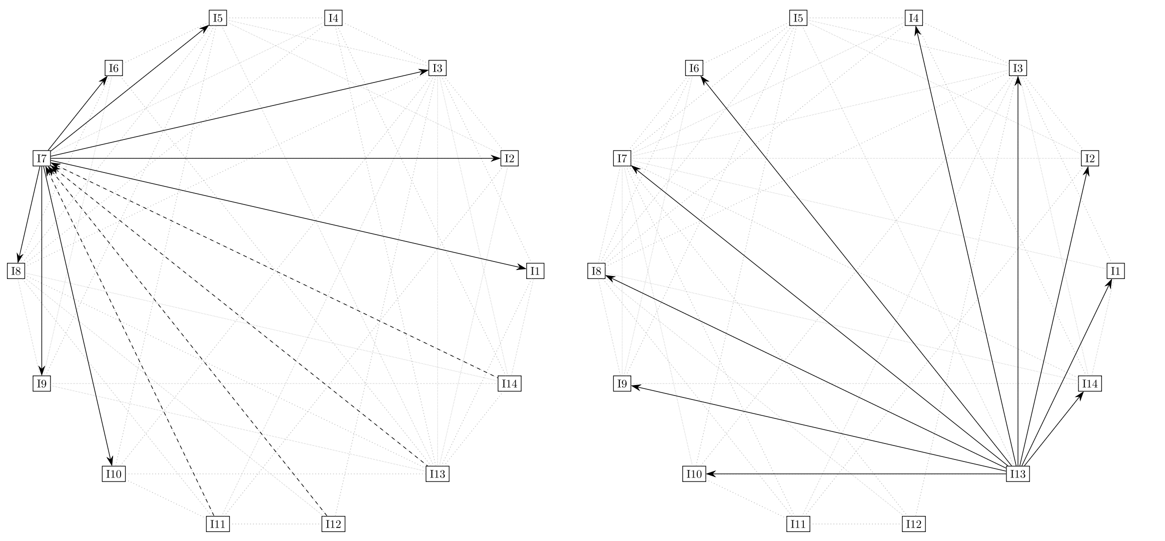
Furthermore, it may be worthwhile to test operators from the low-performing groups if there is evidence that the operator might work for the specific problem at hand. A fair share of operators did not even make it into the ranking, because they have not been analyzed in three or more articles. Still, it may be relevant to check the performance of unranked operators as well.

Take runtime into account. The benchmark of operators should assess the trade-off between solution quality and runtime. When only looking at the contribution of an operator to the solution quality, runtime intensive operators are favored. For example, an insertion operator that evaluates all insertion positions of a customer in all routes is computationally more demanding compared to an insertion operator that

Table 8
Comparing insertion operators: Number of wins/total number of comparisons ($x_{i,j}/z_{i,j}$).

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12	I13	I14
I1			0/1				0/3						0/2	1/1
I2			0/1		0/2		0/4				0/2		0/2	
I3	1/1	1/1		2/4	0/1		2/5	0/1		1/1	3/5	1/1	1/6	0/3
I4			2/4		0/1		1/2	0/1					0/3	0/3
I5		2/2	1/1	1/1		3/4	2/6	1/1	2/2	0/1			2/4	
I6					1/4		0/1	0/1	2/2				0/2	
I7	3/3	4/4	3/5	1/2	4/6	1/1		5/8	2/2	2/3	3/13	0/2	8/18	2/5
I8			1/1	1/1	0/1	1/1	3/8		2/2		1/2	1/2	0/6	1/5
I9					0/2	0/2	0/2	0/2					0/2	0/2
I10			0/1		1/1		1/3				0/1		0/1	
I11		2/2	2/5				10/13	1/2		1/1		0/3		
I12			0/1				2/2	1/2			3/3			
I13	2/2	2/2	5/6	3/3	2/4	2/2	10/18	6/6	2/2	1/1				7/7
I14	0/1		3/3	3/3			3/5	4/5	2/2				0/7	

I1: In removal order - at random position, I2: In random order - at random position, I3: In removal order - at best position, I4: In removal order - at best position - with noise, I5: In random order - at best position, I6: In random order - at best position - with noise, I7: Customers with lowest cost first - at best position, I8: Customers with lowest cost first - at best position - with noise, I9: Customers with the n lowest cost first - at best position, I10: In random order - at best position - restricted to random route, I11: Customer with highest route regret - at best position, I12: Customer with highest route regret - at best position - with noise, I13: Customer with highest position regret - at best position, I14: Customer with highest position regret - at best position - with noise



(a) Most frequently used: *I7: Customers with lowest cost first - at best position* (b) Most wins: *I13: Customer with highest position regret - at best position*

Fig. 4. Graphs representing the incomplete pairwise comparison matrix for two insertion operators.

Table 9
Ranking of insertion operators.

#	Insertion operator	w_i	z_i
1	I13: Customer with highest position regret - at best position	18.47	53
2	I12: Customer with highest route regret - at best position - with noise	11.79	8
3	I5: In random order - at best position	9.27	22
4	I14: Customer with highest position regret - at best position - with noise	9.12	26
5	I7: Customers with lowest cost first - at best position	8.62	72
6	I11: Customer with highest route regret - at best position	8.07	26
7	I8: Customers with lowest cost first - at best position - with noise	6.60	29
8	I3: In removal order - at best position	6.31	29
9	I10: In random order - at best position - restricted to random route	5.30	7
10	I1: In removal order - at random position	4.63	7
11	I6: In random order - at best position - with noise	3.76	10
12	I4: In removal order - at best position - with noise	3.68	14
13	I2: In random order - at random position	2.31	11
14	I9: Customers with the n lowest cost first - at best position	2.08	12

evaluates only the insertion positions in one route randomly chosen, but it is likely that the unrestricted operator improves on the solution quality (and therefore is chosen more often). See Table 9 where there is a clear tendency for good ranks for computationally more demanding insertion operators.

We suggest using a measure like the primal integral (PI) as used in the *DIMACS Vehicle Routing Implementation Challenge* (http://dimacs.rutgers.edu/programs/challenge/vrp/). The PI (Berthold, 2013) measures the area under the search trajectory, i.e., it considers the solution process as a function over time and computes the integral of that function. The search trajectory is a step function which decreases every time the algorithm finds a new best solution. The lower the PI, the better the algorithm, as the algorithm either finds a better solution within the same time or the same solution within shorter time.

The runtime of operators could also be taken into account when updating the weights of operators during the search, i.e., runtime intensive operators are called less often. See also Turkeš et al. (2021) for a further discussion on different weight adjustment procedures.

Assess operator and algorithmic component performance through ablation studies. Despite the importance of assessing the performance of algorithmic components, including operators, relatively few articles conduct appropriate analyses. Therefore, we stress the need for conducting performance analyses. We argue that ablation studies are preferable to frequency-based analyses for two reasons. First, frequency-based analyses are not always stable, meaning that the frequencies of operator usage can vary significantly between runs, as pointed out by Mara et al. (2022). Second, ablation-based performance assessments allow for the comparison of the impact of multiple algorithmic components, not just operators. However, when only one component/operator is assessed at a time, interaction effects cannot be captured.

6.2. Improving the presentation of ALNS research

The following recommendations draw inspiration from the limitations of our study as discussed in Section 3.3. These limitations could be remedied to a large extent by improving the presentation of ALNS research.

Include information on unsuccessful operators. Authors often only show results for the “final” ALNS, i.e., the version that they propose to solve the problem under consideration. This means that the reader gets no information on other (less efficient) operators that were implemented and tested but did not make it into the final ALNS. Authors should present information on all operators that were implemented and tested, including the less efficient ones that did not make it into the final ALNS. This would help narrow down the field of potential insertion and removal operators and prevent researchers from repeatedly wasting time and energy on less-efficient operators.

Present standard errors in results. Authors should not only present mean values but also standard errors when reporting the change in objective value when one operator is excluded. This would provide a measure of the robustness of the results and help to assess the validity of pairwise comparisons.

Provide specific details on operators implemented. Authors should be more specific about the operators they implemented to facilitate reproducibility. The clear classification and naming convention can help indicate the most important implementation details.

Publish ALNS code. The best way to avoid misunderstandings and facilitate further development of the ALNS is to publish the code. This would allow for quick testing of operators or concepts on existing implementations without the need for a time-consuming reimplementation based on a potentially inaccurate description in the article. A notable example is the open-source implementation of the Hybrid Genetic Search (HGS) for the CVRP as introduced by Vidal (2022). Subsequent to this publication, the HGS was adopted and improved by various researchers, resulting in an exceptional metaheuristic for VRPs.

7. Research opportunities

This section discusses various research opportunities in the field of ALNS algorithms. Section 7.1 suggests three avenues for exploration within the domain of insertion operators. Moving on to operator selection in Section 7.2, there are several research directions to pursue. Lastly, in Section 7.3 we emphasize the importance of considering the bigger picture of metaheuristics.

7.1. Insertion operators

Implement modular insertion operators. The naming convention of insertion operators highlights the decisions taken to insert customers, i.e., order, position, application of noise, and restrictions on available positions. Instead of defining insertion operators statically, it could be appropriate to define operators by combining these components. The ALNS may then use four modules (possibly each with its own roulette wheel) to dynamically construct the insertion operator. Some modules or options may be (de)activated depending on the instance. For example, to save computation time, it may be worthwhile to restrict the available insertion positions to near routes for large-scale instances.

The following enumeration summarizes several options for each module derived from the previous survey of insertion operators.

- Module **insertion order**
  - removal order
  - random order
  - (n) lowest/highest cost first
  - most constrained first
  - highest demand first
  - farthest/closest to depot first
  - route/position regret
  - impact value
- Module **insertion position**
  - random position
  - best position
  - position with least waiting time
  - position with lowest duration increase
- Module **application of noise**
  - no noise
  - with noise
- Module **restriction on available positions**
  - no restriction
  - restricted to random route
  - restricted to routes in zone
  - restricted to positions on new routes
  - restricted to positions (not) on the previous route
  - restricted to positions on the closest/loosest/shortest route
  - blink mechanism: probabilistically skips the evaluation of positions (Christiaens & Vanden Berghe, 2020).

Implement granular insertion operators. Upon reviewing the list of restrictions on available positions, it becomes evident that there is no insertion operator restricting the insertion of customers solely to related neighbors. Determining related customers can be achieved, for example, by evaluating the distance between them, with closer proximity indicating greater relatedness and an increased likelihood that the edge between the related customers is found in the optimal solution. This approach is inspired by granular tabu search, which constrains local search operators to granular neighborhoods (Toth & Vigo, 2003). Notably, Santana et al. (2022) discovered that granular operators significantly enhance the performance of the hybrid genetic search (Vidal et al., 2012). Therefore, we anticipate that applying this concept to the ALNS will yield comparable benefits.

Use machine learning to determine the insertion order. Regret operators have been widely used and proven to be successful in identifying the insertion order (refer to Table 9). Regret operators give an estimate of how strong the costs will increase if the customer is not inserted right now but another one is preferred. However, they are computationally intensive. A promising research direction could be to estimate the impact of inserting a customer on the subsequent insertion process without calculating actual regret-values. Possibly, machine learning can be utilized to learn from previous insertion steps which customers should be inserted earlier.

7.2. Operator selection

Validate the proposed selection process. An interesting avenue for future research is to assess the proposed selection process based on the ranking of operators (see Section 6.1) and compare it with alternative approaches.

Analyze the performance of operators on different problem and instance types. Building on the selection process, further research is needed into how operators perform across a spectrum of problems and instances. If there is a strong dependency machine learning could facilitate the automated selection of operators based on these characteristics. SATzilla provides an example of this approach (Xu et al., 2008), as it employs a portfolio of algorithms to solve satisfiability problems and uses machine learning to determine the most suitable algorithm for a specific instance based on its characteristics.

Analyze the impact of ALNS implementation and parameters on operator performance. Further extending the analysis of operators, it is also worth investigating if the version of ALNS implemented or the parameter settings have an effect on the operator performance.

7.3. The bigger picture: Metaheuristics in the scientific period

When considering the bigger picture, this article represents a next step towards the scientific period of metaheuristics as termed by Sörensen et al. (2018), where the acquisition of knowledge is more important than the introduction of yet another *novel* metaheuristic (or operator, in our case) with only a marginal increase in performance. Future research should focus on the following aspects.

Analyze metaheuristic components to identify well-performing concepts. Like this article and the works of Santini et al. (2018) and Turkeš et al. (2021), more reviews are needed to synthesize knowledge on metaheuristic concepts in general. This should not be limited to ALNS but should focus on all metaheuristic concepts. The goal is to combine well-performing concepts into powerful hybrid heuristics.

Use and evaluate existing components. Given the already extensive list of removal and insertion operators, one may question whether there was really a need to propose yet another operator, or if previously existing operators would have sufficed. Researchers should work with existing components and evaluate their performance before proposing a new type of metaheuristic/operator for another variant of a VRP, unless there is compelling evidence that the existing concepts do not work. This is especially relevant for application-centered articles where there is nothing wrong with *simply* applying an existing (and well-performing) metaheuristic to a new problem. It is essential to carefully evaluate whether a proposed *novel* component/operator is genuinely new or already exists under a different name, as pointed out by Sörensen (2013) in the context of metaphor-based metaheuristics. To facilitate the comparison of metaheuristics, de Armas et al. (2021) suggest employing a pool template.

Conduct and participate at implementation challenges. Recent implementation challenges such as the DIMACS VRP (http://dimacs.rutgers.edu/programs/challenge/vrp/), NeurIPS dynamic VRP (https://euro-neurips-vrp-2022.challenges.ortec.com/), and Amazon (https://routingchallenge.mit.edu/) challenges have provided excellent opportunities to test algorithmic components within a standardized environment. The research community should make a greater effort to host such challenges to evaluate specific algorithmic components.

8. Conclusion

We analyzed 211 articles to gain insights into the removal and insertion operators used in the ALNS in VRPs. A total of 1266 removal operators (an average of 6 per ALNS) and 788 insertion operators (an average of 3.73) were identified. Of those, 980 removal operators and 608 insertion operators were found to be applicable to VRPs in general. Among those are 57 distinct removal operators and 42 insertion operators. A consistent and simple nomenclature for both types of operators to clearly explain and distinguish the different operators was given. Based on an incomplete pairwise comparison matrix, a ranking of the operators is derived and a slight discrepancy between the most frequently used and the most effective operators was found. Specifically, sequence-based removal operators appear to be superior but are underrepresented. For insertion operators, the most commonly used operator *Insertion: Customers with lowest cost first - at best position* ranks only fifth and has a similar performance to the less computationally demanding *Insertion: In random order - at best position*. Following the ranking of removal and insertion operators we identified a *standard* set of operators consisting of four removal operators (*random customers*, *worst cost customers*, *a posteriori score related customers - to seed customer*, and a sequence-based removal operator, e.g., *all customers - from one of two Kruskal clusters from randomly selected route*) and two insertion operators (*customer with highest position regret - at best position* and *in random order - at best position*). Furthermore, we introduce a more structured process for selecting and evaluating operators based on the ranking derived from network meta-analysis. However, the existing ranking and associated articles usually do not consider runtime. Therefore, we suggest using a criterion that assesses the trade-off between runtime and solution quality for selecting operators in future ALNSs. To improve the presentation of ALNS research, we suggest including information on unsuccessful operators, presenting standard errors, providing more details on operators, and publishing the code. Future research is needed to improve insertion operators and operator selection in the ALNS domain.

Our study demonstrates the possibility of deriving insights and a ranking through network meta-analysis in the context of metaheuristics. Further meta-analyses are required to identify well-performing concepts, for example analyzing components of population-based metaheuristics (e.g., genetic algorithms). Future implementations of metaheuristics may build on these findings by combining several well-performing components from different domains into powerful hybrids.

CRediT authorship contribution statement

Stefan Voigt: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

Acknowledgments

I would like to express my gratitude to Sándor Bozóki for generously sharing the code required to analyze the pairwise comparison matrices, as described in Bozóki et al. (2016). Furthermore, I thank the anonymous reviewers whose very constructive comments helped to significantly improve the paper.

Appendix A. VRP, VRPTW and notation

The basic variants of VRPs are the capacitated VRP (CVRP) and the VRP with time windows (VRPTW). More complex variants include these basic VRPs as special case. Therefore, we briefly introduce both CVRP and VRPTW, as well as the notation used throughout the review. The CVRP constrains the $\mathcal{NP}$-hard traveling salesman problem by introducing a capacity limit. The goal of the CVRP is to find a set of routes K starting from and returning to a depot, that serve all customers $j \in C$ with minimal distance costs. A route k in K, is a sequence of customer visits, starting and ending at the depot, and covering customers in between. Each customer has a demand d_j and is to be served by one vehicle with limited capacity Q. The VRPTW builds upon the CVRP by adding a temporal constraint. Specifically, it imposes hard time windows during which customers must be serviced. Each customer is associated with a time window, defined by the earliest arrival time e_j and the latest arrival time l_j. The service time required at each customer is denoted as s_j. It is possible for a vehicle to arrive earlier than the earliest arrival time, however, it would need to wait. The actual start of service time at customer j in a solution is denoted by S_j. A solution is considered infeasible if a vehicle arrives at a customer after the latest arrival time or if the vehicle exceeds its capacity limit.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ejor.2024.05.033.

References

Adulyasak, Y., Cordeau, J. F., & Jans, R. (2014). Optimization-based adaptive large neighborhood search for the production routing problem. *Transportation Science*, *48*(1), 20–45. http://dx.doi.org/10.1287/trsc.1120.0443.

Alinaghian, M., & Shokouhi, N. (2018). Multi-depot multi-compartment vehicle routing problem, solved by a hybrid adaptive large neighborhood search. *Omega*, *76*, 85–99. http://dx.doi.org/10.1016/j.omega.2017.05.002.

Avci, M. G., & Avci, M. (2019). An adaptive large neighborhood search approach for multiple traveling repairman problem with profits. *Computers & Operations Research*, *111*, 367–385. http://dx.doi.org/10.1016/j.cor.2019.07.012.

Azi, N., Gendreau, M., & Potvin, J. Y. (2014). An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Computers & Operations Research*, *41*, 167–173. http://dx.doi.org/10.1016/j.cor.2013.08.016.

Bakkehaug, R., Rakke, J. G., Fagerholt, K., & Laporte, G. (2016). An adaptive large neighborhood search heuristic for fleet deployment problems with voyage separation requirements. *Transportation Research Part C (Emerging Technologies)*, *70*, 129–141. http://dx.doi.org/10.1016/j.trc.2015.06.019.

Berthold, T. (2013). Measuring the impact of primal heuristics. *Operations Research Letters*, *41*(6), 611–614. http://dx.doi.org/10.1016/j.orl.2013.08.007.

Bozóki, S., Csató, L., & Temesi, J. (2016). An application of incomplete pairwise comparison matrices for ranking top tennis players. *European Journal of Operational Research*, *248*(1), 211–218. http://dx.doi.org/10.1016/j.ejor.2015.06.069.

Bozóki, S., Fülöp, J., & Rónyai, L. (2010). On optimal completion of incomplete pairwise comparison matrices. *Mathematical and Computer Modelling*, *52*(1–2), 318–333. http://dx.doi.org/10.1016/j.mcm.2010.02.047.

Braaten, S., Gjønnes, O., Hvattum, L. M., & Tirado, G. (2017). Heuristics for the robust vehicle routing problem with time windows. *Expert Systems with Applications*, *77*, 136–147. http://dx.doi.org/10.1016/j.eswa.2017.01.038.

Christiaens, J., & Vanden Berghe, G. (2020). Slack induction by string removals for vehicle routing problems. *Transportation Science*, *54*(2), 417–433. http://dx.doi.org/10.1287/trsc.2019.0914.

Coelho, L. C., Cordeau, J. F., & Laporte, G. (2012). The inventory-routing problem with transshipment. *Computers & Operations Research*, *39*(11), 2537–2548. http://dx.doi.org/10.1016/j.cor.2011.12.020.

Contardo, C., Hemmelmayr, V., & Crainic, T. G. (2012). Lower and upper bounds for the two-echelon capacitated location-routing problem. *Computers & Operations Research*, *39*(12), 3185–3199. http://dx.doi.org/10.1016/j.cor.2012.04.003.

Dayarian, I., Crainic, T. G., Gendreau, M., & Rei, W. (2016). An adaptive large-neighborhood search heuristic for a multi-period vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, *95*, 95–123. http://dx.doi.org/10.1016/j.tre.2016.09.004.

de Armas, J., Lalla-Ruiz, E., Tilahun, S. L., & Voß, S. (2021). Similarity in metaheuristics: a gentle step towards a comparison methodology. *Natural Computing*, *21*(2), 265–287. http://dx.doi.org/10.1007/s11047-020-09837-9.

Demir, E., Bektaş, T., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, *223*(2), 346–359. http://dx.doi.org/10.1016/j.ejor.2012.06.044.

Dönmez, S., Koç, Ç., & Altıparmak, F. (2022). The mixed fleet vehicle routing problem with partial recharging by multiple chargers: Mathematical model and adaptive large neighborhood search. *Transportation Research Part E: Logistics and Transportation Review*, *167*, Article 102917. http://dx.doi.org/10.1016/j.tre.2022.102917.

Emeç, U., Çatay, B., & Bozkaya, B. (2016). An adaptive large neighborhood search for an E-grocery delivery routing problem. *Computers & Operations Research*, *69*, 109–125. http://dx.doi.org/10.1016/j.cor.2015.11.008.

Erdem, M. (2022). Designing a sustainable logistics network for hazardous medical waste collection a case study in COVID-19 pandemic. *Journal of Cleaner Production*, *376*, Article 134192. http://dx.doi.org/10.1016/j.jclepro.2022.134192.

Franceschetti, A., Demir, E., Honhon, D., Woensel, T. V., Laporte, G., & Stobbe, M. (2017). A metaheuristic for the time-dependent pollution-routing problem. *European Journal of Operational Research*, *259*(3), 972–991. http://dx.doi.org/10.1016/j.ejor.2016.11.026.

Friedrich, C., & Elbert, R. (2022). Adaptive large neighborhood search for vehicle routing problems with transshipment facilities arising in city logistics. *Computers & Operations Research*, *137*, Article 105491. http://dx.doi.org/10.1016/j.cor.2021.105491.

Ghiami, Y., Demir, E., Woensel, T. V., Christiansen, M., & Laporte, G. (2019). A deteriorating inventory routing problem for an inland liquefied natural gas distribution network. *Transportation Research, Part B (Methodological)*, *126*, 45–67. http://dx.doi.org/10.1016/j.trb.2019.05.014.

Ghilas, V., Demir, E., & Woensel, T. V. (2016). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers & Operations Research*, *72*, 12–30. http://dx.doi.org/10.1016/j.cor.2016.01.018.

Goeke, D., & Schneider, M. (2015). Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research*, *245*(1), 81–99. http://dx.doi.org/10.1016/j.ejor.2015.01.049.

Grangier, P., Gendreau, M., Lehuédé, F., & Rousseau, L.-M. (2016). An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. *European Journal of Operational Research*, *254*(1), 80–91. http://dx.doi.org/10.1016/j.ejor.2016.03.040.

Grimault, A., Bostel, N., & Lehuédé, F. (2017). An adaptive large neighborhood search for the full truckload pickup and delivery problem with resource synchronization. *Computers & Operations Research*, *88*, 1–14. http://dx.doi.org/10.1016/j.cor.2017.06.012.

Gullhav, A. N., Cordeau, J. F., Hvattum, L. M., & Nygreen, B. (2017). Adaptive large neighborhood search heuristics for multi-tier service deployment problems in clouds. *European Journal of Operational Research*, *259*(3), 829–846. http://dx.doi.org/10.1016/j.ejor.2016.11.003.

Gunawan, A., Widjaja, A. T., Vansteenwegen, P., & Yu, V. F. (2021). A matheuristic algorithm for the vehicle routing problem with cross-docking. *Applied Soft Computing*, *103*, Article 107163. http://dx.doi.org/10.1016/j.asoc.2021.107163.

Hellsten, E. O., Sacramento, D., & Pisinger, D. (2020). An adaptive large neighbourhood search heuristic for routing and scheduling feeder vessels in multi-terminal ports. *European Journal of Operational Research*, *287*(2), 682–698. http://dx.doi.org/10.1016/j.ejor.2020.04.050.

Hemmelmayr, V. C., Cordeau, J. F., & Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, *39*(12), 3215–3228. http://dx.doi.org/10.1016/j.cor.2012.04.007.

Hof, J., & Schneider, M. (2019). An adaptive large neighborhood search with path relinking for a class of vehicle-routing problems with simultaneous pickup and delivery. *Networks*, *74*(3), 207–250. http://dx.doi.org/10.1002/net.21879.

Hurkmans, S., Maknoon, M. Y., Negenborn, R. R., & Atasoy, B. (2021). An integrated territory planning and vehicle routing approach for a multi-objective residential waste collection problem. *Transportation Research Record: Journal of the Transportation Research Board*, *2675*(7), 616–628. http://dx.doi.org/10.1177/03611981211030262.

Hutton, B., Salanti, G., Caldwell, D. M., Chaimani, A., Schmid, C. H., Cameron, C., Ioannidis, J. P., Straus, S., Thorlund, K., Jansen, J. P., Mulrow, C., Catalá-López, F., Gøtzsche, P. C., Dickersin, K., Boutron, I., Altman, D. G., & Moher, D. (2015). The PRISMA extension statement for reporting of systematic reviews incorporating network meta-analyses of health care interventions: Checklist and explanations. *Annals of Internal Medicine*, *162*(11), 777–784. http://dx.doi.org/10.7326/m14-2385.

Jiang, D., & Li, X. (2020). Order fulfilment problem with time windows and synchronisation arising in the online retailing. *International Journal of Production Research*, *59*(4), 1187–1215. http://dx.doi.org/10.1080/00207543.2020.1721589.

Johnson, D. (2002). A theoretician's guide to the experimental analysis of algorithms. In *Data structures, near neighbor searches, and methodology: Fifth and sixth DIMACS implementation challenges*.

Kancharla, S. R., & Ramadurai, G. (2018). An adaptive large neighborhood search approach for electric vehicle routing with load-dependent energy consumption. *Transportation in Developing Economies*, *4*(2), 1–9. http://dx.doi.org/10.1007/s40890-018-0063-3.

Keskin, M., & Çatay, B. (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C (Emerging Technologies)*, *65*, 111–127. http://dx.doi.org/10.1016/j.trc.2016.01.013.

Koç, Ç. (2016). A unified-adaptive large neighborhood search metaheuristic for periodic location-routing problems. *Transportation Research Part C (Emerging Technologies)*, *68*, 265–284. http://dx.doi.org/10.1016/j.trc.2016.04.013.

Lahyani, R., Gouguenheim, A. L., & Coelho, L. C. (2019). A hybrid adaptive large neighbourhood search for multi-depot open vehicle routing problems. *International Journal of Production Research*, *57*(22), 6963–6976. http://dx.doi.org/10.1080/00207543.2019.1572929.

Lei, H., Laporte, G., & Guo, B. (2011). The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, *38*(12), 1775–1783. http://dx.doi.org/10.1016/j.cor.2011.02.007.

Li, B., Krushinsky, D., Woensel, T. V., & Reijers, H. A. (2016). An adaptive large neighborhood search heuristic for the share-a-ride problem. *Computers & Operations Research*, *66*, 170–180. http://dx.doi.org/10.1016/j.cor.2015.08.008.

Liu, X., Laporte, G., Chen, Y., & He, R. (2017). An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time. *Computers & Operations Research*, *86*, 41–53. http://dx.doi.org/10.1016/j.cor.2017.04.006.

Majidi, S., Hosseini-Motlagh, S.-M., & Ignatius, J. (2017). Adaptive large neighborhood search heuristic for pollution-routing problem with simultaneous pickup and delivery. *Soft Computing*, *22*(9), 2851–2865. http://dx.doi.org/10.1007/s00500-017-2535-5.

Mancini, S. (2016). A real-life multi depot multi period vehicle routing problem with a heterogeneous fleet: Formulation and adaptive large neighborhood search based matheuristic. *Transportation Research Part C (Emerging Technologies)*, *70*, 100–112. http://dx.doi.org/10.1016/j.trc.2015.06.016.

Mara, S. T. W., Norcahyo, R., Jodiawan, P., Lusiantoro, L., & Rifai, A. P. (2022). A survey of adaptive large neighborhood search algorithms and applications. *Computers & Operations Research*, *146*, Article 105903. http://dx.doi.org/10.1016/j.cor.2022.105903.

Masson, R., Lehuédé, F., & Péton, O. (2013). An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, *47*(3), 344–355. http://dx.doi.org/10.1287/trsc.1120.0432.

Moher, D., Liberati, A., Tetzlaff, J., & and, D. G. A. (2009). Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement. *BMJ*, *339*(jul21 1), b2535. http://dx.doi.org/10.1136/bmj.b2535.

Ortega, E. J. A., Schilde, M., & Doerner, K. F. (2020). Matheuristic search techniques for the consistent inventory routing problem with time windows and split deliveries. *Operations Research Perspectives*, *7*, Article 100152. http://dx.doi.org/10.1016/j.orp.2020.100152.

Özarık, S. S., Veelenturf, L. P., Woensel, T. V., & Laporte, G. (2021). Optimizing e-commerce last-mile vehicle routing and scheduling under uncertain customer presence. *Transportation Research Part E: Logistics and Transportation Review*, *148*, Article 102263. http://dx.doi.org/10.1016/j.tre.2021.102263.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, *34*(8), 2403–2435. http://dx.doi.org/10.1016/j.cor.2005.09.012.

Qu, Y., & Bard, J. F. (2012). A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers & Operations Research*, *39*(10), 2439–2456. http://dx.doi.org/10.1016/j.cor.2011.11.016.

Real, L. B., Contreras, I., Cordeau, J. F., de Camargo, R. S., & de Miranda, G. (2021). Multimodal hub network design with flexible routes. *Transportation Research Part E: Logistics and Transportation Review*, *146*, Article 102188. http://dx.doi.org/10.1016/j.tre.2020.102188.

Ribeiro, G. M., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, *39*(3), 728–735. http://dx.doi.org/10.1016/j.cor.2011.05.005.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455–472. http://dx.doi.org/10.1287/trsc.1050.0135.

Ropke, S., & Pisinger, D. (2006). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, *171*(3), 750–775. http://dx.doi.org/10.1016/j.ejor.2004.09.004.

Sacramento, D., Pisinger, D., & Ropke, S. (2019). An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C (Emerging Technologies)*, *102*, 289–315. http://dx.doi.org/10.1016/j.trc.2019.02.018.

Sacramento, D., Solnon, C., & Pisinger, D. (2020). Constraint programming and local search heuristic: a matheuristic approach for routing and scheduling feeder vessels in multi-terminal ports. *SN Operations Research Forum*, *1*(4), http://dx.doi.org/10.1007/s43069-020-00036-x.

Santana, I., Lodi, A., & Vidal, T. (2022). Neural networks for local search and crossover in vehicle routing: A possible overkill?. http://dx.doi.org/10.48550/ARXIV.2210.12075.

Santini, A., Ropke, S., & Hvattum, L. M. (2018). A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics*, *24*(5), 783–815. http://dx.doi.org/10.1007/s10732-018-9377-x.

Schiffer, M., & Walther, G. (2018). An adaptive large neighborhood search for the location-routing problem with intra-route facilities. *Transportation Science*, *52*(2), 331–352. http://dx.doi.org/10.1287/trsc.2017.0746.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In M. Maher, & J.-F. Puget (Eds.), *Principles and practice of constraint programming — CP98* (pp. 417–431). Berlin, Heidelberg: Springer Berlin Heidelberg, http://dx.doi.org/10.1007/3-540-49481-2_30.

Smith, S. L., & Imeson, F. (2017). GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem. *Computers & Operations Research*, *87*, 1–19. http://dx.doi.org/10.1016/j.cor.2017.05.010.

Sörensen, K. (2013). Metaheuristics-the metaphor exposed. *International Transactions in Operational Research*, *22*(1), 3–18. http://dx.doi.org/10.1111/itor.12001.

Sörensen, K., Sevaux, M., & Glover, F. (2018). A history of metaheuristics. In R. Martí, P. M. Pardalos, & M. G. C. Resende (Eds.), *Handbook of heuristics* (pp. 791–808). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-319-07124-4_4.

Soriano, A., Gansterer, M., & Hartl, R. F. (2018). The two-region multi-depot pickup and delivery problem. *OR Spectrum*, *40*(4), 1077–1108. http://dx.doi.org/10.1007/s00291-018-0534-2.

Ticha, H. B., Absi, N., Feillet, D., & Quilliot, A. (2019). Multigraph modeling and adaptive large neighborhood search for the vehicle routing problem with time windows. *Computers & Operations Research*, *104*, 113–126. http://dx.doi.org/10.1016/j.cor.2018.11.001.

Toth, P., & Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, *15*(4), 333–346. http://dx.doi.org/10.1287/ijoc.15.4.333.24890.

Turkeš, R., Sörensen, K., & Hvattum, L. M. (2021). Meta-analysis of metaheuristics: Quantifying the effect of adaptiveness in adaptive large neighborhood search. *European Journal of Operational Research*, *292*(2), 423–442. http://dx.doi.org/10.1016/j.ejor.2020.10.045.

Vidal, T. (2022). Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers & Operations Research*, *140*, Article 105643. http://dx.doi.org/10.1016/j.cor.2021.105643, URL https://www.sciencedirect.com/science/article/pii/S030505482100349X.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., & Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, *60*(3), 611–624. http://dx.doi.org/10.1287/opre.1120.1048.

Voigt, S., Frank, M., Fontaine, P., & Kuhn, H. (2022). Hybrid adaptive large neighborhood search for vehicle routing problems with depot location decisions. *Computers & Operations Research*, *146*, Article 105856. http://dx.doi.org/10.1016/j.cor.2022.105856.

Voigt, S., Frank, M., Fontaine, P., & Kuhn, H. (2022). The vehicle routing problem with availability profiles. *Transportation Science*, http://dx.doi.org/10.1287/trsc.2022.1182.

Voigt, S., & Kuhn, H. (2021). Crowdsourced logistics: The pickup and delivery problem with transshipments and occasional drivers. *Networks*, *79*(3), 403–426. http://dx.doi.org/10.1002/net.22045.

Wang, L., Kinable, J., & van Woensel, T. (2020). The fuel replenishment problem: A split-delivery multi-compartment vehicle routing problem with multiple trips. *Computers & Operations Research*, *118*, Article 104904. http://dx.doi.org/10.1016/j.cor.2020.104904.

Wen, M., Linde, E., Ropke, S., Mirchandani, P., & Larsen, A. (2016). An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem. *Computers & Operations Research*, *76*, 73–83. http://dx.doi.org/10.1016/j.cor.2016.06.013.

Xu, L., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2008). SATzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, *32*, 565–606. http://dx.doi.org/10.1613/jair.2490.