Yugoslav Journal of Operations Research
25 (2015), Number 1, 3–31
DOI: 10.2298/YJOR140217009K

Invited survey

ADAPTIVE SEARCH TECHNIQUES FOR PROBLEMS IN VEHICLE ROUTING, PART I: A SURVEY

Stefanie KRITZINGER
Department of Production and Logistics, Johannes Kepler University Linz, Austria
stefanie.kritzinger@jku.at

Karl F. DOERNER
Christian Doppler Laboratory for Efficient Intermodal Transport Operations, Department of Business Administration, University of Vienna, Austria
karl.doerner@univie.ac.at.at

Fabien TRICOIRE, Richard F. HARTL
Department of Business Administration, University of Vienna, Austria
fabien.tricoire@univie.ac.at, richard.hartl@univie.ac.at

Received: February 2014 / Accepted: May 2014

Abstract: Research in the field of vehicle routing is often focused on finding new ideas and concepts in the development of fast and efficient algorithms for an improved solution process. Early studies introduce static tailor-made strategies, but trends show that algorithms with generic adaptive policies - which emerged in the past years - are more efficient to solve complex vehicle routing problems. In this first part of the survey, we present an overview of recent literature dealing with adaptive or guided search techniques for problems in vehicle routing.

Keywords: Adaptive Strategies, Local Search, Metaheuristics, Vehicle Routing.

MSC: 90B06, 90C05, 90C08.

1. INTRODUCTION

Metaheuristics and vehicle routing problems (VRPs) are on the one hand, solution procedures and on the other hand, problem types which are strongly

connected. Most of the VRPs are $\mathcal{NP}$-hard and so efficient solution techniques do not exist, but as shown by Garey and Johnson [23], there are no adequate solution techniques for solving them. Therefore, these problem types are perfect applications where metaheuristic search techniques can provide substantial support in tackling them. In fact, with the invention of metaheuristic search a vast range of different VRPs could be solved in a reasonable manner. In the past years, many variations of the classical VRP were introduced and studied. The classical VRP has a central depot and a set of customers which have to be visited by a set of vehicles. Each vehicle has a certain capacity and it can also have a maximum tour length. Several variants of the classical VRP exist, e.g., VRPs with time windows (VRPTW) or open VRPs with (OVRPTW) and without time windows (OVRP). Also, different objective functions, different side constraints, and also different problem structures are considered. Due to the availability of data for the current traffic situation, the problems become even richer.

The application area of VRP has many different problem settings, and therefore a large number of scientists are working on the development of different solution procedures. Some workshops or conferences dedicated to special topics of VRPs have almost 500 participants (e.g. the TRISTAN workshop series), most of the participating researchers are working on solution techniques for variants of the VRP. Although a number of different problem settings exist, some aspects of the problem characteristics are the same in many VRPs. This feature makes the applicability of generic search concepts possible. Nevertheless, it is not easy to find the appropriate search technique or the appropriate operator of a specific problem type. In the past years, adaptive search techniques where introduced to overcome the problem of selecting the most appropriate design decisions a priori.

The first paper introducing a heuristic approach for solving a previous VRP variant is presented by Dantzig and Ramser [18]. They present a procedure based on a linear programming formulation for obtaining near optimal solutions. Shortly after, one of the most popular construction heuristics for routing problems is introduced by Clarke and Wright [14]: the savings algorithm. Starting with single customer routes, routes are merged in a feasible way subject to maximize the cost savings. Few years later, the sweep algorithm is developed by Gillett and Miller [27]. With this approach, routes are generated according to the polar-coordinate angle of each node. At the time of the first calculating machines, the sweep algorithm was already seen as an efficient construction algorithm that competes with similar approaches.

As computers influence the progress in approaches for VRP positively, learning mechanisms are included in search strategies as Ghaziri shows in [26]. Artificial intelligence is used to learn from the previous performance of the algorithm by incrementally adjust their weights in an iterative fashion with mediocre success.

The fundamental ideas of tabu search (TS) are described by Glover [28] in the late eighties for solving various combinatorial problems. Through introducing a *tabu list*, containing moves are forbidden for a certain number of iterations in order to prevent these moves from being reversed. In doing so, cycling behavior around local optima should be avoided. Unlike most other metaheuristics, TS

is a deterministic (non-randomized) algorithm in its basic form. Fundamental ideas, principles, and applications of the TS are summarized by Glover in [29, 30]. Research in TS application to problems in vehicle routing is done through several independent research groups, e.g. Taillard [78], Osman [57], or Gendreau et al. [24]. The novelty of the contemporaneous developed contributions is in the neighborhood type. Gendreau et al. [24] move one customer to another tour. Osman [57] either moves one customer from one tour to another or swaps two customers of two diverse tours. Taillard [78] suggests an exchange of at least two consecutive cities of each tour. In addition, the so-called *taburoute* of Gendreau et al. [24] differs from the TS in the *tabu list*: each move individually receives a random number of iterations being forbidden.

In order to implement an efficient candidate-list strategy, Toth and Vigo [85] introduced the *granular* TS. Restricted neighborhood are called *granular* if they involve only elements that are seen as inefficient in finding promising solutions.

In the beginning of the twentieth century, memory- and evolutionary-based approaches are applied to routing problems. The so-called *BoneRoute*, an adaptive memory-based method, is presented by Tarantilis and Kiranoudis [82]. This method produces new solutions out of sequences of nodes (bones) to receive a population of solutions. In order to guarantee that the pool of solutions does not explode, worse solutions are removed and new solutions are obtained from the remaining. In this period, nature inspired techniques were also applied to VRPs. One nature inspired approach is the ant system. The concept of artificial trail laying, and artificial trail following behavior with pheromone used by ant colonies have been studied in computer science for several years. Reimann et al. [67, 68] present a savings based ant algorithm for solving the capacitated VRP (CVRP).

An efficient evolutionary algorithm for solving VRPs is the genetic algorithm (GA) introduced by Prins [66]. The GA generates solutions using techniques which are inspired by natural evolution, such as inheritance, mutation, selection, and crossover. The GA in Prins [66] outperforms all known metaheuristics that solves large-scale instances with high solution quality. A recent contribution with adaptive strategies by Vidal et al. [87] shows similar achievements. The proposed metaheuristic merges three different search strategies: (i) a complex exploration of population-based evolutionary search, (ii) a neighborhood-based metaheuristic with strong improvement strategies, and (iii) advanced population diversity management schemes. Using this combination, the authors generate new best solutions for all available benchmark instances for the proposed problems. An extension of the resulting multi-attribute VRPs is recently given in Vidal et al. [88].

A metaheuristic, the so-called variable neighborhood search (VNS) proposed by Mladenović and Hansen [53], has gained popularity because of its ability to solve combinatorial problems across a wide range of applications [31, 32, 33, 53]. In particular, the VNS is used to solve various variants of vehicle routing, e.g. the multi-depot VRPTW (MDVRP) [64], the periodic VRP (PVRP) [35], or the dial-a-ride problem [58].

Finally, recent trends show that algorithms with generic adaptive mechanisms

are widely-used. The most popular and very successful adaptive approach is the adaptive large neighborhood search (ALNS) developed by Ropke and Pisinger in [70] and [71]. A clever selection mechanism is used to favor the most successful operators. This strategy is adapted to various metaheuristics, e.g., the VNS. Recently, Stenger et al. [74] implement an adaptive VNS (AVNS) using a similar selection method inspired by ALNS. A very simple way to add an adaptive manner to a metaheuristic is done with TS: a parameter, which guides the solution process, is updated in every iteration.

The focus of Part I of this survey is on recent contributions of algorithms with generic adaptive mechanisms. We consider as *adaptive* if it modifies the parameters of an optimization algorithm during the search, based on information that was not available before the beginning of the search. To begin in Section 2, basic local search-based concepts are presented. In Section 3, hybrid local search-based methods, e.g. iterated local search (ILS) and AVNS, are discussed. We describe the large neighborhood search (LNS) and proceed with the ALNS in Section 4. Section 5 presents adaptive mechanisms in population-based methods. A list of abbreviations of all used routing variants is provided in the appendix in Table 11.

2. BASIC LOCAL SEARCH CONCEPTS

Since TS is a very popular algorithm based on local search, an important portion of the mechanisms described in this section are either based on TS or applied to TS. Before discussing adaptive strategies in TS, two general mechanisms of basic local search concepts, the reactive search (RS) and the guided local search (GLS), are described.

RS is a general mechanism to adapt and tune the parameters of local search methods based on search history. General descriptions can be found in Battiti [7] and Battiti and Brunato [8]. An important idea of RS is to dynamically modify the behavior of a basic algorithm according to contextual needs in diversification or intensification. In the case of TS, both diversification and intensification are decided by the tabu tenure, also called tabu list size; therefore applying RS to TS requires to dynamically modify the tenure. This is done, e.g., in Battiti and Tecchiolli [9]: the tabu tenure is increased when previously visited configurations are repeated, thus providing extra diversification. If no previously visited configuration is repeated for some time, the tenure is decreased in order to rebalance the search towards more intensification. Additionally, when it occurs too often that previous states are revisited, an escape mechanism is triggered, which consists in performing random moves. Overall, this method could also be seen as ILS, which includes random moves fulfilling the role of perturbation (see Section 3).

Another adaptive generalization of local search, the GLS, as described in Voudouris et al. [89], aims at guiding the local search towards promising regions of the search space. This is implemented by analyzing so-called *features*, e.g. the use of arcs in routing optimization, and penalizing some of these features in order to drive the search toward more promising regions of the search space. An

important step in GLS is to define these features. Penalized features are those that have a bad contribution to the objective function and have not been penalized too much yet, i.e., those features i of solution x that maximize the following utility function (assuming a minimization problem):

$$U_i(x) = I_i(x)\frac{c_i}{1+p_i} \tag{1}$$

where c_i is the cost of feature i, p_i is the current penalty of feature i, and $I_i(x)$ is 1 if x exhibits feature i, and 0 otherwise. An extension to GLS, the extended GLS (EGLS), adds aspiration criteria and random moves to GLS as in Mills et al. [52].

The definition of the use of arcs as features is done, for instance, in Leung et al. [46], in Tarantilis et al. [83], in Zachariadis et al. [91, 92]. In [91, 92], GLS is embedded in a TS: the evaluation function of TS is modified following the GLS paradigm. In [83], GLS is used in a steepest descent fashion and called multiple times with different neighborhoods after subsequent changes to the solution. In [46], EGLS is applied to TS.

There are also other kinds of adaptive TS (ATS) approaches in the literature. A common concept in local search is to accept infeasible solutions during the search process, while incurring a penalty in the objective value, typically by multiplying a measure of constraint violation by a certain factor. Several contributions adapt such factors dynamically during the search. In Potvin [65], the capacity constraint is relaxed and excess load is multiplied by a factor, and added up in the evaluation function. At each iteration, this factor is either increased (if the incumbent solution is infeasible) or decreased (if it is feasible). In Di Gaspero and and Schaerf [20], two constraints are relaxed and the weights for penalizing them are increased or decreased depending on (in)feasibility of the solution. However, such modifications only happen after (in)feasibility is consistent over several iterations. This is a similar mechanism to that introduced by Anagnostopoulos et al. [1].

Interestingly, all these ATS methods consist in modifying some parameters after solution evaluation, such as penalty factors for infeasibility, penalty for attributes, or tabu list size. Therefore, a very simple way to express ATS in a general manner is to add a parameter update step after the solution evaluation step in each iteration. For the sake of completeness, we provide an abstract algorithm fitting all previously mentioned adaptive tabu search methods in Algorithm 1. It is freely inspired from the generic TS algorithm from Stützle and Hoos [75].

After initializing the starting solution and the best solution found (lines 1 and 2), the main loop consists in iteratively constructing the set of admissible neighbors (line 4), selecting one best admissible neighbor as a new incumbent (line 5), updating the best solution found (lines 6-8) and updating the adaptive mechanisms (line 9). It is noteworthy that the construction of the admissible neighbors of x takes both the search history and x^* as parameters. The history allows prohibiting tabu neighbors, while x^* allows overriding tabu status for aspiration criteria.

Algorithm 1 (Adaptive tabu search).

$x \leftarrow$ *constructionHeuristic*()
$x^* \leftarrow x$
while *stopping criterion not met* ***do***
  $X' \leftarrow$ *admissibleNeighbors*$(x, history, x^*)$
  $x \leftarrow$ *selectBest*(X')
  if $z(x) < z(x^*)$ ***then***
    $x^* \leftarrow x$
  end if
  updateAdaptiveMechanisms(*history*)
end while
return x^*

Recent trends show that the GLS approach is the most popular adaptive generalization of TS. In Table 1, we present a selection of the last years contribution on GLS-based approaches, and in Table 2, the used acronyms are described. The values titled $\{n_{min}; n_{max}\}$ in Tables 1, 3, 6, 7, and 10 indicate the minimum and maximum number of nodes considered in the particular contribution.

The hybrid framework in Tarantilis et al. [83] combines three different metaheuristic strategies: VNS introduced by Mladenović and Hansen [53], TS by Glover [28], and GLS by Voudouris et al. [89]. After defining the neighborhood structures and generating an initial solution, the TS, which acts as local descent within the VNS block, achieves an efficient interplay between diversification and intensification. The VNS systematically changes the neighborhood operators while the local search is applied by TS. The GLS method removes low-quality features from the solution and reinserts the removed nodes. The source of inspiration comes from Mester and Bräysy [50]: (i) low quality features of the solution are selected, (ii) modified penalization terms are used for augmentation of the objective function, and (iii) a different customer removal and reinsertion procedure is used to rearrange the routing schedule. Arc (ij) with cost c_{ij} is penalized with the utility function

$$U(ij) = \frac{c_{ij}/avg_{ij}}{1 + p_{ij}}, \tag{2}$$

where p_{ij} is the number of times that arc (ij) has been penalized and avg_{ij} is a cost measure of the relative distance of nodes i and j. Compared to a methodology based on adaptive memory and TS Crevier et al. [16], some best known solutions can be improved up to 0.41% by the proposed metaheuristic.

Table 1: Guided local search

Contribution $\{n_{min}; n_{max}\}$	Problems tackled	Objective function	Operators	Guided mechanism	Parameters	Solution quality	Termination criteria
Tarantilis et al. [83], 2008 {48;216}	VRPIRF	minimize total travel costs	*CR, RSC, RSI*	penalize arc with highest value of utility function	utility values	competitive algorithm finds best solutions	VNS: 20 it. TS: 30 it. GLS: 6000 it.
Zachariadis et al. [91], 2009 {15;255}	2L-CVRP	minimize total travel costs	*CRR, RE, RSC*	penalize arc with highest value of utility function	utility values	competitive algorithm solves a wide variety of benchmark instances	7000 non-improving TS it.
Zachariadis et al. [92], 2009 {50;400}	VRPSPD	minimize total travel costs	*CR, RSC, RSI*	penalize arc with highest value of utility function	utility values	competitive algorithm to state-of-the-art methods	6000 non-improving TS it.
Leung et al. [46], 2011 {15;255}	2L-CVRP	minimize total travel costs	*CRR, RE, RSC*	penalize arc with highest value of utility function	utility values	effective performance	10000 it.

Table 2: List of acronyms of guided local search neighborhoods

CR	Customer relocation relocates a customer from its current place to another.
CRR	Customer relocation relocates a customer from its route to another route.
RE	Route exchange swaps two customers of two different routes.
RSC	Route segment crossover is described as a 2-opt move or a customer exchange within a route, or a 2-opt* move within two routes.
RSI	Route segment interchanging exchanges route segments that cover a pair of customers; in other words: two customers are exchanged within two segments.

In Zachariadis et al. [91], an interaction between TS and GLS is proposed to solve a capacitated VRP with two-dimensional loading constraints. The guiding mechanism within the TS controls the objective function by penalizing low-quality featured arcs resulted through the utility function (see Equation (2)). Due to the guiding mechanism, the solution cost can be reduced about 4% compared to the same TS without any guiding strategy. Compared to a competitive TS of Gendreau et al. [25], the GLS-based TS is able to improve some of the best known solutions, but it is not successful for every instance.

A similar approach as in Tarantilis et al. [83] and in Zachariadis et al. [91] is used by Zachariadis et al. [92]. The TS- and GLS-based hybrid metaheuristic was successfully applied to various benchmark instances and, e.g., compared to the TS algorithm of Tang and Galvão [81], the average solution value can be improved by 0.6%.

Leung et al. [46] present a metaheuristic methodology that incorporates theories of TS and EGLS. The authors follow Zachariadis et al. [91] to implement the guiding strategy within the TS algorithm. Results show that optimizing by using the aspiration criterion leads to significant improvements.

It has to be mentioned, that the work of Cordeau and Maischberger [15] is classified here as well (see Section 3).

3. HYBRID LOCAL SEARCH CONCEPTS

As the name indicates, ILS consists of iterative calls to a local search method. In each iteration, the incumbent solution is perturbed and the local search is performed on the perturbed solution. Then a decision is made as to whether the newly found local optimum should become the incumbent solution or not. This whole process is iterated a number of times, and then the best solution found during the whole process is returned. Readers interested in details and discussions about ILS should consult Lourenço et al. [48].

Algorithm 2 (Iterated local search).

1: $x \leftarrow constructionHeuristic()$
2: $x \leftarrow localSearch(x)$
3: $x^* \leftarrow x$
4: ***while*** *stopping criterion not met* ***do***
5: $\quad x' \leftarrow perturbation(x, history)$
6: $\quad x'' \leftarrow localSearch(x')$
7: $\quad$ ***if*** $acceptanceDecision(x, x'', history)$ ***then***
8: $\quad\quad x \leftarrow x''$
9: $\quad\quad$ ***if*** $z(x) < z(x^*)$ ***then***
10: $\quad\quad\quad x^* \leftarrow x$
11: $\quad\quad$ ***end if***
12: $\quad$ ***end if***
13: ***end while***
14: ***return*** x^*

We outline the basic steps of ILS in Algorithm 2, which is inspired by the algorithm provided in [48]. It is assumed that the optimization problem at hand is a minimization problem. First of all, an initial solution x has to be constructed (line 1) before a local search mechanism improves it (line 2). The incumbent solution is denoted as x, while the best found solution is x^*; $z(x)$ denotes the objective value of solution x. As long as the stopping criterion is not met, a perturbation is performed to get x' (line 5), and a local search heuristic improves the solution to x'' (line 6). If x'' passes the acceptance decision, it becomes the new incumbent (line 7–8). If the new incumbent improves the best found solution x^*, then x^* is updated accordingly (line 9–10).

We can already note that the search history is incorporated in the perturbation as well as in the acceptance decision, which allows for adaptive versions of ILS. The stopping criterion can be for instance a predetermined number of iterations, a predetermined CPU budget, or a given number of iterations without improvement.

In Table 3, recent literature dealing with efficient ILS-based algorithms are listed. The operators of Table 3 are explained in Table 4.

An ILS algorithm combined with a variable neighborhood descent and random neighborhood ordering (ILS-RVND) is discussed by Penna et al. [59], and shortly after by Subramanian and Battarra in [76]. As long as the neighborhood list is not empty, a neighborhood is randomly selected and the best admissible move is determined. The neighborhood list varies in the following way: if a neighborhood does not improve the solution, the neighborhood is removed from the list; otherwise, all removed neighborhoods are returned to the list for being again randomly selected. The ILS-RVND in [59] is compared with various algorithms, e.g., two instances, of which the solution was not proven to optimality of the unified exact method of Baldacci and Mingozzi [5], could be improved.

Compared to a heuristic approach based on a branch-and-cut procedure of Hernandez-Perez and Salazar-González [36], the ILS-RVND of Subramanian et al. [76] finds new best solutions, but the execution time is higher. An extension of the ILS-RVND algorithm [76] with an exact procedure based on a set partitioning formulation (ILS-RVND-SP) is developed by Subramanian et al. [77]. The interaction between a mixed integer programming solver and an ILS-based approach allows that different benchmark instances of VRP variants. As a unified framework, the performance of ILS-RVND-SP is compared with several metaheuristics and hybrid approaches, for example the ALNS in Pisinger and Ropke [62] and Ropke and Pisinger [70] or a hybrid genetic algorithm of Vidal et al. [87].

Table 3: Iterated local search

Contribution $\{n_{min}; n_{max}\}$	Problems tackled	Objective function	Operators	Additional mechanism	Parameters	Solution quality	Termination criteria
Penna et al. [59], 2011 {20;100}	HFVRPFD HVRPD FSMFD FSMF FSMD	minimize total traveling costs	shift(λ,0), k-shift, swap(λ_1,λ_2), cross, reinsertion, Or-opt, 2-opt, split, exchange	updating neighborhoods due to behavior: repopulation of neighborhood, if improvement; removing neighborhood if non-improving	neighborhood list	4 new best solutions	depending on n and m[1]
Cordeau et al. [15], 2012 {27;1008}	CVRP VRPTW PVRP(TW) MDVRP(TW) SDVRP(TW)	minimize total travel costs	relocate, 2-opt	violation weights: increasing if violation, decreasing if no violation	penalizing non-improving moves; self-adaption of violation weights	competitive with most recent heuristics	10^5 it. 10^6 it.
Subramanian et al. [76], 2012 {20;500}	TSPPD	minimize total traveling costs	Or-opt, 2-opt, exchange, relocate, double bridge	updating neighborhoods due to behavior: repopulation of neighborhood, if improvement; removing neighborhood if non-improving	neighborhood list	new best solutions	$25 * n/4$ it.
Subramanian et al. [77], 2013 {34;483}	CVRP ACVRP OVRP VRPSPD MDVRP MDVRPMPD	minimize total travel costs	shift(λ,0), shiftDepot, swap(λ_1,λ_2), swapDepot, cross, reinsertion, 2-opt, Or-opt, exchange	updating neighborhoods due to behavior: repopulation of neighborhood, if improvement; removing neighborhood if non-improving	neighborhood list	new best solutions	depending on number of customers; 2000 it.
Michallet et al. [51], 2014 {5;150}	PVRPTS	minimize total travel costs while dispersing arrival times	relocate, swap, 2-opt, 2-opt*	updating penalty coefficients of evaluation function	penalty coefficients	new best solutions	max. number of nonimproving it.

[1] m . . . number of routes

Table 4: List of operators used in ILS

2-opt	The 2-opt heuristic in Croes [17] iteratively inverts sequences of nodes.
2-opt*	The 2-opt* removes two arcs in the same rout or in two different routes with two other arcs.
cross	The cross operator in Penna et al. [59] is a 2-opt* operator and exchanges the last parts of two routes.
double bridge	The double bridge operator in Martin et al. [49] removes four randomly chosen edges and reconnects with four alternative edges.
exchange	The exchange move swaps two nodes within a tour.
k-shift	A subset of consecutive nodes k is transferred from a route to another one in Penna et al. [59].
Or-opt	The Or-opt of Or [56] heuristic iteratively moves subsequences up to a sequence length of three nodes.
reinsertion	The reinsertion operator moves a node from a position to another position within its route.
relocate	The relocate operator moves one node to another place in the tour.
shift(λ,0)	λ consecutive nodes are moved from one route to another one in Penna et al. [59].
shiftDepot	The ShiftDepot operator of Subramanian et al. [77] moves a depot node from one route to another one.
split	A route is divided into smaller routes in Penna et al. [59].
swap(λ_1,λ_2)	The swap(λ_1,λ_2) operator in Penna et al. [59] is a cross exchange heuristic (see Table 5).
swapDepot	The swapDepot operator in Subramanian et al. [77] swaps two depots of two routes.

In their recent article, Cordeau and Maischberger [15] describe a parallel iterated TS (ITS) heuristic for solving four different routing problems. As this is a hybrid search technique of ILS and TS, we decided to mention it in this section. The method combines TS with a simple perturbation mechanism. It competes with recent heuristics designed for each particular problem. The objective function is the sum of total routing costs plus the total weighted violation of capacity, duration, and time window constraints. The corresponding weights of the violation terms are self-adapting: if violation occurs, the value is increased; otherwise, the value is decreased. Furthermore, non-improving moves are penalized depending on the current iteration number. Compared to the best known methods for the classical VRPTW benchmark instances, it performs fast and competitive results.

Michallet et al. [51] solve the PVRP with time spread constraints on services (PVRPTS) with a hybrid combination of a mixed integer linear model and a multi-start ILS.

The VNS relies on a systematic change of neighborhoods to escape local optima and provide a broad exploration of the search space. After designing a set of *shaking* neighborhoods and constructing an initial solution, it consists in iteratively (i) shaking an incumbent solution, (ii) performing local search on it, and (iii) deciding whether to accept it as a new incumbent or not. The name of the method comes from the fact that the neighborhood used for the shaking phase changes systematically during the search process.

Assuming κ neighborhoods named $N_1, \ldots, N_\kappa$ are designed, then the search starts by using N_1. If no improvement is found when using N_k ($k \in 1, \ldots, \kappa$), then the next neighborhood used for shaking will be N_{k+1}; on the other hand, whenever an improvement is found, the shaking neighborhood is reset to N_1. VNS is outlined in Algorithm 3.

Algorithm 3 (Variable neighborhood search).

```
1:  x ← constructionHeuristic()
2:  x* ← x
3:  k ← 1
4:  while stopping criterion not met do
5:     x' ← N_k^r(x)
6:     x'' ← localSearch(x')
7:     if acceptanceDecision(x, x'') then
8:        x ← x''
9:        k ← 1
10:       if z(x) < z(x*) then
11:          x* ← x
12:       end if
13:    else
14:       k ← 1 + (k mod κ)
15:    end if
16: end while
17: return x*
```

After the initial solution x is constructed and the parameters are initialized (line 1–3), a random neighbor of x using neighborhood N_k is generated (line 5). We use the additional notation $N_k^r(x)$. Local search is performed to improve the solution to x'' (line 6). If x'' passes the acceptance decision, it becomes the new incumbent (line 7–8) and the shaking neighborhood is reset to N_1 (line 9). If the new incumbent improves the best found solution x^*, then x^* is updated accordingly (line 10–11). Otherwise, if x'' fails the acceptance decision, the search continues with the next neighborhood N_{k+1} (line 12).

For a good introduction to VNS, see Hansen and Mladenović [31]. A common practice is to use so-called *nested* neighborhoods $N_1 \subseteq N_2 \subseteq, \ldots, \subseteq N_k$. This way, research is biased towards smaller neighborhoods, and large neighborhoods are only used when the small ones fail to provide a new acceptable solution.

Over the last years, a number of VNS methods integrating adaptive aspects have been developed. In most cases, the adaptive aspect concerns the shaking phase: shaking is performed differently depending on the context. For instance, in Pillac et al. [60] and in Stenger et al. [74], the shaking method is selected using a roulette wheel where the weight for each neighborhood is based on its success rate in previous iterations. In Polacek et al. [63], the size of the neighborhood as well as the acceptance rate of ascending moves are automatically adapted through the search. However, other possibilities exist. In Hsiao et al. [38], for instance, the CPU budget allocated to local search is adapted dynamically. There is no unified

approach for AVNS so, we suggest a generic outline in Algorithm 4, which covers all of the contributions we are aware of. It involves potential adaptive mechanisms at all three steps of any iteration, namely shaking (line 4), local search (line 6) and acceptance decision (line 7).

Algorithm 4 (Adaptive variable neighborhood search).

```
1:  x ← constructionHeuristic()
2:  x* ← x
3:  while stopping criterion not met do
4:      N ← selectShaking(history)
5:      x' ← N^r(x)
6:      x'' ← localSearch(x', history)
7:      if acceptanceDecision(x, x'', history) then
8:          x ← x''
9:          if z(x) < z(x*) then
10:             x* ← x
11:         end if
12:     end if
13: end while
14: return x*
```

The operators of Table 6 are explained in Tables 4 and 5.

Table 5: List of operators used in VNS

3-opt	The 3-opt operator of Lin [47] removes three edges and reconnects with three alternative edges.
cross exchange	The cross exchange operator of Taillard et al. [79] takes two segments of different routes and exchanges the sequences.
cyclic-exchange	The cyclic-exchange operator in Thompson and Psaraftis [84] simultaneously moves nodes among routes in a cyclic way.
icross exchange	The icross exchange operator in Bräysy and Gendreau [11] takes two segments of different routes, exchanges and inverts the sequences.
λ-interchange	The λ-interchange operator of Osman [57] moves a sequence of λ nodes from one route to another.
relocate op.	The relocate operator is a special case of the λ-interchange operator of Osman [57] and moves one node from one route to another.
sequence displacing	The sequence displacing displaces a sequence of nodes with or without inversion.
string exchange	The string exchange operator in Irnich et al. [39] takes two segment of nodes within one route and exchanges them.
swap	The swap operator in Irnich et al. [39] exchanges two nodes.

Table 6: AVNS

Contribution $\{n_{min}; n_{max}\}$	Problems tackled	Objective function	Operators	Adaptive mechanism	Parameters	Solution quality	Termination criteria
Polacek et al. [63], 2008 {48;288}	MDVRPTW	minimize total traveled distance	cross exchange, icross exchange, 3-opt	roulette wheel selection depending on success for neighborhood and ascending move parameter	weights of roulette wheel	solution quality improved by 1.12% compared to other results	10^6, 10^8 it.
Hosny et al. [37], 2010 {20;500}	1-PDP	minimize total distance * (1 + violation of capacity)	sequence displacing, 2-opt	maximum neighborhood size depends on the current stage of the search	maximum neighborhood size	improvement compared to recent methods	no benefit or given number of it.
Stenger et al. [74], 2012 {5;360}	MDVRPPC MDVRP VRPPC	minimize vehicles' fixed costs + travel costs + subcontracting costs	51 neighborhood structures based cyclic-exchange, 2-opt, Or-opt, swap, relocate op.	roulette wheel selection mechanism (inspired by ALNS: adaption of probability depending on the success of a neighboring solution)	weights of roulette wheel mechanism	simple adaptive mechanism considerably improves the performance of VNS on various related RPs	2500 sek.
Pillac et al. [60], 2012 {30;60}	DVRPSD	minimize total costs	swap, string exchange, 2-opt, Or-opt	neighborhoods are explored randomly selected with a bias depending on their previous performance; an average ratio of the improvement to time as a metric of neighborhood performance is used to maintain the information between calls to the optimization procedure	neighborhood size	competitive with state-of-the-art methods	no impr. of all neighborhoods

The AVNS presented by Polacek et al. [63] has the ability to self-adapt the most influential parameters of a VNS algorithm: the selection of neighborhood structures and the threshold parameter used in the acceptance decision. For both self-adapting parameters, a counter x_i is used to score the success of the parameter i. If an improvement can be achieved, the particular counter is increased by 1. The roulette wheel method selects the parameter values for each defined part of the solution process. The probability $P(i)$ for every parameter value i is calculated by the following function:

$$P(i) = \frac{\ln(x_i)}{\sum_{j\in\Omega} \ln(x_j)} \; \forall \; j \in \Omega \tag{3}$$

where Ω describes the set of all possible parameter values of each self-adapting parameters. In order to avoid the dominance of a specific parameter setting, the natural logarithm is applied. Compared to the previous VNS version, Polacek et al. [64] without self-adapting parameters, an average improvement of 0.28% can be obtained.

Hosny et al. [37] also use a simple adaptive strategy for the neighborhood size. The authors perform VNS runs repeatedly, while the initial solution is the final solution of the previous VNS run. They find out that in the beginning of the whole solution process, comprising multiple runs, larger neighborhood sizes quickly provide improvements, whereas later on smaller neighborhoods seem to be more beneficial. After each VNS run, the neighborhood size is reduced by a quarter of the initial maximum neighborhood size, until the predefined lower bound of a quarter of the initial neighborhood size is met. The initial neighborhood size is set to $2 \times \sqrt{n}$, where n is the total number of nodes. An AVNS run stops after a fixed number of iterations or a given number of non-improving iterations. To put it simply, the multiple VNS runs can be interpreted as one VNS run with reducing maximum neighborhood size after a given number of iterations or a fixed number of non-improving iterations. Improvements up to 3% are obtained compared to the genetic algorithm in Zhao et al. [94].

Recently, Stenger et al. [74] present an AVNS algorithm obtained by incorporating an adaptive mechanism inspired by the roulette wheel method in the ALNS of Pisinger and Ropke [62] which is described in Section 3. Due to the roulette wheel selection method, the AVNS guides the shaking step to areas where high quality solutions are expected by biasing the random shaking step of VNS. In particular, the authors define two selection decisions that are independently performed: (i) a route selection chooses the routes to be involved, and (ii) a customer selection chooses the customers to be exchanged. In total, 51 different neighborhood structures are used. After applying a method in the shaking phase, a scoring system evaluates the success of each neighborhood i in a segment of 30 iterations and adds scores to the counter x_i: (i) a score of nine is added whenever a new overall best solution is found, (ii) a score of three is added, if the current solution is improved, and (iii) a score of one is added if the solution is worse than the current, but is accepted by the simulated annealing criterion. A reaction

factor $\rho = 0.3$ allows for controlling the adaptive behavior of the algorithm to the recent trend with exponential smoothing. Before the roulette wheel mechanism is performed, the weights π_i, which are initialized equally, need to be adapted:

$$\pi_i = \rho \frac{x_i}{\chi_i} + (1 - \rho)\pi_i, \tag{4}$$

where χ_i is the number of times the neighborhood has been called in the current segment. The probability $P(i)$ for every neighborhood structure in the set Ω of all possible parameters Ω is calculated by the following function:

$$P(i) = \frac{\pi_i}{\sum_{j \in \Omega} \pi_j}. \tag{5}$$

The adaptive mechanism considerably improves performances with respect to both solution quality of real-world instances and convergence speed compared to a commercial solver.

An AVNS is used within an event-driven optimization framework by Pillac et al. [60]. In this algorithm, neighborhoods are not explored in sequential order, as it is defined in the original VNS algorithm, Mladenović and Hansen [53], but are rather selected randomly with a bias depending on their previous success. Neighborhoods with higher success are chosen frequently, while neighborhoods which lead to less improvements are chosen to a lesser extent. As in Polacek et al. [63] and Stenger et al. [74], the parameters are adapted with a roulette wheel method. The stopping criterion is the same as in Penna et al. [59] and in Subramanian et al. [76]: the process iterates until all neighborhoods have been explored with no improvement. Computational experiments show that this approach is competitive with state-of-the-art algorithms, e.g., a dynamic programming approach in Novoa et al. [55].

4. LARGE NEIGHBORHOOD SEARCH

The LNS is a specialization of the concept of local search to so-called *large* neighborhoods. In LNS, the neighborhood considered is the set of solutions that can be obtained by destroying large portions of an incumbent solution x, and then repairing this partial solution to make it a feasible solution to the whole optimization problem at hand. The terms *destroy* and *repair* can be substituted with *ruin* and *recreate*, as similar concepts were published under different names in Schrimpf et al. [72] and in Shaw [73]. Since there are many ways of destroying and repairing a solution, the neighborhood is very large. Hence, it is explored heuristically and destroy and repair heuristics are designed for that purpose. Then LNS consists in iteratively (i) selecting a pair of destroy and repair operators, (ii) applying them to the incumbent solution, and (iii) deciding to accept or not the new solution.

ALNS, introduced in Ropke and Pisinger [71], adds an adaptive mechanism to the step where the operators are selected, by using search history to favor the

most successful operators. Using the previously introduced notation and style, we outline ALNS in Algorithm 5. The algorithm has to be initialized with the construction of a starting solution x (line 1). At every iteration, a destroy operator d and a repair operator r need to be selected (line 4). The adaptive aspect of ALNS lies in the *history* parameter on line 4: without this parameter, the algorithm describes LNS. Then d destroys x and r repairs $d(x)$ (line 5). A new solution x' is obtained. If x' passes the acceptance decision, it becomes the new incumbent (line 6–7). If the new incumbent improves the best found solution x^* then x^* is updated accordingly (line 8–9).

Algorithm 5 (Adaptive large neighborhood search).

```
1:  x ← constructionHeuristic()
2:  x* ← x
3:  while stopping criterion not met do
4:      (d, r) ← selectOperators(history)
5:      x' ← r(d(x))
6:      if acceptanceDecision(x, x', history) then
7:          x ← x'
8:          if z(x) < z(x*) then
9:              x* ← x
10:         end if
11:     end if
12: end while
13: return x*
```

A collection of recent and important ALNS contribution is summarized in Table 7, and a description of the destroy and the repair operators can be found in Table 8.

Although the ALNS is introduced in Ropke and Pisinger [70, 71], the mostly cited paper discussing ALNS is presented by Pisinger and Ropke [62]. The algorithm is able to solve several variants of VRPs. The key to success of this unified framework is the strategy of choosing the destroy and the repair neighborhoods due to their success in the past. The adaptiveness lies in a simple roulette wheel mechanism to update the probability for each operator to be chosen: the more successful an operator N_i is, the more its score x_i is increased; the less contribution an operator N_i has, the less its score x_i is increased. Scores are updated every time a time segment of 100 iterations is started. Information from past time segments is kept by updating the score parameters using the reaction factor $\rho = 0.1$ (see Equation (4)). The probability $P(i)$ for choosing operator $N_i \in \omega$ is calculated as in Equation (5). The ALNS algorithm has also been applied to different research areas only with slight parameter changes. Furthermore, destroy and repair operators are previously treated independently, but recent trends, e.g. Kovacs et al. [42], have shown that dependent considerations of neighborhoods pairs are efficient as well. The discussed ALNS is a competitive approach solving different variants of VRPs and obtaining new best solutions, e.g., for the VRP with time windows.

Table 7: Adaptive large neighborhood search

Contribution $\{n_{min}; n_{max}\}$	Problems tackled	Objective function	Operators	Adaptive mechanism	Parameters	Solution quality	Termination criteria
Pisinger et al. [62], 2007 {100;1000}	CVRP,OVRP MDVRP RPDPTW SDVRP VRPTW	min. travel costs and the number of vehicle used	*RaR, WR, RelR, CR, TOR, HNPR, HRPR, GI, RI*	roulette wheel selection mechanism; destroy and repair operators are weighted and chosen independently	weights of roulette wheel	new best solutions for the benchmark instances of VRPTW	25000, 50000 it.
Lei et al. [45], 2011 {12;50}	CVRPSDTW	min. sum of total routing and expected cost of recourse because of overload	*SimR, FOR, EWR, RaR, GI, FOI, DFSI, TFSI*	roulette wheel selection depending on success; destroy and repair operators are weighted and chosen independently	weights of roulette wheel	no comparative data available	numb. of non-impr. it.
Azi et al. [3], 2012 {72;144}	DVRPM	max. total profit: total gain of served customers minus total traveled distance	*RaR, RelR, RouR, RelRou, WoR, LCH, RI*	roulette wheel selection depending on success; destroy and repair operators are weighted and chosen independently	weights of roulette wheel	comparison between myopic and non myopic approach	24000 it.
Demir et al. [19], 2012 {10;200}	PRP	min. fuel consumption, emissions and driver costs	*RaR, WDR, PR, TR, DR, HR, NR, ZR, NNR, GI, RI, GIN, RIN, ZI*	roulette wheel selection depending on success	weights of roulette wheel	percentage deviation smaller than 0.72 %	25000 it.
Hemmelmayr et al. [34], 2012 {21;200}	2E-VRP LRP	minimize opening cost of depots and routing costs	*SatR, SatO, SatS, RaR, WR, RelR, RouR, RouRe, GI, GIP, GIF, RI, FLLS*	roulette wheel selection depending on success; destroy and repair operators are weighted and chosen independently	weights of roulette wheel	new best solutions, competitive results	$5x10^5$, $5x10^6$
Kovacs et al. [42], 2012 {25;100}	STRSP	min. sum of total routing and outsourcing costs	*RaR, WR, ReR, CR, GI, SIH, RI*	roulette wheel selection depending on success; scores and weights for pairs of destroy and repair operators	weights of roulette wheel	as good or even slightly better than previous algorithms	25000 it.
Ribeiro et al. [69], 2012 {50;420}	CCVRP	min. sum of arrival times at customers	*SRHBAT, SRHBD, RaR, WR, CR, NGR, ReqGR, GI, DGI, RI*	roulette wheel selection depending on success	weights of roulette wheel	improves best known solutions	50000 it., SA is 0.01
Azi et al. [4], 2014 {100;1000}	VRPMTW	max. the number of served customers; min. the total distance traveled	*RaR, RelR, RouR, RelRou, WoR, LCH, RI*	roulette wheel selection depending on success; destroy and repair operators are weighted and chosen independently	weights of roulette wheel	comparison of procedures based on diff. operators	24000 it.

Lei et al. [45] use four different removal and four different insertion heuristics, which are treated independently. Contrary to Pisinger and Ropke [62], each segment is 50 iterations. As there is no competing heuristic for the considered capacitated VRP with stochastic demands and time windows (CVRPSDTW), the authors compare the solutions with the deterministic case to prove that the invented ALNS performs efficiently.

In Azi et al. [3], different removal operators, which are treated independently of insertion operators, are defined to cover three different operation levels: the customers, the routes, and the workdays. Each segment of collecting scores is 200 iterations. The exact scoring system is not specified. A comparison of the previous version of the algorithm in Azi et al. [2] to the optimal solution results a gap less than 1%. Recently, Azi et al. [4] show that the classical approach with the use of customer-based operators is not as beneficial as the use of operators based on the three defined levels.

A sophisticated ALNS approach is presented by Hemmelmayr et al. [34] to solve the two-echelon VRP that considers two levels: Level 1 is the delivery from the central depot to the satellite facilities, and Level 2 is the delivery from the satellites to the customers. Besides the well known destroy and repair operators, mechanisms to open and close satellites are necessary to guarantee high quality solutions. As the authors deal with destroy and repair operators of two different levels, a hierarchical structure needs to be defined: an operator to open and close satellites following a local search phase is executed whenever a given number of iterations have been performed without improvement. The destroy and repair operators are updated independently every 100 iterations. If a new best solution is obtained, a score of 1 is added to the score parameter. Several new best solutions can be found, e.g., compared to a hybrid metaheuristic based on VNS combined with integer linear programming Pirkwieser and Raidl [61]. Several new best solutions for standard benchmark instances can be found compared to Duhamel et al. [22], e.g..

A service technician routing and scheduling problem (STRSP) is solved by Kovacs et al. [42] using ALNS. Appropriate destroy and repair algorithms with and without team building inspired by Ropke and Pisinger [71] are designed for solving real-world problem instances, including lunch break requirements and shift length related labor costs. Contrary to the popular ALNS approach, the scores and weights for each destroy and each repair operator are not considered independently, but destroy-repair heuristic pairs are used. In total, ten pairs are used for the STRSP without team building and eleven pairs are used for the STRSP with team building. Every 100 iterations, the probabilities of the destroy-repair heuristic pairs are updated as in Equation (5). The new adaptive mechanism is as good as or even slightly better than the one proposed in [71]. Compared to the real-world solutions of the manual planning, an improvement of about 10% can be obtained.

Table 8: List of acronyms for destroy and repair heuristics

CR	Cluster removal is a *RelR* that removes requests that are, e.g., geographically clustered.
DFSI	Demand and failure sorting insertion inserts requests in order of the largest expected demand to the route with the smallest probability of failure
DGI	Deep greedy insertion inserts requests which cause the minimal objective costs
DR	Demand-based removal based on demands is a *RelR*
EWR	Expected worst removal removes requests that have a critical effect on the costs
FLLS	First level local search improves the transportation from the depot to the satellites
FOI	Feasibility-oriented insertion moves toward feasibility of infeasible solutions by inserting requests into the routes
FOR	Feasibility-oriented removal moves toward feasibility of infeasible solutions by removing vertices from the routes
GI	Greedy insertion inserts a requests in the cheapest position
GIF	Greedy insertion forbidden works like *GI* but is not allowed to serve a customer from the same satellite from which it was removed
GIN	Greedy insertion with noise function is an extension of *GI* but uses a degree of freedom in selecting the insertion place
GIP	Greedy insertion perturbation works like GI but penalizes the insertion cost by a factor
HNPR	Historical node-pair removal uses historical information when removing requests
HRPR	Historical request-pair removal uses historical success of paired requests
HR	Historical knowledge node removal is similar the *HNPR*
LCH	Least-cost heuristic inserts requests at a feasible position with minimum detour
NGR	Neighbor graph removal is similar to the *HNPR*
NNR	Node neighborhood removal removes randomly a request as well as requests in its neighborhood
NR	Neighborhood removal removes requests which are remarkably different compared to the average distance of a route
PR	Proximity-based removal removes a set of requests that are similar in terms of distance
RaR	Random removal removes requests, e.g. customers, randomly
RelR	Related removal removes requests, e.g. customers, with common characteristics; a relatedness measure has to be defined before
RelRou	Related route removal removes routes with common characteristics
ReqGR	Request graph removal is similar to *HPRP*
RI	Regret insertion uses a look-ahead information when selecting the request to insert
RIN	Regret insertion with noise function works as *RI* but uses the same noise function as *GIN*
RouR	Route removal removes a random route
RouRe	Route redistribution removes routes from each open satellite and reassigns the requests due to penalized distances
SatO	Satellite opening opens a random satellite that is closed
SatR	Satellite removal closes a random satellite
SatS	Satellite swap replaces a satellite with a new one which is close to the old one
SIH	Sequential insertion heuristic considers one route at a time; two criteria define which request at which position should be inserted
SimR	Similarity removal is based on a cost measure is a *RelR*
SRHBAT	Shaw removal heuristic based on arrival times is a *RelR*
SRHBD	Shaw removal heuristic based on distances is a *RelR*
TFSI	Time and failure sorting insertion inserts requests in order of the width of their time window to the route with the smallest probability of failure
TOR	Time-oriented removal is a *RelR* that removes requests which are served at roughly the same time.
TR	Time-based removal is based on *SRHBAT*
WDR	Worst-distance removal removes requests with high distance costs
WoR	Workday removal removes randomly workdays
WR	Worst removal removes requests with high costs
ZI	Zone insertion inserts requests at the best insertion due to time windows rather than distance
ZR	Zone removal removes requests of a predefined area in the Cartesian coordinate system

An ALNS algorithm is recently discussed by Demir et al. [19] to deal with the important topics of fuel, emission, and driver costs. The authors present twelve removal operators, where nine are adapted or inspired by Pisinger and Ropke [62] and Shaw [73], and three are newly invented, as well as five insertion neighborhoods, where four are adapted and one is new. Destroy and repair neighborhoods are treated independently. Each segment is 450 iterations. In order to evaluate the effectiveness of the heuristic algorithm, different sets of real geographic instances are tested and the results show a highly productive approach.

Following Ropke and Pisinger [71] and Shaw [73], Ribeiro et al. [69] use ten destroy and repair operators to solve different instance classes of the cumulative capacitated VRP. The operators are selected independently according to the adaptive mechanism as described in Pisinger and Ropke [62]. A score of 50 is added to the score parameter if a new best solution is obtained; a score of 20 is added to the score parameter if the current solution can be improved; and a score of 5 is added to the score parameter if a non-improving solution is accepted. After 50 iterations the weights and probabilities are updated with a reaction factor $\rho = 0.01$. The algorithm stops either after 50000 iterations, or if the temperature of the simulated annealing acceptance criterion reaches 0.01. Compared to a memetic algorithm of Ngueveu et al. [54], the proposed ALNS improves the best known solutions up to 22 %.

5. POPULATION-BASED METHODS

Adaptiveness can be also found in some population-based approaches. Interestingly, some population-based methods are adaptive by nature. This is the case with ant colony optimization (ACO) [21] and the methods based on the concept of *adaptive memory programming* [80]. ACO relies on repetitively calling a probabilistic construction heuristic. At a given stage of this construction heuristic, solution components have a certain probability of being selected, this probability being influenced by a so-called *pheromone* value. This pheromone value is regularly updated based on search history and on the quality of solutions previously using the same solution component. When solving routing problems, such components are typically arcs. Arcs which have been present in good solutions have higher probabilities of being selected. ACO is a constructive metaheuristic, therefore efficient solutions, e.g. generated with the savings-based concepts in Reimann et al. [67], have to be obtained. There is a significant literature on ACO methods for routing problem, see e.g. [6, 12, 67].

The general idea behind *adaptive memory programming* is to keep a number of good solutions encountered during the search, and use this memory to build new solutions. Every time a new solution is built, the memory is *adapted* in order to integrate the new solution if necessary (that is, if it is interesting to add this solution to the adaptive memory). This memory can be seen as a pool or population of solutions, which is why we mention it here. Adaptive memory has been used to solve routing problems. For instance, [93] develop an adaptive memory

methodology for the vehicle routing problem with simultaneous pickups and deliveries (VRPSPD). Adaptive memory has been hybridized with particle swarm optimization (another population-based method) to solve a dynamic vehicle routing problem [41]. Recently, Khebbache-Hadji et al. [40] have improved heuristics with a memetic algorithm to solve the capacitated vehicle routing problem with two-dimensional loading constraints and time windows (2L-CVRPTW).

Evolutionary algorithms integrate adaptive mechanisms in a number of ways, e.g. in [10], Berger uses a sparsification parameter β in order to restrict the search by considering only insertions of neighboring nodes. This parameter is dynamically modified based on the search history, in order to favor intensification or diversification. In [44], a genetic algorithm is guided by using fuzzy logic. More precisely, the crossover and mutation rate are dynamically adapted based on the recent search history. For instance, when the average solution quality in the population increases, crossover increases and mutation decreases in order to favor intensification. In recent contributions [86, 87, 88], a genetic algorithm which also uses infeasible solutions during the search is presented. A certain proportion of infeasible solutions in the population is targeted, in order to explore interesting and potentially improvement-bringing solutions. The evaluation function uses penalty coefficients to deal with infeasibility, where coefficients are dynamically adapted in order to steer the search towards the desired proportion of infeasible solutions. In [13], a very similar framework is developed but without the adaptive mechanism. However, a local search phase is introduced. Several neighborhoods are used, and at each iteration of the local search, one of them is probabilistically selected. The probabilities associated with each neighborhood are adapted at every 100 iterations based on search history, in a fashion similar to ALNS albeit simpler.

In Table 10, we present a selection of the last years contributions on adaptive population-based approaches, and in Table 9, the used operators are described.

Table 9: List of operators used in population-based methods

1-0 exchange	A node is moved from its position in one route to another position in either the same or a different route [90].
1-1 exchange	Two nodes are swapped from either the same or different routes [90].
2-opt	The 2-opt heuristic in Croes [17] iteratively inverts sequences of nodes.
2-opt*	The 2-opt* heuristic exchanges the last parts of two routes.
move	Nodes are moved to another position in either the same or different route.
swap	Nodes are swapped from either the same or different routes.

Table 10: Population-based concepts

Contribution $\{n_{min}; n_{max}\}$	Problems tackled	Objective function	Operators	Adaptive mechanism	Parameters	Solution quality	Termination criteria
Zachariadis et al. [93], 2010 {50; 400}	VRPSPD	minimize total travel costs	1-0 exchange, 1-1 exchange, 2-opt	adaptive memory	constant memory size	robust solution quality	10,000 cycles of adaptive memory exploitation
Vidal et al. [87], 2012 {50; 417}	MDVRP PVRP MDPVRP	minimize total travel costs	move, swap, 2-opt, 2-opt*	crossover and mutation rate are dynamically adapted based on the recent search history	crossover and mutation rate	new best solutions are obtained	$10^4 - 5 \times 10^4$ it.
Vidal et al. [86], 2013 {48; 1000}	VRPTW PVRPTW MDVRPTW SDVRPTW	minimize total travel costs	move, swap, 2-opt, 2-opt*	crossover and mutation rate are dynamically adapted based on the recent search history	crossover and mutation rate	new best solutions are obtained	5×10^3 it.
Vidal et al. [88], 2014 {48; 1000}	ACVRP CCVRP CVRP GVRP LDVRP MDVRP MDVRPTW OVRP OVRPTW PVRP PVRPTW SDVRPTW TDVRPTW VFMP-F VFMP-FV VFMPTW VFMP-V VRPB VRPBTW VRPMDP VRPSDP VRPSTW VRPTW VRTDSP	minimize total travel costs	move, swap, 2-opt, 2-opt*	crossover and mutation rate are dynamically adapted based on the recent search history	crossover and mutation rate	new best solutions are obtained	$it_{max} = 5000$, $t_{max} = 30$ min

6. CONCLUSION

This research paper presents an overview of recent adaptive mechanisms when solving vehicle routing problems (VRPs) with metaheuristics. Starting with basic local search-based methods, e.g. adaptive tabu search (ATS) or guided local search (GLS), we progress to hybrid local search methods, e.g. iterated local search (ILS), adaptive variable neighborhood search (AVNS) and adaptive large neighborhood search (ALNS). For the sake of completeness, we concluded the survey with population-based methods, e.g. ant colony optimization (ACO), memetic and genetic algorithms (GAs).

The most popular and very successful adaptive approach is the ALNS using a clever selection mechanism favor the most successful operators. Also, recent work in population-based methods, e.g. Vidal et al. [88] achieve, by using, adaptive crossover and mutation rate competitive results.

In order to further investigate which of the possible adaptive strategies are particularly useful, Part II of this survey [43] will consider several ways of making a VNS algorithm adaptive and will investigate numerically, which ones are useful and promising for solving the open VRP instances.

Acknowledgements. This work received support from the Austrian Science Fund (FWF) under grant and L628-N15 (Translational Research Programs).

REFERENCES

[1] Anagnostopoulos, A., Michel, L., Van Hentenryck, P., Vergados, Y., "A simulated annealing approach to the traveling tournament problem", *Journal of Scheduling*, 9 (2006) 177-193.

[2] Azi, N., Gendrau, M., Potvin, J.-Y., "An Adaptive Large Neighborhood Search for Vehicle Routing Problem with Multiple Trips", *Tech. Rep. CIRRELT-20012-08*, Interuniversity Research Center on Enterprise Networks, Logistics and Transportation, Canada, (2010).

[3] Azi, N., Gendrau, M., Potvin, J.-Y., "A dynamic vehicle routing problem with multiple delivery routes", *Annals of Operations Research*, 199(1) (2012) 103-112.

[4] Azi, N., Gendrau, M., Potvin, J.-Y., "An adaptive large neighborhood search for a vehicle routing problem with multiple routes", *Computers & Operations Research*, 41 (2014) 167-173.

[5] Baldacci, R., Mingozzi, A., "A unified exact method for solving different classes of vehicle routing problems", *Mathematical Programming*, 120(2) (2009) 347-380.

[6] Balseiro, S. R., Loiseau, I., Ramonet, J., "An Ant Colony algorithm hybridized with insertion heuristics for the Time Dependent Vehicle Routing Problem with Time Windows", *Computers & Operations Research*, 38(6) (2011) 954-966.

[7] Battiti, R., "Reachtive Search: Toward Shelf-Tuning Heuristics", in: V. J. Rayward-Smith, I. H. Osman, C. R. Reeves, G. D. Smith (eds.) *Modern Heuristic Search Methods*, John Wiley and Sons Ltd (1996) 61-83.

[8] Battiti, R., Brunato, M., "Reactive Search Optimization: Learing While Optimizing", in: M. Gendreau, J.-Y. Potvin (eds.) *Handbook of Metaheuristics, Second Edition*, International Series in Operations Research & Management Science, Springer Science+Business Media LCC, vol. 146 (2010) 543-571.

[9] Battiti, R., Tecchiolli, G., "The Reactive Tabu Search", *ORSA Journal on Computing*, 6(2) (1994) 126-140.

[10] Berger, J., Barkaoui, M., "A new hybrid genetic algorithm for the capacitated vehicle routing problem", *Journal of the Operational Research Society*, 54 (2003) 1254-1262.

[11] Bräysy, O., "A Reactive Variable Neighborhood Search for the Vehicle-Routing Problem with Time Windows", *INFORMS Journal on Computing*, 15(4) (2003) 347-368.

[12] Bullnheimer, B., Hartl, R. F., Strauss, C., "An improved ant system algorithm for the vehicle routing problem", *Annals of Operations Research*, 89 (1997) 319-328.
[13] Cattaruzza, D., Absi, N., Feillet, D., Vidal, T., "A memetic algorithm for the Multi Trip Vehicle Routing Problem", *European Journal of Operational Research*, 236(3) (2014) 833-848.
[14] Clarke, G., Wright, J. W., "Scheduling of vehicles from a central depot to a number of delivery points", *Operations Research*, 12(4) (1964) 568-581.
[15] Cordeau, J. F., Maischberger, M., "A parallel iterated tabu search heuristic for vehicle routing problems", *Computers & Operations Research*, 39(9) (2012) 2033-2050.
[16] Crevier, B., Cordeau, J. F., Laporte, G., "The multi-depot vehicle routing problem with inter-depot routes", *European Journal of Operational Research*, 176(2) (2007) 756-773.
[17] Croes, G. A., "A Method for Solving Traveling-Salesman Problems", *Operations Research*, 6(6) (1958) 791-812.
[18] Dantzig, G. B., Ramser, J. H., "The truck dispatching problem", *Management Science*, 6 (1959) 80-91.
[19] Demir, E., Bektaş, T., Laporte, G., "An adaptive large neighborhood search heuristic for the Pollution-Routing Problem", *European Journal of Operational Research*, 223(2) (2012) 346-359 .
[20] Di Gaspero, L., Schaerf, A., "A composite-neighborhood tabu search approach to the traveling tournament problem", *Journal of Heuristics*, 13(2) (2007) 189-207.
[21] Dorigo, M., Stützle, T., "Ant Colony Optimization: Overview and Recent Advances", in: M. Gendreau, J.-Y. Potvin (eds.) *Handbook of Metaheuristics, Second Edition*, International Series in Operations Research & Management Science, Springer Science+Business Media LLC, vol. 146 (2010) 227-263.
[22] Duhamel, C., Lacomme, P., Prins, C., Prodhon, C., "A GRASP × ELS approach for the capacitated location-routing problem", *Computers & Operations Research*, 37(11) (2010) 1912-1923.
[23] Garey, M. R., Johnson, D. S., "Computers and Intractability; A Guide to the Theory of NP-Completeness", W. H. Freeman & Co., New York, NY, USA, (1990).
[24] Gendreau, M., Hertz, A., Laporte, G., "A Tabu Search Heuristic for the Vehicle Routing Problem", *Management Science*, 40(10) (1994) 1276-1290.
[25] Gendreau, M., Iori, M., Laporte, G., Martello, S., "A Tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints", *Networks*, 51(1) (2007) 4-18.
[26] Ghaziri, H., "Solving routing problems by a self-organizing map", in: T. Kohonen, K. Makisara, O. Simula, J. Kangas (eds.) *Artificial Neural Networks*, North-Holland Amsterdam, (1991) 829-834.
[27] Gillett, B. E., Miller, L. R., "A Heuristic Algorithm for the Vehicle-Dispatch Problem", *Operational Research*, 22(2) (1974) 340-349.
[28] Glover, F., "Future paths for integer programming and links to artificial intelligence", *Computers & Operations Research*, 13(5) (1986) 533-549.
[29] Glover, F., "Tabu Search – Part I", *ORSA Journal on Computing*, 1(3) (1989) 190-206.
[30] Glover, F., "Tabu Search – Part II", *ORSA Journal on Computing*, 2(1) (1990) 4-32.
[31] Hansen, P., Mladenović, N., "Variable Neighborhood Search", in: P. M. Pardalos, M. G. C. Resende (eds.) *Handbook of Applied Optimization*, Oxford University Press New York, (2000) 221-234.
[32] Hansen, P., Mladenović, N., "Variable Neighborhood Search: Principles and applications", *European Journal of Operational Research*, 130(3) (2001) 449-467.
[33] Hansen, P., Mladenović, N., Moreno Pérez, J. A., "Variable neighborhood search: methods and applications", *Annals of Operations Research*, 175(3) (2010) 367-407.
[34] Hemmelmayr, V. C., Cordeau, J. F., Crainic, T. G., "An adaptive large neighborhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics", *Computers & Operations Research*, 195(3) (2009) 803-809.
[35] Hemmelmayr, V. C., Doerner, K. F., Hartl, R. F., "A variable neighborhood search heuristic for periodic routing problems", *Computers & Operations Research*, 39(16) (2012) 3215-3228.
[36] Hernández-Pérez, H., Salazar-González, J. J., "Heuristics for the One-Commodity Pickup-and-Delivery Traveling Salesman Problem", *Transportation Science*, 38(2) (2004) 245-255.
[37] Hosny, M. I., Mumford, C. L., "Solving the One-Commodity Pickup and Delivery Problem Using an Adaptive Hybrid VNS/SA Approach", in: R. Schaefer, C. Cotta, J. Kolodziej, G. Rudolph (eds.) *Parallel Problem Solving from Nature, PPSN XI, Lecture Notes in Computer Science*, Springer

Berlin Heidelberg, vol. 6239 (2010) 189-198.
[38] Hsiao, P. C., Chiang, T. C., Fu, L. C., "A VNS-based Hyper-heuristic with Adaptive Computational Budget of Local Search", *IEEE Congress on Evolutionary Computation*, (2012) 1-8.
[39] Irnich, S., Funke, B., Grünert, T., "Sequential search and its application to vehicle-routing problems", *Computers & Operations Research*, 33(8) (2006) 2405-2429.
[40] Khebbache-Hadji, S., Prins, C., Yalaoui, A., Reghioui, M., "Heuristics and memetic algorithm for the two-dimensional loading capacitated vehicle routing problem with time windows", *Central European Journal of Operations Research*, 21(2) (2013) 307-336.
[41] Khouadjia, M., Jourdan, L., Talbi, E., "Adaptive particle swarm for solving the dynamic vehicle routing problem", *IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, (2010) 1-8.
[42] Kovacs, A. A., Parragh, S. N., Doerner, K. F., Hartl, R. F., "Adaptive large neighborhood search for service technician routing and scheduling problems", *Journal of Scheduling*, 15(5) (2012) 579-600.
[43] Kritzinger, S., Tricoire, F., Doerner, K. F., Hartl, R. F., "Adaptive search techniques for problems in vehicle routing, Part II: A numerical comparison", to appear in *Yugoslav Journal of Operations Research* (2014).
[44] Lau, H. C. W., Chan, T. M., Tsui, W. T., Pang, W. K., "Application of Genetic Algorithms to Solve the Multidepot Vehicle Routing Problem", *IEEE Transactions on Automation Science and Engineering*, 60(2) (2010) 383-392.
[45] Lei, H., Laporte, G., Guo, B., "The capacitated vehicle routing problem with stochastic demands and time windows", *Computers & Operations Research*, 38(12) (2011) 1775-1783.
[46] Leung, S. C. H., Zhou, X., Zheng, J., "Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem", *Computers & Operations Research*, 38(1) (2011) 205-215.
[47] Lin, S., "Computer Solutions of the Traveling Salesman Problem", *Bell System Technical Journal*, 44 (1965) 2245-2269.
[48] Lourenço, H. R., Martin, O. C., Stützle, T., "Iterated Local Search: Framework and Applications", in: M. Gendreau, J.-Y. Potvin (eds.) *Handbook of Metaheuristics, Second Edition*, International Series in Operations Research & Management Science, Springer Science+Business Media LLC, vol. 146 (2010) 363-397.
[49] Martin, O., Otto, S. W., Felten, F. W., "Large-step Markov chains for the traveling salesman problem", *Complex Systems*, 5(3) (1991) 299-326.
[50] Mester, D., Bräysy, O., "Active-guided evolution strategies for large-scale capacitated vehicle routing problems", *Computers & Operations Research*, 34(10) (2007) 2964-2975.
[51] Michallet, J., Prins, C., Amodeo, L., Yalaoui, F., Vitry, G., "Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services", *Computers & Operations Research*, 41 (2014) 196-207.
[52] Mills, P., Tsang, E., Ford, J., "Applying an Extended Guided Local Search to the Quadratic Assignment Problem", *Annals of Operations Research*, 118(1-4) (2003) 121-135.
[53] Mladenović, N., Hansen, P., "Variable Neighborhood Search", *Computers & Operations Research*, 24(11) (1997) 1097-1100.
[54] Ngueveu, S. U., Prins, C., Wolfler Calvo, R., "An effective memetic algorithm for the cumulative capacitated vehicle routing problem", *Computers & Operations Research*, 37(11) (2010) 1877-1885.
[55] Novoa, C., Storer, R., "An approximate dynamic programming approach for the vehicle routing problem with stochastic demands", *European Journal of Operational Research*, 196(2) (2009) 509-515 .
[56] Or, I., "Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking", Ph.D. thesis, North Western University, Chicago, USA (1976).
[57] Osman, I. H., "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem", *Annals of Operations Research*, 41(4) (1993) 421-451.
[58] Parragh, S. N., Doerner, K. F., Hartl, R. F., "Variable neighborhood search for the dial-a-ride problem", *Computers & Operations Research*, 37(6) (2010) 1129-1138.
[59] Penna, P. H. V., Subramanian, A., Ochi, L. S., "An Iterated Local Search heuristic for the Heterogeneous Fleet Vehicle Routing Problem", *Journal of Heuristics*, 19(2) (2013) 201-232.

[60] Pillac, V., Guéret, C., Medaglia, A. L., "An event-driven optimization framework for dynamic vehicle routing", *Decision Support Systems*, 54(1) (2012) 414-423.
[61] Pirkwieser, S., Raidl, G. R., "Variable neighborhood search coupled with ILP-based very large neighborhood searches for the (periodic) location-routing problem", in: M. Blesa, C. Blum, G.R. Raidl, A. Roli, M. Sampels (eds.) *Hybrid metaheuristics, Lecture Notes in Computer Science*, Springer Berlin Heidelberg, vol.6373 (2010) 174-189.
[62] Pisinger, D., Ropke, S., "A general heuristic for vehicle routing problems", *Computers & Operations Research*, 34(8) (2007) 2403-2435.
[63] Polacek, M., Benkner, S., Doerner, K. F., Hartl, R. F., "A Cooperative and Adaptive Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows", *Business Research Journal*, 1(2) (2008) 207-218.
[64] Polacek, M., Hartl, R. F., Doerner, K. F., Reimann, M., "A Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows", *Journal of Heuristics*, 10(6) (2004) 613-627.
[65] Potvin, J.-Y., Naud, M. A., "Tabu search with ejection chains for the vehicle routing problem with private fleet and common carrier", *Journal of the Operational Research Society*, 62 (2011) 326-336.
[66] Prins, C., "A simple and effective evolutionary algorithm for the vehicle routing problem", *Computers & Operations Research*, 31(12) (2004) 1984-2002.
[67] Reimann, M., Doerner, K. F., Hartl, R. F., "D-Ants: Savings Based Ants divide and conquer the vehicle routing problem", *Computers & Operations Research*, 31(4) (2004) 563-591.
[68] Reimann, M., Stummer, M., Doerner, K. F., "A savings based ant system for the vehicle routing problem", in: W. B. Langdon, et al. (eds.) *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*. San Francisco: Morgan Kaufmann (2002).
[69] Ribeiro, G. M., Laporte, G., "An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routring problem", *Computers & Operations Research*, 39(3) (2012) 728-735.
[70] Ropke, S., Pisinger, D., "A unified heuristic for a large class of Vehicle Routing Problems with Backhauls", *European Journal of Operational Research*, 171(3) (2006) 750-775.
[71] Ropke, S., Pisinger, D., "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows", *Transportation Science*, 40(4) (2006) 455-472.
[72] Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., Dueck, G., "Record breaking optimization results using the ruin and recreate principle", *Journal of Computational Physics*, 159 (2000) 139-171.
[73] Shaw, P., "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, Lecture Notes in Computer Science*, Springer Berlin Heidelberg, vol. 1520 (1998) 417-431.
[74] Stenger, A., Vigo, D., Enz, S., Schwind, M., "An Adaptive Variable Neighborhood Search Algorithm for a Vehicle Routing Problem Arising in Small Package Shipping", *Transportation Science*, 47(1) (2013) 64-80.
[75] Stützle, T., Hoos, H., "Stochastic Local Search", in: *SLS Methods*, Morgan Kaufmann, (2005) 61-112.
[76] Subramanian, A., Battarra, M., "An iterated local search algorithm for the Travelling Salesman Problem with Pickups and Deliveries", *Journal of the Operational Research Society*, 64(3) (2013) 402-409.
[77] Subramanian, A., Uchoa, E., Ochi, L. S., "A hybrid algorithm for a class of vehicle routing problems", *Computers & Operations Research*, 40(10) (2013) 2519-2531.
[78] Taillard, É. D., "Parallel Iterative Search Methods for Vehicle Routing Problems", *Networks*, 23(8) (1993) 661-673.
[79] Taillard, É. D., Badeau, P., Gendreau, M., Potvin, J.-Y., "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows", *Transportation Science*, 31(2) (1997) 170-186.
[80] Taillard, É. D., Gambardella, L. M., Gendreau, M., Potvin, J.-Y., "Adaptive memory programming: A unified view of metaheuristics", *European Journal of Operational Research*, 135(1) (2001) 1-16.
[81] Tang Montané, F. A., Galvão, R. D., "A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service", *Computers & Operations Research*, 33(3) (2006) 595-619.
[82] Tarantilis, C. D., Kiranoudis, C. T., "Bone-Route: An Adaptive Memory-Based Method for

Effective Fleet Management", *Annals of Operations Research*, 115 (2002) 227-241.
[83] Tarantilis, C. D., Zachariadis, E. E., Kiranoudis, C. T., "A Hybrid Guided Local Search for the Vehicle-Routing Problem with Intermediate Replenishment Facilities", *INFORMS Journal on Computing*, 20(1) (2008) 154-168.
[84] Thompson, P. M., Psaraftis, H. N., "Cyclic transfer algorithms for multivehicle routing and scheduling problems", *Operations Research*, 41(5) (1993) 935-946.
[85] Toth, P., Vigo, D., "The Granular Tabu Search and Its Application to the Vehicle-Routing Problem", *INFORMS Journal on Computing*, 15(4) (2003) 333-346.
[86] Vidal, T., Crainic, T. G., Gendreau, M., Prins, C., "A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows", *Computers & Operations Research*, 40(1) (2013) 475-489.
[87] Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., Rei, W., "A Hybrid Genetic Algorithm for Multi-Depot and Periodic Vehicle Routing Problems", *Operations Research*, 60(3) (2012) 611-624.
[88] Vidal, T., Crainic, T. G., Gendreau, M., Prins, C., "A Unified Solution Framework for Multi-Attribute Vehicle Routing Problems", *European Journal of Operational Research*, 234(3) (2014) 658-673.
[89] Voudouris, C., Tsang, E. P. K., Alsheddy, A., "Guided Local Search", in: M. Gendreau, J.-Y. Potvin (eds.) *Handbook of Metaheuristics, Second Edition*, International Series in Operations Research & Management Science, Springer Science+Business Media LLC, vol. 146 (2010) 321-361.
[90] Waters, C. D. J., "A solution procedure for the vehicle-scheduling problem based on iterative route improvement", *Journal of the Operational Research Society*, 38(9) (1987) 833-839.
[91] Zachariadis, E. E., Tarantilis, C. D., Kiranoudis, C. T., "A Guided Tabu Search for the Vehicle Routing Problem with two-dimensional loading constraints", *European Journal of Operations Research*, 195(3) (2009) 729-743.
[92] Zachariadis, E. E., Tarantilis, C. D., Kiranoudis, C. T., "A hybrid metaheuristic algorithm for the vehicle routing problem with simulataneous delivery and pick-up service", *Expert Systems with Applications*, 36(2) (2009) 1070-1081.
[93] Zachariadis, E. E., Tarantilis, C. D., Kiranoudis, C. T., "An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries", *European Journal of Operational Research*, 202 (2010) 401-411.
[94] Zhao, F., Li, S., Sun, J., Mei, D., "Genetic algorithm for the one-commoditiy pickup-and-delivery traveling salesman problem", *Computers & Industrial Engineering*, 56(4) (2008) 1642-1648.

APPENDIX

Table 11: List of acronyms for problem variants

1-PDP	One-commodity pickup and delivery problem
2E-VRP	Two-echelon vehicle routing problem
2L-CVRP	Capacitated vehicle routing problem with two-dimensional loading constraints
ACVRP	Asymmetric capacitated vehicle routing problem
CCVRP	Cumulative capacitated vehicle routing problem
CVRP	Capacitated vehicle routing problem
CVRPSDTW	Capacitated vehicle routing problem with stochastic demands and time windows
DVRPM	Dynamic vehicle routing problem with multiple delivery routes
DVRSD	Dynamic vehicle routing problem with stochastic demands
FSM	Fleet size and mix is a HFVRP with unlimited fleet
FSMD	Fleet size and mix with dependent costs is a HFVRPD with unlimited fleet
FSMF	Fleet size and mix with fixed costs is a HFVRPF with unlimited fleet
FSMFD	Fleet size and mix with fixed and dependent costs is a HFVRPFD with unlimited fleet
GVRP	Generalized vehicle routing problem
HFVRP	Heterogeneous fleet vehicle routing problem
HFVRPD	Heterogeneous fleet vehicle routing problem with dependent costs
HFVRPF	Heterogeneous fleet vehicle routing problem with fixed costs
HFVRPFD	Heterogeneous fleet vehicle routing problem with fixed and dependent costs
LDVRP	Load-dependent costs vehicle routing problem
LRP	Location routing problem
MDVRP	Multi depot vehicle routing problem
MDVRPMPD	Multi depot vehicle routing problem with mixed pickup and delivery
MDVRPPC	Multi depot vehicle routing problem with private fleet and common carriers
MDVRPTW	Multi depot vehicle routing problem with time windows
MRP	Multicast routing problem
OVRP	Open vehicle routing problem
OVRPTW	Open vehicle routing problem with time windows
PRP	Pollution-routing problem
PVRP	Periodic vehicle routing problem
PVRPTW	Periodic vehicle routing problem with time windows
RPDPTW	Rich pickup and delivery problem with time windows
SDVRP	Site-dependent vehicle routing problem
SDVRPTW	Site-dependent vehicle routing problem with time windows
STRSP	Service technician routing and scheduling problem with and without team building
TDVRPTW	Time-dependent vehicle routing problem with time windows
TSP	Traveling salesperson problem
TSPPD	Traveling salesperson problem with pickups and deliveries
VFMP-F	Vehicle fleet mix problem with fixed vehicle costs
VFMP-FV	Vehicle fleet mix problem with fixed and variable vehicle costs
VFMP-V	Vehicle fleet mix problem with variable vehicle costs
VFMPTW	Vehicle fleet mix problem with time windows
VRP	Vehicle routing problem
VRPB	Vehicle routing problem with backhauls
VRPBTW	Vehicle routing problem with backhauls and time windows
VRPIRF	Vehicle routing problem with intermediate replenishment facilities
VRPMPD	Vehicle routing problem with mixed pickup and delivery
VRPPC	Vehicle routing problem with private fleet and common carriers
VRPSPD	Vehicle routing problem with simultaneous pickups and deliveries
VRPSTW	Vehicle routing problem with soft time windows
VRPTW	Vehicle routing problem with time windows
VRPTWMD	Vehicle routing problem with time windows and multiple deliverymen
VRTDSP	Vehicle routing and truck-driver scheduling problems