Information Technology and Management Science
ISSN 2255-9094 (online)
2020, vol. 23, pp. 55–63
https://doi.org/10.7250/itms-2020-0008
https://itms-journals.rtu.lv

1862
RIGA TECHNICAL UNIVERSITY

Parameter Tuning of a Local Search Heuristic for a Vehicle Routing Problem with Loading Constraints

Hanne Pollaris[1], Gerrit K. Janssens[2], Kris Braekers[3], An Caris[4]
[1–4] *Hasselt University, Hasselt, Belgium*

***Abstract* – A vehicle routing problem (VRP) with sequence-based pallet loading and axle weight constraints is introduced in the study. An Iterated Local Search (ILS) metaheuristic algorithm is used to solve the problem. Like any metaheuristic, a number of parameters need to be set before running the experiments. Parameter tuning is important because the value of the parameters may have a substantial impact on the efficacy of a heuristic algorithm. While traditionally, parameter values have been set manually using expertise and experimentation, recently several automated tuning methods have been proposed. The performance of the routing algorithm is mostly improved by using parameter tuning, but no single best tuning method for routing algorithms exists. The tuning method, Iterated F-race, is chosen because it seems to be a very robust method and it has been shown to perform well on the ILS metaheuristic and other metaheuristics. The research aims at developing an algorithm, which performs well over a wide range of network sizes.**

***Keywords* – Decision making, evolutionary computation, operations research, supply chain management.**

I. Introduction

Vehicle routing is a well-studied problem in operations research as it makes up an important part of distribution logistics decision making. The increase in computing power as the development of efficient algorithms has made that complex variants of the Vehicle Routing Problem (VRP) can be solved in a reasonable time. Therefore, the research is still active, because nowadays real-life cases can be handled. The more complex variants with multiple types of constraints are called 'rich vehicle VRPs' in literature. There is no strict definition of a Rich VRP: some of them are mentioned in [1], [2] and [3]. They may include, for example, the introduction of time windows, heterogeneous vehicles, incompatibility of products in vehicles, etc.

Apart from the routing of vehicles, logistics companies also need to consider the loading of the vehicles. A feasible loading plan is not guaranteed when only total capacity of a vehicle is considered. The loading problem considers efficiency in unloading at the customers' site, stability of the vehicles, incompatibility of storage in the vehicle with respect to potential damage of the goods, and fulfilment of rules and regulations regarding mass distribution within the vehicle. Authors of the study [4] mention a survey among several Belgian logistics service providers pointing out that they are faced with complex loading problems when planning their route (e.g., multi-dimensional packing constraints, unloading sequence constraints, stability constraints and axle weight limits). Ignorance of these constraints may compromise planning and induce last minute changes resulting in additional costs. Axle weight limits, in particular, impose a challenge for transportation companies since they are faced with high fines when violating the limits. Weigh-In-Motion (WIM) systems on highways monitor axle weight violations of trucks while driving, which increases the probability that axle weight violations are detected [5]. Furthermore, trucks with overloaded axles represent a threat for traffic safety and may cause serious damage to the road surface. Since the weight on the axles changes when items are loaded and unloaded, it is important that axle weights are considered during the entire trip of the vehicle and not only at the time the vehicle departs from the depot.

In their guide to vehicle routing problem variants, scientists [6] dedicate a section on specificities of drivers and vehicles. They focus on heterogeneous vehicles, working hour regulations, recharging stops and loading constraints. Problems including heterogeneous vehicles typically deal with different capacities of vehicles, but the scope has shifted towards multi-modal transportation systems involving bikes, scooters, vans and drones. Working hour regulations deal with the responsibility of ensuring that driving plans can be safely performed. Various countries impose daily and weekly rest periods as well as limits on driving and working hours. Recharging stops relate to battery-powered electric vehicles because of their limited range. Loading constraints relate to specific load restrictions, which need to be considered during route planning. They relate primarily to geometric constraints (pallet loading, 2D packing and 3D packing), but also to fragility, orientation or equilibrium.

In this paper, a variant of the classical Capacitated VRP is analysed. The vehicle fleet consists of homogeneous vehicles and the demand of the customers is defined in terms of the number of pallets. Sequence-based loading is imposed, which ensures that, when arriving at a customer, no pallets belonging to customers served later on the route, block the removal of the pallets of the current customer. Furthermore, the capacity of a truck is not only expressed in total weight and the number of pallets but also in terms of a maximum weight on the axles of the truck.

II. Literature Review

To our knowledge, [7] and [8] are the only authors that address axle weight constraints in a container loading problem. The researchers [7] developed a heuristic method to tackle the single container loading problem with axle weight constraints. In turn, the scientists [8] developed integer linear programming models to solve multi-container loading problems with axle weight constraints, in which items are first packed on pallets and, afterwards, pallets are placed onto the trucks. The integration of axle weight constraints in a Capacitated VRP (CVRP) has been introduced in [9]. A Mixed Integer Linear Programming (MILP) model is proposed to solve the CVRP with sequence-based pallet loading and axle weight constraints to optimality for networks of up to 20 nodes. Authors [10] propose an Iterated Local Search (ILS) metaheuristic to solve instances with up to 100 customers for the same problem.

For state-of-the-art review of the literature concerning the combination of Vehicle Routing Problems and loading problems, the reader is referred to [11] and [4].

Iterated Local Search (ILS) is a metaheuristic framework that iteratively applies local search, perturbation, and evaluation of the solution against the acceptance criterion [12]. The local search embeds a heuristic procedure; the perturbation and the acceptance criterion allow exploring the search space as well as escaping from local optima. The ILS metaheuristic has been applied to many optimisation problems, such as vehicle routing problems [13], [10], [14], scheduling problems, facility location problems, and inventory routing problems, among others.

Situation Description and Assumptions

The assumptions of the problem under study are listed here. The demand of the customers is expressed as a number of euro pallets (80 x120 cm). Pallets are packed dense in a truck in two horizontal rows (a left one and a right one). This means that no gap is allowed between two consecutive pallets in the container and that all pallets are alternately packed in the left and right row. Dense packing also entails that there is an open space allowed between the front of the container and the first pallets that are packed. Dense packing is often imposed to increase the stability of the load since it restricts the moving area of the pallets considerably. The driver therefore needs to spend less time on securing the cargo. It is assumed that all pallets of a single customer have the same weight and that the weight is uniformly distributed inside each pallet, i.e., the centre of gravity of a pallet is situated in its geometric midpoint. A container can only be unloaded at the rear side. To avoid moving pallets of other customers, when arriving at a customer, sequence-based loading is imposed. Vertical stacking is not allowed. The vehicle types in the fleet are different in terms of tare weight and measurements. Consequently, the capacity in terms of number of pallets and payload is different as well as the weight capacity of the axles.

Axle weight is defined as the weight that is placed on the axles of the truck. In Europe, heavy goods vehicles, buses and coaches must comply with certain rules on weights and dimensions for road safety reasons and to avoid damaging roads, bridges and tunnels. A directive of the European Union (EU) has set maximum dimensions and weights for international traffic, also ensuring that Member States cannot restrict the circulation of vehicles which comply with these limits for performing international transport operations within their territories. The values of the upper bounds on the weight on the axles of the tractor and on the axles of the trailer depend on the vehicle characteristics and are specified in legislation. The lower bound of the weight on the axles of the tractor may also be fixed in legislation.

The Iterated Local Search Algorithm

The proposed solution method is based on an Iterated Local Search (ILS) framework, which is proven to be a highly effective heuristic for routing problems [12]. The ILS consists of four procedures (Generate Initial Solution, Local Search, Perturbation, Acceptance). The general structure is presented in Algorithm 1. The algorithm is described in detail in [10], but some parts are reproduced here because the parameters, which will be tuned in this research, need to be recognised.

Algorithm 1. Steps of the ILS

Initialization
1: $s^0 \leftarrow$ Generate initial solution
2: $s; s^b \leftarrow$ Local search on s^0
3: **repeat**
  $s \leftarrow$ Perturbation on s
  $s \leftarrow$ Local search on s
  $s; s^b \leftarrow$ Acceptance criterion
4: **until** *non improving it* $> \alpha$

First, an initial solution (s^0) is constructed. Second, this solution is improved using local search until a local optimum is reached. The local search is performed by a Variable Neighbourhood Descent (VND) method. Third, the following steps are performed iteratively. In order to escape from the local optimum, a new starting point for the local search is generated by perturbing the current solution (s). This solution is improved using local search. Then, the acceptance criterion determines with which solution the process continues. The ILS stops after a number of α consecutive non-improving iterations. A non-improving iteration (*non_improving_it*) is an iteration in which no new best solution was found. Note that since the local search is performed by a VND, the algorithm may also be called an Iterated Variable Neighbourhood Descent, as used in [15].

Initial Solution

The generation of initial solutions requires no parameters, so it is of less relevance to repeat the procedure in this document. The interested reader can find all details in the 'Initial Solution' section of [10].

Local Search

The local search is performed by a VND method, in which four neighbourhoods are used. The exchange operator [16]

swaps two nodes, which can be either from the same route or from different routes. The 2-opt operator [17] removes two arcs of a single route and generates two new arcs in such a way that the section between the removed arcs is reversed. The cross-exchange operator [18] interchanges two segments of different routes while preserving the orientation of the segments and the routes. Finally, the relocate operator [16] removes a node from its route and reinserts it in another place in its original route or in another route. An overview of the local search procedure is presented in Algorithm 2. Further details can be found in the 'Local Search' section of [10].

Algorithm 2. Local search

Neighbourhoods = {exchange, 2-opt, cross-exchange, relocate}
s = initial solution
$s' := s$
stop := 0
repeat
  for i := 1 to 4 **do**
    next neighbourhood = 0
    repeat
      s'' ← Local search with *Neighbourhoods*[i] on s'
      if $s' = s''$ **then**
        next neighbourhood := 1
      else
      $s' := s''$
      end if
    until *next neighbourhood* = 1
  end for
  if $s = s'$ **then**
    stop := 1
  else
    $s := s'$
  end if
until *stop* = 1

Perturbation

In the perturbation phase, the relocate operator is considered once for each customer, using a randomized objective function. The effect of relocating a customer to another position is randomized by adding a noise factor to the insertion cost. The insertion cost is calculated as the sum of the costs of the arcs that are created when inserting a customer in a new position minus the costs of the arcs that are removed. Similar to [19], the noise value is calculated as a random number in the interval $[-\eta * maxD; \eta * maxD]$ where $\eta \in]0, +\infty[$ is a parameter to control the amount of noise and *maxD* is the maximum distance between two nodes in the network. The general framework of the perturbation procedure is presented in Algorithm 3. Other details can be found in the 'Perturbation' section of [10]. Here the focus lies on the parameter η.

Initially, the value for η is determined by the value of parameter η^0. If the perturbation does not change the solution s, η is increased with the value of parameter η^{incr} and the perturbation is repeated. After δ consecutive non-improving iterations (*non_improving_it*) of the ILS, a heavy perturbation is applied. This means that η increases with the parameter value of η^{heavy} to increase the level of diversification. When improvement is found after the local search procedure, the number of consecutive non-improving iterations becomes 0 (as may be seen in Algorithm 4, in the subsection "Acceptance Criterion") and the value for η is set to the value of η^0.

Algorithm 3. Perturbation

if *non_improving_it* > δ **then**
  $\eta := \eta^0 + \eta^{heavy}$
else
  $\eta := \eta^0$
end if
repeat
  s' ← Relocate for each customer on s with noise η
  $\eta := \eta + \eta^{incr}$
until $s \neq s'$
$s = s'$

Acceptance Criterion

The acceptance criterion that is used in this ILS algorithm is based on record-to-record travel introduced by Dueck (1993). The solution s obtained after local search is always accepted to become the new incumbent solution s of the next ILS iteration if the cost is lower than the cost of the current best solution s^b. When the cost of s is higher than the cost of s^b and no heavy perturbation is applied in the next iteration of the ILS (i.e., *non_improving_it* < δ), the solution is still accepted if the worsening is smaller than a certain threshold value. This threshold value corresponds to a fraction β of the cost of s^b. In case a heavy perturbation is applied in the next ILS iteration, worsening is never accepted in order not to deviate too far from the current best solution. In case s is not accepted to become the new incumbent solution, the search continues from s^b. The acceptance criterion procedure is described in Algorithm 4.

Algorithm 4. Acceptance Criterion

if *cost*[s] < *cost*[s^b] **then**
  $s^b := s$
  non_improving_it := 0
else
  non_improving_it := *non_improving_it* + 1
  if (*cost*[s] > *cost*[s^b] · (1 + β)) or (*non improving it* > δ)
then
    $s := s^b$
  end if
end if

Six parameters have been chosen to be tuned. They are {α, δ, η^0, η^{incr}, η^{heavy}, β}. Parameter α appears in Algorithm 1. Parameters δ, η^0, η^{incr}, η^{heavy} appear in Algorithm 3. Parameter β appears in Algorithm 4.

III. Parameter Tuning

This section presents the parameter tuning of the algorithm described in the previous section. Parameter tuning is important because the value of the parameters may have a substantial impact on the efficacy of a heuristic algorithm [20]. Authors [21] compare the performance of five metaheuristics (tabu search, simulated annealing, genetic algorithm, iterated local search and ant colony optimization) with and without automated parameter tuning on a VRP with stochastic demands. The parameters from the non-tuned algorithms were randomly drawn within a given range, while the parameters from the tuned versions were obtained through an automatic configuration process based on the F-Race algorithm [22]. For every metaheuristic, the tuned version achieves significantly better results than the corresponding non-tuned version. While traditionally parameter values have been set manually using expertise and experimentation, recently several automated tuning methods have been proposed. Authors [23] compare the performance of seven state-of-the-art algorithm configuration methods on different routing metaheuristics. Their findings confirm the results of [21] that the performance of the routing algorithm can be clearly improved by using parameter tuning. The results also reveal that there is no single best tuning method for routing algorithms, but that the Iterated F-Race algorithm seems to be the most robust one. Iterated F-race also performed very well on the ILS metaheuristic. Therefore, Iterated F-race will be used for the parameter tuning of the ILS in the present research.

For the tuning of the parameters, 20 test instances with sizes ranging from 20 to 75 customers are used. The value for parameter α is determined based on the results of a single run of the test instances, in which no substantial improvement was found after more than 220 consecutive non-improving iterations. The values of the other parameters were determined for this run based on preliminary tests. Since this value was the result of a single run, α is set to 250, to incorporate a margin of 10%. The parameter space for the remaining five parameters of the ILS algorithm is denoted by $X = \{ \delta, \eta^0, \eta^{incr}, \eta^{heavy}, \beta \}$. Each parameter $X_d \in X$ may take different values within a specified range $\left[\underline{x_d}, \overline{x_d}\right]$. A configuration of the algorithm $\theta = \{x_1; x_2; x_3; x_4; x_5\}$ is a unique assignment of values to these parameters [24]. The tuning problem is stated by [25] as the problem of finding the configuration θ that provides the lowest expected cost on a set of problem instances. In order to find this configuration, the irace package provided by [24] is used. The irace package is designed for automatic algorithm configuration and implements the iterated racing procedure, which is an extension of the Iterated F-race procedure [24]. Irace is also successfully used by other authors for the tuning of heuristic algorithms in similar applications to ours such as by [26] and [27]. The iterated racing procedure is presented in Algorithm 5.

The input of the iterated racing procedure consists of an instance set I, parameter space X, a cost function C and a tuning budget B. The cost function returns the cost of configuration θ on instance i. The tuning budget refers to the number of calls to the ILS that irace will perform. Based on the number of parameters that are tuned (N^{param}), the estimation of the number of iterations N^{iter} is made with $N^{iter} = \lfloor 2 + \log 2\ N^{param} \rfloor$. Since five parameters are tuned in our algorithm, the value for $N^{iter} = 4$. Note that N^{iter} is the estimation of the number of iterations. In case, after N^{iter} iterations, there is still enough budget to perform a new race, the algorithm continues. The tuning budget B_j for iteration j depends on the tuning budget B, the tuning budget already used in previous iterations B_{used}, N^{iter} and the iteration number:

$$B_j = \frac{B - B_{used}}{N^{iter} - j + 1}. \quad (1)$$

Algorithm 5. Iterated Racing [24]

Require: $I = \{I_1; I_2; \ldots\}$,
1: parameter space: X,
2: cost measure: $C(\theta, i) \in \mathrm{R}$,
3: tuning budget: B
4: $\Theta_1 \sim SampleUniform(X)$
5: $\Theta^{elite} := Race(\Theta_1, B_1)$
6: $j := 2$
7: **while** $B_{used} \leq B$ **do**
8: $\quad \Theta^{new} \sim Sample(X, \Theta^{elite})$
9: $\quad \Theta_j := \Theta^{nes} \cup \Theta^{elite}$
10: $\quad \Theta^{elite} := Race(\Theta_j, B_j)$
11: $\quad j := j + 1$
12: **end while**

In the first step of the first iteration ($j = 1$) of the iterated racing procedure, candidate configurations Θ_1 are sampled according to a uniform distribution in the parameter space X (line 4 in Algorithm 5). In the second step, a racing procedure is used to select elite candidate configurations Θ^{elite} from the configurations sampled in the previous step (line 5). The racing procedure consists of several steps. In each step of the race, the candidate configurations are evaluated on a set of instances by means of a cost measure C. The order in which the instances are considered is randomized. For the evaluation of the candidate configurations, the rank-based Friedman test is used. If a candidate configuration performs statistically worse than at least one other configuration, this configuration is discarded in the next step. The racing procedure is terminated when a minimum number of surviving configurations is reached ($N_j^{surv} \leq N^{min}$) or when the number of surviving configurations N_j^{surv} exceeds the remaining tuning budget B_j for race j.

At the end of the race, the surviving configurations are ranked according to the mean cost and assigned to a rank value r_z. The set of the elite configurations Θ^{elite} is composed of the $N_j^{elite} = min\ (N_j^{surv}, N^{min})$ configurations with the lowest rank. For the generation of a new candidate configuration for the next race, a parent configuration θ_z is sampled from the set of elite configurations Θ^{elite} with a probability p_z proportional to its rank r_z. A higher ranked elite configuration has a higher probability of being selected as a parent:

$$p_z = \frac{N_{j-1}^{elite} - r_z + 1}{N_{j-1}^{elite} \cdot \frac{\left(N_{j-1}^{elite} + 1\right)}{2}}. \quad (2)$$

A new value is sampled for each parameter X_d within the given range according to a normal distribution $N(x_d^{\bar{}}, \sigma_d^{\bar{}})$ (line 8). For the integer parameter δ, the sampled value is rounded to the nearest integer. The mean of the distribution $x_d^{\bar{}}$ is the value of parameter X_d in elite configuration θ_z. The standard deviation σ_d^j decreases at each iteration j and depends on the value of the standard deviation in the previous iteration (σ_d^{j-1}), the number of newly sampled configurations (N_j^{new}) and the number of parameters to be tuned (N^{param}). The parameter σ_d^1 is set to $\underline{(x_d} - \overline{x_d})/2$.

$$\sigma_d^j = \sigma_d^{j-1} \cdot \left(\frac{1}{N_j^{new}}\right)^{1/N^{param}}. \quad (3)$$

The new set of candidate configurations consists of N_{j-1}^{elite} elite configurations from the previous iteration Θ_{elite} and the N_j^{new} newly sampled configurations (line 9). These configurations are used in a new racing procedure to select elite candidate configurations Θ_{elite} (line 10). This process (lines 8 to 11) is repeated until the number of experiments B_{used} exceeds the maximum number of experiments specified in the tuning budget B. The number of candidate configurations N_j in iteration j depends on the tuning budget B_j, the iteration number j and parameter μ:

$$N_j = \frac{B_j}{\mu + \min(5, j)}. \quad (4)$$

Parameter μ has a default value of 5 in irace. This value may be changed in order to influence the ratio between budget and number of configurations. At each iteration, the number of candidate configurations Θ_j decreases to allow for more evaluations per candidate configuration in later iterations. After five iterations, the number of candidate configurations remains constant at value Θ_5 in order to avoid having too few configurations in a single race. For more information regarding the tuning process, the reader is referred to [24].

Irace is run with default parameter values on the ILS heuristic. A tuning budget B of 5000 runs is specified. The parameters tuned by irace may be found in Table I, along with the type, range and tuned value. δ is integer (i), while η_0, η_{heavy}, η_{incr}, β are real parameters (r). The ranges for the parameters are intuitively determined. The upper bound of δ is equal to 250, which is the value for α. For the other parameters, the upper bound was increased when the tuned value was close to an upper bound. The lower bound of β is set to zero to test the case in which a deterioration of the solution value is not accepted as a new incumbent solution. The lower bound of η_{heavy} is also set to zero to test the case without heavy perturbation. The lower bound of η_0 is greater than 0 since there must be a non-zero value for noise in the perturbation phase to escape from the local optimum. For the real parameters (η_0, η_{heavy}, η_{incr}, β), two decimal places are considered.

TABLE I
PARAMETER LIST

Name	Description	Type	Range	Tuned value
δ	# non-improving_it heavy perturbation	i	(1, 250)	196
η_0	Initial value η	r	(0.01, 0.80)	0.33
η_{heavy}	Increase η heavy perturbation	r	(0.0, 1.0)	0.14
η_{incr}	Increase η when solution is not changed during perturbation	r	(0.0, 0.50)	0.22
β	Threshold value	r	(0.0, 0.50)	0.10

The introduction of heavy perturbation as the acceptance of a worse solution based on record-to-record travel appears to have a positive influence on solution quality since η_{heavy} and β have non-zero tuned values. In the next section, this influence is further analysed by means of a sensitivity analysis.

IV. SENSITIVITY ANALYSIS

In this section, the performance of the algorithm in terms of solution quality is tested with respect to different parameter values. The aim of this analysis is to verify whether the parameter tuning produced logical results as well as to test the importance of the parameter values for the efficacy of the ILS algorithm. For each parameter, different values are tested on various instance sets while keeping other parameters at their tuned value. We illustrate the working of the sensitivity analysis with 12 instances of size 50.

The instances in the instance sets only differ in network size. Other characteristics, such as pallet weight and number of pallets per customer, are assigned in a similar way to all instance sets. The performance of each parameter setting is measured with five independent runs. The best and average increase in solution cost compared to the lowest cost found over all experiments for that instance are plotted. Note that for each instance set a different scale is used on the vertical axis of the graph because of the large difference in cost increase between the different instance sets. For all graphs of the same instance set, however, the same scale on the vertical axis is maintained for the sensitivity analyses of the average and best cost of all parameters.

A. Sensitivity Analysis of η^0

The impact of η^0, the initial value for noise, on the solution quality of the instance set is plotted in Fig. 1. The following values are considered for η^0: 0.01, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.80 and 1.0. The tuned configuration, in which η^0 has a value of 0.33, is also included. We see that when η^0 exceeds 0.40, the solution quality deteriorates.

The variation in solution quality for the instance set with respect to changes in the initial value of noise is explored by means of a boxplot presented in Fig. 2. In each boxplot the first, second (median) and third quartile as well as the minimum and maximum cost increase over all instances are indicated in the instance set. Note that a different scale is used on the vertical axis since the maximum cost increase highly exceeds the average cost increase over all instances in the instance set

reported in the previous graphs. The figure shows that the variation in solution quality for the instances of the set decreases when η^0 reaches 0.40.

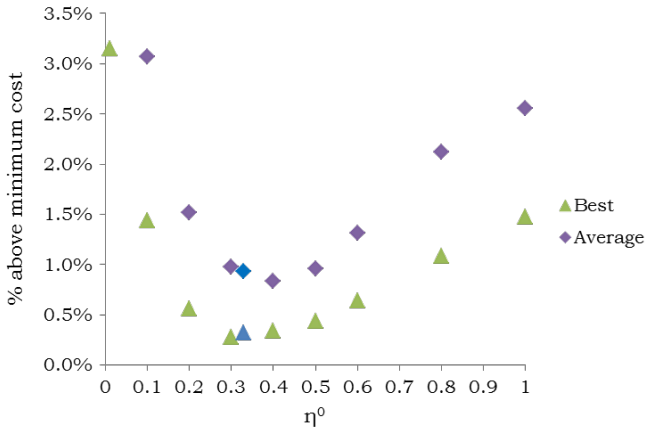

Fig. 1. Sensitivity of η^0 on the instance set.

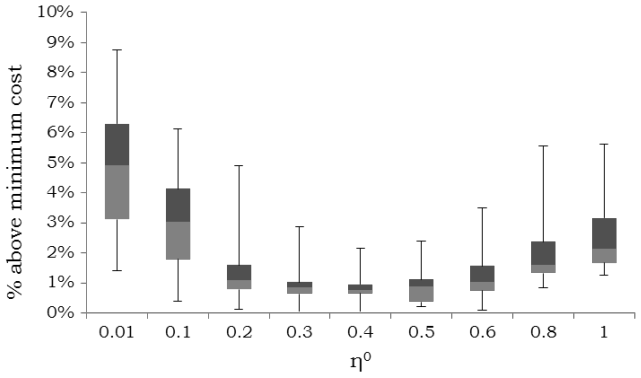

Fig. 2. Boxplot of the objective value increase in function of η^0.

B. Sensitivity Analysis of β

For the sensitivity analysis of β, the factor that determines the threshold value in the acceptance criterion, values between 0.0 and 0.5 are considered in steps of 0.10. To have an idea of the effect of a very high value of β, the value 1.0 is also considered. In this setting, a solution with a cost twice as high as the cost of the best-known solution will still be accepted as the incumbent solution in the next iteration of the ILS. For the instance set, depicted in Fig. 3, the solution quality improves when worsening is accepted ($\beta > 0.0$). Based on the figure, it appears that the tuned value 0.10 leads to the best solution quality although the difference with higher values of β is very small. The solution quality clearly benefits from accepting worse solutions and, in addition, the tuned value ($\beta = 0.1$) renders the best solution quality although the difference with higher values of β is small.

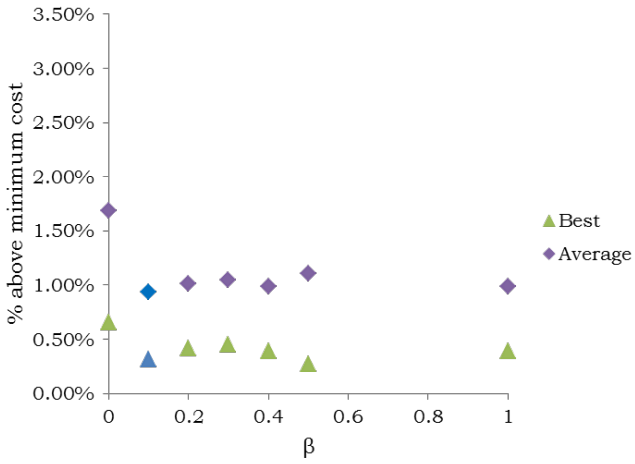

Fig. 3. Sensitivity of β on the instance set.

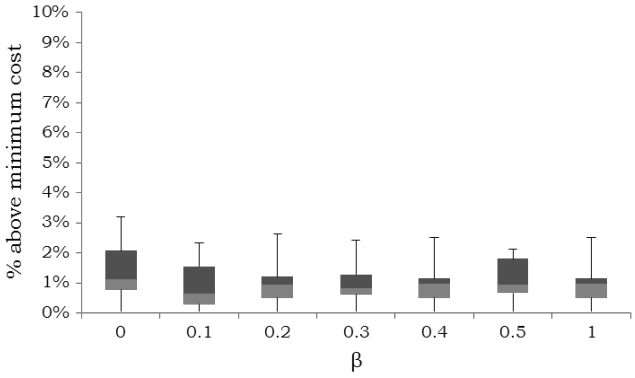

Fig. 4. Boxplot of the objective value increase in function of β.

Figure 4 presents the variation in solution quality for the instances of the set with respect to changes in the threshold value. The boxplot of the cost increase of the tuned value of β clearly lies lower than the boxplot of the configuration with a threshold value of $\beta = 0.0$, except for the minimum values which are equal for both configurations. When β increases to a value higher than 0.1, no clear trend can be distinguished in the graph.

C. Sensitivity Analysis of δ and η_{heavy}

The impact of δ, the number of non-improving iterations of the ILS after which heavy perturbation is applied, on solution quality is measured simultaneously with the impact of η_{heavy}, the increase in η during heavy perturbation. To this end, ten combinations are created by varying the values for δ and η_{heavy}. The first combination represents the situation, in which no heavy perturbation is applied. In Table II, the remaining nine combinations are presented. δ may have a low (50), medium (125) or high (200) value. Similarly, η_{heavy} is assigned to a low (0.10), medium (0.30) and high (0.60) level. The tuned values of δ and η_{heavy} are 196 and 0.14, which correspond most to combination 4 with a high value of δ combined with a low value of η_{heavy}.

TABLE II
COMBINATIONS FOR THE HEAVY NOISE ANALYSIS

	$\delta = 50$	$\delta = 125$	$\delta = 200$
$\eta_{heavy} = 0.1$	Combination 2	Combination 3	Combination 4
$\eta_{heavy} = 0.3$	Combination 5	Combination 6	Combination 7
$\eta_{heavy} = 0.6$	Combination 8	Combination 9	Combination 10

Figure 5 plots the average solution quality of the instances of the instance set with respect to the different combinations. A trend may be distinguished in which the combinations with a low value of η_{heavy} have a higher solution quality than the other combinations. Furthermore, when a moderate or high value of η_{heavy} is considered, the value of δ also seems to influence the solution cost. A low value of δ appears to have a negative effect on the solution quality. Note that, surprisingly, combination 1, in which no heavy perturbation is applied, has the best solution quality although the difference in the combinations with a low value of heavy noise is negligible. The introduction of heavy noise therefore does not appear to have a positive impact on the solution quality for the instances.

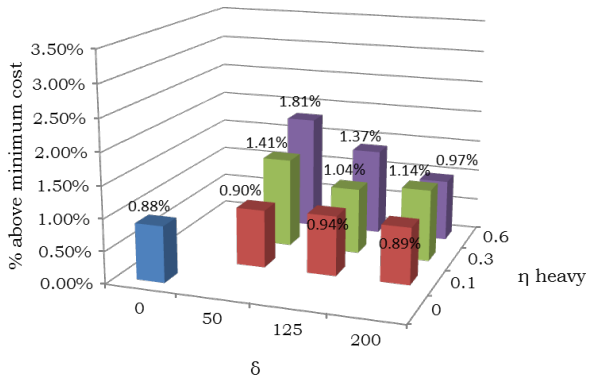

Fig. 5. Sensitivity of δ and η_{heavy} on the instance set.

D. Sensitivity Analysis of η_{incr}

As discussed earlier, in case the solution is not changed during the perturbation phase, the value of noise is incremented with η_{incr} and the perturbation is repeated. However, in all runs of the instances under study, the solution changes during every perturbation. Therefore, the value of η_{incr} does not have an impact on the solution quality of the instances.

E. Sensitivity Analysis of α

The impact of the number of consecutive non-improving iterations after which the ILS is stopped (α) on the solution quality and CPU time is illustrated in Fig. 6 for the instance set. Values between 50 and 500 with a step of 50 are considered for α. As one can expect, if α increases, CPU time and solution quality also increase. The largest gains in solution quality are obtained when α is small. For the instance set, the solution quality does not increase much beyond $\alpha = 300$.

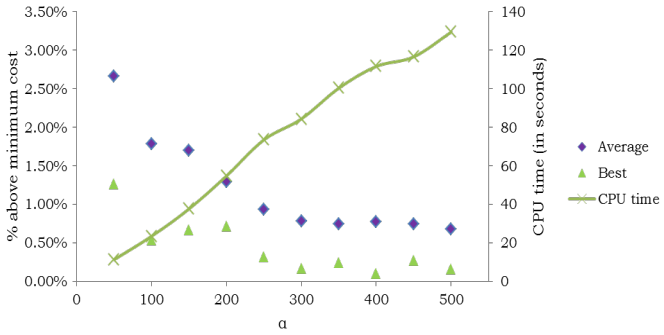

Fig. 6. Sensitivity of α on the instance set.

F. Contribution of the Local Operators

In this section, the contribution to solution quality of the local search algorithm and the different local search operators is analysed. Five variants of the ILS are analysed on the instance sets (not only the set with 50 customers as used in the previous sections, but also of sizes 10, 20 and 75). In the first variant, only an initial solution is generated. No local search is performed. In the other variants, each time a single local search operator is removed from the search. Five independent runs of the variants of the ILS are performed on each instance. Table III gives an overview of the results. When the local search is removed, the average gap with the original algorithm is very large, ranging from 69.51 % to 98.24 % as can be expected. Individually, the local search operators also have a contribution to solution quality although much smaller. The influence of the local search operators is larger on realistic size instances with 50 and 75 customers. The relocate operator seems to have the largest influence on the results for the realistic-size instances. Note that interaction effects between local search operators are ignored in this analysis.

TABLE III
CONTRIBUTION OF THE LOCAL SEARCH OPERATORS

	Average gap with original algorithm			
	Instance set 1 (tuning 20–75 cust)	Instance set 2 (20 cust)	Instance set 3 (50 cust)	Instance set 4 (75 cust)
No local search	69.51 %	69.60 %	96.50 %	98.24 %
No exchange	0.03 %	0.00 %	0.25 %	0.53 %
No 2-opt	0.01 %	0.00 %	0.20 %	0.52 %
No cross-exchange	−0.05 %	-0.01 %	0.10 %	0.46 %
No relocate	−0.01 %	0.02 %	0.27 %	0.68 %

G. Conclusion on the Sensitivity Analysis

From the sensitivity analysis of different size instance sets, it appears that there is an interaction between the size of the network and the effect of the parameters on the solution quality of the metaheuristic. As a result, the acceptance of worse solutions based on record-to-record travel, the heavy noise perturbation and the noise increment do not have an added value for instances of all sizes. The analysis indicates, however, that in these cases the solution quality is also not negatively influenced. The algorithm needs to perform well on instances

of all sizes and these mechanisms all have proven to contribute to the solution quality for at least a subset of instances.

The initial value of noise η^0 has the largest impact on the solution quality. For all instance sets, the solution quality greatly improves when η^0 increases to a value of 0.30. For the tuning instances and the instances of size 20, the solution quality does not change when η^0 further increases. For the instances of size 50 and 75, however, the solution quality reaches a maximum when η^0 reaches 0.30 and 0.40, respectively. The solution quality decreases when η^0 further increases.

The value of β has a considerable impact on instances with 20 or 50 customers. For these instances, the solution quality greatly improves when β has a non-zero value. The tuned value $\beta = 0.10$ renders the best solution quality, although the difference in higher values of β is very small. For the instances of size 75, the tuned value also renders the best solution quality, where higher values of β produce worse solutions than the configuration where $\beta = 0.0$.

The impact of δ and η_{heavy} on the solution quality is investigated simultaneously. The impact of these parameters on small-size instances of set 2 is rather small, while for the instances of sets 1, 3 and 4, the configurations with a low value of η_{heavy} produced a higher solution quality than the configurations with a moderate and high value of η_{heavy}. Furthermore, in these instance sets a low value of δ appears to have a negative effect on the solution quality, especially when combined with a moderate or a high value for η_{heavy}. The value of η_{incr} is only relevant in very small-size instances. The reason for this is that an increment is performed only when the perturbation does not change the solution, which does not occur in the instances with 50 or 75 customers and occurs only rarely in the instances of size 20.

As a result, the sensitivity analysis shows that the tuned setting for the parameters tuned by irace $\{\delta, \eta^0, \eta^{incr}, \eta^{heavy}, \beta\}$ renders a good solution quality for all instance sizes.

With regards to the value of α, it may be concluded that on instance sets 1 and 2 the increase in α beyond the tuned value of 250 does not yield a quality increase. For the instances of sets 3 and 4, the solution quality does not increase much beyond $\alpha = 300$ and $\alpha = 350$, respectively. Based on these results, α is set to 300.

The relevance of the algorithmic components of the ILS has been demonstrated in the foregoing analyses. The contribution of the acceptance of worse solutions based on record-to-record travel, the heavy noise perturbation and the noise increment is analysed in the sensitivity analysis of the parameters. Record-to-record travel has a positive influence on the solution quality since for all instance sets, a threshold value (β) of 0.10 leads to a higher solution quality than a threshold value equal to zero. A high value of noise after a number of consecutive non-improving iterations also has a positive effect on the solution quality, although this effect appears to depend on the instance size. A noise increment after perturbation that did not change the solution is only relevant in very small-size instances. The contribution of this component can only be demonstrated in instances of size 10. For larger instances, this component has no influence on the solution quality. For the contribution of the local search operators, it appears that the relocate operator has the largest impact on the solution quality for realistic size instances. The impact on solution quality of the local search operators individually does not seem to be very large. Interaction effects among local search operators have, however, not been considered. Furthermore, there is an interaction effect between the size of the network and the contribution of the local search operators.

Note that although the analysis shows that all algorithmic components contribute to the solution quality for at least of a subset of instances, it may be interesting to look at the possibility of using a simplified heuristic method consisting of fewer local search operators while maintaining or even increasing the efficiency of the solution method.

V. Conclusion

A metaheuristic algorithm for the capacitated vehicle routing problem with sequence-based pallet loading and axle weight restrictions has been proposed in the paper. The design and analysis of an Iterated Local Search algorithm has also been presented. The parameters of the metaheuristic have been tuned with automatic algorithm configuration software, which implements an iterated racing procedure that is an extension of the Iterated F-race procedure. Furthermore, a sensitivity analysis has been performed to analyse the impact of the values of the parameters of the ILS and to test the contribution of the algorithmic components of the ILS on the solution quality. Although the results show that the tuned configuration renders a good solution quality for all instance sizes, the sensitivity analysis also indicates that the impact of the parameter values depends on the size of the network. Furthermore, the contribution of the algorithmic components seems to interact with the instance size. The aim of the parameter tuning has been to develop an algorithm, which performs well on instances of all sizes. However, future research may focus on making a distinction, during the tuning process, between networks of different sizes. The differences from the current algorithm to algorithms designed for a specific network size may also be analysed.

References

[1] J. Caceres-Cruz, *et al*. "Rich Vehicle Routing Problem: Survey", *ACM Computing Surveys*, vol. 47, no. 2, Article 32, 2014. https://doi.org/10.1145/2666003

[2] R. F. Hartl, G. Hasle, and G. K. Janssens, "Special issue on rich vehicle routing problems", *Central European Journal of Operations Research*, vol. 14, no. 2, pp. 103–104, 2006. https://doi.org/10.1007/s10100-006-0162-9

[3] G. Hasle, and O. Kloster, "Industrial vehicle routing". In *Geometric Modelling, Numerical Simulation, and Optimization*, G. Hasle, K. A. Lie, and E. Quak (Eds). Springer, Berlin, pp. 397–435, 2007. https://doi.org/10.1007/978-3-540-68783-2_12

[4] H. Pollaris, *et al*. "Vehicle routing problems with loading constraints: State-of-the-art and future directions", *OR Spectrum,* vol. 37, pp. 297–330, 2015. https://doi.org/10.1007/s00291-014-0386-3

[5] B. Jacob, and V. F.-de La Beaumelle, "Improving truck safety: potential of weigh-in-motion technology", *IATSS (International Association of Traffic and Safety Science) Research*, vol. 34, no. 1, pp. 9–15, 2010. https://doi.org/10.1016/j.iatssr.2010.06.003

[6] T. Vidal, G. Laporte, and P. Matl, "A concise guide to existing and emerging vehicle routing problem variants", *European Journal of Operational Research,* vol. 286, no. 2, pp. 401–416, 2020. https://doi.org/10.1016/j.ejor.2019.10.010
[7] A. Lim, *et al*. "The single container loading problem with axle weight constraints", *International Journal of Production Economics*, vol. 144, no. 1, pp. 358–369, 2013. https://doi.org/10.1016/j.ijpe.2013.03.001
[8] M. T. Alonso, *et al*. "Mathematical models for multicontainer loading problems", *Omega,* vol. 66, Part A, pp. 106–117, 2013. https://doi.org/10.1016/j.omega.2016.02.002
[9] H. Pollaris, *et al*. "Capacitated vehicle routing problem with sequence-based pallet loading and axle weight constraints", *EURO Journal of Transportation and Logistics,* vol. 5, no. 2, pp. 231–255, 2016. https://doi.org/10.1007/s13676-014-0064-2
[10] H. Pollaris, *et al*. "Iterated local search for the capacitated vehicle routing problem with sequence-based pallet loading and axle weight constraints", *Networks,* vol. 69, no. 3, pp. 304–316, 2017. https://doi.org/10.1002/net.21738
[11] M. Iori, and S. Martello, "Routing problems with loading constraints", *TOP,* vol. 18, pp. 4–27, 2010. https://doi.org/10.1007/s11750-010-0144-x
[12] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search: framework and applications". In Gendreau, M. and Potvin, J. Y. (eds), *Handbook of Metaheuristics.* International Series in Operations Research & Management Science, vol. 146, pp. 363–397, 2010. Springer, Boston. https://doi.org/10.1007/978-1-4419-1665-5_12
[13] A. A Juan, *et al*. "Combining biased randomization with iterated local search for solving the multidepot vehicle routing problem", *International Transactions in Operational Research,* vol. 22, no. 4, pp. 647–667, 2015. https://doi.org/10.1111/itor.12101
[14] J. Brandao, "Iterated local search algorithm with ejection chains for the open vehicle routing problem with time windows", *Computers & Industrial Engineering,* vol. 120, pp. 146–159, 2018. https://doi.org/10.1016/j.cie.2018.04.032
[15] P. Chen, H. Huang, and X.-Y. Dong, "Iterated variable neighbourhood descent algorithm for the capacitated vehicle routing problem", *Expert Systems with Applications,* vol. 37, no. 2, pp. 1620–1627, 2010. https://doi.org/10.1016/j.eswa.2009.06.047
[16] C. Waters, "A solution procedure for the vehicle scheduling problem based on iterative route improvement", *Journal of the Operational Research Society*, vol. 38, no. 9, pp. 833–839, 1987. https://doi.org/10.1057/jors.1987.137
[17] G. A. Croes, "A method for solving traveling salesman problems", *Operations Research,* vol. 6, no. 6, pp. 791–812, 1958. https://doi.org/10.1287/opre.6.6.791
[18] E. Taillard, *et al*. "A tabu search heuristic for the vehicle routing problem with soft time windows", *Transportation Science*, vol. 31, no. 2, pp. 170–186, 1997. https://doi.org/10.1287/trsc.31.2.170
[19] S. Ropke, and D. Pisinger, "An adaptive neighbourhood search heuristic for the pickup and delivery problem with time windows", *Transportation Science*, vol. 40, no. 4, pp. 455–472, 2006. https://doi.org/10.1287/trsc.1050.0135
[20] H. H. Hoos, "Automated algorithm configuration and parameter tuning". In Hamadi, Y., Monfroy, E., Saubion, F. (Eds), *Autonomous Search*. Springer, Berlin, Heidelberg, pp. 37–71, 2012. https://doi.org/10.1007/978-3-642-21434-9_3
[21] P. Pellegrini, and M. Birattari, "Out-of-the-box and custom implementation of metaheuristics. A case study: The vehicle routing problem with stochastic demand." In Köppen, M., Schaefer, G., Abraham, A. (Eds), *Intelligent Computational Optimization in Engineering Techniques and Applications*. Springer, Berlin, Heidelberg, pp. 273–295, 2011. https://doi.org/10.1007/978-3-642-21705-0_10
[22] M. Birattari, *et al*. "A racing algorithm for configuring metaheuristics." In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO '02*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 11–18, 2002.
[23] J. Rasku, N. Musliu, and T. Kärkkäinen, T. "Automating the parameter selection in VRP: an off-line parameter tuning tool comparison." In Fitzgibbon, W., Kuznetsov, A. Y., Neittaanmäki, P., Pironneau, O. (Eds), *Modeling, Simulation and Optimization for Science and Technology*, vol. 34, pp. 191–209, 2014. Springer, Netherlands. https://doi.org/10.1007/978-94-017-9054-3_11
[24] M. Lopez-Ibanez, *et al*. "The irace package Iterated racing for automatic algorithm configuration", *Operations Research Perspectives,* vol. 3, pp. 43–58, 2016.
[25] M. Birattari, *Tuning Metaheuristics: A Machine Learning Perspective*, 1st Edition. Springer Publishing Company, 2009. https://doi.org/10.1007/978-3-642-00483-4
[26] S. Ceschia, A. Schaerf, and T. Stützle, "Local search techniques for a routing packing problem", *Computers & Industrial Engineering*, vol. 66, no. 4, pp. 1138–1149, 2013. https://doi.org/10.1016/j.cie.2013.07.025
[27] V. François, *et al*. "Large neighborhood search for multi-trip vehicle routing", *European Journal of Operational Research,* vol. 255, no. 2, pp. 422–441, 2016. https://doi.org/10.1016/j.ejor.2016.04.065

Hanne Pollaris received the M. Sc. degree in Business Engineering in 2012 from Hasselt University. In March 2017, she obtained a PhD degree in Business Economics from Hasselt University.
Her PhD research focused on the integration of axle weight constraints in vehicle routing problems. From then on, she has been employed as a Lecturer at Hasselt University and at Hogeschool PXL.
Address: Faculty of Business Economics, Research Group Logistics, Agoralaan 1, B-3590 Diepenbeek, Belgium.
E-mail: hanne.pollaris@uhasselt.be
ORCID iD: https://orcid.org/0000-0003-3176-4057

Gerrit K. Janssens received the M. Sc. degree in Business Engineering from the University of Antwerp (RUCA), Belgium, M. Sc. degree in Computer Science from the University of Ghent (RUG), Belgium, and Dr. sc. ing. in Computer Science in 1989 from the Free University of Brussels (VUB), Belgium. After some years of work at General Motors Continental, Antwerp, he joined the University of Antwerp.
He was a Professor of Operations Management and Logistics at Hasselt University (UHasselt) within the Faculty of Business Economics from 2000 to 2016. He is now an Emeritus Professor within the Research Group Logistics.
He was the President of the Belgian Operations Research Society (ORBEL) in the period of 2006–2007. During the past twenty-nine years, he has been several times visiting universities in South-East Asia and Africa. His main research interests include the development and application of operations research models in production and distribution logistics.
Address: Faculty of Business Economics, Research Group Logistics, Agoralaan 1, B-3590 Diepenbeek, Belgium.
E-mail: gerrit.janssens@uhasselt.be
ORCID iD: https://orcid.org/0000-0002-9857-5130

Kris Braekers received the degree of M. Sc. in Business Engineering and his PhD in Business Economics from Hasselt University in 2008 and 2012, respectively. Later, he was appointed at Hasselt University as a postdoctoral researcher (2012–2014) and as a postdoctoral fellow of the Research Foundation Flanders (2014–2017).
Since 2015, he has been an Assistant Professor at the Research Group Logistics, Faculty of Business Economics, Hasselt University.
His main research interest is modelling and solving complex combinatorial optimization problems in the context of transport and logistics, mainly using meta-heuristic solution approaches. In the past years, he has particularly been focusing on complex vehicle routing problems in a distribution or healthcare context. Other ongoing research topics include warehouse optimization and intermodal transport planning.
Address: Faculty of Business Economics, Research Group Logistics, Agoralaan 1, 3500 Hasselt, Belgium.
Email: kris.braekers@uhasselt.be
ORCID iD: https://orcid.org/0000-0001-6052-6846

An Caris is an Associate Professor at the Research Group Logistics of the Faculty of Business Economics, Hasselt University. She wrote her Doctoral Thesis in 2010 on the topic "Simulation and Optimisation of Intermodal Barge Transport Networks" and was a postdoctoral research fellow of the Research Foundation Flanders (FWO, 2011–2015).
She is passionate about synchromodal transport and the physical internet. Other research interests are decision support in warehouse operations and health care logistics. From a technical viewpoint, she is also involved in the statistical analysis of algorithms for efficient parameter setting of meta-heuristic algorithms.
Address: Faculty of Business Economics, Research Group Logistics, Agoralaan 1, B-3590 Diepenbeek, Belgium.
E-mail: an.caris@uhasselt.be
ORCID iD: https://orcid.org/0000-0001-6004-8776