Route Optimisation for Winter Maintenance

Nikmal Raghestani and Carsten Keßler

Abstract In many countries, winter maintenance is a requirement to keep public life going throughout the cold season. This paper investigates the optimization of salt spreading routes in Denmark in terms of service time and cost. It looks at salting as a capacitated arc routing problem and proposes a greedy randomized adaptive search procedure to this end. At the core of the proposed approach is a heuristic algorithm based on simulated annealing that improves the initial route by searching for alternatives within a predefined search space, taking into account a number of constraints and criteria at each iteration of the procedure. The performance of the optimization approach is tested on three different existing service routes, where it is shown to reduce route length by an average of 8.7% and service time by an average of 9.5%.

Keywords Capacitated Arc Routing Problem, Simulated Annealing, Route Optimization

1 Introduction

The nordic winter weather requires the authorities to remove snow and take measures against road icing in order to maintain accessibility and traffic safety. In Denmark, The Danish Road Directorate (Vejdirektoratet, VD) and the municipalities are responsible for these tasks. This paper investigates the potential for optimisation of the winter maintenance services in terms of service time and cost. We focus on salting, as this is by far the most common winter road maintenance activity (Knudsen et al., 2014). Planning of winter maintenance activities consists of various types of decision-making problems at the strategic (facility locations such as plants, depots, material storages), tactical (fleet size, fleet combination and other vehicle/material

Nikmal Raghestani, e-mail: nikmaljuve@gmail.com · Carsten Keßler
Department of Planning, Aalborg University Copenhagen, Denmark e-mail: kessler@plan.aau.dk

related issues) and operational (determination of routing and scheduling of each vehicle) levels (Bodin et al., 1981). These levels are interconnected, but the strategical and tactical levels are often bound by a number of inflexible circumstances, so that the largest potential for optimisation is in the vehicle routing at the operational level.

This routing problem is based on a set of arcs that a fleet of vehicles has to traverse. The objective is to determine feasible routes for each vehicle with respect to the cost variables. Since one of the cost variables in road salting is the salt capacity on a vehicle, this kind of vehicle routing problem is a Capacitated Arc Routing Problem (CARP) (Eiselt et al., 1995). Since CARPs fall into the class of NP-hard, *non-deterministic polynomial-time hard* problems (Golden and Wong, 1981), heuristic procedures are required in order to approach a near-optimal solution within acceptable time. This paper applies an approach based on Simulated Annealing (SA) for this purpose.

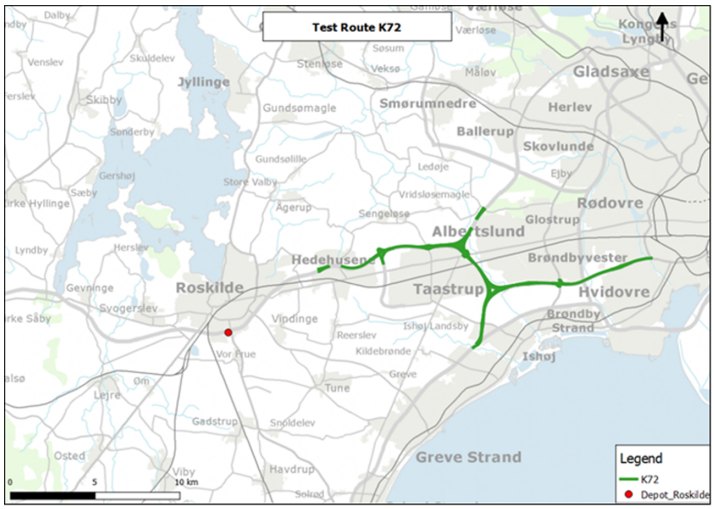

Fig. 1 Winter maintenance route K72, West of Copenhagen, with depot location in Roskilde.

Figure 1 shows one of the winter maintenance routes served by VD. During the optimization process, the route itself, shown in green, cannot be changed due to service contracts that VD and the municipalities have with private road maintenance firms, which they can bid on in public tenders. The optimisation is therefore limited to the routing between the depot and the route, the order in which the green segments are serviced, as well as the *deadheading* during turns and between different segments of the route, during which no salt is brought onto the road. Despite these limitations for route optimisation, manual inspection of the existing routing instructions has shown potential for optimisation due to the fact that the current routing instructions, including deadheading and getting from depot to route, have been created manually. The remainder of this paper will show that despite these limitations for optimisation, a heuristic approach based on simulated annealing could reduce route length by an

average of 8.7% and service time by an average of 9.5% across a set of three test routes.

The next section introduces relevant related work. Section 3 formalizes the problem, followed by an introduction of the heuristic algorithm in Section 4. Section 5 outlines the implementation and presents the achieved results, followed by concluding remarks in Section 6.

2 Related Work

Scientific research on winter maintenance in terms of road salting and snow removal first began in the 1960s. In the 1980s and 1990s, winter maintenance as Arc Routing Problems (ARPs) got more attention (Lemieux and Gampagna, 1984, Evans and Weant, 1990, Goode and Nantung, 1995, Haslam and Wright, 1991, Gendreau et al., 1992). Eglese (1994) conducted research with the objective to determine the most cost-effective management of salt spreading. The approach was based on a heuristic algorithm with respect to practical constraints such as service time, deadheading time, vehicle capacities, multiple depot locations, etc. The heuristic procedure was initiated with a construction algorithm, first introduced by Male et al. (1977), and an SA heuristic was then used for improvement. The study showed that the use of SA improved the initial results (Eglese, 1994).

The idea of SA was introduced by Metropolis et al. (1953) based on the cooling of molten materials, where the cooling rate determines the structural properties of the solid mass. Metropolis et al. (1953) proposed the use of SA in order to simulate the physical system's cooling process and change in energy until a state of thermal equilibrium is obtained. This is ensured with the stopping conditions of either a minimum temperature or a maximum number of iterations, thus retaining the possibility of accepting a non-improving solution in order to escape *local* minima while searching for a *global* minimum. Kirkpatrick et al. (1983) proposed the use of SA to search for near-optimal solutions in the context of routing. They proposed that the search be carried out in a wide solution space with respect to decreasing probability depending on the progresses of the algorithm by moving arcs between the routes.

The Capacitated Arc Routing Problem (CARP) has been defined based on a connected undirected graph $G = (N, A, C, Q)$, where N is a set of nodes, and A is a set of arcs. C denotes a cost vector, indicating the cost of traversing each arc, and Q is a corresponding demand matrix, indicating the demand for each arc. Given a number of identical vehicles with capacity W, the problem is to find a number of tours so that 1) Each arc with positive demand is serviced by exactly one vehicle, 2) The sum of demand of those arcs serviced by each vehicle does not exceed W, and 3) The total cost of the tours is minimized. (Golden and Wong, 1981).

Existing heuristic algorithms for CARP can be divided into construction methods and improvement methods. Simple construction methods include the *Construct-Strike Algorithm* (Christofides, 1973, Pearn, 1989), which is one of the simplest

CARP construction methods. This method progressively constructs and removes feasible routes from the original graph without disconnecting the graph until no more feasible routes can be determined. The *Path Scanning Algorithm* (Bodin et al., 1981) uses a set of five arc selection rules. This is done by selecting one rule at a time and determining the optimal routes for each of the five rules. This method was extended by Pearn (1989), who suggested random use of the rules and then selecting the optimal solution. Furthermore, Bodin et al. (1981) proposed an *Augment-Merge Algorithm* for CARP which heuristically creates feasible routes based on incomplete routes. Using the same method, Chapleau et al. (1984) proposed the *Parallel-Insert Algorithm*, which routes in parallel, while balancing the costs of the different routes. Moreover, there are a number of heuristics that follow a two-phase construction approach by either clustering route segments first and then creating routes from those clusters, or vice versa (Eiselt et al., 1995).

The heuristics for these construction methods are problem-dependent and often too greedy. This can cause trapping in a local minimum, thus failing to obtain the global minimum. To avoid this, there is a need for meta-heuristics that are not greedy and can accept a temporary solution deterioration. This allows exploration of the solution space in order to find a better solution towards the global minimum. The meta-heuristics are improvement methods for an initially constructed solution and there are a number of meta-heuristics for various problem statements within the CARP. In this paper, we use a two-phase algorithm consisting of a construction phase that generates an initial route and a local search phase, where an SA-based meta-heuristic is used to improve the initial route. While we focus on an SA-based approach in this paper, recent work on the CARP also employs genetic algorithms (Arakaki and Usberti, 2018, Kurniawan et al., 2015).

3 Problem Formulation

In order to generalise the approach presented in this paper beyond salt spreading in Denmark, the development will be based on the following problem formulation:

Given a graph with both directed and undirected arcs, some of which may not require service but can be used for deadheading, and given a number of vehicles with different capacities, find a total least time set of routes that start and end at the concerned depot, while satisfying the distance, capacity, time and frequency constraints.

An existing exact approach to this kind of combinatorial optimization is integer linear programming (Nemhauser and Wolsey, 1988). Integer linear programs (ILPs) can be solved with existing mathematical solvers, which allow computing globally optimal solutions for small problem instances. However, since our real-world problem instances are large and the existing algorithms for solving ILPs have an exponential worst-case runtime, we developed a heuristic method based on simulated annealing. Nevertheless, in the following, we define the problem in the form of an ILP.

The problem formalization is based on a model proposed by Golden and Wong (1981) and modified by Haghani and Qiao (2001). The parameters, variables and indexes used are shown in Table 1. The objective is to minimise the total travel distance (and by that, the total time, observing speed limits and assuming different maximum speeds for salting and deadheading), for servicing all required arcs with the vehicle(s) in use, including deadheading:

$$Minimize \sum_{p=1}^{k} \sum_{i,j \in A} C_{ij} X_{ij}^{p} \tag{1}$$

Notation	Definition	Unit
A	The network's set of arcs	$[number]$
C_{ij}	Distance between the nodes	$[m]$
D	The maximum total distance a vehicle can cover	$[m]$
i, j	Node index	$[number]$
F_{ij}^{p}	This decision variable ensures the elimination of sub-tours and is assigned to all arc (i, j) and vehicle p in the direction i to j.	$[number]$
G_{ij}	The difference between deadheading time and servicing time for an arc (i, j) in the direction i to j	$[minutes]$
k	The number of available vehicle(s) for service	$[number]$
L_{ij}^{p}	This variable indicates if an arc needs to be serviced and $L_{ij}^{p} = 1$ if an arc (i, j) has to be serviced by a vehicle p in the direction i to j	$[number]$
n	The total number of nodes in the network	$[number]$
N_{ij}	The number of times an arc (i, j) in the direction i to j needs to be serviced	$[number]$
p	Vehicle index	$[number]$
Q_{ij}	The demanded salt for an arc (i, j) in the direction i to j	$\left[m^2 * \frac{kg}{m^2}\right]$
T	The maximum total time a vehicle can cover	$[minutes]$
T_{ij}	Time for travel without servicing (deadheading), arc (i, j) in the direction i to j	$[minutes]$
W	Vehicle capacity	$[kg]$
X_{ij}^{p}	This variable indicates if an arc is "used" and $X_{ij}^{p} = 1$ if an arc (i, j) is used by vehicle p for either servicing or deadheading in the direction i to j	$[number]$

Table 1 Parameters, variables and indexes used in the problem formalization.

The minimisation has to be carried out with respect to the following constraints. The vehicle capacity constraint ensures that the vehicle capacity is not exceeded:

$$\sum_{i,j \in A} L_{ij}^{p} Q_{ij} \leq W \ \forall p = 1..k \tag{2}$$

The continuity constraint ensures that the vehicle enters and exits a node every time one is used, thus maintaining connectivity:

$$\sum_{\substack{r=1 \\ r \neq i}}^{n} X_{ri}^{p} - \sum_{\substack{r=1 \\ r \neq i}}^{n} X_{ir}^{p} = 0 \ \forall i = 1..n \ p = 1..k \tag{3}$$

The frequency constraint ensures the service of the demanded arcs. These arcs can either require service in one direction:

$$\sum_{p=1}^{k} L_{ij}^{p} + L_{ji}^{p} = 1 \ \forall (i,j) \in A \tag{4}$$

. . . or a finite number of times, N_{ij} in either direction if the width of the carriageways exceeds the salting width:[1]

$$\sum_{p=1}^{k} L_{ij}^{p} = \sum N_{ij} \ \forall (i,j) \in A, N_{ij} >= 1 \tag{5}$$

The service/deadheading constraint ensures that the service of an arc can only be carried out when a vehicle traverses the arc:

$$X_{ij}^{p} \geq L_{ij}^{p} \ \forall (i,j) \in A, p = 1..k \tag{6}$$

The time constraint ensures that the total travel time is not exceeded:

$$\sum_{i,j \in A} T_{ij} X_{ij}^{p} + \sum_{i,j \in A} G_{ij} L_{ij}^{p} \leq T \ \forall p = 1..k \tag{7}$$

The distance constraint ensures that the maximum total distance (service and deadheading) is not exceeded:

$$\sum_{i,j \in A} X_{ij}^{p} C_{ij} \leq D \ \forall p = 1..k \tag{8}$$

[1] Based on the salting vehicles used in Denmark, we assume a salting width of 8m.

The sub-tour cancellation constraint ensures the cancellation of sub-tours, which is a loop of tours without connection or inclusion of the depot, adapted from Golden and Wong (1981). When the network has at least one arc that requires service, then the right side of Eq. 9 will have a value greater than 0. To satisfy this equation, the left side containing the decision (flow) variables must be positive. According to Eq.10, an arc's decision variable can only be positive if a vehicle travels the arc. Thereby, Eq. 9 prevents sub-tours by ensuring that at least one arc which is connected to the depot is used when traveling.

$$\sum_{\substack{r=1 \\ r \neq i}}^{n} F_{ir}^{p} - \sum_{\substack{r=1 \\ r \neq i}}^{n} F_{ri}^{p} = \sum_{j=1}^{n} L_{ij}^{p} \ \forall i = 2..n,\ p = 1..k \tag{9}$$

$$F_{ij}^{p} \leq n^2 X_{ij}^{p} \ \forall (i,j) \in A, p = 1..k \tag{10}$$

$$F_{ij}^{p} \geq 0 \ \forall (i,j) \in A,\ p = 1..k \tag{11}$$

Due to the computational complexity of solving the equations shown above, a heuristic algorithm has to be developed that finds a near-optimal solution with respect to those constraints.

4 The Heuristic Algorithm

The heuristic proposed here is adapted from Resende and Ribeiro (1997) and based on the *Greedy Randomized Adaptive Search Procedure* (GRASP). GRASP consists of two phases: In the first phase, a feasible solution is generated by a *Route Construction Algorithm* (RCA) which is then improved in the second phase that searches for a better solution in its neighbourhood using a *Simulated Annealing* (SA) meta-heuristic to execute the local search for all the iterations. The aim of the iterative GRASP process is to reach the global minimum, i.e., to minimize overall salting time.

4.1 Phase One: Route Construction Algorithm

RCA traces a route from the depot by adding arcs incrementally with respect to the constraints (Section 3) for each iteration until a route is constructed. In every iteration of RCA, a list of arc candidates, a *Restricted Candidate List* (RCL), is generated by evaluating possible arc candidates that meet a greedy evaluation function. Furthermore, these arc candidates must be able to meet the feasibility of the partial solution, thus enabling arc exchange. This is used by the greedy evaluation function, which

calculates the change in total cost and only considers the candidates that contribute with an incremental cost below the threshold value. The selection of candidates from RCL is random and repeated until a final route is obtained. The *cost* refers to the adjusted lengths, which are then adjusted for the traverse time based on the speed limit. In addition, the arcs with a service demand will be termed the required arcs.

Variable	Definition
α	The threshold parameter between 0 and 1
C_{max}	Highest incremental cost
C_{min}	Lowest incremental cost
E	Possible candidate list
No_{Routes}	The number of developed routes

Table 2 The nomenclature of RCA

The RCA consists of the following steps (see Table 2 for the nomenclature):

Step 1: The shortest distance between each pair of nodes is calculated and NO_{Routes} is set to 0. The shortest distance can be calculated using Floyd-Warshall's algorithm (Cormen et al., 1990, pp. 558–565), which computes the shortest distances between all nodes in the graph.

Step 2: The current node is set to the depot node, partial route termed as no route, partial cost set to 0, the remaining capacity is set to the capacity of vehicle and the depot node is set as start node. If all the arcs' requirements are 0, i.e. no arcs require service in the network, then the process goes to step 7, otherwise to step 3.

Step 3: The process goes to step 4 if any required arcs are connected to the current node, otherwise step 6.

Step 4: The RCL with a cost in the range of $R = [C_{min}, C_{min} + \alpha * (C_{max} - C_{min})]$ is generated. α is a value between 0 and 1.

Step 5: If the RCL contains at least one arc, then a random selection is made and the selected arcs are added to the partial route. Then, the partial route is updated by setting it to the current partial route including the added arc and partial cost is set as partial cost including the cost of the added arc. Furthermore, the remaining capacity is set to the remaining capacity minus the required capacity for servicing the added arc. If the added arc has a traverse demand >= 1, then the demand is reduced by 1. In addition, the current node is set to end node of the recently added arc and the process returns to step 3, otherwise to step 6.

Step 6: If the addition of the required arc(s) is not possible due to violation of the constraints, then the shortest return route to the depot has to be determined. While doing this, the partial route is updated with the shortest return path to the depot, route cost is set to partial cost including the distance back to the depot and NO_{Routes} is set to NO_{Routes} + 1. After doing this, the process returns to step 2. Otherwise, the closest node connected to a required arc that does not exceed the constraints has to be determined and the shortest distance calculated. Then, the partial route is updated to the partial route including the distance to the closest

node, route cost is set to partial cost including the cost of the path to the closest node and the current node is set to the closest node and the process can go back to step 4.

Step 7: The solution is a set of routes that fulfills the demand of the required arcs in the network. NO_{Routes} is set to the total number of solution routes in the network.

4.2 Phase Two: Simulated Annealing Heuristic

The SA process starts with the best overall solution obtained from RCA and looks for a better one in a predefined neighbourhood. Since SA is a meta-heuristic, it allows for uphill moves, i.e. moves that generate solutions with higher costs than the current solution. This enables the process to escape local minima and find better solutions. The following pseudo-code outlines SA (Rere et al., 2015):

Algorithm 1 The local search

procedure SIMULATED ANNEALING
  Select the best solution vector x_0 to be optimised. Initialise the parameters:
  Temperature T, Boltzmann's constant k, reduction factor c
  while termination criterion is not satisfied **do**
    for number of new solution select a new solution: $x_0 + \delta x$
      if $f(x_0 + \delta x) < f(x_0)$ **then** $f_{new} = f(x_0 + \delta x); x_0 = x_0 + \delta x$
      else $\delta f = f(x_0 + \delta x) - f(x_0)$
        random $r(0, 1)$
        if $r < exp(-\delta f / kT)$ **then** $f_{new} = f(x_0 + \delta x)$, $x_0 = x_0 + \delta x$
        else $f_{new} = f(x_0)$, **do**
        end if
      end if
      $f = f_{new}$
      Decrease the temperature periodically: $T = c * T$
    end for
  end while
end procedure

The SA algorithm itself is embedded in a loop of moves of arcs in the route, followed by the execution of a route improvement algorithm, as outlined in the following.

The Movements

The movements for the local search are adapted from Pearn (1989), following the idea of "shuffling" subsets of arcs between two possible routes in order to find a better solution. This method randomly applies five arc selection rules to then select

the optimal solution. This is done by defining sets of moves where each contains every move only once. The randomness is achieved by sequencing the order of the moves in the sets randomly. Each move of each set is performed for a number of finite iterations. When all iterations for all moves in one set have been performed, the process continuous to a different set. Table 3 shows the nomenclature used for the moves in the following steps.

Variable	Definition
$Level_1$	The obtained routes, when the input route is splitting at 2 points and rearranged.
$Level_2$	The obtained routes, when the input route is splitting at 3 points and rearranged.
N_1	The number of best $Level_1$ routes.
N_2	The number of best $Level_2$ routes.
$Routex_1$	The route from which one or more arcs are removed
$Routex_2$	The route in which one or more arcs are inserted

Table 3 The nomenclature of the movements and the route improvement.

Move 1 moves one arc from $Routex_1$ to $Routex_2$ with respect to the constraints of the succeeding route by executing the following four steps:

Step 1: First, a candidate list containing arcs that can be moved from one route to another, is generated. These arcs have to respect the constraints of the route to which they are moved to. For this move, there are following three categories of such arcs; **a)** Both arcs being non-required arcs. **b)** One of the preceding or succeeding arcs being a service arc. **c)** Both arcs being service arcs.
This move can be executed with the preference hierarchy (probability of selection) of **a)** followed by **b)** and then **c)**.

Step 2: For every candidate arc, all the candidate routes, in respect to the constraints, are determined.

Step 3: Then, an arc is randomly selected from the candidate list in respect to the probability of selection.

Step 4: For the selected arc, a candidate route is selected randomly, cf. step 2.

Move 2 exchanges a pair of arcs between $Routex_1$ and $Routex_2$ with respect to the constraints of both routes by executing the following two steps:

Step 1: First, a candidate list containing all pair of arcs that can be exchanged between two routes is generated by complete enumeration. This is done with respect to the constraints of both routes.

Step 2: Then, a random selection from the candidate list is made and the move is carried out by inserting the two arcs in their receptively counter route.

Move 3 exchanges two arcs from $Routex_1$ with one arc from $Routex_2$ with respect to the constraints of both routes by executing the same steps as in move 2.
Move 4 exchanges three arcs from $Routex_1$ with one arc from $Routex_2$ with respect to the constraints of both routes by executing the same steps as in move 2.

Move 5 exchanges three arcs from $Routex_1$ with two arcs from $Routex_2$ with respect to the constraints of both routes by executing the same steps as in move 2.

The Route Improvement

When a move is executed, a *Route Improvement* (RI) algorithm is used to generate a new route, starting and ending at the depot, including the moved arcs. The RI-algorithm uses the set of arcs obtained from the different moves and works with one set at a time. When the RI-algorithm has constructed a route, it splits the route at different points, creating multiple elements. Then, the algorithm changes the sequence of elements, by moving the elements, creating multiple routes with different arranged sequence of elements. Among these routes, the cheapest route with rearranged sequence of elements is the final result of the RI-algorithm, which then is used in the SA as the result of the move. The steps of the RI-algorithm are as follows;

Step 1: The RCA, as described in subsection 4.1, is used to create a route that starts and ends at the depot.

Step 2: The route obtained from RCA is then divided in all possible ways at two points creating three segments ($Level_1$-analysis). The sequence of these three segments is then changed and rearranged in all possible ways with respect to the orientation of the arcs. This step ends with an evaluation of the cost of each route after the rearrangement.

Step 3: The N_1 routes of the $Level_1$-analysis are stored in a $Level_1$-list. Every time a cheaper route than N_1 is found, the list is updated by including the new route.

Step 4: The top N_1 routes are then randomly selected from the $Level_1$-list, termed $Level_2$-routes, and stored as a $Level_2$-list. Each route from this list is then divided in all possible ways at three points creating four segments, $Level_2$-analysis. The sequence of these four segments are then changed and rearranged as in step 2.

Step 5: If the $Level_2$-analysis creates a cheaper route, N_2, than the routes in the $Level_1$-list, then the N_2-route is used in the $Level_1$-analysis, step 2. If the new $Level_1$-analysis of the N_2-route creates a cheaper route, then a new $Level_2$-analysis is generated, step 4.

Step 6: The final result is the cheapest route after the performance of $Level_1$- and $Level_2$-analysis.

The low cost routes for both $Routex_1$ and $Routex_2$ are thereby determined. After creating the sets of moves, the SA can be performed, consisting of the following steps and using the nomenclature from Table 4:

Variable	Definition
a	The temperature reduction factor
$Best_{Cost}$	The lowest cost
$Best_{Routes}$	The corresponding route to the lowest cost
$Curr_{Cost}$	The current cost
$Curr_{Routes}$	The set of corresponding routes to the current cost
$Max_{Iteration}$	The maximum number of iterations
N	The number of iterations at the current temperature (counter)
$N_{Iteration}$	The number of ongoing iterations (counter)
N_{max}	The maximum number of iterations at each temperature
N_{move}	The maximum number of iterations for each type of move
R	A random value between 0 and 1
RCA_{Cost}	The total cost obtained from RCA
RCA_{Routes}	The set of routes obtained from RCA
SA_{Cost}	The cost of SA
SA_{Routes}	The set of SA routes
T	The temperature
ΔTC	The value of $(Curr_{Cost} - SA_{Cost})$

Table 4 The nomenclature of SA

Step 1: Step 1: The values of the parameters; $a, N_{max}, N_{move}, Max_{Iteration}$ and T have to be set. In addition, the following parameters have to be; $N = 0$ and $N_{iteration} = 0$. The process continuous to step 2.
The cooling of SA is managed by N_{max} and a i.e. for every N_{max}, the temperature reduces by factor a. In addition, the temperature T allows the exchange of the neighbouring solutions, enabling the heuristic search space, thus the value of T has to be high.
Furthermore, in this step, a set of moves has to be generated, as described in section moove.

Step 2: The following conditions have to be set; $Best_{Cost} = RCA_{Cost}$, $Best_{Routes} = RCA_{Routes}$, $Curr_{Cost} = RCA_{Cost}$ and $Curr_{Routes} = RCA_{Routes}$. The process continuous to step 3.
Here, the route(s) and solution values obtained from the RCA are assumed and set to be the current and the best route with the related values.

Step 3: The following condition is set; $N_{iteration} = N_{iteration}+1$. The process continuous to step 4.
In this step, the overall iteration counter increases by 1.

Step 4: The following condition is set; $N = N + 1$. The process continuous to step 5.
In this step, the annealing iteration counter increases by 1.

Step 5: The following conditions are set; SA_{Cost} = the cost of the performed move and SA_{Routes} = the route obtained after performing the move. The process continuous to step 6.
In this step, the cost and set of route(s) of the move are set as the SA values.

Step 6: $\Delta TC = Curr_{Cost} - SA_{Cost}$ is calculated and depending on the result, the process continuous to either step 7 or step 8.

In order to assess the solution obtained after a move, the difference between the current cost and the SA cost has to be calculated.

Step 7: The following conditions are set if $\Delta TC >= 0$: $Curr_{Cost} = SA_{Cost}$ & $Curr_{Routes} = SA_{Routes}$ and the following conditions are set if $SA_{Cost} < Best_{Cost}$, $Best_{Cost} = SA_{Cost}$ and $Best_{Routes} = SA_{Routes}$. The process continuous to step 9.
Here, based on the evaluation of the difference between the current cost and the SA cost a number of conditions are set. In case of the difference being greater or equal then 0, the solution of SA is better the current solution. At the same time, the values of SA are set as the best values, if the cost of SA is less than the cost of the best solution.

Step 8: The following conditions are set in case of $\Delta TC < 0$ and $\exp(\Delta TC/T) > R$; $Curr_{Cost} = SA_{Cost}$ and $Curr_{Routes} = SA_{Routes}$. The process continuous to step 9.
In contrast to step 7, this step deals with current solutions being worse than the SA solutions. If so, the non-improving SA solutions have to, based on the probability function, either be accepted or rejected. As $\Delta TC \mapsto 0$, the chance of the non-improving SA solutions being accepted increases. At the same time, as $T \mapsto 0$ the probability of acceptance decreases.

Step 9: The following conditions are set if $N = N_{max}$; $T = a * T$ and $N = 0$. The process continuous to step 10.
If the maximum number of iterations at the current temperature has been reached, the temperature is reduced by a and the N counter is set to 0

Step 10: If $N_{Iteration} = Max_{Iteration}$, the algorithm process ends. Otherwise, the process continuous to step 11.
If the total number of ongoing iterations has reached the maximum number of iterations (the algorithm's stop condition), then the best overall solution of the iterations is the final solution.

Step 11: If $N_{Iteration} MOD(5 * N_{move}) = 0$, then a new set of moves has to be generated and the process returns to step 3. Otherwise, the process continuous to step 12.
If the number of ongoing iterations has reached five times the maximum iterations for each type of move, then all solutions for that set of moves have been searched. Since the $N_{iteration}$ has not reached the number of maximum iteration of the algorithm, then a new set of moves can be assessed.

Step 12: If $N_{Iteration} MOD N_{move} = 0$, then the type of move, in the same set of moves, has to be updated and the process returns to step 3. If that is not the case, the process still returns to step 3.
Here, it is determined whether there has to be a change in the type of move or if the algorithm has to run the next iteration of the same type of move.

The best overall solution becomes the final solution when one of the stop conditions has been reached.

5 Implementation and Results

In order to implement and test the approach outlined in the previous sections, three of VD's winter maintenance routes have been modeled based on a traffic direction-based schema. This model is an abstraction of vehicle driving directions rather than physical objects (Ruas and Gold, 2008, p. 622). Figure 2, shows an example in which the northern carriageway is represented as a single line due to same direction lanes and the southbound carriageway is represented by five modeling objects based on their traffic directions; off-ramp, on-ramp, right-turn, left-turn and ahead flow.

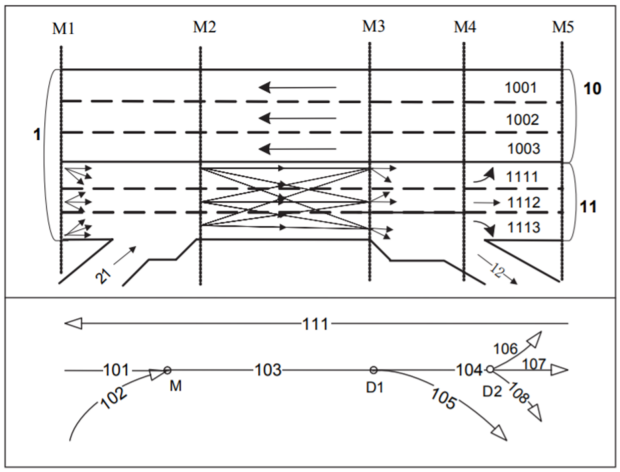

Fig. 2 An example of a traffic direction-based model (Ruas and Gold, 2008, p. 623).

The data for the three routes (and their vicinity; see Figure 3 for an example) were obtained from VD's road management system[2] and, after extensive cleaning and manual fixing of disconnected arcs, translated to the traffic direction-based schema. RCA and SA have then been implemented in Python to work on this model.

[2] See http://vejman.dk and https://trafikkort.vejdirektoratet.dk

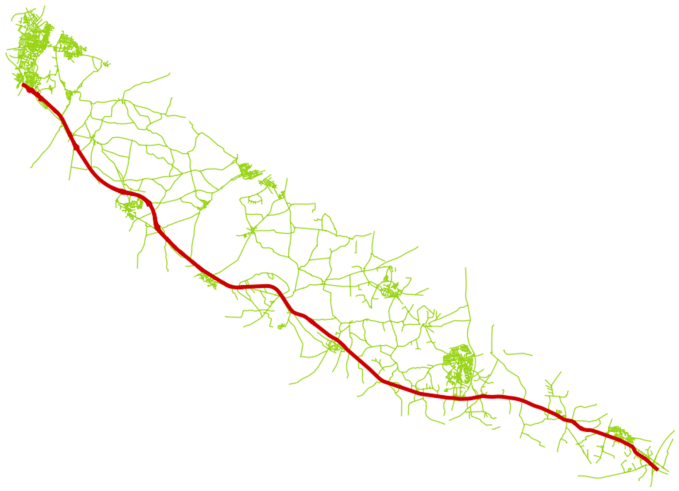

Fig. 3 A maintenance route with neighbourhood, obtained by clipping a buffered convex hull.

The three test routes used for evaluation of the approach are shown in Figures 1 and 4, respectively. After some experiments, the optimization procedure was run with an initial temperature T of 100, a temperature reduction factor α of $0,995$, a maximum number of iterations for each type of move N_{move} of 1000, a maximum number of iterations at each temperature of 300, and an overall maximum number of iterations of 125.000. The SA progress with these parameters for each route is shown if Figure 5. In addition to a fixed salting width of 8m and the arcs' respective speed limits, the evaluation is based on a maximum speed of 80 km/h when deadheading, 30 km/h when salt spreading the ramps and 70 km/h when salting elsewhere.

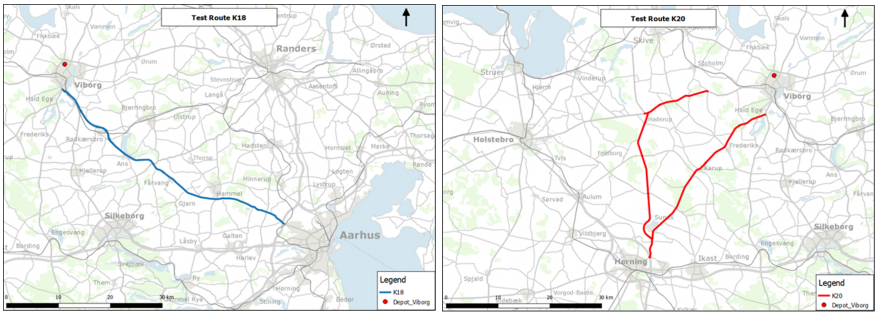

Fig. 4 Winter maintenance routes K18 and K20.

The optimization produces significantly shorter routes for all three routes, despite the fact that the maintenance route itself was fixed and optimization was only possible in the deadheading and in the order of the segments of the maintenance route. For test route K18, the route length was reduced by $6,3\%$ and service time by $7,1\%$.

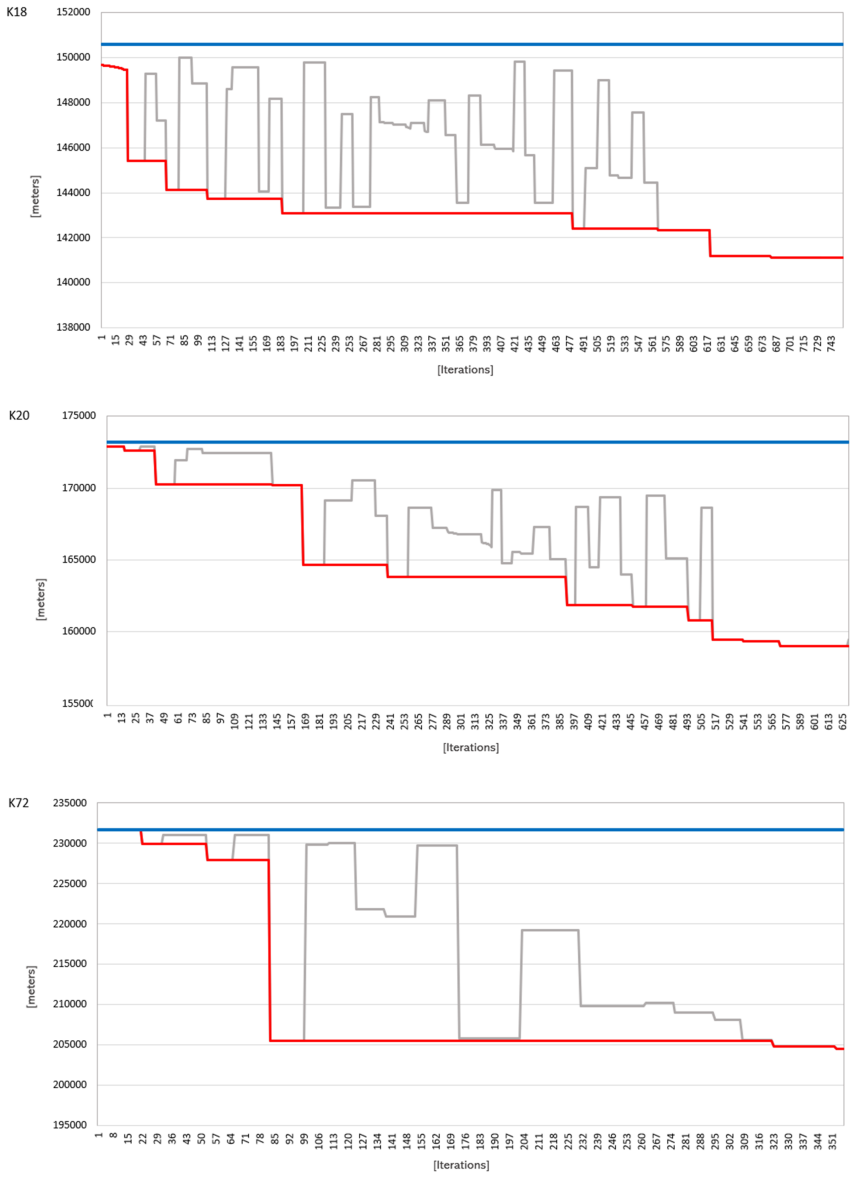

Fig. 5 SA progress for the three test routes, with the blue line indicating the route length before optimization, the gray line indicating the route length at the current iteration, and the red line indicating the minimum route length found up to the current iteration.

For K20, route length was reduced by 8, 2% and service time by 10, 7%, and for K72, route length was reduced by 11, 7% and service time by 10, 6%. Assuming the

current average salting cost per hour and that similar improvements can be expected for the remaining roads VD has to maintain during the winter, we expect savings in the order of 2,8 million DKK (about 380.000 Euros). This estimate does not include savings due to the automation of route planning.

Concerning the route optimization procedure itself, there is still room for improvement in computation time. The result shown here were obtained on a standard laptop and took 7h 28min (K18), 12h 41min (K20), and 71h 3min (K72). While these are clearly substantial computation times, they are not unusual for such heuristic procedures on complex problems. The computation time of almost three full days for route K72 is a prime example, as this is the only route that requires two vehicles to be serviced, resulting in a substantially increased processing time. However, the focus of this research has been on demonstrating that substantial improvements of the routes themselves are possible using this approach, without a big focus on computation times, since these procedures are only run once in the rare event that the routes are changed. Nonetheless, we believe that further testing and tuning of the parameters would allow us to improve computation times.

6 Conclusions

This paper has shown how the Danish Road Directorate's salt spreading can be optimized by treating it as a Capacitated Arc Routing problem (CARP). Based on CARPs categorisation as NP-hard, heuristic procedures were investigated and a heuristic algorithm based on a Greedy Randomized Adaptive Search Procedure (GRASP) was developed. In the proposed procedure, each iteration constructs a solution which is further optimised by performing a local search with a Simulated Annealing (SA) based heuristic algorithm. This is done by constructing an initial solution with a Route Construction Algorithm (RCA), that incrementally traces a route with the depot as the start and end node, in order to service the required arcs with respect to a number of constraints and criteria at each iteration. The final obtained solution of RCA is used to perform a Simulated Annealing (SA) based local search, which improves the RCA-obtained solution by exchanging arcs in terms of five different moves that define the solution space. When a move is performed, a Route Improvement (RI) algorithm is used to rearrange and reconstruct a new solution that, if better than the initial solution, is used as the new initial solution for the next iteration. The final result becomes the best overall solution that is obtained when the GRASP has reached one of the stop conditions. The proposed algorithms were implemented and tested on three different service routes, showing that they were able to improve the existing routing of all three test routes significantly. Future work on this approach should focus on the reduction of the processing time, as well as the inclusion of further parameters in the optimization, such as variable salting widths or the inclusion of typical morning traffic conditions, when most salting activities take place.

References

Arakaki RK, Usberti FL (2018) Hybrid genetic algorithm for the open capacitated arc routing problem. Computers Operations Research 90:221 – 231, DOI https://doi.org/10.1016/j.cor.2017.09.020, URL http://www.sciencedirect.com/science/article/pii/S0305054817302502

Bodin LD, Golden BL, Assad AA, Ball MO (1981) The State of the Art in the Routing and Scheduling of Vehicles and Crews. U.S. Department of Transportation, Urban Mass Transportation Administrationh, https://babel.hathitrust.org/cgi/pt?id=ien.35556021333117;view=1up;seq=13

Chapleau L, Ferland JA, Lapalme G, Rousseau JM (1984) A parallel insert method for the capacitated arc routing problem. Operations Research Letters 3(2):95–99, used: 04-12-2017

Christofides N (1973) The optimum traversal of a graph. Omega 1(6):719–732, used: 03-12-2017

Cormen TH, Leiserson CE, Rivest RL (1990) Introduction to Algorithms, 1st edn. MIT Press and McGraw-Hill

Eglese R (1994) Routeing winter gritting vehicles. Discrete Applied Mathematics 48(3):231–244, used: 08-12-2017

Eiselt HA, Gendreau M, Laporte G (1995) Arc routing problems, part ii: The rural postman problem. Operations Research 43(3):399–414, used: 15-12-2017

Evans J, Weant M (1990) Strategic planning for snow and ice control vehicles using computer-based routing software. Public Works 121(4):60–64, used: 09-12-2017

Gendreau M, Hertz A, Laporte G (1992) New insertion and postoptimization procedures for the traveling salesman problem. Oper Res 40(6):1086–1094, used: 13-11-2017

Golden BL, Wong RT (1981) Capacitated arc routing problems. Networks 11(3):305–315, used: 15-12-2017

Goode L, Nantung T (1995) CASPER: The Friendly, Efficient Snow Routes Planner. http://onlinepubs.trb.org/onlinepubs/trnews/rpo/rpo.trn181.pdf

Haghani A, Qiao H (2001) Decision support system for snow emergency vehicle routing: Algorithms and application. Transportation Research Record: Journal of the Transportation Research Board 1771:172–178, used: 02-02-2017

Haslam E, Wright JR (1991) Application of routing technologies to rural snow and ice control. Transportation Research Board 1304:202–211, used: 10-09-2017

Kirkpatrick S, C D Gelatt J, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680, used: 29-11-2017

Knudsen F, Eram MM, Jansen KB (2014) Denmark, Technical Committee 2.4, pp 52–60

Kurniawan R, Sulistiyo MD, Wulandari GS (2015) Genetic algorithm for capacitated vehicle routing problem with considering traffic density. In: 2015 International Conference on Information Technology Systems and Innovation (ICITSI), pp 1–6

Lemieux PF, Gampagna L (1984) The snow ploughing poblem solved by a graph theory algorithm. Civil Engineering Systems 1(6):337–341, used: 29-11-2017

Male JW, Liebman JC, Orloff CS (1977) An improvement of orloff's general routing problem. Networks 7(1):89–92, used: 11-12-2017

Metropolis N, Rodenbluth AW, Rosenbluth MN, Teller AH (1953) Equation of state calculations by fast computing machines. The Journal of Chemical Physics 21(6):1087–1092, used: 02-02-2018

Nemhauser GL, Wolsey LA (1988) Integer programming and combinatorial optimization. Wiley, Chichester GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992) Constraint Classification for Mixed Integer Programming Formulations COAL Bulletin 20:8–12

Pearn WL (1989) Approximate solutions for the capacitated arc routing problem. Computers & Operations Research 16(6):589 – 600, used: 03-12-2017

Rere LR, Fanany MI, Arymurthy AM (2015) Simulated annealing algorithm for deep learning. Procedia Computer Science 72(1):137–144, used: 23-11-2017

Resende MGC, Ribeiro CC (1997) A grasp for graph planarization. Networks 29(3):173–189, used: 16-10-2017

Ruas A, Gold C (2008) Headway in Spatial Data Handling: 13th International Symposium on Spatial Data Handling. Springer