Synchronized arc routing for snow plowing operations

Article *in* Computers & Operations Research · July 2012

DOI: 10.1016/j.cor.2011.08.014 · Source: DBLP

Angélica Salazar-Aguilar
Autonomous University of Nuevo León

André Langevin
Polytechnique Montréal

Synchronized Arc Routing for Snow Plowing Operations

Angélica Salazar-Aguilar
André Langevin
Gilbert Laporte

May 2011

CIRRELT-2011-29

Synchronized Arc Routing for Snow Plowing Operations

Angélica Salazar-Aguilar[1,2,*], André Langevin[1,3], Gilbert Laporte[1,2]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Department of Management Sciences, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

[3] Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal, Canada H3C 3A7

Abstract. This paper introduces a synchronized arc routing problem for snow plowing operations. In this problem, routes must be designed in such a way that street segments with two or more lanes in the same direction are plowed simultaneously by different synchronized vehicles. A mixed integer formulation and an adaptive large neighborhood search heuristic are proposed. The performance of the proposed algorithm is evaluated over a large instance set, including artificial and real data. Computational results confirm the efficiency of the algorithm.

Keywords. Arc routing, snow plowing, road maintenance, synchronized routes, adaptive large neighborhood search.

Acknowledgements. This work was supported by CONACYT under the program of postdoctoral training abroad 2009-2010, and by the Natural Sciences and Engineering Council of Canada (NSERC) under grants RGPIN41978 and 39682-10. This support is gratefully acknowledged.

* Corresponding author: Angelica.Salazar@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2011

1. Introduction

Cities with severe winters face each year the difficult task of clearing snow and ice from their streets. Problems arising in winter road maintenance are complex, costly, and site-specific because of the variation of climatic conditions, demographics, economics, and technology. According to Perrier et al. (2006a,b, 2007a,b, 2011), the importance of winter road maintenance is due to the magnitude of the expenditures associated to these operations, and to the indirect costs resulting from the loss of productivity and decreased mobility. In the United States alone these operations consume over $2 billion yearly in direct costs. In Japan and Europe snow removal expeditures are two to three times those of the United States (Perrier et al. (2006a)). Moreover, driving conditions can deteriorate and vary dramatically due to snowfalls and ice formation, which causes a significant reduction in pavement friction, increases the risk of accidents, and generates additional costs to motorists and businesses (Usman et al. (2010)).

The estimation of the cost and benefits of winter road maintenance has been studied by several researchers (Norrman et al. (2005); Venlinen and Kangas (2003); Usman et al. (2010)). Snow removal costs can be very high. For example, in Montreal the average cost of a 20 cm snow storm in 2010 was $17 million Canadian dollars (see Ville de Montréal (2010)). Each year, the city has to clear 6,550 km of sidewalks and 4,100 km of streets. On average, there are 65 weather events calling for response every winter. Snow clearing is performed in four stages: salting, plowing, removal, and disposal. Plowing operations begin as soon as there is an accumulation of 2.5 cm of snow on the ground and continue as long as the storm lasts, ending about eight hours after the snow stops falling.

Winter road maintenance operations such as plowing and spreading de-icers play an indispensable role in maintaining good road surface conditions and keeping roads safe (Norrman et al. (2005); Usman et al. (2010)). In these operations, the efficient application of mathematical optimization techniques can result in substantial savings, improved mobility, and reduced societal impacts.

There exists a relatively limited scientific literature on the practical aspects of snow removal operations (Perrier et al. (2006a)). Here we concentrate on the most important publications of the past ten years. Golbaharan (2001) has studied a multi-depot snow removal routing problem with time windows. This problem consists of designing a set of least cost routes for homogeneous snow plows, while covering every required road segment exactly once within its associated time window. Every snow plow starts its route at a depot and returns to the same depot. The problem was formulated as a constrained set covering problem and was solved by a column generation algorithm. The performance of the proposed solution procedure was evaluated on real-life data from the district of Eskilstuna, Sweden, involving seven depots, 21 snow plows, 362 nodes, and 814 edges, of which 707 were required. Similar problems have been studied by Sochor and Yu (2004), and Razmara (2004).

Perrier et al. (2008) have proposed a formulation and two solution approaches based on mathematical optimization techniques for the routing of snow plowing vehicles in urban areas. Given a district and a single depot at which a number of plows are based, the problem is to determine a set of routes, each performed by a single vehicle starting and ending at the district depot, such that all road segments are serviced, while satisfying a set of operational constraints and minimizing a completion time objective. The model contains general precedence relation constraints with no assumption on class connectivity, different service and deadhead speed possibilities, separate pass requirements for multilane road segments, class upgrading possibilities, and vehicle road segment dependencies. The authors have proposed a model based on a multicommodity network flow structure to impose the connectivity of the route performed by each vehicle, as well as constraints to model a hierarchical objective. The problem is solved by means of two constructive algorithms. The first constructs several routes in parallel by sequentially solving a multiple vehicle rural postman problem with side constraints. The second is a cluster-first, route-second algorithm, which first determines a partition of the arcs into clusters, each having approximately the same work load. A hierarchical rural postman problem with class upgrading possibilities, vehicle road segment dependencies, and turn restrictions is then solved on each cluster.

Fu et al. (2009) have developed a real-time optimization model to evaluate alternative resources allocation plans for winter road maintenance operations. Given a fleet of plowing vehicles and a set of predefined maintenance routes, the problem consists in developing an operations plan for the available service vehicles that specify for each of them a route assignment, a service type, and a start time. The scheduling model takes into account both operating costs and quality service requirements, as well as road network topology, road classes, and weather forecasts.

The paper by Dali (2009) proposes a sequential constructive heuristic for designing snow plow routes in a multi-depot network. The minimization of the total deadhead distance is carried out under some side constraints such as service continuity, both-sides service, vehicle capacity, and maximum time for service completion. The problem is modeled as a capacitated arc routing problem.

A project focused on an optimal workforce planning and shift scheduling for snow and ice removal in St. Louis County, Minnesota, was undertaken by Gupta et al. (2010). In the first part of the project, the authors have developed a model to calculate the best way to group road segments belonging to each plow route into passes such that high priority road segments are in the same group, and each group can be processed by a single pass of the plow, and can be plowed and sanded with a single vehicle. The problem was solved by means of a set partitioning formulation to select the arcs that a snowplow should traverse in each pass until all arcs have been serviced. The number of arcs that a snowplow can traverse is constrained by two factors: 1) the total time the snowplow can be away from the depot and 2) the maximal amount of sand or salt that a snowplow can carry. If a pass contains subtours, a heuristic is used to eliminate them by treating each one as a node and solving the underlying Traveling Salesman Problem (TSP).

Finally, Jang et al. (2010) have proposed a formulation and a heuristic for a combined depot location, sector design, spreading and plowing route design, fleet configuration and vehicle scheduling problem. Their heuristic integrates depot and sector selection, initial route construction, route improvement, fleet configuration, and scheduling. It iteratively solves these problems until no better solution can be found. The model, which assumes a heterogeneous fleet, takes into account road class service frequency and spreader capacity constraints. Service routes must be established exclusively for each class, but a vehicle can service multiple routes in different classes with different frequencies. Multiple lane roadways are represented with one arc for each traffic lane. The objective considered is to minimize the number of working vehicles. The authors have applied their algorithm to a real transportation network in Boone County, Missouri. This network contains 138 vertices and 452 arcs. Computational results have shown the good performance of their solution procedure: the number of snow plows could be reduced by 21.8% with respect to the current operations.

An important practical consideration absent from the literature on the planning of snow plowing operations is the need to synchronize the vehicles required at the same time on the same street segment. For example, when clearing a two-lane or three-lane street, several snow plows must follow one another to push the snow by the side of the street. Synchronization is important to avoid building snow mounds in the middle of the street (see Figure 1) In this paper we formally introduce, model, and solve the *Synchronized Arc Routing Problem* (SyARP) for snow plowing operations. The SyARP consists of determining a set of routes such that all street segments are serviced within the least possible time, subject to a synchronization constraint. Each segment is cleaned by snow plow vehicles that start and end their journey at the depot. The street segments have different numbers of lanes in one or two directions, and all lanes in the same direction should be plowed by the required vehicles at the same time.

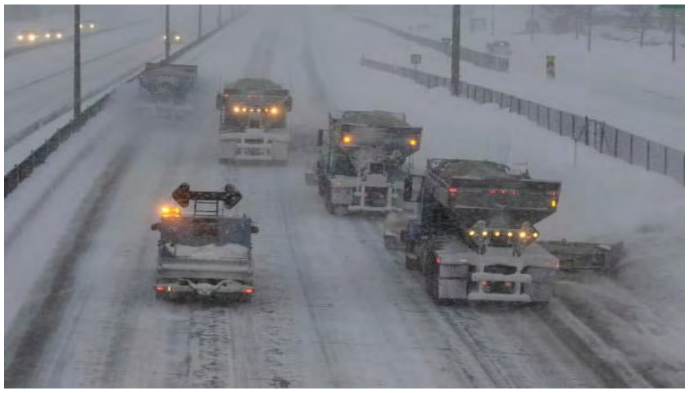


Figure 1: Highway 720 during a late day snow storm in Montreal. The Montreal Gazette, March 7, 2011 6:43 PM.

To the best of our knowledge, the SyARP has never been addressed in the arc routing literature, apart from a conference presentation on which this paper expands (Salazar-Aguilar et al. (2011)). Here, we introduce a mixed integer programming formulation for this problem and we develop a heuristic solution technique based on the adaptive large neighborhood search (ALNS) metaheuristic, which is one of the most successful metaheuristics for solving complex routing problems (Ropke and Pisinger (2006)). ALNS was recently applied by Laporte et al. (2010) to a capacitated arc routing problem with stochastic demands.

The remainder of this paper is organized as follows. Section 2 is devoted to the detailed description, formulation, and complexity of the problem. The proposed solution procedure is described in Section 3. Experimental work is presented in Section 4, followed by conclusions in Section 5 .

2. Problem description, formulation and complexity

Given a network of streets and a fleet of snow plowing vehicles based at a depot, the SyARP consists of determining a set of routes such that all streets, some of which have multiple lanes, are plowed by using fleets of synchronized vehicles in order to minimize the duration of the longest route, called the *makespan*. The street segments have one or two directions, and each direction has a number of lanes, typically between one and three. All lanes belonging to the same segment in the same direction must be plowed simultaneously, and the number of vehicles servicing a given segment is equal to its number of lanes. Deadheading is allowed, which means that any vehicle can traverse a street segment one or several times without servicing it.

2.1. Example

In the city map depicted in Figure 2(a), the arrows show the street directions, and the numbers on the arrows represent the number of lanes. In this example, all streets have one, two or three lanes.

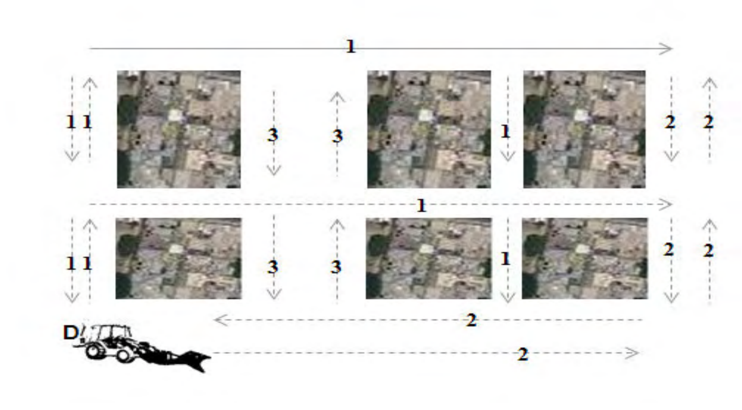
Figure 2(b) depicts a set of three synchronized vehicle routes. Let v_1, v_2, and v_3 the vehicle assigned to the routes R_1, R_2, and R_3, respectively. Vehicle v_1 leaves the depot at 9:00 h and the first two arcs on R_1, shown in gray, are not serviced by it. Service only starts with the third arc of R_1. Vehicle v_1 services the third and fourth arcs, after which it returns to the depot to continue its route, moving north, east and south, and servicing all one-lane segments on its trajectory. At 10:15 h vehicle v_2 meets v_1 in order to plow the two-lane streets (from the bottom right corner), until 10:31 h, at which time v_3 joins v_1 and v_2 to plow the three-lane streets. The three vehicles remain synchronized until all three-lane streets have been plowed. The service performed by these three vehicles is completed at 11:40 h.

2.2. Mathematical model

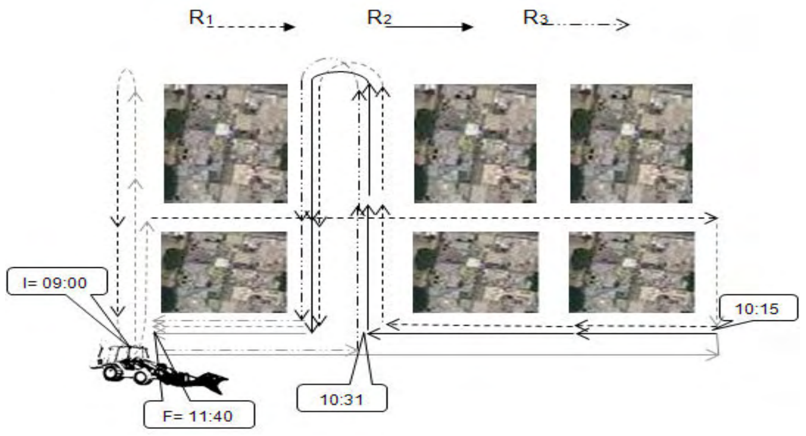
We now provide a mathematical integer programming formulation of the SyARP. The interest of the model is to provide a non-ambiguous statement of the problem. This formulation is non-linear and could be linearized through standard techniques, but we have chosen not to do so because the resulting model would then become very large and still untractable, even for small size instances.

Let $G = (V, A)$ be a directed multi-graph where $V = \{0, ..., n\}$ is the vertex set and $A = \{(i, j) : i, j \in V \text{ and } i \neq j\}$ is the arc set. Vertex 0 is the depot. An additional vertex $0'$ represents an artificial depot, used as the end point of all routes. This vertex is necessary because any route may pass through the depot several times. Artificial arcs $(0', 0)$ and $(0, 0')$ are used to start and end the routes. The set of available vehicles is denoted by R. The maximal number of arcs included in any route is given by e and we define K as $\{1, ..., e\}$. Let n_{ij} be the number of lanes on arc (i, j) and let λ be the maximal number of lanes on any arc $(i, j) \in A$. Each arc $(i, j) \in A$ has two associated times called t_{ij} and t'_{ij} for service and traversal, respectively. We define the following decision variables:

$$x^k_{ijr} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is traversed by vehicle } r \text{ and appears} \\ & \text{in the } k^{th} \text{ position of the route while deadheading} \\ 0 & \text{otherwise;} \end{cases}$$



(a) City in which streets have one, two, and three lanes in one or two directions.



(b) Synchronized routes for three vehicles, servicing some of the streets of the city depicted in Figure 2(a).

Figure 2: Example of synchronized snow plowing operations.

$$y_{ijr}^k = \begin{cases} 1 & \text{if arc } (i,j) \text{ is serviced by vehicle } r \text{ and appears} \\ & \text{in the } k^{th} \text{ position of the route} \\ 0 & \text{otherwise.} \end{cases}$$

Note that variables $y_{0'0r}^1$ and $y_{00'r}^k$ are equal to zero given that the artificial arcs do not require service. In addition,

t_{ijr}^k	is the starting time of service or traversal of arc (i,j) by vehicle r and this arc appears in the k^{th} position of the route;
w_{ijr}^k	is the waiting time of vehicle r after service or traversal of arc (i,j) when it appears in the k^{th} position of the route.

The objective is to minimize the makespan:

$$\text{Minimize} \quad z = \max_{r\in R, k\in K} \left\{ t_{00'r}^k \right\} \tag{1}$$

subject to

$$x_{0'0r}^1 + y_{0'0r}^1 = 1 \qquad r \in R \tag{2}$$

$$\sum_{(0,j)\in A} x_{0jr}^2 + y_{0jr}^2 = 1 \qquad r \in R \tag{3}$$

$$\sum_{k\in K} x_{00'r}^k + y_{00'r}^k = 1 \qquad r \in R \tag{4}$$

$$\sum_{k\in K}\sum_{r\in R} y_{ijr}^k = n_{ij} \qquad (i,j) \in A \tag{5}$$

$$(t_{ijr}^k + t_{ij})y_{ijr}^k + (t_{ijr}^k + t'_{ij})x_{ijr}^k + w_{ijr}^k = \sum_{(j,h)\in A\cup(0,0')} t_{jhr}^{k+1}(x_{jhr}^{k+1} + y_{jhr}^{k+1}) \qquad r \in R, (i,j) \in A, k \in K \setminus \{1,e\} \tag{6}$$

$$\sum_{(i,j)\in A\cup\{(0',0),(0,0')\}} x_{ijr}^k + y_{ijr}^k \leq 1 \qquad r \in R, k \in K \tag{7}$$

$$y_{ijr}^k \left(t_{ijr}^k - \sum_{q\in R}\sum_{c\in K\setminus\{1\}} t_{ijq}^c y_{ijq}^c \right) / n_{ij} = 0 \qquad (i,j) \in A, r \in R, k \in K \setminus \{1\} \tag{8}$$

$$\sum_{(i,j)\in A} (x_{ijr}^k + y_{ijr}^k) \leq \sum_{(j,h)\in A\cup(0,0')} (x_{jhr}^{k+1} + y_{jhr}^{k+1}) \qquad r \in R, j \in V, k \in K \setminus \{1,e\} \tag{9}$$

$$x_{ijr}^k \in \{0,1\} \qquad k \in K, r \in R, (i,j) \in A \tag{10}$$

$$y_{ijr}^k \in \{0,1\} \qquad k \in K, r \in R, (i,j) \in A \tag{11}$$

$$t_{ijr}^k \geq 0 \qquad k \in K, r \in R, (i,j) \in A \tag{12}$$

$$w_{ijr}^k \geq 0 \qquad k \in K, r \in R, (i,j) \in A. \tag{13}$$

Objective (1) is the makespan. Constraints (2)−(4) guarantee that all routes start and end at the depot. Constraints (5) state that all lanes of arc (i,j) must be serviced. Constraints (6) ensure time consistency of service and traversal times. Constraints (7) state that at most one arc appears in each position of a route. By constraints (8), each vehicle servicing an arc (i,j) starts at the same time as all vehicles that service the same arc. Constraints (9) ensure that each vertex is connected to some vertex. Note that when a vertex j is reached in the k^{th} position of route r, the outgoing arc (j,h) should be in the $(k+1)^{st}$ position of the same route. The remaining constraints impose conditions on the variables.

2.3. Complexity of the SyARP

To show that the SyARP is NP-hard, we consider an instance in which $\lambda = 1$, all arcs are incident to the depot, i.e, the graph is a star, and the servicing and traversing times on any arc are such that $t_{0i} = t'_{0i}$ and $t_{i0} = t'_{i0}$, $i \in V$. The recognition version of this instance consists of determining whether there exists a solution of cost at most equal to $\bar{z}$. It is equivalent to the recognition version of a *Bin Packing Problem* with $|R|$ bins of capacity $\bar{z}$ and items of sizes $(t_{0i} + t_{i0}), i \in V$, which is NP-complete (see, for example, Martello and Toth (1990)).

3. Adaptive large neigborhood heuristic

We have developed a local search heuristic based on the ALNS metaheuristic. Our heuristic constructs an initial set of feasible routes, and then attempts to improve them by mean of different destroy/repair operators. The ALNS metaheuristic was proposed in Ropke and Pisinger (2006) and extends the local neighborhood search heuristic of Shaw (1998) by allowing multiple *destroy/repair* operators to be used within the same search. The choice of operators is controlled dynamically according to their past performance. Each operator has a weight which controls how often it is used during the search. These weights are adjusted as the search progresses so that the heuristic adapts to the instance at hand and to the state of the search.

3.1. Construction phase

In the SyARP, one of the main difficulties is the synchronization of the routes. To construct an initial solution with synchronized routes we proceed as follows. Let $\{C_1, ..., C_\lambda\}$ be a partition of A where C_l is the set of arcs with l lanes, $l = 1, ..., \lambda$. Let μ_l be the minimum number of vehicles required to service the arcs of C_l: it is equal to l if $C_l \neq \emptyset$ and to 0 otherwise. We assume that $|R| \geq \mu_\lambda$ for otherwise no feasible solution exists. Also, since the objective is the minimization of the makespan, it is always optimal to service the different arc classes simultaneously provided the available number of vehicles allows it, i.e. wherever $|R| \geq \sum_{l=1}^{\lambda} \mu_l$. If this condition is not satisfied, we determine the largest α such that $\sum_{l=\lambda-\alpha}^{\lambda} \mu_l \leq |R|$ and we merge the classes $C_1, ..., C_{\lambda-\alpha}$ into a single class $C_{\lambda-\alpha}$. Then all arc classes can be serviced simultaneously. Otherwise, we define α as $\lambda - 1$.

The number v_l of vehicles assigned to C_l must be a multiple f_l of μ_l, where f_l is the number of fleets assigned to C_l, and $\sum_{l=\lambda-\alpha}^{\lambda} v_l = |R|$. Because there are usually several ways to satisfy this equation, there also exists several possible fleet distributions $(f_1, ..., f_\lambda)$. For each distribution, consider in turn each arc class C_l and let $D_l = \{(i_1, j_1), ..., (i_{f_l}, j_{f_l})\}$ be a set of *seed arcs* of C_l, used to initialize f_l routes. In our implementation, these arcs are as geographically dispersed as possible. To this end, the set D_l first includes the arc of C_l farthest from the depot, and this arc is then removed from C_l. The arc of C_l farthest from any arc of D_l is then added to D_l, and so on until f_l seed arcs have been selected. The algorithm then extends a route from each seed arc of D_l by keeping route lengths balanced. This is done in a greedy fashion by assigning the next arc to either end of the shortest current route. In this process, arcs yielding circuits are related with a probability β of being selected ($\beta = 0.3$ in our implementation). The process continues until all arcs of C_l have been assigned to a route, at which point the first and the last vertex of each route are connected to the depot by means of shortest paths. The construction algorithm is formally described in Algorithm 1.

To illustrate, consider an instance with 12 vehicles and arcs having one, two or three lanes, i.e., $|R| = 12, \lambda = 3, \mu_1 = 1, \mu_2 = 2, \mu_3 = 3$, and $\sum_{l=1}^{\lambda} \mu_l = 6$. Then, one can plow the various classes of arcs simultaneously, which requires three fleets and six vehicles: one fleet with one vehicle, another one with two vehicles, and the last one with three vehicles. The remaining six vehicles can be distributed in various ways among the three fleets, and all combination of fleets requiring 12 vehicles would be considered to generate multiple initial solutions. In this example, with 12 available vehicles and streets with one, two, or three lanes, there are seven possible fleet combinations: (1,1,3), (1,4,1), (2,2,2), (3,3,1), (4,1,2), (5,2,1), and (7,1,1). For example, the fifth combination (4,1,2) means that four fleets of one vehicle each are used to service the arcs with one lane, one fleet of two vehicles is used to service the arcs with two lanes, and two fleets of three vehicles each are used to service the arcs with three lanes. Our solution procedure constructs routes for each fleet of each distribution. Every solution generated by Algorithm 1 is used successively as an input to the improvement phase, which we now describe.

3.2. Improvement phase

Given a feasible initial solution P, the ALNS heuristic is applied until a stopping criterion is reached. The algorithm outputs the best solution encountered during the search. During the iterative process, each destroy/repair operator $\omega \in \Omega$ has a weight ρ_ω which is adjusted dynamically during the search. The search is divided into a number of segments of δ iterations (in our case $\delta = 100$) and the weights ρ_ω are re-initialized every δ iterations to the value they had at the first iteration of the segment. At the beginning of each segment, all operators ω have the same weight $\rho_\omega = 1/|\Omega|$. At each iteration of the same segment, these weights are adjusted dynamically, based on the past performance of the operator ω, computed as

$$\rho_\omega = \frac{S(\omega)}{U(\omega)},$$

where $U(\omega)$ is the number of times operator ω has been selected, and $S(\omega)$ is the number of times for which ω has improved the current solution in the past iterations of the segment.

At each iteration, the selection of an operator ω is achieved by following a *roulette wheel mechanism* by which operator ω has a probability

$$\tau_\omega = \rho_\omega / \sum_{\phi \in \Omega} \rho_\phi$$

of being selected. Once an operator has been chosen, it is applied until three consecutive non-improving solutions have been performed. As in record-to-record travel (Dueck (1993)), a non-improving solution can be randomly accepted if its value does not exceed that of the incumbent solution by more than 10%. In addition, for each operator ω, at most 50 non-improving solutions can be accepted. Let Π be the set of non-improving solutions accepted in the previous iterations and let $P' \in \Pi$ be a non-improving solution whose value does not exceed that of the incumbent solution by more than 10%. A number θ is randomly selected in $[0, 50]$ according to a discrete uniform distribution; if $|\Pi| < \theta < 50$ then P' is

Algorithm 1 Construction phase of the ALNS heuristic

```
Input:
  $G = (V, A)$: Instance graph
  $|R|$: Number of vehicles
  $\lambda$: Maximum number of lanes
  $\beta$: arc selection probability
Output: $P$: set of routes for all fleets
  Arcs classification, $C_l = \{(a, b) \in A$ and $(a, b)$ has $l$ lanes$\}$
  Set $\mu_l$ equal to $l$ if $C_l \neq \emptyset$ and to 0 otherwise
  if $|R| < \sum_{l=1}^{\lambda} \mu_l$ then
    Compute the largest $\alpha$ satisfying $\sum_{l=\lambda-\alpha}^{\lambda} \mu_l \leq |R|$ and merge $C_1, ..., C_{\lambda-\alpha}$ into a single
    class $C_{\lambda-\alpha}$
  else
    Set $\alpha = \lambda - 1$
  end if
  Generate a fleet distribution $f = \{f_{\lambda-\alpha}, ..., f_\lambda\}$
  for $(l = \lambda - \alpha, ..., \lambda)$ do
    Set $P(l)_k = \emptyset, k = 1, ..., f_l, D_l = \emptyset$
    Identify $f_l$ seed arcs: $D_l = \{(i_1, j_1), ..., (i_{f_l}, j_{f_l})\}, D_l \subset C_l$
    Initialization of routes, $P(l)_k \leftarrow (i_{f_l}, j_{f_l}), k = 1, ..., f_l$
    Set $C_l \leftarrow C_l \setminus \{(i_1, j_1), ..., (i_{f_l}, j_{f_l})\}$
    while $C_l \neq \emptyset$ do
      Set $\gamma = \arg\min_{k=1,..,f_l}\{$path length $(P(l)_k)\}$
      Let $(i, j)$ be the first arc of $P(l)_\gamma$ and $(g, m)$ be the last arc of $P(l)_\gamma$
      Let $N_\gamma$ be the union of the incoming arcs to $(i, j)$ and outgoing arcs from $(g, m)$.
      if $N_\gamma \neq \emptyset$ then
        Choose $(a, b) \in N_\gamma$, if $(b, a) \in P(l)_\gamma$, then $(a, b)$ has a probability $\beta$ of being
        selected
      else
        Choose $(a, b) \in C_l$ such that the path length from $(a, b)$ to $P(l)_\gamma$ is minimized
      end if
      $P(l)_\gamma \leftarrow P(l)_\gamma \cup \{(a, b)\}$
      $C_l \leftarrow C_l \setminus \{(a, b)\}$
    end while
  end for
  return $P$
```

Algorithm 2 Improvement phase of the ALNS heuristic

```
Input:
  $P$: Initial feasible solution
  $\Omega$: Set of destroy/repair operators
Output:  $P_b$: Incumbent solution
  Initialization
  $P_b \leftarrow P$, $\Pi = \emptyset$, $U(\omega) = 0$, $S(\omega) = 0$, $\rho_\omega = 1/|\Omega|$, for all $\omega \in \Omega$
  while Stop criterion is not met do
    Choose a destroy/repair operator $\omega \in \Omega$ using the roulette wheel selection mechanism
    based on current weights $\rho_\omega$
    while Three consecutive non-improvement solutions are not found do
      Set $U(\omega) \leftarrow U(\omega) + 1$
      Generate $P_n$ from $P$ by applying operator $\omega$.
      if $z(P_n) < z(P)$ then
        Set $P \leftarrow P_n$, $S(\omega) \leftarrow S(\omega) + 1$
      else
        if $z(P_n) \leq 1.10 z(P_b)$ then
          Generate a number $\theta \in [0, 50]$ according to a discrete uniform distribution
          if $|\Pi| < \theta < 50$ then
            Set $P \leftarrow P_n$, $\Pi \leftarrow \Pi \cup P_n$
          end if
        end if
      end if
      if $z(P_n) < z(P_b)$ then
        Update the incumbent solution, set $P_b \leftarrow P_n$
      end if
    end while
    if the end of the search segment is reached then
      $\rho_\omega = 1/|\Omega|$
    else
      $\rho_\omega = S(\omega)/U(\omega)$
    end if
  end while
  return $P_b$
```

accepted, otherwise P' is rejected. Algorithm 2 summarizes the general framework of the improvement phase of our ALNS heuristic.

We have developed five destroy/repair operators. For each of them, the insertion of any sequence of arcs is allowed if and only if the fleet size associated with a given arc is at least equal to its number of lanes. The destroy/repair operators are described below. Note that most of our proposed operators are based on insertions and removals of arcs, and route reconnections are all performed by means of shortest paths. Let $(..., (i, j)_k, ..., (g, h)_k, ..., (u, v)_k, ...)$ be a subsequence of arcs from a route P_k, where $(i, j)_k$, $(g, h)_k$ and $(u, v)_k$ are three serviced arcs and there are no serviced arcs between $(i, j)_k$ and $(g, h)_k$, and between $(g, h)_k$ and $(u, v)_k$, respectively. If (g, h) is removed from P_k, then, a new route $\bar{P}_k$ with the subsequence of arcs $(..., (i, j)_k, ..., (u, v)_k, ...)$ is obtained, where the subroute between j and u corresponds to a shortest path from j to u, and all arcs on this path are only traversed in $\bar{P}_k$.

N1: Best Arcs Sequence Removal-Best Insertion.

The aim of this destroy/repair operator is to reduce the makespan. A sequence of arcs is randomly chosen and removed from the longest route. It is then inserted into another route which traverses the maximum number of arcs belonging to the sequence. The insertion point is after the arc having the shortest path from the initial arc of the sequence. Finding the best insertion point requires $O(n)$ comparisons.

N2: Random Arcs Sequence Removal-Insertion.

The goal of this destroy/repair operator is to diversify the search. A sequence of arcs is removed from the longest route and randomly inserted into another route. Because the insertion point is randomly chosen, this operator requires $O(1)$ operations.

N3: Best Interchange of Arc Status (Serviced-Traversed).

This destroy/repair operator attempts to intensify the search process. An arc (i, j) serviced by the longest route P_d is randomly chosen. If another route uses this arc just for traversal, the statuses of this arc in these two routes are interchanged, which requires $O(1)$ operations. The route P_d is reoptimized by computing shortest paths in the links affected by the change of status of arc (i, j).

N4: Worst Interchange of Arc Status (Traversed-Serviced).

This destroy/repair operator attempts to diversify the search. The route with the shortest duration is chosen and an arc (i, j) which is only traversed by the route is randomly selected. Another route P_d servicing (i, j) is identified and an interchange of statuses in the arc (i, j) is carried out in both routes. This operator requires $O(1)$ operations. As in N3, P_d is reoptimized by computing shortest paths in the links affected by the change of status of arc (i, j).

N5: First Traversed-First Serviced.

The aim of this destroy/repair operator is to intensify the search process. All routes in the current solution P are reoptimized. Recall that each arc can be traversed several times by the same route but should be serviced only once. Therefore, this procedure is applied to each route $P_d \in P$ in such a way that the arcs that are both serviced and traversed by P_d are reordered in an attempt to reduce the makespan. Thus, each arc that is both serviced and traversed by P_d will be forced to be serviced the first time it is traversed in route P_d. This process requires $O(|K||R|)$ operations.

Our ALNS heuristic stops with the best known solutions after a given number of iterations. The routes originally designed for a fleet of vehicles are split into individual vehicle routes.

4. Computational results

The algorithm just described was coded in C++ and compiled on a 2592.574 MHz AMD Opteron(tm) processor 285 with 1GB of RAM under the Linux operating system.

4.1. Computational results on randomly generated instances

We have first tested our algorithm on several set of randomly generated instances. To this end, we have generated three instance sets with 42, 180, and 300 vertices on the same grid shape. The minimum number of arcs for each set was 113, 499, and 795, respectively. Fifteen instances were generated for each set, and all arcs have one, two or three lanes. The number of available vehicles for each set was 12, 18, and 25, respectively. The maximum number of iterations in the ALNS heuristic was set to 1500. The instance names provide information on the number of vertices, vehicles, and maximum number of lanes. For example, "N300V25L3-1" denotes an instance with 300 vertices, 25 vehicles, and at most three lanes. The last field corresponds to the replica number.

Because this problem has never been previously studied, no comparative data and no competing heuristic exist. We have evaluated the performance of our ALNS heuristic by using different combinations of our destroy/repair operators: N1, (N1,N2), N\(N4,N5), N\N5, and N, where N is the set of all operators. For each tested instance, we have identified the best solution of the different combinations of neighborhoods. The values of these solutions are boldfaced in Tables 1, 2, and 3. We have then computed the percent gap associated to each combination of neighborhoods with respect to the best found solution identified during the experiments. Tables 4, 5, and 6 display these results for various neighborhood choices. Observe that in general, the best solutions are reported when the ALNS applies all our proposed destroy/repair operators (last column).

Tables 4, 5, and 6 provide useful information on the contribution of each operator. For instance, the second row in Table 4 clearly shows how the average percent gap decreases when the number of destroy/repair operators increases. A similar behavior is observed in Table 5. This means that better solutions are found when the search space is diversified with different neighborhoods. Table 7 shows the time required by our solution procedure. The generator of initial solutions is very efficient and requires only 9.31 seconds for the largest instances tested. In contrast, the neighborhood N5 is the most time consuming. This was expected given that the exploration of this neighborhood requires a higher computational effort than that of the other neighborhoods.

In addition, we have analyzed the performance of the improvement phase over the construction phase in our ALNS heuristic. Table 8 shows the average percent improvement for each instance set. The average improvement ranges from 16.13% in the largest instance

Table 1: Best found solution values reported by ALNS, instance set (42,12).

Instance	**N1**	**(N1,N2)**	**N\(N4,N5)**	**N\N5**	**N**
N42V12L3-1	379.78	380.81	369.68	363.21	**357.45**
N42V12L3-2	408.29	408.29	392.80	392.80	**353.57**
N42V12L3-3	377.84	377.84	369.65	**361.07**	365.19
N42V12L3-4	473.30	473.30	473.30	473.30	**458.46**
N42V12L3-5	400.90	400.90	**377.02**	381.08	381.21
N42V12L3-6	462.15	462.15	461.00	450.72	**411.25**
N42V12L3-7	381.65	378.90	369.47	361.16	**358.74**
N42V12L3-8	405.50	405.50	379.41	383.83	**368.39**
N42V12L3-9	406.33	406.33	**379.16**	383.58	383.92
N42V12L3-10	414.49	414.49	410.16	**400.05**	414.49
N42V12L3-11	379.78	380.81	369.68	363.21	**357.45**
N42V12L3-12	506.28	506.28	506.28	502.78	**486.10**
N42V12L3-13	403.69	403.69	396.94	390.08	**377.10**
N42V12L3-14	414.09	414.09	396.81	396.81	**383.67**
N42V12L3-15	379.59	380.62	**352.91**	**352.91**	368.13

Table 2: Best found solution values reported by ALNS, instance set (180,18).

Instance	**N1**	**(N1,N2)**	**N\(N4,N5)**	**N\N5**	**N**
N180V18L3-1	753.36	713.95	606.98	606.34	**593.90**
N180V18L3-2	660.10	660.10	605.97	591.01	**570.59**
N180V18L3-3	493.95	502.65	488.60	488.60	**481.39**
N180V18L3-4	561.17	574.48	**525.70**	**525.70**	538.94
N180V18L3-5	559.91	566.48	506.44	506.44	**488.74**
N180V18L3-6	540.96	540.96	522.86	525.22	**506.35**
N180V18L3-7	611.79	611.79	594.95	592.75	**580.13**
N180V18L3-8	626.83	626.83	**613.05**	625.87	625.87
N180V18L3-9	653.55	653.55	634.77	634.77	**593.75**
N180V18L3-10	680.45	680.45	**618.25**	634.67	636.40
N180V18L3-11	818.47	818.47	**767.47**	784.63	779.30
N180V18L3-12	742.25	751.03	**684.24**	697.79	689.18
N180V18L3-13	680.73	680.73	673.88	673.09	**669.38**
N180V18L3-14	782.63	790.45	705.95	699.15	**698.28**
N180V18L3-15	661.36	661.36	633.71	**614.67**	650.06

Table 3: Best found solution values reported by ALNS, instance set (300,25).

Instance	**N1**	**(N1,N2)**	**N\(N4,N5)**	**N\N5**	**N**
N300V25L3-1	784.43	784.43	773.04	773.04	**757.99**
N300V25L3-2	688.14	663.25	655.85	656.51	**612.86**
N300V25L3-3	628.78	628.78	627.58	627.58	**627.84**
N300V25L3-4	606.63	606.63	573.61	**560.50**	590.84
N300V25L3-5	727.14	727.14	659.31	665.66	**636.86**
N300V25L3-6	734.79	734.79	663.94	669.32	**649.29**
N300V25L3-7	819.86	824.62	728.61	746.52	**721.98**
N300V25L3-8	554.75	567.06	549.41	549.20	**536.06**
N300V25L3-9	651.10	651.10	612.13	627.78	**601.26**
N300V25L3-10	651.61	651.61	**615.25**	627.99	641.52
N300V25L3-11	836.55	841.59	**731.04**	784.10	777.29
N300V25L3-12	769.25	766.82	**733.32**	743.99	735.74
N300V25L3-13	738.86	738.86	680.16	672.21	**656.34**
N300V25L3-14	663.34	673.93	599.67	593.91	**577.23**
N300V25L3-15	578.13	578.13	516.40	506.39	**493.04**

Table 4: Gap with respect to the best found solution value, instance set (42,12).

Gap(%)	**N1**	**(N1,N2)**	**N\(N4,N5)**	**N\N5**	**N**
Min	3.00	3.24	0.00	0.00	0.00
Ave	7.23	7.24	3.80	2.97	0.76
Max	15.47	15.47	12.10	11.09	4.31

Table 5: Gap with respect to the best found solution value, instance set (180,18).

Gap(%)	**N1**	**(N1,N2)**	**N\(N4,N5)**	**N\N5**	**N**
Min	1.70	1.70	0.00	0.00	0.00
Ave	9.17	9.27	2.07	2.22	1.04
Max	26.85	20.21	6.91	6.91	5.76

Table 6: Gap with respect to the best found solution value, instance set (300,25).

Gap(%)	**N1**	**(N1,N2)**	**N\(N4,N5)**	**N\N5**	**N**
Min	0.19	0.19	0.00	0.00	0.00
Ave	9.79	9.86	2.31	3.05	1.09
Max	17.26	17.26	7.01	7.26	6.33

set (N300V25L3) to 46.78% in the smallest instance set (N42V12L3). This indicates that the improvement phase in our method makes an important contribution to our ALNS heuristic.

Finally, we have investigated the effect on computation time of reducing the maximum

Table 7: Computation time for all instance sets.

ALNS Average Time (s)						
Instances	**Initial**	**N1**	**(N1,N2)**	**N\(N4,N5)**	**N\N5**	**N**
N42V12L3	0.36	8.73	12.47	8.40	8.13	13.67
N180V18L3	2.20	232.00	275.13	202.87	202.00	698.87
N300V25L3	9.31	1372.53	1443.40	1018.93	998.40	4843.00

Table 8: Average percent improvement of the improvement phase over the construction phase in our ALNS heuristic.

Instances	**% Improvement**
N42V12L3	46.78
N180V18L3	21.91
N300V25L3	16.13

number of lanes from three to two. To this end, we have modified the instance set (180,18) by setting the number of lanes equal to two on all arcs which previously had three lanes. Here we have used the full set N of operators. The results of this experiment are displayed in Table 9. The first column is the computation time obtained when the maximal number of lanes is three, and the second column corresponds to the case where the maximal number of lanes is two. The computation time decreases dramatically when the maximum number of lanes goes down from three to two.

Table 9: Computation time (seconds), instance set (180,18)

ALNS(N)		
Instance	L3	L2
N180V18-1	772	313
N180V18-2	744	270
N180V18-3	637	270
N180V18-4	643	331
N180V18-5	849	334
N180V18-6	794	258
N180V18-7	792	287
N180V18-8	627	329
N180V18-9	679	298
N180V18-10	717	387
N180V18-11	694	302
N180V18-12	643	327
N180V18-13	606	330
N180V18-14	623	388
N180V18-15	663	313

4.2. Case study

The experimental work described in Section 4.1 was carried out over randomly generated instance sets. However, real-life instances often behave differently. We now present a case of study for the city of Dieppe, a suburb of Moncton in New Brunswick, Canada, with a population of about 20,000. Dieppe currently has approximately 144 km of roads to plow, representing 363 lane-km. The municipality has experienced rapid growth in recent years, and the snow clearing plan is adjusted yearly to incorporate the new residential areas. Figure 3 depicts the current Dieppe street network. It contains 430 vertices and 1056 arcs, all of which have one or two lanes.

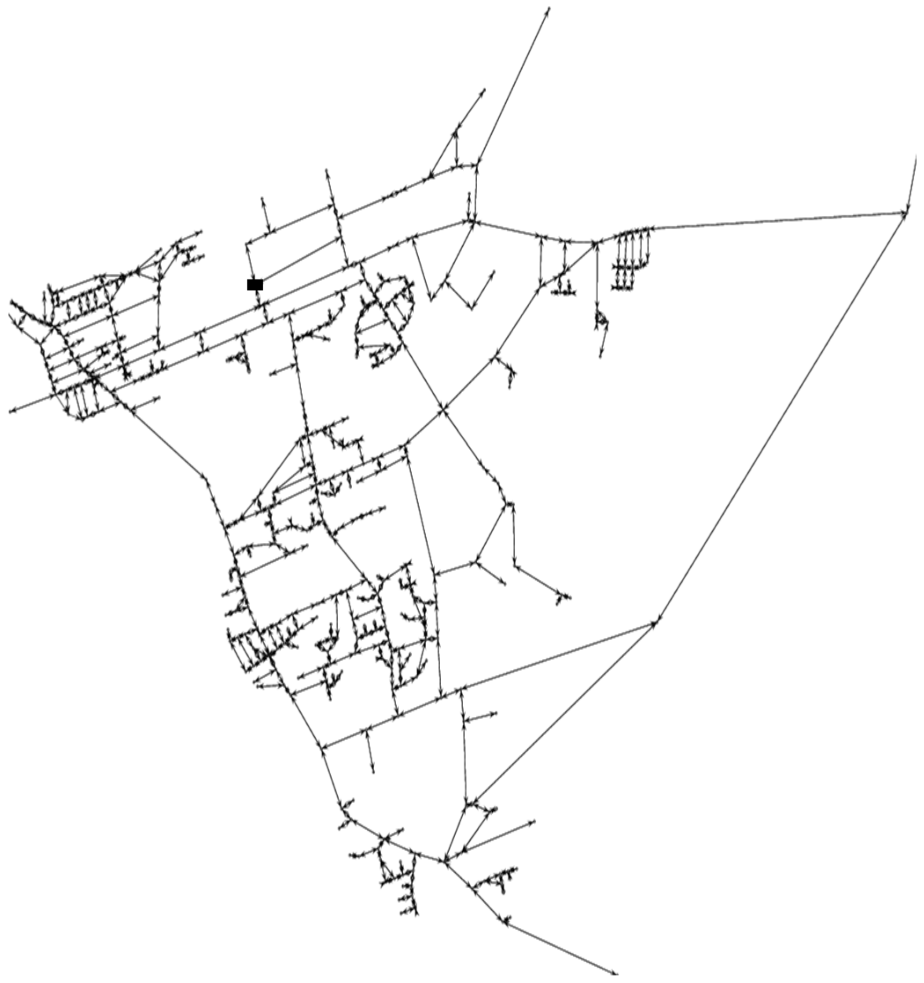
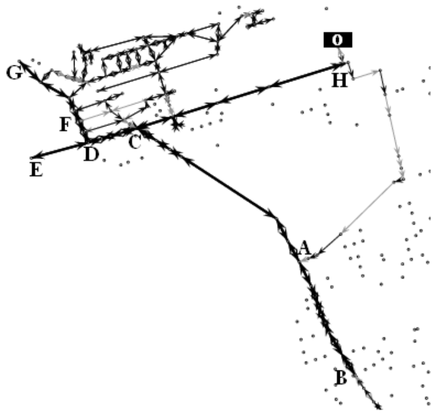
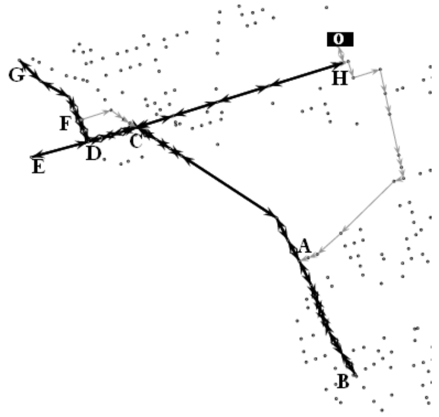


Figure 3: Street network of Dieppe.

We have analyzed the case where there are eight available vehicles traveling and servicing at a speed of 48 and 12 km/h, respectively. Our algorithm was able to generate in 538 seconds a set of routes with a makespan of 3 hours and 28 minutes. Figure 4 depicts two synchronized routes obtained by our algorithm: the gray arrows are traversed arcs and the bold arrows are serviced arcs. Both routes leave from the depot ("0") at the same time and take the same clockwise direction. Figure 4(a) displays the route followed by vehicle 1 and Figure 4(b) corresponds to the route followed by vehicle 2. Vehicle 1 starts providing service before vehicle 2 and continues servicing some one-lane streets until the first two-lane street appears on its route (point "A"). The two vehicles meet for the first time at point "A", service the path "AB", return to "A" to service the paths "AC", "CD", and "DE", return to "D" to service "DF" and "FG", then both return to "F" where each of them follows an independent route. Vehicle v_2 returns to "C" where it waits for vehicle v_1, and both vehicles service "CH" and finish their routes at the depot. Note that the thickest arcs in 4(a) and 4(b) represent two-lane streets.



(a) Route 1



(b) Route 2

Figure 4: Synchronized routes for vehicles 1 and 2.

5. Conclusions

We have introduced, modeled and solved a synchronized arc routing problem arising in snow plowing operations. This problem was shown to be NP-hard. A mixed integer non-linear problem was proposed, and a metaheuristic based on adaptive large neighborhood search was developed. The ALNS algorithm first constructs a good set of feasible routes and then applies an improvement phase. Five destroy/repair operators were used in the improvement phase of the heuristic. The performance of the proposed ALNS procedure was evaluated on large artificial and real instances. The relative performance of each operator was first evaluated and all tend to be useful. The average improvement yielded by the improvement phase of our algorithm is significant, which is an indication of the efficiency of this phase. Finally, we have shown that the algorithm produces realistic routes on a difficult real example.

References

Dali, Z., 2009. Optimization of vehicle routing for plowing and snow disposal. In: Wang, Y., Yi, P., An, S., Wang, H. (Eds.), Proceedings of the 9th International Conference of Chinese Transportation Professionals. ICCTP 2009: Critical Issues in Transportation System Planning, Development, and Management, Harbin, China, pp. 2738–2744.

Dueck, G., 1993. The great deluge algorithm and the record-to-record travel. Journal of Computational Physics 104, 86–92.

Fu, L., Trudel, M., Kim, V., 2009. Optimization winter road maintenance operations under real time information. European Journal of Operational Research 196, 332–341.

Golbaharan, N., 2001. An application of optimization to the snow removal problem - A Column generation approach. Ph.D. thesis, Linkping University, Linkping, Sweden.

Gupta, D., Tokar-Erdemir, E., Kuchera, D., Mannava, A. K., Xiong, W., 2010. Optimal workforce planning and shift scheduling for snow and ice removal. Tech. rep., Industrial and Systems Engineering Program, Department of Mechanical Engineering, University of Minnesota.

Jang, W., Noble, J. S., Hutsel, T., 2010. An integrated model to solve the winter asset and road maintenance problem. IIE Transactions 42, 675–689.

Laporte, G., Musmanno, R., Vocaturo, F., 2010. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. Transportation Science 44, 125–135.

Martello, S., Toth, P., 1990. Knapsack Problems: Algorithms and Computer Implementations. Wiley, New York.

Norrman, J., Eriksson, M., Lindqvist, S., 2005. Relationship between road slipperiness, traffic accident risk and winter road maintenance activity. Climate Research 15, 185–193.

Perrier, N., Campbell, J. F., Gendreau, M., Langevin, A., 2011. Vehicle Routing Models and Algorithms for Winter Road Spreading Operations, in Hybrid Algorithms for Service, Computing and Manufacturing Systems: Routing, Scheduling and Availability Solutions. IGI Books, Hershey, PA.

Perrier, N., Langevin, A., Amaya, C. A., 2008. Vehicle routing for urban snow plow operations. Transportation Science 42, 44–56.

Perrier, N., Langevin, A., Campbell, J. F., 2006a. A survey of models and algorithms for winter road maintenance. Part I: System design for spreading and plowing. Computers & Operations Research 33, 209–238.

Perrier, N., Langevin, A., Campbell, J. F., 2006b. A survey of models and algorithms for winter road maintenance. Part II: System design for snow disposal. Computers & Operations Research 33, 239–262.

Perrier, N., Langevin, A., Campbell, J. F., 2007a. A survey of models and algorithms for winter road maintenance. Part III: Vehicle routing and depot location for spreading. Computers & Operations Research 34, 211–257.

Perrier, N., Langevin, A., Campbell, J. F., 2007b. A survey of models and algorithms for winter road maintenance. Part IV: Vehicle routing and fleet sizing for plowing and snow disposal. Computers & Operations Research 34, 258–297.

Razmara, G., 2004. Snow removal routing problems - Theory and applications. Ph.D. thesis, Linkping University, Linkping Studies in Science and Technology, Linkping, Sweden.

Ropke, S., Pisinger, D., 2006. An adaptive large neighbourhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40, 455–472.

Salazar-Aguilar, M. A., Langevin, A., Laporte, G., 2011. An adaptive large neighborhood search heuristic for a snow plowing problem with synchronized routes. In: Network Optimization - International Network Optimization Conference (INOC 2011). Lecture Notes in Computer Science, Springer, Hamburg, Germany.

Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. Lecture Notes in Computer Science 1520, Springer, Berlin, pp. 417–431.

Sochor, J., Yu, C., 2004. A heuristic method for routing snowplows after snowfall. Ph.D. thesis, Division of Optimization, Department of Mathematics, Linkping Institute of Technology, Linkping, Sweden.

Usman, T., Fu, L., Miranda-Moreno, L. F., 2010. Quantifying safety benefot of winter road maintenance: Accident frequency modeling. Accident Analysis and Prevention 42, 1878–1887.

Venlinen, A., Kangas, M., 2003. Estimation of winter road maintenance costs using climate data. Meteorological Applications 10, 69–73.

Ville de Montréal, 2010. Snow removal 101.
URL `http://ville.montreal.qc.ca`