Snow Plow Route Optimization: A Constraint Programming Approach

Joris Kinable[a,c,*], Willem-Jan van Hoeve[b] and Stephen F. Smith[c]

[a]OPAC, Dept. of Industrial Engineering & Innovation Sciences, Eindhoven University of Technology, Den Dolech 2, 5612 AZ Eindhoven, Netherlands; [b]Tepper School of Business, Carnegie Mellon University, 4765 Forbes Ave, Pittsburgh, PA 15213, USA; [c]Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA

ARTICLE HISTORY
Compiled September 25, 2020

ABSTRACT
Many cities have to cope with annual snowfall, but are struggling to manage their snow plowing activities efficiently. Despite the fact that winter road maintenance has been a popular research subject for decades, very few papers propose scalable models that can incorporate side constraints encountered in real-life applications. In this work we propose a Constraint Programming formulation for a Snow Plow Routing Problem (SPRP). The SPRP under consideration involves finding a set of vehicle routes to service a street network in a pre-defined service area, while accounting for various vehicle constraints and traffic restrictions. The fundamental mathematical problem underlying SPRP is the well-known Capacitated Arc Routing Problem (CARP). Common Mathematical Programming (MP) approaches for CARP are typically based on (1) a graph transformation, thereby transforming CARP into an equivalent node routing problem, or (2) a sparse network formulation. The CP formulation in this paper is based on the former graph transformation. Using geospatial data from the city of Pittsburgh, we empirically show that our CP approach outperforms existing MP formulations for SPRP. For some of the larger instances, our CP model finds 26% shorter plowing schedules than alternative Integer Programming formulations. A test pilot held with actual vehicles proves the applicability of our approach in practice: our routes are 3%-156% shorter than the routes the city of Pittsburgh generated with commercial routing software.

KEYWORDS
winter road maintenance; arc routing; vehicle routing; mathematical programming

1. Introduction

Each year, many northern cities face significant expenditures pertaining winter road maintenance. Snow removal constitutes a large part of these costs. According to a report by the Office of the New York City Comptroller (Stringer, 2015), the costs for ice and snow removal in NYC alone averages $55.3 million a year, with a low of $25.4 million in FY 2008, to a high of $130.7 million in FY 2014. The direct measurable costs related to material, equipment, maintenance, resources and personnel, are however largely exceeded by indirect expenses accounting for the societal, environmental and economical impact of adverse driving conditions. Snow storms have a disruptive impact

*Corresponding author: j.kinable@tue.nl

on mobility and transportation, leading to a significant increase in traffic accidents and congestions, and reduce access to critical infrastructure such as hospitals and airports (Rubin et al., 2010; Usman et al., 2010). Moreover, excessive usage of snow plows as well as salt and chemicals required for deicing, damages road surfaces, corrodes cars and metal structures, and pollutes soil and local water systems (Environmental Protection Agency, 1999). Clearly, the monetary costs of winter road maintenance, as well as the number of people negatively impacted by winter driving conditions motivate the need for a robust, data-driven and highly-optimized system to effectively perform these maintenance operations in a resource constraint environment.

In this work we study a realistic Snow Plow Routing Problem (SPRP) where routes must be computed for a set of heterogeneous vehicles such that they collectively cover a geographical area. The vehicles remove snow from the roads, and simultaneously spread a mixture of salt and chemicals. Vehicles only have a finite capacity of salt available, and must return to a depot whenever they run out of salt. The routes must comply with a number of side-constraints, involving service time restrictions, one-way streets, road priorities and turn restrictions. This work is part of a larger initiative to develop an adaptive system for snow plow optimization and management. This route planning system stores pre-computed snow plow routes in a database, and issues real-time turn-by-turn instructions to the snow plow drivers. The snow plows are equipped with GPS trackers, which allow the system to monitor the plows' progress according to a predefined schedule. Periodically, the system must be able to re-optimize the routes, thereby re-balancing the workload of the individual snow plows. Moreover, when deviation of the original route is necessary, for instance due to an unforeseen road obstruction, equipment failure or emergency request, the system must adequately adjust the routes and issue new instructions to the drivers. In this paper, we focus on the routing and modeling aspects of the problem; real-time adjustments and coping with unexpected events is the subject of future work.

Existing Mathematical Programming (MP) formulations for SPRP are typically classified into sparse models, defined over a sparse routing graph, and dense models which rely on graph transformations. The CP formulation used to solve the SPRP in this paper belongs to the latter category. The fundamental mathematical problem underlying our SPRP is the well-known Capacitated Arc Routing Problem (Golden and Wong, 1981) which is shown to be NP-hard by Golden and Wong (1981). Next to snow plowing, CARP arises in a wide variety of applications, including refuse collection, street sweeping, winter gritting, postal delivery, inspection of roads, power lines, bridges and pipeline systems, meter reading, and road surface marking. In the past, a significant amount of research has been devoted to the design of efficient methods to solve Capacitated Arc Routing Problems, for which a large amount of synthetic benchmark data exists.[1] From an operational perspective, it remains however largely unclear how existing methods for CARP or related Snow Plow Routing Problems can be exploited by a system that must efficiently compute routes under realistic conditions for an actual fleet of vehicles. The latter involves answering practical questions such as: can the proposed methodology be extended with additional side-constraints necessary to generate viable routes? Does the methodology scale to realistically sized instances? Prior to answering these questions, we must however

[1] An overview of benchmark instances can be found here: https://logistik.bwl.uni-mainz.de/forschung/benchmarks/

address more fundamental problems such as how to extract input data for such a routing system, or how to quantify and compare the quality of vehicle schedules generated by the system. As it turns out, simply solving an optimization problem which minimizes total travel time subject to CARP constraints, as is routinely done with synthetic benchmark data, leads to unbalanced and undriveable vehicle itineraries in practice.
This paper sets itself apart from existing work by its strong focus on the *operational* aspects of winter road maintenance. The contributions of this paper are as follows. We present a Constraint Programming (CP) formulation which incorporates a large number of side constraints commonly encountered in winter road maintenance. We show that this CP model scales better than conventional Mathematical (Integer) Programming formulations. To demonstrate the efficacy of our CP formulation, an extensive computational evaluation is performed. Unlike many traditional works that perform computations on synthetic benchmark data, we utilize geospatial data from the city of Pittsburgh to compute realistic schedules. The usage of actual map data requires us to investigate how to extract, process and store this data efficiently. The data used in the experiments covers different geographical settings, ranging from residential neighborhoods to industry and business districts. In the computational evaluation we compare our CP formulation against the two-stage improvement heuristic from Kinable et al. (2016), a commercial snow plow routing system, and alternative MP formulations including the sparse formulation by Perrier et al. (2008), and a dense model based on a Generalized VRP transformation proposed by Bartolini et al. (2013). It is worth noting that, although this paper is explicitly written in the context of winter road maintenance, the methodology can be extended to other arc routing applications as well. Moreover, although the experimental evaluation is conducted using data from the city of Pittsburgh, a very similar problem setting has recently been described for Kentucky state (USA) in Blandford et al. (2018). This again underlines the need for efficient snow plow optimization algorithms.

The remainder of this paper is structured as follows. First Section 2 provides an overview of related literature. Next, Section 3 formally defines our SPRP and introduces notation. The same Section also covers a transformation from SPRP into an equivalent Generalized VRP used by our CP model. The CP model is presented in Section 4, followed by two alternative Mathematical Programming formulations in Section 5. Essential to the computation of practical routes is the calculation of turn restrictions (Section 6). An extensive experimental evaluation, covering data extraction and preprocessing, computational results of our CP model, and a pilot test, is provided in Section 7. Finally, Section 8 offers the conclusion and discusses opportunities for future research.

2. Related Literature

A rich body of literature pertaining the Capacitated Arc Routing Problem, as well as related snow plow optimization problems exists. For an excellent literature overview of CARP problems, applications and solution approaches, we refer to the books by Corberán and Laporte (2014a); Dror (2000) and Wøhlk (2008). An extensive overview of decision and optimization problems involving winter road maintenance can be found in the survey articles by Corberán and Prins (2010); Perrier et al. (2006a,b, 2007a,b). In this section we provide a concise overview of existing mathematical programming

formulations as well as heuristic solution approaches for CARP and SPRP.

Mathematical programming formulations for CARP are broadly divided into Branch-and-bound approaches based on combinatorial lower bounds, Branch-Bound-Cut formulations, and Branch-and-Price formulations (Corberán and Laporte, 2014b). The class of Branch-Bound-Cut formulations can be further refined into models that are directly defined on the arc routing graph, so-called sparse models, and formulations that transform the arc-routing problem into an equivalent node-routing-problem.
Belenguer and Benavent (1998, 2003) study the polyhedron associated with CARP and identify a large number of valid inequalities. These inequalities are used to strengthen an LP formulation for CARP by iteratively invoking separation procedures. The optimal cost of the LP at any iteration is a lower bound for the cost of the optimal CARP solution. The algorithm stops when no more violated inequalities are found, or the lower bound is equal to a known upper bound provided by some primal heuristic. In the latter case, the lower bound and the corresponding heuristic solution are optimal. In the former case, the procedure only yields a valid bound but no optimal solution; it would be necessary to employ a branching scheme to continue the process. Perrier et al. (2008) presents a rich MIP model for a SPRP that, except for the lack of resource constraints, is identical to the problem studied in this paper. The MIP model from Perrier et al. (2008) is defined on a sparse routing graph and incorporates several commonly used side-constraints. To improve the scalability of their model, the authors present two decomposition approaches: a cluster-first-route-second approach and parallel decomposition approach. The former cluster-first-route-second approach first partitions the arcs into clusters, each having approximately the same work load, and then solves a Hierarchical Rural Postman Problem. The parallel algorithm on the other hand constructs several routes in parallel by sequentially solving a Multiple Vehicle Rural Postman problem for each class of vehicles. A limited computational study is performed on real-world data involving the Canadian city of Dieppe.

Column generation (CG) approaches to solve CARP have been studied by Bartolini et al. (2013); Bode and Irnich (2012); Letchford and Oukil (2009); Pecin and Uchoa (2019). Letchford and Oukil (2009) propose a set-covering formulation for CARP which is solved with CG. The CG approach uses a specialized pricing routine which exploits the sparsity of arc routing graphs. Longo et al. (2006) use a graph transformation to transform CARP into an equivalent CVRP and solve the resulting problem with the CG algorithm of Fukasawa et al. (2006). Bartolini et al. (2013) use a cut-and-column-generation procedure to solve CARP with a set partitioning formulation. Their method executes a sequence of bounding procedures to obtain progressively stronger lower bounds on the optimal solution. The final dual solution computed by the last bounding procedure is then used to generate a reduced integer problem which is guaranteed to contain an optimal solution, and is solved using an integer programming solver. A detailed comparison of the various CG algorithms for CARP can be found in Pecin and Uchoa (2019). In addition, the authors generalize existing CG formulations and combine several of their core features into a new Branch-and-Price formulation. To date, their method produces the strongest bounds. Extensive research is however required to determine whether their methods can be extended with side-constraints encountered in real-world SPRPs without breaking the structure of their pricing problem, which, to a large extend, determines the performance of their algorithm.

Next to mathematical solution approaches, heuristic approaches have been proposed for variants of CARP. Of particular interest to us are heuristics which (1) scale to sufficiently large instances, and (2) can be adapted to solve SPRP. A Late Acceptance (LA) heuristic as well as a basic CP model for a related SPRP have been proposed in Kinable et al. (2016). The latter paper involves servicing roads with a homogeneous fleet of vehicles while accounting for renewable resource constraints such as salt and fuel. Unlike the work by Kinable et al. (2016), this paper does not take fuel constraints into account, as in practice salt is a much more stringent resource and the majority of salt depots also provide fuel. Likewise there are a number of significant differences in problem definition pertaining the operational constraints. Our problem must be solved for a heterogeneous fleet (as opposed to a homogeneous fleet) of vehicles, must satisfy turn restrictions, as well as restrictions on the types of vehicles that can be used to service certain roads and limits on the duration of the routes. Finally, to ensure that arterial roads as well as roads that lead to critical infrastructure such as hospitals are serviced prior to residential neighborhood roads, in this paper, similar to Perrier et al. (2008), we explicitly account for road priority classes. In Kinable et al. (2016) the authors conclude that their CP model finds good solutions for instances up to a 1000 single-lane road segments, but does not scale well beyond that; their LA heuristic scales considerably better. Experiments conducted in this paper (Section 7.3) reveal that our CP model can outperform the LA heuristic, from (Kinable et al., 2016) even for some of the largest problems in our benchmark set.
Vidal (2017) proposes a Large Neighborhood Search (LNS) heuristic which is able to incorporate many practical side-constraints including turn penalties. Irnich (2008) uses an alternative LNS procedure to solve very large mail delivery routing problems for the Deutsche Post. In particular, the heuristic considers turn and street crossing restrictions, cluster constraints, and the use of alternative modes of transportation. Quirion-Blais et al. (2017) present an ALNS for a winter road maintenance problem where vehicles must both plow and grit roads. A challenging aspect in their problem is that some vehicles can both plow and grit, whereas others can only perform one of these tasks, so synchronization among vehicles is required. Wang and Liu (2019) propose a Location-Allocation-Routing heuristic for a snow plow problem that involves both resource allocation and vehicle routing. In order to improve the resilience of a road network after a snow storm, their approach prioritizes servicing road segments that bring the largest improvements in average network throughput. In Zhou and Wu (2018), the authors propose a cluster-first-route-second heuristic for a CARP variant which assumes that roads are impassable whenever they have not been serviced. Most related works, on the other hand, assume that vehicles can traverse road segments even when they are not yet cleared of snow, especially in a context where roads of higher priority have to be serviced first, but not all high-priority roads are connected. Next to the aforementioned heuristic approach, alternative heuristics to solve CARP problems include an adaptive ILS (Dell'Amico et al., 2016), tabu search (Brandão and Eglese, 2008), GRASP (Usberti et al., 2013) and Variable Neighborhood Descent (Hertz and Mittaz, 2001).

Several extensions to CARP have appeared in the literature which include side constraints from real-world applications. Common CARP extensions for snow plow optimization problems are: intermediate resupplies, and vehicle synchronization.
Ghiani et al. (2001) introduce CARP with Intermediate Facilities (CARP-IF). In CARP-IF, vehicles visit Intermediate Facilities throughout their tour to replenish resources or to unload. Examples arise in resupplying salt or gritting material for

snow plows or emptying garbage trucks. Ghiani et al. (2001) propose two lower bounding as well as two heuristic procedures for CARP-IF. The first lower bounding procedure is based on the Rural Postman Problem whereas the second one is based on a LP relaxation of a related CARP problem. A variable neighborhood heuristic for CARP-IF can be found in (Polacek et al., 2008).
One practical issue encountered in solutions to CARP-IF problems involves a strong imbalance between the workload of vehicles. This is due to the fact that CARP-IF does not define a limit on the number of times a vehicle can resupply at IFs. As a result, one vehicle could potentially do all the work. To circumvent this issue, (Ghiani et al., 2004) extended the CARP-IF problem with tour length restrictions (CLARP-IF) and solve the problem with a tabu search heuristic.
Thus far, CARP-IF problems have only been solved using synthetic benchmark data. It remains to be investigated whether intermediate resupplying for snow plows is beneficial in terms of the overall schedule. Moreover, IFs also pose some managerial challenges, as supply depots are often managed by different districts which would have to implement a resource sharing policy.
Salazar-Aguilar et al. (2012a,b) study a related snow plow optimization problem where plowing operations are synchronized to service multiple lanes of the same street segment simultaneously with multiple vehicles. This so-called 'tandem plowing' pushes snow from one lane to the next and eventually to the side of the road, thereby avoiding snow mounts building up between lanes. Tandem plowing typically occurs on high-ways where the driving speed is high enough for snow plows to throw the snow from one lane to the next, or when special equipment such as snow blowers are available. The problem in Salazar-Aguilar et al. (2012a) is defined through a MIP model and solved with an efficient Adaptive Neighborhood Search. Although tandem plowing has its merits, it is not frequently applied within city limits because of the added level of planning complexity. An alternative solution to tandem plowing can be found in Gundersen et al. (2017) where the authors propose a MIP for a SPRP with precedence constraints between different driving lanes. These precedence constraints ensure that streets are plowed from the inside out: snow plows service the inner most lane first, and gradually push the snow towards outer lanes and eventually off the road.

3. Problem description

The snowplow optimization problem discussed in this paper is a generalization of the classical Capacitated Arc Routing Problem (CARP). Let $G(V, A, E)$ be a strongly connected, mixed multigraph, where $V = \{v_0, \ldots, v_{n-1}, v_n, v_{n+1}\}$ is a set of vertices including a vehicle source depot vertex v_0 and a vehicle target depot v_{n+1}. $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ a set of directed arcs, and $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ a set of undirected edges $(v_i, v_j)(i < j)$. Vertices in the graph represent intersections in the road network or depots; the arcs and edges represent resp. directed and undirected road segments. A road segment consisting of 2 lanes in each direction, is represented with 4 directed arcs in the graph. Edge set E typically consists of small roads which can be cleared by a snow plow in one pass from either direction. Each edge $e = (v_i, v_j) \in E$ or arc $e = (v_i, v_j) \in A$ has an associated non-negative integer demand $q_e \geq 0$ for salt, a nonnegative c_{ij}^{serv} servicing cost, and a nonnegative c_{ij} traversal cost. In this work the servicing and traversal costs are to be interpreted as travel times. Edges (resp. arcs) with positive demand ($q_e > 0$) form a subset $E_R \subseteq E$ (resp. $A_R \subseteq A$) of *required* edges

(arcs). Each required edge has to be serviced exactly once. Any edge can be traversed multiple times. The act of traversing an edge, i.e. without servicing it, is commonly referred to as *deadheading*. Graph G will be denoted as the *routing graph*, whereas the graph induced by the required edges E_R and arcs A_R is called the *plowing graph*.
The set V contains a subset I of supply depots where vehicles can load salt. There is no requirement that the target depot v_{n+1}, to which the vehicles must return at the end of their trip, is included in I. It is assumed that vehicles are loaded to full capacity when they leave the depot v_0. If v_{n+1} is not included in I, a vehicle must visit a supply depot prior to returning to v_{n+1} to ensure that the vehicle's salt is replenished before its next trip.
Servicing is performed by a heterogeneous set of vehicles K. Each vehicle $k \in K$ has a salt capacity Q^k. Moreover, weight and width restrictions on some of the streets prohibit some vehicles from servicing some streets. We use superscript ^{k} to denote subsets of edges and vertices that can be reached by vehicle k, e.g. $A^k \subseteq A$ is the subset of arcs traversable by vehicle $k \in K$.
Each road segment has an associated *priority class*. Priority classes are denoted in the range $1, \ldots, P$, where roads in class 1 have the highest priority. Typically, roads of priority p have to be serviced before roads of priority $p+1$. The priority class of a road is often decided by the volume of traffic that uses that particular road (e.g. highways are considered more important than residential roads). Similarly, roads to critical facilities such as hospitals may also be included in the highest priority class. Using the priority classes, the set of edges E_R and arcs A_R can be partitioned into disjoint subsets $\{A_1, A_2, \ldots, A_P\}$. We also include a fictional priority class $P+1$ which includes the shortest return path from the last serviced edge in class P back to the depot. The set of all priority classes is denoted as $\mathscr{P} = \{1, \ldots, P, P+1\}$. Certain subscripts and superscripts can be combined together, e.g. $A_R^{kp} \subseteq A$ is the set of required arcs, belonging to priority class $p \in \mathscr{P}$ which can be traversed by vehicle $k \in K$.
The goal is to compute routes for every vehicle $k \in K$ such that every edge $e \in E_R$ and every arc $a \in A_R$ is serviced exactly once. A vehicle route must start at the source depot v_0, and terminate at the target depot v_{n+1}. The sum of salt demands of edges serviced by a single vehicle $k \in K$ between visits to a resupply depot cannot exceed the vehicle capacity Q^k. Different objective functions will be considered, including minimizing of the total schedule duration (makespan), minimizing the total amount of deadheading, or minimizing the weighted completion time of the road priority classes. A summary of the sets and parameters defining SPRP is given in Table 1.

3.1. Notation

Throughout this paper, the following additional notation is used. Given a subset of vertices $S \subseteq V$, the cutset $\delta(S)$ denotes the set of edges with exactly one endpoint in S. The cutset $\delta^+(S)$ denotes the set of directed edges (arcs) having their tail in S and their head not in S. Similarly, the cutset $\delta^-(S)$ denotes the set of edges with their head in S and their tail outside S. For undirected edges $\delta(S) = \delta^+(S) = \delta^-(S)$. When S is a singleton vertex, we use the shorthand $\delta(i)$ instead of $\delta(\{v_i\})$. The set $E(S)$ denotes the set of edges with both endpoints in S. We denote subsets of required edges using subscript $_R$, e.g., $\delta_R(S) = \delta(S) \cap E_R$. To prevent ambiguity, in some rare cases we will explicitly specify the graph G for the cutset operator. For instance, $\delta(S, G)$ denotes the cutset $\delta(S)$ in graph G. Finally, since the routing and plowing graphs are *mixed* graphs consisting of both arcs and edges, we will occasionally use composite sets consisting

Parameter	Description
$V = \{v_0, v_1, \ldots, v_n\}$	Set of routing nodes, with v_0 being the vehicle depot.
$I \subseteq V$	Set of resupply depots
E	Set of undirected road segments
A	Set of directed road segments
$\mathscr{P} = \{1, \ldots, P, P+1\}$	Set of road priority classes. Roads in class 1 have the highest priority. $P+1$ is a fictional class containing return paths to the depot.
$E_R \subseteq E, A_R \subseteq A$	Subset of edges, resp. arcs that require servicing.
$V_R \subseteq V$	Subset of vertices incident to a required edge: $\{v_i \in V : \delta_R(i) \neq \emptyset\}$. Note that V_R does not necessarily contain depot v_0.
K	Set of heterogeneous vehicles
q_{ij}	Demand of road segment (v_i, v_j). $q_{ij} = 0$ if $(v_i, v_j) \notin E_R \cup A_R$, $q_{ij} > 0$ otherwise.
Q^k	Capacity of vehicle k.
c_{ij}^{serv}	Time required to service road segment (v_i, v_j).
c_{ij}	Time required to deadhead road segment (v_i, v_j).
r_i	Resupply time at depot $i \in I$

Table 1.: A summary of the sets and parameters defining the Snow Plow Routing Problem (SPRP).

of both arcs and edges (e.g. $A_R \cup E_R$). This deliberate but minor abuse in notation significantly reduces the verbosity of this paper.

3.2. Node routing transformation

Many formulations for arc routing problems, including some of the models in this paper, rely on the fact that arc routing problems can be cast into traditional node routing problems through appropriate transformations of the routing graph. The advantage of such a transformation is that the resulting node routing problems can be solved using common algorithms for the Capacitated Vehicle Routing Problem. These transformations are however not free, in the sense that they increase the size of the problem instances significantly. The number of *vertices* in the node routing problems are polynomial in the number of *edges* in the corresponding arc routing problems. Moreover, while an arc routing problem is typically defined on a sparse graph, the node routing problem is defined on a dense graph. Since our CP model will be based on a Generalized Vehicle Routing Problem formulation, we review node routing transformation procedures and introduce some additional notation used in the subsequent sections.

Several alternative node routing transformations for CARPs can be found in the literature, including Baldacci and Maniezzo (2006); Bartolini et al. (2013); Longo et al. (2006); Pearn et al. (1987). Each of these transformations relies on the property that in an optimal CARP solution, deadheading between two locations $u, v \in V$ always occurs along a shortest $P_{u,v}$ path. In this work, we utilize the transformation introduced by Bartolini et al. (2013) to transform SPRP into an asymmetric, Generalized Vehicle Routing Problem (GVRP) on a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{A})$. Recall that in the GVRP, nodes are grouped into disjoint clusters, and that an optimal solution consists of a set of routes which include exactly one node from each cluster. Informally, to encode the SPRP as a GVRP problem, one has to construct a new graph $\mathcal{G}$ consisting of a single vertex for every depot $v \in I \cup \{v_0, v_{n+1}\}$, and one resp. two vertices for every *orientation* of an edge in A_R resp. E_R. Vertices in $\mathcal{G}$ representing different orientations of the same edge are grouped together in a cluster. An example GVRP encoding for

a SPRP problem instance is depicted in Figure 1. Here, the arcs represent shortest deadheading paths from a vehicle's terminal location after servicing one edge, to the vehicle's starting location of the next edge. In this example solution, the vehicle starts at node v_0, deadheads along a shortest path to intersection v_1, then services consecutively edges (v_1, v_2) and (v_2, v_3), then deadheads from v_3 to v_1 and services (v_1, v_4) and so on. It follows that a solution to SPRP can be concisely represented by an ordered sequence and orientation of the edges serviced by the vehicles.

Given an SPRP instance, the GVRP graph $\mathcal{G}(\mathcal{V}, \mathcal{A})$ is formally defined as follows. The node set $\mathcal{V}$ is composed of two vertices s_{ij} and s_{ji}, one for each orientation of an edge $(i, j) \in E_R$, one vertex s_{ij} for each arc $(i, j) \in A_R$, and one vertex for each node $v_i \in I \cup \{v_0, v_{n+1}\}$. The set of nodes $\mathcal{V}$ is partitioned into $|E_R| + |A_R| + |I| + 1$ disjoint clusters $\mathcal{C} = \mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_m, \mathcal{I}_1, \ldots, \mathcal{I}_{|I|}$. Cluster $\mathcal{V}_0$ contains the depot nodes v_0 and v_{n+1}. Clusters $\mathcal{V}_1, \ldots, \mathcal{V}_m$, with $m = |E_R| + |A_R|$, correspond with the edges $E_R \cup A_R = e_1, \ldots, e_m$; a cluster associated with edge $e = (i, j) \in E_R$ contains nodes s_{ij} and s_{ji} whereas a cluster associated with arc $a = (i, j) \in A_R$ contains only a single node. Clusters $\mathcal{I} = \mathcal{I}_1, \ldots, \mathcal{I}_{|I|}$ each contain a single vertex v_i representing a supply depot $v_i \in V$. The function $\pi(v_i) : \mathcal{V} \to \mathcal{C}$ provides a mapping from a node $v_i \in \mathcal{V}$ to the unique cluster $\mathcal{V}_c \in \mathcal{C}$ containing v_i. For a given subset of arcs and edges $U \subseteq E_R \cup A_R$, the shorthand $\mathcal{V}(U) = \bigcup_{(i,j)\in U} \{s_{ij}\}$ will be used to denote the subset of vertices in $\mathcal{G}$ induced by U.

The arc set $\mathcal{A}$ in $\mathcal{G}(\mathcal{V}, \mathcal{A})$ can now be defined as follows:

(1) An arc (u, v) for every pair of nodes $u, v \in \bigcup_{i=0}^{m} \mathcal{V}_i$ that are taken from *different* clusters.
(2) An arc (u, v) for every pair of nodes $u \in \bigcup_{i=1}^{m} \mathcal{V}_i$, $v \in \mathcal{I}$.
(3) An arc (u, v) for every pair of nodes $u \in \mathcal{I}$, $v \in \bigcup_{i=0}^{m} \mathcal{V}_i$.

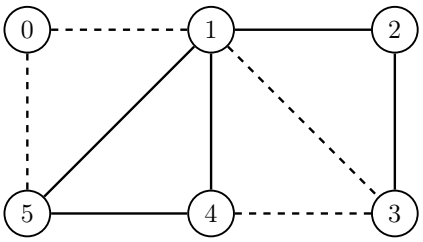
Each node $v_i \in \mathcal{V}$ has an associated demand $q(v_i)$ and a duration $\tau(v_i)$:

- $q(v) = q_{ij}$ for every node v that represents an edge $(i, j) \in E_R \cup A_R$; $q(v) = 0$ otherwise.
- $\tau(v) = c_{ij}^{serv}$ for every node v that represents an edge $(i, j) \in E_R \cup A_R$, $\tau(v) = r_v$ if v represents a supply depot in I; $\tau(v) = 0$ otherwise.

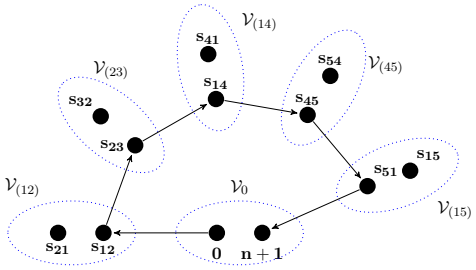
Due to the unique mapping between a node and the cluster it belongs to, we extend the same notation to clusters: $q(\mathcal{V}_{\pi(v)}) = q(v)$ and $\tau(\mathcal{V}_{\pi(v)}) = \tau(v)$. Finally, each arc $a = (v_u, v_w) \in \mathcal{A}$ has an associated cost $w(a) : \mathcal{A} \to \mathbb{R}_0^+$ representing the deadheading cost along a shortest path P_{uw} from vertex v_u to vertex v_w. More precisely, let $i(v)$ resp. $j(v)$ denote the initial and terminal endpoints associated with a node $v \in \mathcal{V}$, i.e. $i(v) = j(v)$ if v is a depot, and $i(v) = i$, $j(v) = j$ if v corresponds with a required edge represented by vertex s_{ij} in $\mathcal{G}$. The weight function $w(a)$ of an arc $a = (u, v) \in \mathcal{A}$ is then defined as $w(a) = c(P_{j(u),i(v)})$, where $c(P_{j(u),i(v)})$ is the cost of the shortest deadheading path $P_{j(u),i(v)}$ from vertex $j(u)$ to vertex $i(v)$.

4. Constraint Programming Formulation

The previous sections discussed two main representations for solving the CARP, i.e., one based on the sparse arc routing graph and one based on the transformation into a denser node routing instance, both of which have been successfully applied in previous optimization approaches. In principle, one could design a CP model based on either



(a) Simple SPRP instance; solid lines require servicing, dashed lines can be used for deadheading. For simplicity, weights and distances have been omitted from this figure.



(b) GVRP encoding of a potential solution. The graph represents an ordering and direction of the required edges. For clarity, unused arcs have been omitted.

Figure 1.: SPRP instance and potential solution represented on a GVRP graph

of these representations. In this work, we propose a CP model based on the GVRP transformation introduced in Section 3.2. Before we present the formulation, we provide a motivation for this choice.

Recall that constraint programming technology relies on a modular representation of the problem, for which constraint propagation removes infeasible values from the variable domains based on individual constraints (Bessiere, 2006). Constraint propagation can be particularly effective when applied to so-called 'global constraints' that represent a combinatorial structure (Régin, 2011; van Hoeve and Katriel, 2006); for example, the *alldifferent* constraint over a set of integer variables is much more effective than the decomposition into pairwise not-equal constraints. CP generally works well when the modeler can expose the problem structure to the solver by using such global constraints.

To obtain a CP formulation for SPRP, one could utilize the arc routing representation, and perhaps transform an existing MIP model for CARP into an equivalent CP model with general integer variables and linear constraints. However, this would yield a loosely coupled model as the problem structure gets decomposed. Moreover, propagation algorithms for linear constraints are typically weak because they commonly rely on bound propagation. A significantly stronger CP model, that preserves the underlying problem structure, can be obtained by formulating the SPRP, after it has been transformed into a GVRP, as a *scheduling* problem instead. Treating VRPs as scheduling problems is common practice in the CP literature, see e.g. Kilby and Shaw (2006). Most CP solvers have dedicated syntax and internal data structures to represent and handle scheduling problems, for which advanced propagation methods have been designed (Baptiste et al., 2001; Laborie et al., 2018). By basing our CP model on the denser node routing representation, and the associated CVRP formulation, we can leverage the constraint-based scheduling technology for solving the SPRP more effectively.

The CP model for SPRP treats the vertices of the GVRP graph as *jobs* which have to be assigned to vehicles. Moreover, starting times have to be determined for each job. Jobs in CP are efficiently represented through interval variables (Laborie and Rogerie, 2008; Laborie et al., 2009). We will denote an interval variable as a tuple $\alpha = \{r, d, t, [opt]\}$, where r denotes the earliest start time of the interval, d the latest finish time, t the minimum duration of the interval, and the optional parameter $[opt]$ indicates whether scheduling of the interval is required or not. Optional intervals can

Table 2.: Description of CP constraints. All of these constraints are available in IBM ILOG CP Optimizer by default.

Constraint	Description
presenceOf(α)	Returns 1 if interval α is present, 0 otherwise.
noOverlapSeq($B, dist$)	Sequences the intervals in the set B. Ensures that the intervals in B do not overlap. Furthermore, the two-dimensional distance matrix $dist$ specifies for each pair of intervals a sequence dependent setup time. Absent intervals are ignored. Returns a sequence of the intervals in B.
first(α, seq)	If interval α is present in sequence seq, it must be scheduled before any other interval in the sequence.
last(α, seq)	If interval α is present in sequence seq, it must scheduled after all other intervals in the sequence.
succ(α, seq)	Returns the interval immediately succeeding the interval α in the sequence seq.
pred(α, seq)	Returns the interval immediately preceding the interval α in the sequence seq.
startOf(α)	Returns an expression representing the start time of interval α.
endOf(α)	Returns an expression representing the end time of interval α.
stepAtStart(α, h^-, h^+)	Function in time t which returns a value between h^- and h^+, starting from time $t =$ **startOf**(α). The function returns 0 when t is absent, or before the start of α. When $h^- = h^+$, the shorthand **stepAtStart**(α, h) is used instead.
alwaysEqual(f, α, v)	Function f must equal the value v between the start and end of interval α, if α is present.
alternative(α, B)	If interval α is present, then exactly one of the intervals in set B is present. The start and end of interval α coincides with the start and end of the selected interval from set B.
synchronize(B)	All present interval variables from set B start, respectively end, at the same time.

be either present or absent in the final solution. Absent interval variables are ignored by any constraint or expression they are part of.
To model SPRP as a CP problem, we use two sets of interval variables:

(1) Job variables $j_{\mathcal{V}_c}$ for each cluster $\mathcal{V}_c = \mathcal{V}_1, \ldots, \mathcal{V}_m, \mathcal{I}_1, \ldots, \mathcal{I}_{|I|}$.
(2) Assignment variables a_i^k for all $k \in K$, $i \in \mathcal{V}$

The job variables $j_{\mathcal{V}_c}$ represent road segments, and the role of the assignment variables a_i^k is to associate the job variables with a vehicle; a more detailed explanation is presented below.

A setup time $t_{ij} = w(a)$ is defined between a pair of assignment variables a_i^k, a_j^k, $k \in K$, $a = (i, j) \in \mathcal{A}$. The complete CP model is stated in Algorithm 1. This model is a basic model which captures all SPRP constraints; extensions to this model with additional side constraints are discussed later in this section. A general description of the constraints used in Algorithm 1 can be found in Table 2.

Constraint 4 in Algorithm 1 ensures that every job is assigned to exactly one vehicle. This implies that every street segment gets serviced by one vehicle, and that an edge $(i, j) \in E_R$ represented by two nodes s_{ij} and s_{ji} in $\mathcal{G}$ is only serviced in one direction. Sequencing of the jobs assigned to each vehicle is performed through Constraints 6-8. Resources are managed through cumulative resource constraints (Constraints 9-10): vehicles start with a full load of salt, performing a plow job i consumes $q(v_i)$ salt, and visiting a salt depot replenishes the salt resource (Constraints 9). For each truck, the salt level needs to remain between 0 and Q^k (Constraint 10).
Lines 11-12 are redundant constraints which are used to improve the performance of the model. Constraint 11 links the start and end times of consecutive intervals.

Algorithm 1: Constraint programming model based on GVRP

`Variable definitions:`

1 $j_{\mathcal{V}_c} = \begin{cases} \{0, \infty, \tau(\mathcal{V}_c)\} & \text{if } \mathcal{V}_c = \mathcal{V}_1, \ldots, \mathcal{V}_m \\ \{0, \infty, \tau(\mathcal{V}_c), opt\} & \text{if } \mathcal{V}_c = \mathcal{I}_1, \ldots, \mathcal{I}_{|I|} \end{cases}$

2 $a^k_{v_i}, \forall k \in K = \begin{cases} \{0, 0, 0\} & \text{if } v_i = 0 \\ \{0, \infty, 0\} & \text{if } v_i = n+1 \\ \{0, \infty, \tau(v_i)\} & v_i \in \mathcal{V} \setminus \{0, n+1\} \end{cases}$

3 $obj \in \{0, \infty\}$

`Constraints:`

4 alternative($j_{\mathcal{V}_c}$, $\bigcup_{k \in K} \bigcup_{v_i \in \mathcal{V}_c} a^k_{v_i}$) $\quad \mathcal{V}_c = \mathcal{V}_1, \ldots, \mathcal{V}_m, \mathcal{I}_1, \ldots, \mathcal{I}_{|I|}$

5 **forall** $k \in K$

`Sequencing Constraints:`

6 seq^k = noOverlapSeq($\bigcup_{v_i \in \mathcal{V}} a^k_{v_i}$, $[t_{ij} \mid (i,j) \in \mathcal{A}]$)

7 first($a^k_{v_0}$, seq^k)

8 last($a^k_{v_{n+1}}$, seq^k)

`Salt Constraints:`

9 saltCumulFunc^k =stepAtStart($a^k_{v_0}$, Q^k)$-\sum_{v_i \in \mathcal{V}}$stepAtStart($a^k_{v_i}$, $q(v_i)$) $+\sum_{v_i \in \mathcal{I}}$stepAtStart($a^k_{v_i}$, 0, Q^k)

10 $0 \leq$ saltCumulFunc$^k \leq Q^k$

`Redundant Constraints:` **forall** $k \in K$

11 startOf($a^k_{v_i}$)=endOf(pred($a^k_{v_i}$,seq^k))$+t_{\text{pred}[a^k_{v_i}, seq^k], a^k_{v_i}}$ $\quad \forall v_i \in \mathcal{V} \setminus \{0\}$

12 alwaysEqual(saltCumulFunc, $a^k_{v_i}$, Q^k) $\quad \forall v_i \in \mathcal{I}$

`Objective functions:`

CP_{mkspn} `- Minimize makespan:`

13 `Min` obj

14 **forall** $k \in K$

15 $obj \geq$ endOf($a^k_{v_{n+1}}$)

CP_{deadh} `- Minimize deadheading:`

16 `Min` $\sum_{k \in K} \sum_{v_i \in \mathcal{V} \setminus \{0\}} t_{\text{pred}[a^k_{v_i}, seq^k], a^k_{v_i}}$

$CP_{weighted}$ `- Minimize weighted completion time per priority class:`

17 `Min` $\sum_{k \in K} \sum_{p \in \mathscr{P}} \alpha_p$`Max`$_{v_i \in \mathcal{V}(A^p_R \cup E^p_R)}$endOf($a^k_{v_i}$)$\}$+endOf($a^k_{v_{n+1}}$)

CP_{lex} `- Lexicographic search:`

18 $obj = LexSearch(\forall p \in \mathscr{P}: \; Minimize$ `Max`$_{v_i \in \mathcal{V}(A^p_R \cup E^p_R)}$endOf($a^k_{v_i}$)$\})$

Constraint 12 ensures that vehicles are always reloaded to full capacity. In addition to these constraints, one could add explicit constraints stating that the source depot job $j_{\mathcal{V}_0}$, as well as any job $j_{\mathcal{I}_i}$, $i \in I$ must be followed by a non-supply job, and that the second to last job must be a resupply job if the target depot does not have salt. Experiments however revealed that adding these explicit constraints had a negative impact on the performance of the model and have therefore been omitted.

The constraints in Algorithm 1 can be combined with different objective functions. In Algorithm 1 we distinguish 4 different objective functions (lines 12-18): CP_{mkspn} minimizes the makespan of the schedule, CP_{deadh} minimizes the total amount of deadheading performed by the vehicles, $CP_{weighted}$ minimizes the weighted completion

times of the different priority classes using different weights α_p for each priority class. The objective function CP_{lex} performs a lexicographic search over the different components in the objective function: the search starts by minimizing the first component which minimizes the completion time of the first priority class, then proceeds by minimizing the second component (completion time of the 2nd priority class) while keeping the objective value of the first component fixed, and so on. Technically, by selecting appropriate values for the weight coefficients α_p, $CP_{weighted}$ and CP_{lex} can produce the same optimal outcome, but both models use very different search procedures to reach these solutions, and hence may terminate with different outcomes when the search is terminated prematurely, e.g., by a time limit.

The CP model in Algorithm 1 is flexible enough to accommodate a variety of additional side constraints frequently encountered in the literature. To limit the maximum duration of a route, as is the case in Ghiani et al. (2004), we would redefine variables $a^k_{v_{n+1}} = \{0, \infty, 0\}$ to $a^k_{v_{n+1}} = \{0, L, 0\}$, where L is the maximum route duration. To enforce that a certain street segment cannot be serviced by a specific vehicle, for instance due to weight or width restrictions, we simply omit the corresponding assignment variable $a^k_{v_i}$. In some cases the vehicle return depot does not have salt, i.e. $v_{n+1} \notin I$. Consequently, the vehicles are required to replenish salt prior to returning to the depot. This can be accomplished by setting $q(v_{n+1}) = Q^k$, i.e. setting the salt demand of depot $n+1$ equal to the vehicle capacity. Constraint 9 will then automatically enforce a resupply prior to visiting the return depot. In (Salazar-Aguilar et al., 2012b), a subset of streets with two or more lanes in the same direction have to be plowed simultaneously by different, synchronized vehicles. Since the CP model uses interval variables, synchronization can be implemented through the global constraint *synchronize* which forces intervals to start (resp. end) at the same time. So for a given subset of arcs $A' \subset A_R$ which have to be plowed in tandem formation, we can simply add the constraint $synchronize(\cup_{(i,j)\in A'} j_{\pi(s_{ij})})$ to the CP model.

5. Alternative mathematical programming formulations

To evaluate the performance of the CP model introduced in Section 4, we adapt two alternative mathematical programming formulations to the SPRP discussed in this paper. Most MIP models for CARP can be classified into sparse models, defined over the sparse routing graph G (e.g. Belenguer and Benavent (1998); Ghiani et al. (2001); Letchford (1997); Perrier et al. (2008)), and dense models which rely on a graph transformation (e.g. Baldacci and Maniezzo (2006); Bartolini et al. (2013); Longo et al. (2006); Pearn et al. (1987)). For our empirical evaluation, we selected two representative MIP models, one from each class. In Section 5.1, the sparse model by Perrier et al. (2008) is adapted to our problem setting. Section 5.2 provides a dense formulation defined over the transformed graph $\mathcal{G}$ from Section 3.2.

5.1. Sparse MIP formulation

Perrier et al. (2008) propose an elegant MIP formulation for a related snow plow optimization problem. In contrast to our CP formulation, the MIP model from Perrier et al. (2008) does not rely on an explicit graph transformation; instead, the model is directly defined on the sparse arc routing graph $G(V, E \cup A)$. The model uses x^{kp}_{ij} and

y_{ij}^{kp} variables to count how often vehicle $k \in K$ services, respectively traverses, an edge $(i,j) \in E_R \cup A_R$ in priority class $p \in \mathbb{P}$. In addition, the model uses auxiliary flow variables w_{ij}^{ph} to implement subtour elimination constraints, as well as n_{uvw}^{kp} variables to explicitly count how often vehicle k performs a turn $(u,v,w) : (u,v),(v,w) \in E \cup A$. The priority index p in the variable definitions is used to explicitly distinguish between solutions where road priorities are strictly enforced, i.e. roads of higher priority need to be cleared prior to servicing roads of lower priority, and solutions where milder priority restrictions apply. In the latter case, some roads of lower priority are serviced with a higher priority, so-called class upgrading. This typically leads to schedules having a shorter makespan, at the expense of longer completion times for some of the higher priority classes.

Solving the MIP model results in an assignment of edges to vehicles: for each vehicle $k \in K$ and for each priority class $p \in \mathbb{P}$ it is known how often each vehicle services, resp. deadheads, a specific road segment. Unlike the solutions to the CP model (Section 4), the resulting MIP solution does *not* directly produce a set of vehicle routes because the variables do not encode an explicit ordering of the edges. How to extract actual routes from the assignment variables is not discussed in Perrier et al. (2008). For small instances, one could define a simple CP model to perform the sequencing, but for larger instances, one would have to implement a variant for Hierholzer's Eulerian cycle Algorithm (Hierholzer and Wiener, 1873) which explicitly takes the number of turns identified by the n_{uvw}^{kp} variables into account.

To use the sparse model by Perrier et al. (2008) to solve the SPRP discussed in this paper, the following capacity constraints have to be added to their model:

$$\sum_{p \in \mathbb{P}} \sum_{(v_i,v_j) \in A_R} x_{ij}^{pk} \leq Q^k \qquad \forall k \in K \tag{1}$$

Moreover, the sparse MIP model assumes that all edges in the routing graph are *directed*. Therefore, every edge $(v_i,v_j) \in E$ must be represented as two directed arcs: (v_i,v_j) and (v_j,v_i). For every edge $(v_i,v_j) \in E_R$, a constraint is added to ensure that the edge is only plowed once in one direction: $\sum_{k \in K} \sum_{p \in \mathbb{P}} x_{ij}^{kp} + x_{ji}^{kp} = 1$.

Similar to the CP model, the sparse MIP model can be combined with objective functions to minimize makespan, minimize the total amount of deadheading, or minimize the weighted completion times of the priority classes:

$$MIP_{mkspn} : minimize\ T \tag{2}$$

$$\text{s.t. } T \geq \sum_{p \in \mathscr{P}} \sum_{(v_i,v_j) \in A'} c_{ij} y_{ij}^{pk} + (c_{ij}^{serv} - c_{ij}) x_{ij}^{pk} \qquad \forall k \in K \tag{3}$$

$$MIP_{deadh} : minimize \sum_{(v_i,v_j) \in A} \sum_{p \in \mathscr{P}} c_{ij} (y_{ij}^{pk} - x_{ij}^{pk}) \tag{4}$$

$$MIP_{weighted} : minimize \sum_{p \in \mathscr{P}} \alpha_p T_p^{prior} \tag{5}$$

$$\text{s.t. } T_p^{prior} \geq t_k^p \qquad \forall k \in K, p \in \mathscr{P} \tag{6}$$

$$t_k^p = t_k^{p-1} + \sum_{(v_i,v_j) \in A'} c_{ij} y_{ij}^{pk} + (c_{ij}^{serv} - c_{ij}) x_{ij}^{pk} \qquad p = 1, \ldots, P+1 \tag{7}$$

$$t_k^0 = 0 \qquad \forall k \in K \qquad (8)$$

To strengthen the sparse model, we added the following valid inequalities:

$$\sum_{k \in K} \sum_{p \in \mathscr{P}} \sum_{(v_i, v_j) \in \delta(S)} y_{ij}^{pk} \geq 2 \left\lceil \frac{D(S)}{Q} \right\rceil \qquad S \subseteq V \setminus I \qquad (9)$$

$$\sum_{k \in K} \sum_{p \in \mathscr{P}} \sum_{(v_i, v_j) \in \delta(S)} y_{ij}^{pk} \geq 1 + \sum_{e \in \delta_R(S,G)} \ell_{ij} \qquad S \subset V, S \text{ is } R-odd \qquad (10)$$

In Constraints (9), $Q = \max_{k \in K} q^k$ is the maximum vehicle capacity, and the resource demand $D(S)$ of a partition $S \subseteq V \setminus I$ is defined as $D(S) = \sum_{e \in E_R(S,G) \cup \delta_R(S,G)} \ell_{ij} q_{ij}$. Constraints (9) are known as Capacity inequalities and Constraints (10) as R-odd cuts. Separation algorithms for both inequalities are described in Belenguer and Benavent (2003); Ghiani et al. (2001).

5.2. GVRP MIP model

Analogous to the CP model (Section 4), SPRP can be solved as a GVRP through MIP, using the Single Commodity formulation for asymmetric GVRPs by Bektaş et al. (2011). This model, originally designed for homogeneous vehicles, can be straightforwardly modified to accommodate a heterogeneous fleet of vehicles. The resulting model employs binary z_{uv}^k variables to record whether vertex u is visited immediately prior to vertex v by vehicle $k \in K$, and continuous f_{uv}^k variables ($0 \leq f_{uv}^k \leq Q^k$) to track the remaining capacity of the vehicle traversing arc (u, v). The GVRP model can be linked to the sparse arc routing models from Belenguer and Benavent (1998, 2003); Ghiani et al. (2001) by adding appropriate channeling constraints between the variables in both models. As such, valid inequalities identified for the sparse models can also be used to strengthen the GVRP formulation.

Scalability of GVRP based MIP models is typically limited due to their vast number of variables: GVRP based models have $O(|K||A \cup E|^2)$ variables, whereas sparse models only have $O(|K||A \cup E|)$ variables. Reduced cost based fixing and filtering can be employed to reduce the number of variables in the GVRP model. Here we rely on the observation that, in an optimal solution, many z_{uv}^k variables attain the value zero, especially when $j(u)$ to $i(v)$ are far apart. Moreover, whenever every feasible solution in which some vehicle deadheads along path P_{ij}, is more expensive than some incumbent solution, we can filter out these solutions by fixing all z_{uv}^k variables having $j(u) = i, i(v) = j$ to zero. In our implementation, we use the LP relaxation of the sparse model by Ghiani et al. (2001), strengthened with Capacity inequalities and R-odd cuts, to perform the variable filtering. Through the LP we compute a valid lower bound on the cost of a solution in which some vehicle is *forced* to deadhead path P_{ij}. If this bound is larger than the cost of the incumbent, we can assume that no vehicle deadheads path P_{ij}. The filtering power of this procedure increases when (1) $c(P_{ij})$ is large, (2) the LP relaxation yields a strong lower bound, and (3) the incumbent solution is near optimal.

6. Turn restrictions

All routes must account for forbidden or, in case of large snowplows, physically impossible turns. To this extent, the routing graph must explicitly model turns and driving

directions. An advantage of modeling turns explicitly is that we can both forbid turns (hard constraint), and penalize turns (soft constraint). Routes with frequent U-turns are undesirable from a plowing perspective, due to the limited maneuverability of large snowplows. By adding turn-cost penalties for U-turns in the objective function, the quality of the routes increases considerably. Strictly prohibiting U-turns is not an option, due to the presence of dead-end streets.

Related works often compute deadheading paths through shortest path computations between a pair of vertices u,v in the routing graph $G(V, E, A)$. This however leads to a large number of U-turns. An example is given in Figure 2. A snowplow which is plowing consecutively segments (e, b) and (f, a) would be forced to make a U-turn if the deadheading path corresponds to the shortest path from b to f, because the shortest path algorithm is oblivious to the vehicle's driving direction when entering vertex b, or leaving vertex f. In this section, we discuss how we compute shortest paths while taking vehicle driving directions into account. E.g. for the example in Figure 2, we would like the algorithm to return the blue route, unless the length of this route is significantly larger than the sum of turn penalties incurred by the orange route. The costs of the shortest path between directed segments (u, v) and (w, x), can be directly used in the weight function $w(a)$ described in Section 3.2.

Any turn (u, v, w) is composed of two directed edges (u, v) and (v, w); a U-turn is a special case with $u = w$. To explicitly model turns, a line graph transformation is performed on the routing graph (see e.g. Winter (2002)). More precisely, first the mixed routing graph $G(V, E, A)$ is transformed into a directed graph $G(V, A')$ by replacing every undirected edge $e \in E$ by two directed arcs in opposing directions, i.e. $A' = A \cup \{(i, j), (j, i) | (i, j) \in E\}$. Next a line graph transformation is performed to obtain a directed line graph L_G. Each vertex in L_G corresponds with an arc in A'. Two vertices in L_G are connected by an arc if and only if for their corresponding arcs (u, v) and (i, j) in A' it holds that $v = i$. Finally, an augmented graph L'_G is created by adding to L_G a vertex for every depot in I. A vertex in L'_G corresponding with a depot $u \in I$ is connected with an arc to a vertex in L'_G corresponding with an edge (i, j) in G if $u = i$. Similarly, an arc exists from a vertex in L'_G corresponding with an edge (i, j) to a vertex corresponding with a depot $u \in I$ if $j = u$. An example of a line graph transformation for a T-intersection is provided in Figure 3. In summary, $L'_G(V_L, A_L)$ has vertex set $V_L = I \cup A'$ and arc set $A_L = \{((i, j), (u, v)) \in A' \times A' : j = u\} \cup \{(i, (u, v)) \in I \times A' : i = u\} \cup \{((i, j), u) \in A' \times I : j = u\}$.

For each arc $a \in A_L$, a weight $w(a) : A \to \mathbb{R}_0^+$ is specified as follows:

$$w(a) = \begin{cases} 0, & \text{if } a = (u, (i, j)) \in I \times A' \\ c_{ij} + \sigma_{ijv}, & \text{if } a = ((i, j), (u, v)) \in A' \times A' \\ c_{ij}, & \text{if } a = ((i, j), u) \in A' \times I \end{cases} \tag{11}$$

where $\sigma_{ijv} \geq 0$ is the penalty incurred when making the corresponding turn (i, j, v). In this work, only the arcs in $\{((i, j), (u, v)) \in A' \times A' : j = u \wedge i = v\}$ incur a non-negative U-turn penalty. To prohibit a turn (u, v, w), the arc $((u, v), (v, w))$ is removed from A_L.

Since L'_G is a simple, directed, weighted graph, shortest path computations can be performed using traditional shortest path algorithms (e.g. Dijkstra shortest path). Moreover, by construction, shortest paths in L'_G can be mapped one-to-one to paths in the original routing graph $G(V, A, E)$. Using graph L'_G, it is straightforward to compute feasible deadheading paths from a depot $i \in I$ to the start of an edge $(u, v) \in E_R \cup A_R$,

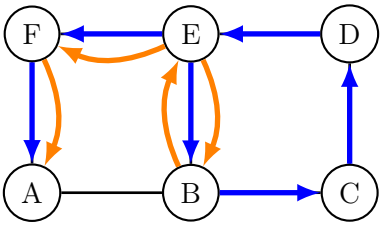
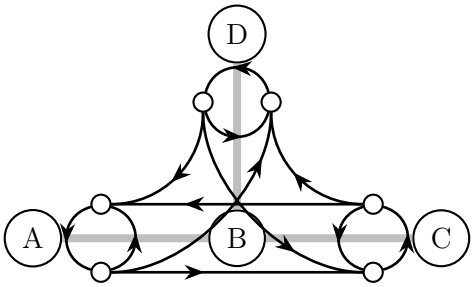
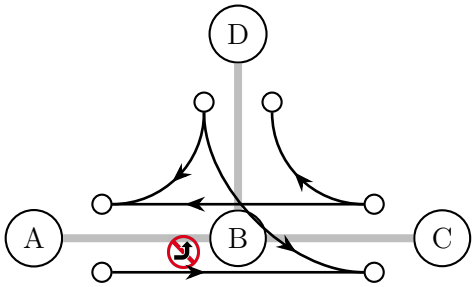


Figure 2.: Example where a vehicle must plow edges (e, b) and (f, a) consecutively. All edges have unit-length distance. A traditional shortest path computation from b to f would yield the orange route, which inevitably introduces a u-turn. A shortest path algorithm which is aware of the vehicle's driving direction could produce the blue route.



(a) T-intersection (grey) without turn-restrictions.



(b) T-intersection (grey) with turn-restrictions. The turn (a, b, d) and U-turns are prohibited.

Figure 3.: Conversion of an undirected routing graph into its corresponding directed line graph L_G (Winter, 2002). Each edge is decoupled into two directed arcs. In the line graph tranformation, each arc in the routing graph is represented by a vertex; the arcs in L_G represent potential turns.

from an edge $(u, v) \in E_R \cup A_R$ to a depot $i \in I$, or between two segments which are plowed consecutively.
Finally, notice that the size of L'_G grows polynomial in the size of $G(V, A')$. L'_G has $|A'| + |I|$ vertices. The number of edges $|A_L|$ in L'_G is bounded by:

$$|A_L| = \sum_{i \in I} |\delta(i)| + \sum_{i \in V \setminus I} |\delta^-(i)| \times |\delta^+(i)| < |I||V| + |V|^3 \tag{12}$$

Since routing graphs are typically sparse graphs, the size of L'_G does not become prohibitively large, even for relatively large routing graphs.

7. Computational evaluation

An extensive computational evaluation is performed to assess the quality of our models. First Section 7.1 describes the different data sources used in the experiments. Next, Section 7.2 explains how we extracted and preprocessed the data to produce resp. the routing and plowing graphs. A comparison of our CP approach (Section 4), against

the two MIP models (Section 5) as well as a commercial route planner for snow plows is provided in Section 7.3. To gain a better insight into the quality of the solutions obtained through the CP model, we compute lower bounds on the optimal solutions of our benchmark instances in Section 7.4. Finally the results from a test pilot held in Greenfield, a neighborhood in Pittsburgh, are discussed in Section 7.7.

7.1. Data

The city of Pittsburgh is partitioned into non-overlapping neighborhoods.[2] Historically, the same neighborhoods are used to delineate snow plow areas. Each neighborhood manages its own resources, and has its own set of snow plow routes; the number of routes required to service a neighborhood typically depends on its size. The city of Pittsburgh uses commercial software from RouteSmart (Routesmart, 2018) to compute routes for their snow plows. To facilitate a comparison between the city's snow plow schedules and the schedules produced by our approaches, the 3rd Division of Pittsburgh's Department of Public Works provided us with 39 routes which cover a total 19 neighborhoods. The city generates routes for each neighborhood independently. Vehicles are permitted to cross neighborhood boundaries for deadheading, but servicing is only permitted in the neighborhood for which the route was generated. All routes start and end at a vehicle depot, possibly located outside the route's service area. The routes provided by the city are designed for vehicles having a salt capacity of approximately 10 tons ($Q = 9000$) per vehicle and cover all primary and secondary roads maintained by the 3rd Division. Intermediate resupplying of the vehicles is currently not allowed. For the experimental evaluation we created 26 benchmark instances: one instance for each of the 19 neighborhoods provided by the city, plus 7 larger instances obtained by combining neighboring neighborhoods. The larger instances allow us to investigate whether potential savings can be incurred when 'artificial' neighborhood boundaries for snowplow areas are lifted. An overview of the instances is provided in Table 3. For each instance, Table 3 states which neighborhoods are covered, the number of arcs and edges, and the number of vehicles employed by the city to service the corresponding neighborhoods. Observe that for these instances, the number of undirected edges, compared to the number of directed arcs, is quite low. We note that capabilities to handle undirected edges have been explicitly added at the request of the Department of Public Works of Pittsburgh. Such edges play a more prominent role in servicing low priority 'service roads', but these roads were not included in the annotated data provided by the city. The last 3 columns of Table 3 provide some coarse information pertaining the total time and distance it takes to service all road segments, as well as the diameter of the plowing graph.

The *routing* graph $G(V, E, A)$ is derived from HERE maps data (HERE, 2018) and extends well beyond the neighborhood boundaries. The same data source is used by RouteSmart to compute the city's routes. For each road segment, HERE maps defines attributes such as: name, length (ft), travel speed (ft/s), number of lanes in each direction, and directionality. In addition, the city has annotated each road segment with information for plowing purposes, including whether a segment needs servicing or not, the priority class of each segment (primary, secondary, tertiary or unclassified), driving speed of the snowplows while servicing, salt demand[3], and maximum vehicle weight and width restrictions.

[2] http://gis.pittsburghpa.gov/pghneighborhoods/

[3] the salt demand of a segment is a linear function of the length of the segment

ID	**neighborhoods**	$\lvert E_R \cup A_R \rvert$	$\lvert E_R \rvert$	$\lvert K \rvert$	**serv. time**	**serv. (km)**	**⌀(km)**
0	WOK	74	0	1	0:40:31	4.8	2.2
1	TER	97	0	1	0:36:33	7.8	1.6
2	SWP	114	0	1	0:59:50	7.7	0.9
3	RSQ	116	0	1	0:22:53	5.5	1.2
4	NEW	117	0	1	1:02:28	10.1	4.4
5	HAY	151	0	2	0:55:50	13.7	2.1
6	UPH	170	0	1	1:07:48	12.6	1.3
7	BLF	173	0	1	1:03:25	12.8	2.7
8	COK	216	0	1	0:47:37	11.5	1.2
9	SOK	243	0	2	1:06:23	16.2	1.9
10	LIN	310	5	2	1:37:33	19.2	1.9
11	CRB	320	0	2	1:17:51	18.9	3.4
12	MHL	354	0	2	1:18:30	17.5	1.4
13	NOK	371	0	2	1:38:10	21.2	2.1
14	PBZ	426	0	2	1:40:13	24.4	2.2
15	CRB,BLF	493	0	3	2:21:16	31.7	4.3
16	HAZ	523	0	3	3:26:48	35.5	2.8
17	SQN	524	0	4	2:55:29	40.4	2.5
18	GNF	629	0	3	2:45:03	36.8	3.4
19	SQS	752	1	7	4:39:34	65.9	4.8
20	WOK,TER,NOK,COK	758	0	5	3:42:51	45.4	3.1
21	CRB,MHL,TER	771	0	5	3:12:54	44.2	3.4
22	SOK,COK,NOK	830	0	5	3:32:10	49	3.4
23	CRB,MHL,UPH	844	0	5	3:44:09	49	3.4
24	SQN,PBZ	950	0	6	4:35:42	64.8	4
25	GNF,HAZ	1152	0	6	6:11:51	72.4	3.7
	Total:	11478	6	74	9:23:22	739	69.3

Table 3.: Pittsburgh 3rd district instances

The experiments are, unless stated otherwise, performed with a homogeneous set of vehicles, each having a capacity of $Q = 9000$. Although the city of Pittsburgh employs a mixed fleet of vehicles, their current routes are all optimized for vehicles with fixed $Q = 9000$ capacity. To service a particular neighborhood, we employ the same number of vehicles as the city (see column $|K|$ in Table 3). In Section 7.3 we conduct an experiment where we vary the vehicle capacity and show that the vehicle size has limited influence on the performance of the models under consideration.

Intermediate resupplying of the vehicles, as proposed in Ghiani et al. (2001), is not allowed; the vehicles allocated to an area collectively have sufficient capacity to service the entire area. Resupplying comes with operational constraints. Each district typically has its own set of snow plows, and manages its own resources including a single salt depot. Vehicle routes are designed to start and end at the district's associated salt depot. For bookkeeping reasons, snow plows in Pittsburgh typically do not resupply at depots belonging to neighboring districts. Moreover, in our instances, resupplying a vehicle takes a considerable amount of time, as the vehicle needs to drive to the nearest depot, reload and return to the service area. Consequently, from an optimization perspective, there is little benefit in resupplying unless the trucks collectively do not have enough salt to service an area.

As a final remark, it is important to note that we use the exact same data for plowing and routing as RouteSmart: we use the same data for routing, and we plow exactly the same street segments with the same frequency. Consequently, we can establish a one-to-one comparison with the city's routes.

7.2. Data preprocessing

Prior to running any computations, preprocessing of the raw routing data is performed. The purpose of this preprocessing step is twofold: to eliminate inconsistencies in the data, and to remove redundant data. Using data provided by the city, we first extract the plowing graph from the the HERE maps routing data which is stored in a GIS file format. To compute the routing graph (Figure 4), we calculate the smallest bounding box (geographical area) containing the plowing graph and the vehicle depot. Next we enlarge the box by 15% and compute the set of intersections V that lie within the enlarged box. The routing graph $G(V, E \cup A)$ is the graph induced by V. An example of a plowing graph and corresponding routing graph are given in Figure 6a, 6c. Each vertex $v \in V_R$ in graph G_R has degree at least one. In addition, the following relations hold: $V_R \subset V, A_R \subset A, E_R \subset E$. The close-up in Figure 5b shows the relevance of this relation: the plowing graph ends at a neighborhood border, but the routing graph extends beyond this border. By expanding upon the plowing graph, the number of U-turns required in the solution can be drastically reduced.

Each of the models discussed in Sections 4,5 requires a distance matrix. Computing distance matrices on large road networks is non-trivial. Therefore, it is beneficial to the keep the routing graph as small as possible. To reduce the size of the routing graph, we first remove all but the largest strongly connected component containing the entire plowing graph. Enumerating all strongly connected components in G is efficiently performed using Gabow's algorithm (Gabow, 2000). Next we proceed by removing all *non-essential* road segments from the routing graph. A road segment $(u, v) \in A \cup E$ is considered non-essential if (1) it is not contained in G_R and (2) no shortest path P_{ij} connecting vertices $i, j \in V_R \cup I$ traverses road segment (u, v). In other words, a non-essential road segment (u, v) does not need servicing, and it will never be used in any deadheading path connecting two plow segments or a plow segment and a depot. Once all non-essential edges have been filtered from the routing graph, any isolated vertices are also removed. The resulting graph is typically sparse and non-planar. Figures 6b (before) and 6c (after) clearly show the impact of this reduction. To complete the preprocessing phase, we perform the transformations described in Section 6 to incorporate turn restrictions and turn penalties directly into the routing graph. The turn cost for U-turns is set to 3 minutes. To efficiently compute an all-pair shortest path matrix for a subset of vertices $S = A' \cup I$ in the augmented line graph L'_G, we implemented a modified version of Johnsons shortest path algorithm. This version terminates as soon as all paths covering the vertices in S have been discovered, and skips phase 1 of Johnsons algorithm since the graph does not contain edges with negative weights.

The routing and plowing graphs are implemented using the graph library JGraphT 1.3.0 (Michail et al., 2020). To reduce memory utilization, lanes in the same direction belonging to the same road segment are represented as a single arc in the graph (as opposed to individual arcs for every lane). The number of lanes in each direction is stored as an attribute on each arc. Shortest path computations are performed using an efficient implementation of Dijkstra's shortest path algorithm; additional performance improvements could be obtained by implementing Highway and Contraction Hierarchies.

Figure 4.: Visualization of a routing graph on a map. The red triangles mark one-way roads. To simplify the visual, we only depict one lane per road.

7.3. Computational Results

In this section we compare the performance of our CP model (Section 4) with 4 alternative solution approaches: the two MIP models (Section 5), the Late Acceptance heuristic from Kinable et al. (2016) and the routes produced by the commercial software RouteSmart used by the city. All computations are performed on a system with an Intel Core i7-4790 CPU, 3.60GHz with 15Gb internal memory. MIP models are solved with ILOG CPLEX v12.8.0 with default parameters; the CP models with IBM CP Optimizer v12.8.0. The inference level of sequence constraints in CP Optimizer has been set to 'extended', and depth-first search with restarts is used as the search algorithm. We note that during the search process, CP Optimizer maintains a lower bound on the objective value which relies in part on an automatic linear programming relaxation of the scheduling constraints (Laborie and Rogerie, 2016). Computation times are limited to 1h per instance, using 4 threads. All models are warm-started with an initial solution produced with the constructive (deterministic) heuristic from (Kinable et al., 2016). When optimizing the weighted completion times of the different priority classes, we set the weight of class $p \in \mathbb{P}$ equal to $10^{|\mathbb{P}|-p}$, i.e. the highest priority class $p = 1$ would get the highest weight.

Figure 7 compares CP_{mkspn} against the sparse model from Perrier et al. (2008) (MIP_{mkspn}, Section 5.1) and the Late Acceptance heuristic (LA Heuristic) from Kinable et al. (2016). The color of each data point corresponds to the instance size measured in the total number of plow jobs. The LA Heuristic is executed with a list length of 1280, and terminates after 1h or when no more improving moves have been observed for 1M iterations. The LA heuristic results are averaged over 5 independent invocations per instance. As can be observed from Figure 7, for the smaller instances (<400 jobs), the sparse MIP model and the CP model have very comparable performance profiles. MIP takes a small lead over CP for some of the smallest instances. However, for the larger instances (>500 jobs) CP drastically outperforms the sparse MIP model. It seems that the vast number of turn variables n^{kp}_{uvw} deteriorates the performance of this model. The LA heuristic which was initially designed for a related problem which included fuel as a resource but excluded turn restrictions, is outperformed by the CP formulation as well: except for one instance,

(a) The plowing graph, marked by the blue rectangle, is a subgraph of the routing graph, delineated by the larger red rectangle.

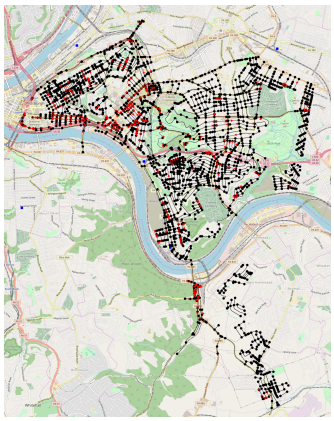
(b) The plowing graph is limited by an artificial district border on the right (dashed). The routing graph (not shown here) however expands beyond the border, thereby drastically reducing the number of U-turns needed if the streets would end at the district border.

Figure 5.: Relation between the routing graph and plowing graph.

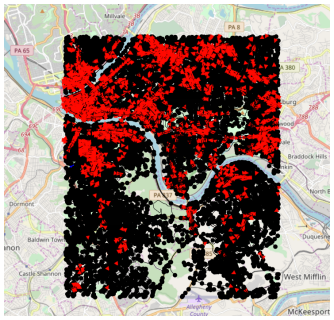
CP yields better solutions.

Next to experiments with the sparse MIP model, we also conducted a number of tests with the dense model (Section 5.2) based on the graph transformation. For all but the smallest instance, our test system ran out of memory while constructing the MIP model. Consequently we must conclude that this type of model is not suitable for practical instances. As elaborated in Section 5.2 we also attempted to reduce the number of variables in these models through variable fixing. This approach was however not successful. We conducted a simple experiment in which we attempted to assess the filtering power of this approach. Even when strong upper and lower bounds were used, very few variables could be filtered. Typically, routes in Capacitated Arc Routing Problems are quite long, covering many segments while deadheading or servicing. The marginal cost increase incurred when fixing a deadheading path P_{ij} was often not enough to determine whether the corresponding variables could be removed from the model. For larger instances, the filtering power reduced even further due to a larger gap between the lower and upper bounds available for such instances.

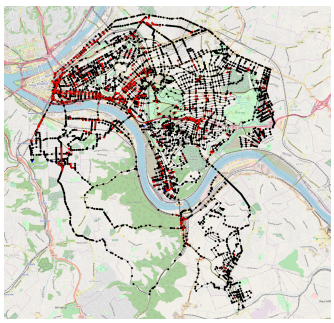
In Figure 10a, we compare CP_{mkspn} against the city's plowing schedules, which are produced with the popular, commercial routing software RouteSmart. It is unknown which objective function RouteSmart uses to optimize its routes. Consequently Figure 10a compares for each schedule the makespan (blue squares), the completion time of the first priority class (P_1, red triangles) and the completion time of the second priority class (P_2, green circles). As can be observed, CP_{mkspn} significantly outperforms RouteSmart on all aspects: all schedules have a shorter makespan, and complete the secondary priority class faster. For all but one schedule, CP_{mkspn} completes servicing the primary roads faster. When simply comparing the makespan of the schedules, the routes produced by our CP approach are 3%-156% (average 33%)



(a) Plowing graph. The red triangles mark one-way roads.



(b) Routing graph *before* preprocessing.



(c) Routing graph *after* preprocessing. Only a subset of the routing edges must be serviced.

Figure 6.: Plowing and routing graph of the 3rd plowing district in Pittsburgh.

shorter than the routes produced by the city.

Figure 8 explorers the relation between different problem features, e.g. the number of vehicles and their capacities, and the performance of the CP and sparse MIP models. Figure 8a shows the solutions of the instances from Table 3 with varying number of vehicles $|K|$ with fixed capacity $Q = 9000$ whereas in Figure 8b Q is changed while keeping $|K|$ fixed to the values in Table 3. Computation times are limited to 30 minutes per instance for a given value of Q and $|K|$. The x-axis shows the number of jobs in an instance. The y-axis depicts the *ratio* between the CP objective and the MIP objective: values *below* 1 indicate that CP outperformed MIP, and vice versa. Figure 8a clear shows that the CP model scales significantly better than the sparse MIP model in the number of vehicles $|K|$. In contrast, the vehicle capacity Q (Figure 8b) has no observable impact on the models' performance: increasing or decreasing the vehicle capacity does not yield favorable circumstances for either the MIP or CP model.

7.4. Bounds

Thus far, the experiments were focused on the comparison of primal methods. To obtain insight in the quality of the solutions, we computed bounds for each of the instances using 4 lower bounding procedures. *CP Bound*, resp. *MIP Bound* are the lower bounds obtained when solving resp. the CP model (Section 4) and MIP model (Section 5.1) and can be queried directly through the corresponding solvers. A third bounding procedure is the bound obtained while solving the MIP model from Section 5.1 but without the turn variables. Finally, the fourth procedure solves a simple bin packing problem to optimality: the plow jobs are assigned to the vehicles while taking vehicle capacity into account and minimizing the maximum duration of jobs assigned to a single vehicle. This last procedure is the weakest procedure because it completely ignores deadheading, but is also the cheapest to compute.
Figure 9 compares these different bounds, as well as the available upper bounds. The 26 benchmark instances, sorted based on size (number of jobs) in ascending order, are

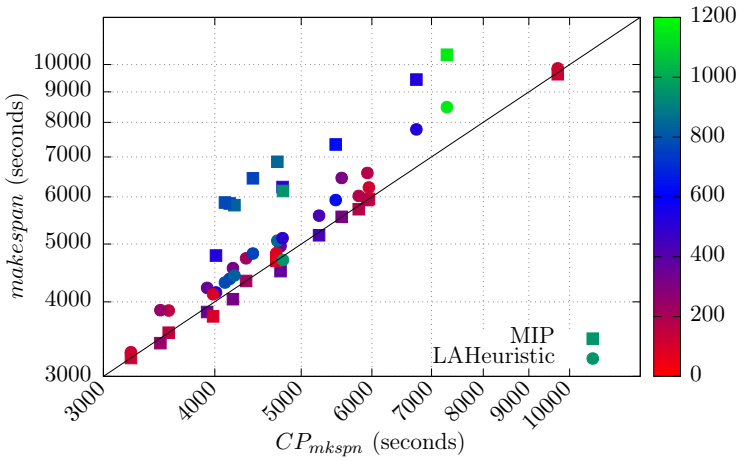
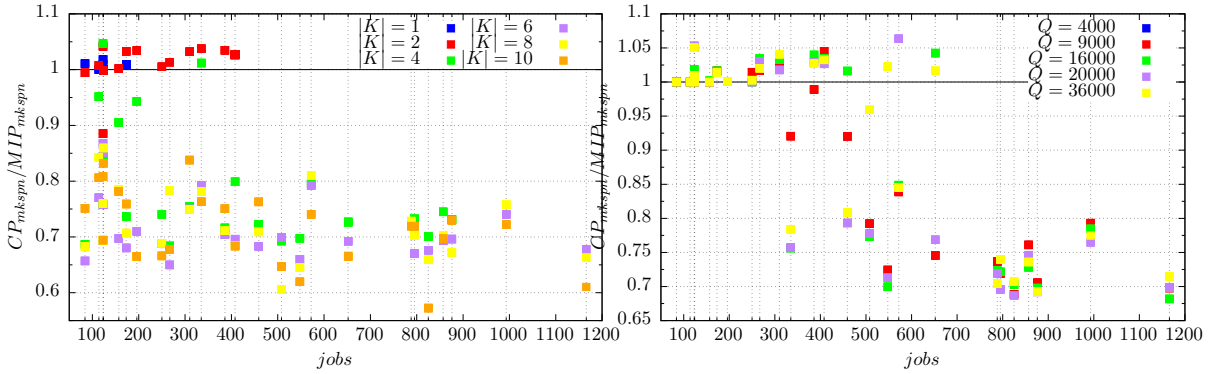


Figure 7.: CP_{mkspn} vs MIP-two index and the LAHeuristic. The colors correspond to the instance size (number of plow jobs). For small instances, MIP slightly outperforms CP, but for the larger instances CP significantly outperforms MIP.



(a) Number of vehicles

(b) Vehicle capacity

Figure 8.: Impact of instance features on CP/MIP performance.

plotted on the x-axis. The IDs of the instances on the x-axis correspond with the IDs in Table 3. The base-line (0% gap) is the best bound (BB) taken over the above 4 lower bounds. All other lines are relative to the BB line. Below the x-axis are the two lower bounds derived from the CP model and the sparse MIP model with turn-cost penalties. Their gap to BB is calculated as $-100\frac{BB-LB}{BB}$. Above the x-axis are the upper bounding methods, calculated as $100\frac{UB-BB}{UB}$. When comparing the Lower Bounding methods it can be observed that in most cases, the MIP model provides the strongest bounds. For the largest instances (19-21, 23-25), the MIP bound is surpassed by the three other bounding procedures.
Similar observations can be made for the upper bounding methods. For the smaller instances (0-9) CP and MIP obtain provably strong solutions, in some cases even optimal. For instances 10-14, the gap increases for both MIP and CP, but their performance remains very comparable. Starting from instance 15, CP starts to outperform MIP, as MIP is no longer able to improve over its warmstart solution. For the larger instances, the optimality gap becomes substantial (around 50%).
From Figure 9 it can be observed that for 5 instances, CP and MIP collectively find

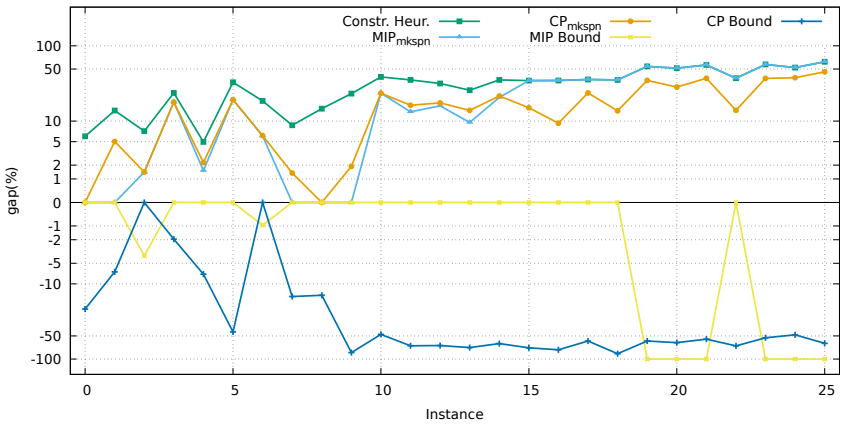


Figure 9.: Bound comparison

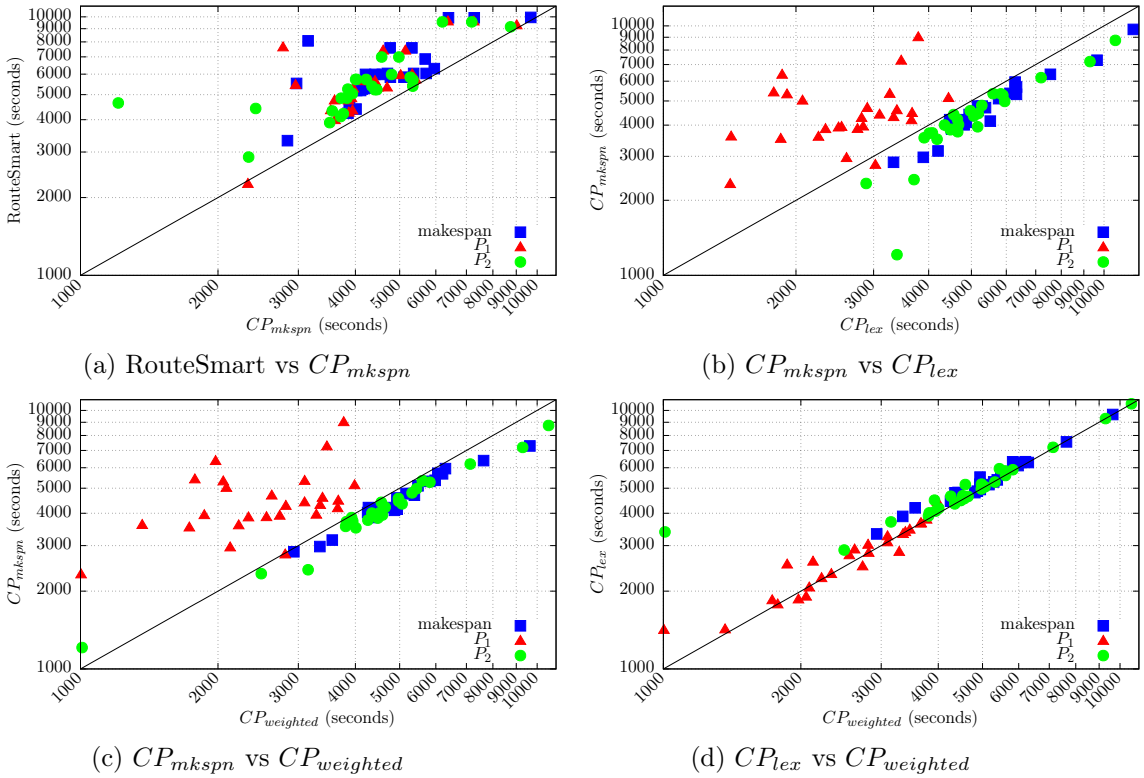
the optimal solutions (i.e some lower bound is equal to either the CP or MIP objective).When for these instances we compare the ratio between time servicing, and time deadheading, we find a 3:2 ratio, which perhaps is surprisingly inefficient. Apart from moving the depots closer to the plowing areas, it is not obvious how to further improve the vehicle efficiency.

7.5. Objective function comparison

In Figures 10b-10d we depict the results when the CP model is ran with different objective functions. Notice that the objective which minimizes total deadheading has been omitted from these figures. Although this is a common CARP objective (e.g. when comparing on synthetic benchmark data), our experiments revealed that this objective is not suitable to produce practical routes for our SPRP. For many instances, the resulting schedules were highly unbalanced: one vehicle would drive a very long, near perfect Eulerian cycle servicing as many streets as possible with minimal deadheading, whereas the other vehicles had very short routes servicing the remaining sections not covered by the first vehicle. Moreover, when minimizing the total amount of deadheading, the cost $\sum_{(i,j)\in E_R \cup A_R} c_{ij}^{serv}$ becomes a constant and is therefore ignored.
In Figure 10b, 10c, we minimize the completion time of the priority classes. Compared to the makespan schedules, we can observe a clear shift in objective values: the completion time of the most important class (primary roads) dramatically improves, but the completion of the second priority class (P_2) is delayed and overall schedules become slightly longer. Finally Figure 10d compares CP_{lex} with $CP_{weighted}$. Albeit both objectives produce very similar results, the length of the schedules produced by $CP_{weighted}$ appears to be slightly shorter.

7.6. District borders

Next we investigate the influence of the district borders on the plowing schedules. Recall from Section 7.1 that the plowing area of an instance is artificially confined by some district border. Would it be possible to improve the schedules when these borders



(a) RouteSmart vs CP_{mkspn}

(b) CP_{mkspn} vs CP_{lex}

(c) CP_{mkspn} vs $CP_{weighted}$

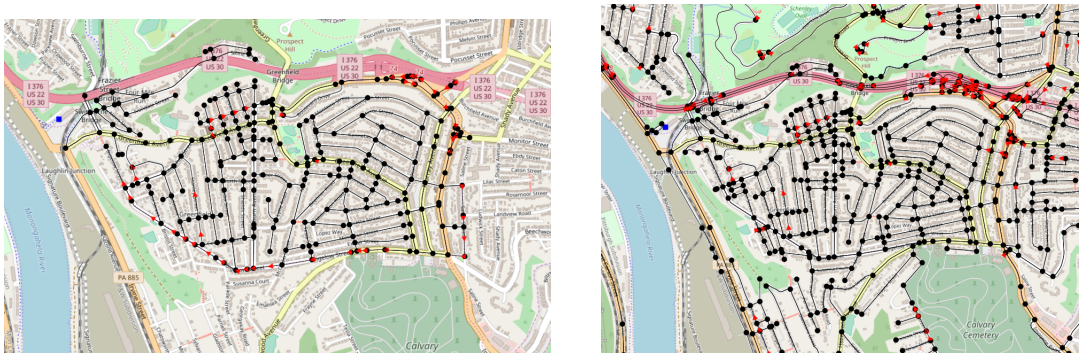
(d) CP_{lex} vs $CP_{weighted}$

Figure 10.: Computational evaluation.

are selected differently? To answer this question, we compared the schedules produced by the instances consisting of multiple neighborhoods (Table 3), to the schedules for each of the neighborhoods individually. Notice that the number of vehicles used for the merged instances, equals the sum of vehicles used for the individual neighborhoods. Consequently, to obtain a feasible schedule for a merged instance, we can simply combine the schedules of its individual neighborhoods. The results are reported in Table 4. For each instance, Table 4 reports the objective value (column CP_{mkspn}), the longest vehicle schedule when considering the neighborhoods individually (column max(mkspn)) and the percentage difference (column reduction). For example, the schedule of 'SOK,COK,NOK', has a makespan of 4208s versus a makespan of 2973s, 4013s, 4583s for SOK, COK and NOK respectively. The percentage difference (8.2%) between 4208s and $max(2973s, 4013s, 4583s) = 4583s$ is the amount of savings that can be realized when the routes are calculated for the 3 neighborhoods collectively. From Table 4, we can observe that savings in the range of 0-18% can be achieved, indicating that it would be beneficial for the city to re-evaluate its district boundaries for plowing purposes. Two instances in Table 4 which are the two largest instances in our dataset, have a negative reduction. This means that the solution produced by CP optimizer for the merged area is worse than the combined solution of the individual areas, indicating that CP is struggling to find high quality solution when the instances become too large.

instance	CP_{mkspn}	max(mkspn)	reduction (%)
SOK,COK,NOK	4208	4583	8.2
CRB,MHL,TER	4106	4026	-2.0
CRB,MHL,UPH	4701	5714	17.7
WOK,TER,NOK,COK	4414	4583	3.7
GNF,HAZ	7283	6394	-13.9
CRB,BLF	4765	5318	10.4
SQN,PBZ	4770	5109	6.6
Average:			9.3

Table 4.: Impact of districting on plow schedules



(a) Plowing graph (b) Routing graph

Figure 11.: Greenfield

7.7. Greenfield Pilot Test

To verify the feasibility of the generated routes, a pilot test was held in a residential neighborhood called Greenfield (Figure 11, and GNF in Table 3). For this neighborhood, we generated 3 routes which were driven by car. Driving instructions for each of the routes were provided to the driver through a custom-made application which issues turn-by-turn instructions using a built-in GPS. The goal of this pilot test was to (1) validate the routes and (2) to test the application which issues the turn-by-turn instructions. Initial tests revealed that some of the routes contained roads that were not traversable by large snow-plows. Fixing a number of data annotation errors which incorrectly marked some roads as passable by all vehicles resolved this issue and made the routes feasible to drive. Our tests also confirmed the necessity of u-turn penalties, as discussed in Section 6, as disabling these penalties produced routes with too many u-turns, which made the routes virtually impossible to drive with a large vehicle.
In Figure 12, three sets of routes for Greenfield are compared; a summary of the results is provided in Table 5. The first set, *RouteSmart* are the routes provided by the city. The 2 remaining sets are the routes we generated using CP_{mkspn} and $CP_{weighted}$. The graph depicts the plowing progress over the course of time: at time $t = 0$, nothing has been plowed, whereas at the end of the schedule 100% of the area has been serviced. The purple and the green lines show for each point in time the percentage of serviced roads belonging to the first resp. second priority class. When comparing the results it is obvious that the best schedules are produced using the CP approaches. Naturally, the shortest schedule is obtained when optimizing towards the makespan objective. However, at the expense of a slightly longer schedule (∼10 minutes), the weighted search

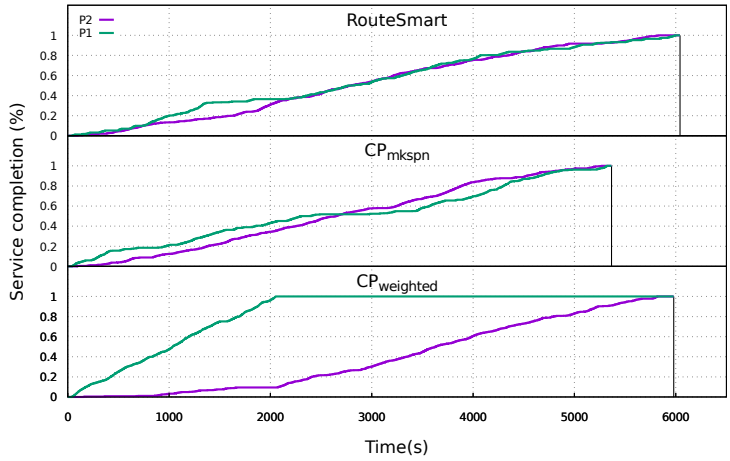


Figure 12.: Greenfield test pilot, comparison of different solution approaches

	Makespan	Completion P_1	Completion P_2	U-Turns
RouteSmart	01:40:42h	6002s	5836s	18
CP_{mkspn}	**01:29:26h**	5326s	**5274s**	**17**
$CP_{weighted}$	01:39:40h	**2054s**	5824s	**17**

Table 5.: Greenfield solution comparison

finds a schedule which clears the primary roads 2.6 times faster than the makespan schedule and 2.9 times faster than the city's schedule.

8. Conclusion

To solve a realistic Snow Plow Routing Problem, an efficient Constraint Programming formulation has been proposed which incorporates many common side-constraints frequently encountered in problems dealing with Winter Road Maintenance. To assess the performance of the CP formulation, two existing, alternative formulations based on conventional MP models for the SPRP have been used for comparison: one formulation depends on a sparse arc routing model, whereas the other formulation relies on a graph transformation to solve the problem as a Generalized VRP. Computational results on data provided by the city of Pittsburgh revealed that the CP model scales better than the alternative MIP formulations. Moreover, when comparing against the city's plowing routes produced by commercial software, we showed that our plowing schedules are 3%-156% shorter while satisfying all necessary routing constraints. In summary we can conclude that our CP model offers a viable alternative to existing approaches.

The routing methodology presented in this work is part of a larger initiative to develop an adaptive system for snow plow optimization and management. Future work will add a real-time component to the system to determine how vehicle schedules have to be modified when deviation of their original routes is necessary, for instance when encountering road obstructions or equipment failures.

Finally, a number of future research directions can be identified. Presently, the city com-

putes routes for each of its neighborhoods seperately. Our experiments revealed that significant savings can be realized when routes are calculated for multiple neighborhoods simultaneously. However, from a computational perspective the plowing graphs cannot be too large either, so some districting is required. Therefore, it would be beneficial to redefine the plowing areas while taking into account the locations of the supply depots. Some work in this area has already been conducted in Butsch et al. (2014). Another potential research avenue involves further preprocessing of the routing and plowing graphs. Currently the plowing graph contains many short road segments, e.g. turn lanes, which are currently treated as separate entities. Merging these sections to form larger road segments could potentially reduce the size of the problem instances considerably. A last potential research direction addresses fleet management problems. The current work assumes that the vehicle fleet is fixed a-priori. In practice the city relies on several freelance drivers, and often has one or more vehicles under repair. It follows that the exact number of vehicles and drivers available at a given point in time is not perfectly predictable. Therefore it is valuable to determine the ideal fleet composition for a given area, but also to compute a number of backup scenarios in case the ideal composition is not available.

Acknowledgements

Hidden for double blind review

References

Baldacci, R. and Maniezzo, V. (2006). Exact methods based on node-routing formulations for undirected arc-routing problems. *Networks*, 47(1):52–60.

Baptiste, P., Le Pape, C., and Nuijten, W. (2001). *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*, volume 39 of *International Series in Operations Research & Management Science*. Springer, New York, NY.

Bartolini, E., Cordeau, J.-F., and Laporte, G. (2013). Improved lower bounds and exact algorithm for the capacitated arc routing problem. *Mathematical Programming*, 137(1):409–452.

Bektaş, T., Erdoğan, G., and Røpke, S. (2011). Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science*, 45(3):299–316.

Belenguer, J. and Benavent, E. (1998). The capacitated arc routing problem: Valid inequalities and facets. *Computational Optimization and Applications*, 10(2):165–187.

Belenguer, J. M. and Benavent, E. (2003). A cutting plane algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 30(5):705 – 728.

Bessiere, C. (2006). Constraint Propagation. In Rossi, F., van Beek, P., and Walsh, T., editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, chapter 3, pages 29–84. Elsevier.

Blandford, B., Lammers, E., and Green, E. (2018). Snow and ice removal route optimization in kentucky. *Transportation Research Record*, 2672(45):294–304.

Bode, C. and Irnich, S. (2012). Cut-first branch-and-price-second for the capacitated arc-routing problem. *Operations Research*, 60(5):1167–1182.

Brandão, J. and Eglese, R. (2008). A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 35(4):1112 – 1126.

Butsch, A., Kalcsics, J., and Laporte, G. (2014). Districting for arc routing. *INFORMS Journal on Computing*, 26(4):809–824.

Corberán, A. and Laporte, G. (2014a). *Arc Routing: Problems, Methods, and Applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Corberán, A. and Laporte, G. (2014b). *Arc Routing: Problems, Methods, and Applications*, chapter 9, pages 183–221. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Corberán, A. and Prins, C. (2010). Recent results on arc routing problems: An annotated bibliography. *Networks*, 56(1):50–69.

Dell'Amico, M., Díaz, J. C. D., Hasle, G., and Iori, M. (2016). An adaptive iterated local search for the mixed capacitated general routing problem. *Transportation Science*, 50(4):1223–1238.

Dror, M. (2000). *Arc Routing Theory, Solutions and Applications*. Springer US, Boston, MA.

Environmental Protection Agency (1999). Storm water management fact sheet, minimizing effects from highway deicing. Technical Report EPA 832-F-99-016, Office of Water Washington, D.C. http://water.epa.gov/scitech/wastetech/upload/2002_06_28_mtb_ice.pdf.

Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M. P. d., Reis, M., Uchoa, E., and Werneck, R. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511.

Gabow, H. N. (2000). Path-based depth-first search for strong and biconnected components. *Information Processing Letters*, 74(3-4):107–114.

Ghiani, G., Guerriero, F., Laporte, G., and Musmanno, R. (2004). Tabu search heuristics for the arc routing problem with intermediate facilities under capacity and length restrictions. *Journal of Mathematical Modelling and Algorithms*, 3(3):209–223.

Ghiani, G., Improta, G., and Laporte, G. (2001). The capacitated arc routing problem with intermediate facilities. *Networks*, 37(3):134–143.

Golden, B. L. and Wong, R. T. (1981). Capacitated arc routing problems. *Networks*, 11(3):305–315.

Gundersen, A. H., Johansen, M., Kjær, B. S., Andersson, H., and Stålhane, M. (2017). Arc routing with precedence constraints: An application to snow plowing operations. In Bektaş, T., Coniglio, S., Martinez-Sykora, A., and Voß, S., editors, *Computational Logistics*, pages 174–188, Cham. Springer International Publishing.

HERE (2018). Here map data. https://www.here.com/en/products-services/map-content/here-map-data.

Hertz, A. and Mittaz, M. (2001). A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. *Transportation Science*, 35(4):425–434.

Hierholzer, C. and Wiener, C. (1873). Über die möglichkeit, einen linienzug ohne wiederholung und ohne unterbrechung zu umfahren. *Mathematische Annalen*, 6(1):30–32.

Irnich, S. (2008). Solution of real-world postman problems. *European Journal of Operational Research*, 190(1):52 – 67.

Kilby, P. and Shaw, P. (2006). Chapter 23 - vehicle routing. In Rossi, F., van Beek, P., and Walsh, T., editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 801 – 836. Elsevier.

Kinable, J., van Hoeve, W.-J., and Smith, S. F. (2016). Optimization models for a real-world snow plow routing problem. In Quimper, C.-G., editor, *Integration of AI and OR Techniques in Constraint Programming*, pages 229–245, Cham. Springer International Publishing.

Laborie, P. and Rogerie, J. (2008). Reasoning with conditional time-intervals. In *FLAIRS Conference*, pages 555–560.

Laborie, P. and Rogerie, J. (2016). Temporal linear relaxation in IBM ILOG CP Optimizer. *Journal of Scheduling*, 19(4):391–400.

Laborie, P., Rogerie, J., Shaw, P., and Vilím, P. (2009). Reasoning with conditional time-intervals. part ii: An algebraical model for resources. In *FLAIRS Conference*.

Laborie, P., Rogerie, J., Shaw, P., and Vilím, P. (2018). Ibm ilog cp optimizer for scheduling. *Constraints*, 23(2):210âĂŞ250.

Letchford, A. (1997). *Polyhedral Results for Some Constrained Arc-Routing Problems*. PhD thesis, Department of Management Science, Lancaster University, Lancaster, UK.

Letchford, A. N. and Oukil, A. (2009). Exploiting sparsity in pricing routines for the capaci-

tated arc routing problem. *Computers & Operations Research*, 36(7):2320 – 2327.

Longo, H., de Aragão, M. P., and Uchoa, E. (2006). Solving capacitated arc routing problems using a transformation to the CVRP. *Computers & Operations Research*, 33(6):1823 – 1837.

Michail, D., Kinable, J., Naveh, B., and Sichi, J. V. (2020). JgraphtâĂŤa java library for graph data structures and algorithms. *ACM Trans. Math. Softw.*, 46(2).

Pearn, W.-L., Assad, A., and Golden, B. L. (1987). Transforming arc routing into node routing problems. *Computers & Operations Research*, 14(4):285 – 288.

Pecin, D. and Uchoa, E. (2019). Comparative analysis of capacitated arc routing formulations for designing a new branch-cut-and-price algorithm. *Transportation Science*, 53(6):1673–1694.

Perrier, N., Langevin, A., and Amaya, C.-A. (2008). Vehicle routing for urban snow plowing operations. *Transportation Science*, 42(1):44–56.

Perrier, N., Langevin, A., and 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., and 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(1):239 – 262.

Perrier, N., Langevin, A., and 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(1):211 – 257.

Perrier, N., Langevin, A., and 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(1):258 – 294.

Polacek, M., Doerner, K. F., Hartl, R. F., and Maniezzo, V. (2008). A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. *Journal of Heuristics*, 14(5):405–423.

Quirion-Blais, O., Langevin, A., and TrÃľpanier, M. (2017). A case study of combined winter road snow plowing and de-icer spreading. *Canadian Journal of Civil Engineering*, 44(12):1005–1013.

Régin, J.-C. (2011). Constraint Propagation. In Milano, M. and van Hentenryck, P., editors, *Hybrid Optimization – The Ten Years of CPAIOR*, pages 63–134. Springer.

Routesmart (2018). Routesmart technologies - the world's most intelligent routing system. https://www.routesmart.com/.

Rubin, J., Garder, P. E., Morris, C. E., Nichols, K. L., Peckenham, J. M., McKee, P., Stern, A., and Johnson, T. O. (2010). Maine winter roads: Salt, safety, environment and cost. Technical report, Margaret Chase Smith Policy Center, University of Maine. http://umaine.edu/mcspolicycenter/files/2010/02/Winter-Road-Maint-Final.pdf.

Salazar-Aguilar, M. A., Langevin, A., and Laporte, G. (2012a). Synchronized arc routing for snow plowing operations. *Computers & Operations Research*, 39(7):1432–1440.

Salazar-Aguilar, M. A., Langevin, A., and Laporte, G. (2012b). Synchronized arc routing for snow plowing operations. *Computers & Operations Research*, 39(7):1432 – 1440.

Stringer, S. M. (2015). The slippery cost slope of ice and snow removal in new york city. Technical report, Office of the New York City Comptroller, Municipal Building, 1 Centre Street, 5th Floor, New York, NY 10007. https://comptroller.nyc.gov/wp-content/uploads/documents/The_Slippery_Cost_Slope_of_Ice_and_Snow_Removal_in_New_York_City.pdf.

Usberti, F. L., França, P. M., and França, A. L. M. (2013). Grasp with evolutionary path-relinking for the capacitated arc routing problem. *Computers & Operations Research*, 40(12):3206 – 3217.

Usman, T., Fu, L., and Miranda-Moreno, L. F. (2010). Quantifying safety benefit of winter road maintenance: Accident frequency modeling. *Accident Analysis & Prevention*, 42(6):1878 – 1887.

van Hoeve, W.-J. and Katriel, I. (2006). Global Constraints. In Rossi, F., van Beek, P.,

and Walsh, T., editors, *Handbook of Constraint Programming*, Foundations of Artificial Intelligence, chapter 6, pages 169–208. Elsevier.

Vidal, T. (2017). Node, edge, arc routing and turn penalties: Multiple problemsâĂŤone neighborhood extension. *Operations Research*, 65(4):992–1010.

Wang, J. and Liu, H. (2019). Snow removal resource location and allocation optimization for urban road network recovery: a resilience perspective. *J. Ambient Intell. Humaniz. Comput.*, 10(1):395–408.

Winter, S. (2002). Modeling costs of turns in route planning. *GeoInformatica*, 6(4):345–361.

Wøhlk, S. (2008). A decade of capacitated arc routing. In Golden, B., Raghavan, S., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 29–48. Springer US, Boston, MA.

Zhou, W. and Wu, B. (2018). Model for snow-fighting vehicle route planning considering deadheading restriction. In *Proceedings of the 2nd International Conference on Computer Science and Application Engineering*, CSAE âĂŹ18, New York, NY, USA. Association for Computing Machinery.