Optimization Models for a Real-World Snow Plow Routing Problem

J. Kinable[1,2], W-J. van Hoeve[2], and S. F. Smith[1]

[1] Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave Pittsburgh, PA 15213, USA
[2] Tepper School of Business, Carnegie Mellon University, 5000 Forbes Ave Pittsburgh, PA 15213, USA
jkinable@cs.cmu.edu, vanhoeve@andrew.cmu.edu, sfs@cs.cmu.edu

Abstract. In cold weather cities, snowstorms can have a significant disruptive effect on both mobility and safety, and consequently the faster that streets can be cleared the better. Yet in most cities, plans for snowplowing are developed using simple allocation schemes that while easy to implement can also be quite inefficient. In this paper we consider the problem of optimizing the routes of a fleet of snow plowing vehicles, subject to street network topology, vehicle operating restrictions, and resource (salt, fuel) usage and replenishment constraints. We develop and analyze the performance of three different optimization models: a mixed-integer programming (MIP) model, a constraint programming (CP) model, and a constructive heuristic procedure that is amplified by an iterative improvement search. The models are evaluated on a set of snow plow routing problems of various sizes, constructed using Open Streets map data of Pittsburgh PA. Experimental results are presented that illustrate the differential strengths and weaknesses of each model, and suggest an alternative hybrid solution approach.

1 Introduction

Each year, many northern cities face significant expenditures pertaining winter road maintenance. Snow removal constitutes a significant part of these costs. For example, the city of Pittsburgh (USA) spent a staggering $4.3M on consumable resources (salt, deicing chemicals), $3.3M on personnel, and $800K on equipment (vehicles, plows, maintenance) during last year's winter season (2014/2015). In addition to these direct costs, a number of indirect costs can also be identified. Slippery roads deteriorate driving conditions thereby increasing the number of traffic accidents. Extensive utilization of snow plows, salt and chemicals damage the roads, corrode cars and metal bridges, and have an overall negative impact on the environment. Consequently, any ability to optimize winter road maintenance and deicing operations offers significant opportunities to realize substantial savings, to improve mobility and to reduce societal and environmental impact (Salazar-Aguilar et al., 2012; Usman et al., 2010; Environmental Protection Agency, 1999; Rubin et al., 2010).
In this work we study the real-world snow plow routing problem (SPRP) faced

by the City of Pittsburgh PA where routes must be computed for a set of heterogeneous vehicles such that they collectively cover a geographical area, and comply with various resource constraints. Here, as in any snow plowing activity, each vehicle removes snow from the streets and simultaneously spreads a mixture of salt and chemicals for deicing purposes. Since each vehicle has only limited fuel and salt capacity, resources have to be periodically replenished. A number of resource depots are available throughout the city: these depots offer fuel, salt or both. The objective is to compute a schedule for each vehicle, which satifies resource constraints and minimizes the overall time it takes to clear all streets (i.e., the schedule makespan).

This work is part of a larger initiative to provide the city with an adaptive approach to snow plow route optimization and management. A route planning system is under development which will ultimately issue optimized turn-by-turn instructions to the vehicles in real-time during snow plowing operations, and dynamically revise these plans as unexpected events force changes. This paper lays the foundations for this project, by formally defining the problem and analyzing both exact (CP and MIP) and heuristic approaches for solving it. The heuristics presented are designed with scalability and adaptivity in mind, such that they can be adapted at a later stage of the project to modify schedules in response to dynamic events such as blocked roads, equipment problems and emergency requests.

The problem under consideration generalizes the well-known Chinese Postman Problem (Mei-Ko, 1962) and relates to other problems such as the Capacitated Arc Routing Problem (Eiselt et al., 1995a,b) and Resource Constrained Project Scheduling.Although an extensive amount of research has been devoted to road maintenance and snow control, only a limited number of works has studied snow plow routing with resource constraints. For an excellent literature overview pertaining winter road maintenance problems in general, and related solution approaches, we refer to the survey series Perrier et al. (2006a,b, 2007a,b).

Salazar-Aguilar et al. (2012) study a related routing problem where routes are computed in such a way that street segments with two or more lanes in the same direction are plowed simultaneously by different synchronized 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. The problem in (Salazar-Aguilar et al., 2012) is first defined through a MIP model. In addition, an efficient Adaptive Neighborhood Search approach is proposed. Although synchronized plowing has certain benefits, it is not being applied in Pittsburgh due to the added level of planning complexity that it implies. (Salazar-Aguilar et al., 2012) primarily focuses on the plowing aspects; management of resources such as salt and fuel is not considered. The performance of their algorithms are evaluated on real-world data, including an instance from the city of Dieppe, New Brunswick, Canada. With a population of roughly 24,000 inhabitants, 462 intersections and 1,234 road segments, the city of Dieppe is less than one fifth the size of downtown Pittsburgh. Consequently, it is not obvious whether their approach can be scaled and adapted to our problem setting.

Perrier et al. (2008) addresses another snow plow routing problem in urban areas. Each area is partitioned into a number of districts. Routes have to be determined for vehicles, parked at the district's depot, such that all road segments are serviced and all operational constraints are satisfied. Routes crossing these boundaries must be avoided from an administrative point of view. A similar situation arises currently in Pittsburgh where plows do not currently cross district boundaries. Although these artificial boundaries simplify the problem, they may also have an negative impact on the solution quality so these boundaries are not considered in this work. In addition to traditional routing constraints, Perrier et al. (2008) considers road priorities, precedence relations between roads belonging to different priority classes, tandem plowing and limitations on the plows which can be used to service certain roads. The authors propose a multicommodity network flow structure to impose the connectivity of the route performed by each vehicle. Two heuristic approaches are presented: the first constructs routes in parallel by solving a multiple vehicle rural postman problem with side constraints, the second is a cluster-first route-second approach.
Gupta et al. (2011) devised an iteration method to solve a snow plow routing problem on a network topography with a single depot. Per iteration, a trip, starting and ending at the depot and servicing a number of street segments is calculated. Every new iteration iteration, the street segments serviced in the previous iteration are removed from the network and a trip covering a (subset of) the remaining edges is calculated. The procedure repeats until all street segments have been serviced. The length of a single trip is limited by a maximum duration. Moreover, the total amount of salt required by the edges in a trip cannot exceed the truck's salt capacity. Although this problem bares strong similarities to our problem, the solution approach is not applicable because in our problem vehicles have to manage both salt and fuel resources, and not every depot offers both resources.
The remainder of this paper is structured as follows. First, Section 2 formally defines the problem and introduces notation. Next, Sections 3 and 4 present a number of exact and heuristic models including a MIP model (Section 3.1), a CP model (Section 3.2), a constructive heuristic (Section 4.1) and a Late Acceptance improvement heuristic (Section 4.2). Finally, Section 5 compares the performance of these methods on real-world data, and draws some conclusions.

2 Problem Description

For a given network of streets and a fleet of snow plows, our SPRP consists of finding a route for each vehicle, such that the routes collectively cover the entire network. The objective is to minimize the duration of the longest route, i.e. to minimize the makespan of the schedule. The road network is modeled as a mixed multigraph. Vertices in the graph represent intersections in the road network, the arcs and edges represent resp. directed and undirected road segments. For instance, a road in between two intersections, consisting of 2 lanes in each direction, translates to 4 directed arcs in the graph. We will refer to these arcs as

Parameter	Description
V^R	Set of intersections
E^R	Set of two-way, single lane residential streets
A^R	Multi-set of directed lanes and one-way streets
K	Set of heterogeneous vehicles
$\mathbb{F}$	Fuel depots
$\mathbb{S}$	Salt depots
d_{ij}	Time or distance it takes to get from intersection i to intersection j, $i, j \in V^R$.
f_{ij}^k	Fuel required to get from intersection i to intersection j, $i, j \in V^R$
s_{ij}^k	Salt required to get from intersection i to intersection j, $i, j \in V^R$
$0, n+1$	Resp. start and end depots of the trucks
$\overline{F}^k$	Maximum fuel capacity of vehicle $k \in K$
$\overline{S}^k$	Maximum salt capacity of vehicle $k \in K$
$\overline{C}$	Time horizon of the problem

Table 1: Parameters defining the snow plow optimization problem

unidirectional plow jobs. Unidirectional plow jobs are typically individual lanes of a multi-lane street, or one-way roads. In addition to unidirectional plow jobs, there also exist bidirectional plow jobs. Road segments in the latter category are small enough to be covered by a single pass of a snow plow, and the plow may come from either direction of the street. Typical examples of bidirectional plow jobs are streets in residential neighborhoods where cars are parked on each side of the road.

More formally, let $G^R(V^R, A^R \cup E^R)$ be a mixed multigraph where vertex set V^R represents the intersections, and E^R, A^R, the edges and arcs representing resp. the uni- and bidirectional street segments. For simplicity, it is assumed that graph G^R is strongly connected.

The roads are serviced by a heterogeneous fleet of snow plows K. Servicing a road segment $(i, j) \in A^R \cup E^R$ takes d_{ij} time. Vehicles may traverse road segments without servicing them. This is called deadheading. Due to the relatively low speed limits within the city, deadheading and servicing a road take equal amounts of time, independent of the road conditions. Each vehicle occasionally needs to refuel and resupply salt. A vehicle $k \in K$ has a fuel capacity $\overline{F}^k$ and salt capacity $\overline{S}^k$, $k \in K$. There are several depots throughout the city. Let $\mathbb{F}$ denote the set of fuel depots, $\mathbb{S}$ the set of salt depots. Some depots may supply both salt and fuel, hence $\mathbb{S} \cap \mathbb{F} \neq \emptyset$. The fuel (salt) consumption per street segment $(i, j) \in A^R \cup E^R$ using vehicle k is denoted by f_{ij}^k (s_{ij}^k). In addition to the fuel and salt depots, we define 0 and $n+1$ as the origin and destination depots where the vehicles are parked resp. before and after the trip.

An overview of the various parameters is provided in Table 1.

3 Mathematical models

In order to construct a MIP or CP model, we first define an auxiliary graph, using a set of unidirectional jobs $\overline{J}$ and bidirectional jobs $\vec{J}$. For every $(u, v) \in A^R$ define a unidirectional plow job $j = (u, v) \in \overline{J}$, which takes $d_j = d_{uv}$ time to complete and requires resp. f_j^k fuel and s_j^k salt when serviced by vehicle $k \in K$. Similarly, for every $(u, v) \in E^R$ define a bidirectional plow job $j \in \vec{J}$. Every bidirectional plow job $j \in \vec{J}$ can be decoupled into two unidirectional plow jobs $\vec{j}, \overleftarrow{j}$, representing the different orientations of the job. Obviously, in order to service a bidirectional road, only $\vec{j}$ or $\overleftarrow{j}$ needs to be executed. Finally, define set J consisting of all jobs, i.e. $J = \overline{J} \cup \{\vec{j}_i, \overleftarrow{j}_i \mid i \in \vec{J}\}$.
Let $i, j \in J$ be two different jobs, representing road segments $i = (u, v)$, $j = (s, t)$. Define d_{ij} as the time it takes to travel from intersection v to intersection s, *plus* the time required to complete job j. The travel time can be computed through a shortest path calculation in the routing graph G^R.
For each fuel depot $i \in \mathbb{F}$, a new ordered set of refuel jobs $F^i = 1, 2, \ldots$, is defined. Furthermore, let $F = \bigcup_{i \in \mathbb{F}} F^i$. A vehicle can refuel at a fuel depot $i \in \mathbb{F}$ by executing one of the fuel jobs $F^i = 1, 2, \ldots$ associated with depot i. Analogous for the salt depots $i \in \mathbb{S}$, we define sets $S^i = 1, 2, \ldots$, $S = \bigcup_{i \in \mathbb{S}} S^i$ representing salt resupply jobs.
We can now define our auxiliary graph, a directed, weighted multigraph $G(V_{0,n+1}, A)$ having vertex set $V_{0,n+1} = \{0\} \cup J \cup F \cup S \cup \{n+1\}$ and arc set A. For shorthand notation, denote $V = V_{0,n+1} \backslash \{0, n+1\}$, $V_0 = V_{0,n+1} \backslash \{n+1\}$, $V_{n+1} = V_{0,n+1} \backslash \{0\}$.
Arc set A is defined as follows:

- there is an arc $(0, j)$ for all $j \in J \cup \{n+1\}$).
- there is an arc $(i, n+1)$ for all $i \in V_0$).
- there is an arc (i, j) for all $i, j \in J$, $i \neq j$.
- there are arcs $(i, j), (j, i)$ for all $i \in J, j \in F \cup S$.
- there is an arc (i, j) for all $i \in F \cup S, j \in J$.

Observe that any resource-feasible vehicle schedule for SPRP can be represented in the auxiliary graph through a simple path from vertex 0 to vertex $n+1$.

3.1 MIP model

A MIP model for SPRP can be constructed through the auxiliary graph. Let binary variables x_{ij}^k denote whether vehicle $k \in K$ travels from i to j, $(i, j) \in A$, *and* executes job j. Integer variables C^i record the time that job $i \in V_{0,n+1}$ is completed. In addition, C^{n+1} records the makespan of the schedule. Finally, integer variables F_i^k, S_i^k indicate resp. the fuel and salt supply levels of vehicle k after leaving node i. For notation purposes, let $\delta^+(i) = \{j \mid (i, j) \in A\}$ and $\delta^-(i) = \{j \mid (j, i) \in A\}$. Table 2 summarizes the various sets and parameters used in the MIP model.

The model, solvable using a traditional branch-bound-cut approach, is as follows:

$$P : min \quad C^{n+1} \tag{1}$$

Param	**Description**
V	$J \cup F \cup S$
V_0	$V \cup \{0\}$
V_{n+1}	$V \cup \{n+1\}$
$V_{0,n+1}$	$V \cup \{0, n+1\}$
d_{ij}^k	Setup time between job $i \in V_{0,n+1}$ and $j \in V_{0,n+1}$, $i \neq j$, plus the time required to perform job j, for vehicle k.
f_{ij}^k	Fuel required to get from $i \in V_{0,n+1}$ to $j \in V_{0,n+1}$, $i \neq j$, plus the fuel required to perform job j, for vehicle k.
s_{ij}^k	Salt required to get from $i \in V_{0,n+1}$ to $j \in V_{0,n+1}$, $i \neq j$, plus the salt required to perform job j, for vehicle k.

Table 2

s.t.

$$\sum_{j \in \delta^+(0)} x_{0j}^k = \sum_{i \in \delta^-(n+1)} x_{i,n+1}^k = 1 \qquad \forall k \in K \tag{2}$$

$$\sum_{j \in \delta^-(i)} x_{ji}^k = \sum_{j \in \delta^+(i)} x_{ij}^k \qquad \forall i \in V \tag{3}$$

$$\sum_{k \in K} \sum_{j \in \delta^+(i)} x_{ij}^k = 1 \qquad \forall i \in \overline{J} \tag{4}$$

$$\sum_{k \in K} \left(\sum_{j \in \delta^+(u)} x_{uj}^k + \sum_{j \in \delta^+(v)} x_{vj}^k \right) = 1 \qquad \forall i \in \overleftrightarrow{J}, u = \overleftarrow{j}_i, v = \overrightarrow{j}_i, \tag{5}$$

$$\sum_{k \in K} \sum_{j \in \delta^+(i)} x_{ij}^k \leq 1 \qquad \forall i \in F \tag{6}$$

$$\sum_{k \in K} \sum_{j \in \delta^+(u+1)} x_{u+1,j}^k \leq \sum_{k \in K} \sum_{j \in \delta^+(u)} x_{u,j}^k \qquad \forall i \in \mathbb{F}, u \in \{1, \ldots, |F^i| - 1\} \tag{7}$$

$$\sum_{k \in K} \sum_{j \in \delta^+(i)} x_{ij}^k \leq 1 \qquad \forall i \in S \tag{8}$$

$$\sum_{k \in K} \sum_{j \in \delta^+(u+1)} x_{u+1,j}^k \leq \sum_{k \in K} \sum_{j \in \delta^+(u)} x_{u,j}^k \qquad \forall i \in \mathbb{S}, u \in \{1, \ldots, |S^i| - 1\} \tag{9}$$

$$C^0 - M(1 - x_{0j}^k) \leq C^j - d_{0j}^k \qquad \forall (0,j) \in A, k \in K \tag{10}$$

$$C^i - M(1 - x_{ij}^k) \leq C^j - d_{ij}^k \qquad \forall (i,j) \in A, i \neq 0, k \in K \tag{11}$$

$$F_j^k \leq F_i^k - f_{ij}^k + \overline{F}^k (1 - x_{ij}^k) \qquad \forall i \in J \cup \{0\}, j \in J \cup \{n+1\}, k \in K \tag{12}$$

$$F_j^k \leq \overline{F}^k - f_{ij}^k x_{ij}^k \qquad \forall i \in F, j \in J \cup \{n+1\}, k \in K \tag{13}$$

$$S_j^k \leq S_i^k - s_{ij}^k + \overline{S}^k (1 - x_{ij}^k) \qquad \forall i \in J \cup \{0\}, j \in J \cup \{n+1\}, k \in K \tag{14}$$

$$S_j^k \leq \overline{S}^k - s_{ij}^k x_{ij}^k \qquad \forall i \in S, j \in J \cup \{n+1\}, k \in K \tag{15}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, k \in K \tag{16}$$

$$0 \leq C^i \leq \overline{C} \qquad \forall i \in V \cup \{0, n+1\} \tag{17}$$

$$0 \leq F_i^k \leq \overline{F}^k \qquad \forall i \in V \cup \{0, n+1\} \tag{18}$$

$$0 \leq S_i^k \leq \overline{S}^k \qquad \forall i \in V \cup \{0, n+1\} \tag{19}$$

Constraints (2) define the starting and ending of the tour: every vehicle must start and end at the depot. Constraints (3) enforce flow preservation. Each unidirectional plow job must be performed exactly once ((4), (5)). Similarly, each bidirectional plow job must be executed, but only in one direction (5). Optional refueling/resupply jobs may be performed at most once (6), (8). Constraint (7) orders the refueling jobs: a refueling job $u \in F^i$ must be performed before $v \in F^i$, $v > u$, can be performed. This constraint reduces the amount of symmetry in the model. Constraint (9) is identical to Constraint (7) in the context of salt resupply jobs. Constraint (10)-(11) relate the completion time variables to the nodes, while taking the setup times and job durations into consideration. Similarly Constraints (12)-(13), (14)-(15) manage resp. the fuel and salt levels of the vehicles at each node. A vehicle leaves a refueling/resupply node with a full tank/salt supply.

3.2 CP Model

To model SPRP efficiently through CP, we will rely on interval variables (Laborie and Rogerie, 2008; Laborie et al., 2009). An interval variable represents an interval during which an activity can be performed. For notation purposes, an interval variable will be denoted 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 optional. Optional intervals can be either present or absent in the final solution. An absent interval variable is ignored by any constraint or expression it is part of. The CP model presented in Algorithm 1 relies on three types of interval variables:

1. Job variables j_i for all $i \in V$ having duration d_i.
2. Assignment variables a_i^k for all $k \in K$, $i \in V_{0,n+1}$
3. Unidirectional plow job variables $\vec{j}_i, \overleftarrow{j}_i$ for all $i \in \overleftrightarrow{J}$ to distinguish the two possible orientations of bidirectional plow jobs.

A summary of the constraints used in Algorithm 1 is given in Table 3.

The objective of the model, minimize the makespan, is modeled through Constraints 5, 9. Constraint 6 states that every bidirectional plowing job has to be performed in only one direction and Constraint 7 ensures that every job is assigned to a single vehicle. Next, a number of constraints per vehicle are specified. Sequencing of the jobs on each vehicle is performed through Constraints 10-12. Resources are managed through a number of cumulative resource constraints (Constraints 13-16). Vehicles start with a full load of salt, performing a plow job i consumes s_i^k salt, and visiting a salt depot replenishes the salt resource (Constraints 13). For each truck, the salt level needs to remain between 0 and $\overline{S}^k$, the maximum salt capacity of the truck (Constraints 14). Similar constraints (15-16) are imposed for the fuel resource. In addition, Constraint 15 also takes the fuel consumption related to traveling in between jobs (deadheading) into account.

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.
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.

Table 3: Description of CP constraints. All of these constraints are available in IBM ILOG CP Optimizer by default.

Finally, lines 17-20 specify a number of redundant constraints which are meant to improve the performance of the model. Constraints 17, 18 reduce the amount of symmetry in the model by imposing an order on the refuel and resupply salt jobs. Constraint 20 links the start and end times of consecutive intervals.

A note on implementation The CP model presented in Algorithm 1 is implemented in IBM ILOG CP Optimizer 12.6.2. To implement this model, a minor modification is required, as CP Optimizer has no direct way to implement the function **stepAtStart**(a_i^k, $f^k_{i,\text{succ}[j_i,seq^k]}$) used in Constraint 15. To resolve this issue, a new variable $fuel_i^k$ is introduced into the model which records the fuel level of vehicle k after performing job i. Constraints 15-16 may now be replaced by the equivalent constraints from Algorithm 2.

4 Heuristic models

4.1 Constructive Heuristic

The constructive heuristic uses a greedy approach to construct a feasible initial schedule. The heuristic works in two stages: stage one sequences all plow jobs while ignoring resource feasibility. Stage two makes the schedule feasible in terms of resources. The heuristic starts off with an empty schedule for every vehicle, that is, each vehicle has a schedule: $[0, n+1]$. The heuristic iterates over all unscheduled *plow* jobs and schedules them one-by-one. To schedule a particular

Algorithm 1: CP model.

`Variable definitions:`

1 $j_i = \begin{cases} \{0, \infty, d_i\} & \text{if } i \in \overrightarrow{J} \cup \overleftarrow{J} \\ \{0, \infty, d_i, opt\} & \text{if } i \in F \cup S \end{cases}$

2 $a_i^k = \begin{cases} \{0, 0, 0\} & \text{if } i = 0 \\ \{0, \infty, 0\} & \text{if } i = n+1 \\ \{0, \infty, d_i\} & \text{otherwise} \end{cases}$

3 $\overrightarrow{j}_i, \overleftarrow{j}_i = \{0, \infty, d_i, opt\} \qquad \forall i \in \overrightarrow{J}$

4 $obj \in \{0, \infty\}$

`Objective:`

5 `Min` obj

6 $\text{alternative}(j_i, \{\overleftarrow{j}_i, \overrightarrow{j}_i\}) \qquad \forall i \in \overrightarrow{J}$

7 $\text{alternative}(j_i, \bigcup_{k \in K} a_i^k) \qquad \forall i \in J \cup F \cup S$

8 **forall** $k \in K$

`Objective Constraints:`

9 $obj \geq \text{endOf}(a_{n+1}^k)$

`Sequencing Constraints:`

10 $seq^k = \text{noOverlapSeq}(\bigcup_{i \in J \cup F \cup S} a_i^k, [d_{ij} - d_j \mid (i,j) \in A])$

11 $\text{first}(a_0^k, seq^k)$

12 $\text{last}(a_{n+1}^k, seq^k)$

`Salt Constraints:`

13 $\text{saltCumulFunc}^k = \text{stepAtStart}(a_0^k, \overline{S}^k) - \sum_{i \in J} \text{stepAtStart}(a_i^k, s_i^k) + \sum_{i \in S} \text{stepAtStart}(a_i^k, 0, \overline{S}^k)$

14 $0 \leq \text{saltCumulFunc}^k \leq \overline{S}^k$

`Fuel Constraints:`

15 $\text{fuelCumulFunc}^k = \text{stepAtStart}(a_0^k, \overline{F}^k) + \sum_{i \in F} \text{stepAtStart}(a_i^k, 0, \overline{F}^k) - \sum_{i \in J \cup F \cup S \cup \{0\}} \text{stepAtStart}(a_i^k, f^k_{i, \text{succ}[j_i, seq^k]})$

16 $0 \leq \text{fuelCumulFunc}^k \leq \overline{F}^k$

`Performance Constraints:`

17 $\text{presenceOf}(j_v) \implies \text{presenceOf}(j_u) \qquad \forall i \in \mathbb{F}, u \in \{1, \ldots, |F^i| - 1\}, v = u + 1$

18 $\text{presenceOf}(j_v) \implies \text{presenceOf}(j_u) \qquad \forall i \in \mathbb{S}, u \in \{1, \ldots, |S^i| - 1\}, v = u + 1$

19 **forall** $k \in K$

20 $\text{startOf}(j_i) = \text{endOf}(\text{pred}(j_i, seq^k)) + t_{\text{pred}[j_i, seq^k], j_i}$
$\forall i \in J \cup F \cup S \cup \{n+1\}$

job, the heuristic evaluates for every vehicle all possible places to insert the job into its schedule. The impact of the job insertion onto the completion time of

Algorithm 2: CP model extension

forall $k \in K$

$$fuel_j^k \in \begin{cases} [\overline{F}^k, \overline{F}^k] & \text{if } j = a_0^k \\ [0, \overline{F}^k] & \text{if } j = a_i^k, i \in J \cup F \cup S \cup \{n+1\} \end{cases}$$

$$fuel^k_{\text{succ}[j,seq^k]} =$$

$$\begin{cases} fuel_j^k - f^k_{j,\text{succ}[j,seq^k]} - f^k_{\text{succ}[j,seq^k]} & \text{if } j = a_0^k \\ \overline{F}^k - f^k_{j,\text{succ}[j,seq^k]} - f^k_{\text{succ}[j,seq^k]} & \text{if } j = a_i^k, i \in J \cup F \cup S \cup \{0\} \end{cases}$$

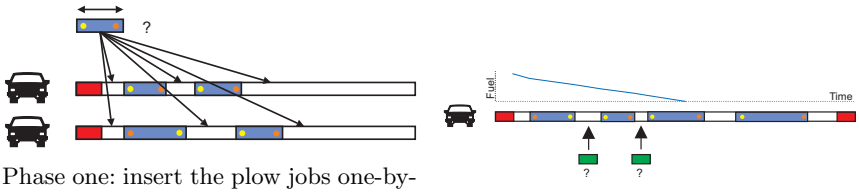
the vehicle's schedule is computed by factoring in the added travel time and job duration. In addition, a lower bound is calculated on the number of refuel and resupply trips the vehicle will have to make based on the amount of salt (fuel) the vehicle will need to complete its schedule. The number of refuel/resupply operations is then multiplied with the duration of a refuel/resupply job, thereby obtaining a lower bound on the time required to refuel and resupply. The actual driving time to a refuel or resupply depot is neglected in these calculations. Finally, recall that the bidirectional plow jobs can be performed from either direction. While evaluating a candidate position to insert the job, the heuristic chooses the best orientation of the plow job in respect to the jobs immediately preceding/succeeding the insert position.

After the plow jobs have been scheduled, phase two of the constructive heuristic will make the schedule resource feasible by inserting refuel and resupply jobs. For a given vehicle $k \in K$, the resupply salt jobs are inserted as follows. Let the plow jobs assigned to vehicle k in phase 1 be indexed from $0, \ldots, n$, and let j be the job for which $\sum_{i=0}^{j} s_i^k > \overline{S}_i^k$. That is, after $j-1$ jobs, the vehicle runs out of salt and as such, cannot complete job j. In such cases, the heuristic schedules a resupply job between jobs $j-1$ and j, thereby choosing the nearest resupply depot. This procedure is repeated until the schedule is feasible in terms of salt. Next, refuel jobs are inserted in a similar fashion. However, before inserting a new fuel job between jobs $j-1$ and j, an extra check has to be performed to verify that after job $j-1$ the vehicle has sufficient fuel to reach the nearest fuel depot. If not, we iterate backwards through the schedule, thereby searching for the nearest feasible position to insert a refuel job. A visual representation of the heuristic is given in Figure 1.

4.2 Late Acceptance improvement heuristic

After executing the first phase of the constructive heuristic, a Late Acceptance (LA) heuristic (Burke and Bykov, 2012) is used to improve the quality of the solution before phase 2 is initiated. To generate new solutions, the heuristic utilizes two simple search neighborhoods:

1. bestSwapMove: randomly choose a vehicle $k_1 \in K$, a job j_1 from the schedule of vehicle k_1 and a target vehicle k_2. For every possible plow job j_2 scheduled



Phase one: insert the plow jobs one-by-one, thereby selecting a insert position and potentially an orientation for the bidirectional jobs.

Phase two: making the schedule resource feasible by inserting resource jobs

Fig. 1: Constructive Heuristic

on vehicle k_2, and for every possible orientation of jobs j_1, j_2, evaluate the impact of swapping jobs j_1 and j_2.

2. bestRemoveInsertMove: randomly choose a vehicle $k_1 \in K$, a job j_1 from the schedule of vehicle k_1 and a target vehicle k_2. For every possible insert position of the schedule of vehicle k_2 and for every possible orientation of job j_1, evaluate the impact of removing job j_1 from the schedule of k_1 and inserting it into k_2.

To move from one solution to a neighboring solution, we randomly select one of the two neighborhoods and evaluate the best candidate solution produced by this neighborhood. Following a standard LA approach, a move is accepted if its cost is better (or equal) to the cost of a solution L iterations ago, where L is a user-controlled parameter of the heuristic. The heuristic is terminated if (a) a maximum time limit is reached or (b) the incumbent solution has not been improved during 10000 consecutive iterations, where 10000 is determined empirically. Notice that when $L = 1$, the heuristic behaves as a greedy heuristic, only accepting improving moves. Selecting a larger value for L generally decreases the convergence rate of the heuristic, but reduces the chance of getting stuck in a local optimum.

5 Computation Experiments

5.1 Setup

Experiments are conducted on real world data, in collaboration with the city of Pittsburgh. Routing data is obtained through Open Street Maps (OSM). To extract data from a geographical area, including information about the roads, lanes, shapes, speed limits, traffic restrictions, etc, rectangular shaped snapshots are taken from an area on the map. In this experimental setup, we captured 21 different regions of Pittsburgh, varying from residential areas, downtown, rural areas, and business districts. Travel times between two neighboring intersections are computed by multiplying the length of the road with the maximum allowed driving speed.

Pittsburgh has 9 depots at different locations, 8 of which have salt, 5 of which

have fuel. For experimental purposes, we use a small heterogeneous fleet of five vehicles to service each area. The smallest pickup-truck in our fleet has a capacity of 2 tons of salt and 26 gallons of fuel, whereas the largest plow has a capacity of 20 tons of salt and 75 gallons of fuel. Currently, the city utilizes about 1 ton of salt per mile, rendering salt the most constraining resource.

5.2 Results

Experiments have been conducted on 22 instances, which are summarized in Table 4. For each instance, the total number of plow jobs, percentage of bidirectional jobs, and total plowing distance (miles) is given. The MIP and CP models have been implemented using Cplex, resp. CP Optimizer 12.6.2. Experiments were run using default parameters and extended inference on the CP sequence variables.

Figure 2 compares the performance of CP and the LA Heuristic. Since each of these methods is warm-started with the solution obtained from the Constructive heuristic, we only show how much either of these approaches could improve the constructive solution. Runtimes for the CP approach were capped at resp. 10 minutes and 1 hour. Similarly, the runtime of the LA Heuristic was capped at 10 minutes, or 10000 non-improving iterations, whichever came first. To measure the impact of the randomization in the LA Heuristic, 8 runs of the heuristic have been performed for each instance. The results of these runs are visualized by boxplots in Figure 2.

The constructive heuristic produces an initial solution of reasonable quality in very little time, usually in the order of milliseconds for instances with less than 1000 jobs. For the smaller instances, up to 1000 jobs, the CP approach is capable of improving upon the constructive heuristic. For the larger instances, we noticed that the CP model ran out of memory and had to fall back on the much slower swap memory, thereby slowing down the CP approach tremendously. The largest instances, Residential Pittsburgh and inst18, could not be solved through CP on our machine due to insufficient memory. The LA approach produces good results in relatively little time. As can be observed from the largest instances, and most notably the Residential instance, the LA approach scales well. An additional advantage of this method is that the convergence rate can be adjusted, depending on the availability of computation time. Occasionally, as for instance inst12, the CP approach significantly outperforms the LA approach. The LA approach tracks for each vehicle how often it needs to resupply fuel and salt based on its resource consumption, and multiplies this with the average distance to a resupply depot to approximate the time spent on resupplying and refueling. This approximates becomes inaccurate when the travel time to a depot varies substantially, depending on the location of the vehicle. Calculating a more accurate approximation on the travel time to a depot, for instance by considering the position of the vehicle at the time it needs to resupply, would help mitigating this issue.

inst	jobs	bidir	dist	inst	jobs	bidir	dist	inst	jobs	bidir	dist
kaminst	45	38	3.4	inst5	631	74	55.2	inst13	529	59	47.3
downtown	724	38	38.1	inst6	632	68	52.9	inst14	498	64	37.2
mntWash.	577	81	52.1	inst7	796	58	50.9	inst15	531	63	36.9
Residential	4073	64	315.5	inst8	500	31	42.8	inst16	498	92	47.5
inst1	233	61	27	inst9	481	38	38.4	inst17	499	75	42.6
inst2	346	93	38.6	inst10	574	64	41.5	inst18	1324	24	80.5
inst3	451	53	32.1	inst11	547	54	42.2				
inst4	287	87	22.9	inst12	339	91	30.7				

Table 4: Instance data: number of jobs, percentage of jobs that are bidirectional, total distance to plow (mi).

In addition to experiments with the CP model, a number of experiments were conducted with the MIP Model. For all but the smallest instance in our data set (Kaminst), the MIP model did not fit into our computer memory (16GB+30GB swap). The latter is mainly attributed to the vast number of variables in each model, namely $|K||V|^2$ flow variables, and $2|K||V|$ resource variables. For the Kaminst instance, after a 1 hour runtime, the MIP model (warm-started by the constructive heuristic) did not manage to improve upon its initial solution and had an optimality gap of 91.98%. The large optimality gap is explained by the presence of the big-M constraints, where the ratio between M and the length of the jobs d_{ij}^k is very large.

Figure 3 shows more details for the 4 named instances in Table 4, and the spreading of the depots (blue squares). Each of these 4 instances represents a different geographical area in Pittsburgh, marked on the map in Figure 5. From left to right: Mnt Washington, Downtown, Residential, Kaminst. The x-axis of the graphs in Figure Figure 3 shows the makespan of the schedule, converted to a HH::MM::SS format. At time 0, 0% of the area has been serviced (y-axis), whereas, by the end of the schedule, 100% of the area has been serviced. Some of the graphs, e.g. the Kamin instance, have a flat section at the beginning and end of the graph. This is where the vehicles travel from the nearest depot to the service area, and eventually back to the depot. The graphs have been generated using the same settings as before, unless mentioned otherwise.

Each graph shows the best CP solution, when one could be found, a solution from the constructive heuristic and LA improvement heuristic. For the LA heuristic, the graphs plot the average solution, as well as the diversity of solutions encountered. The MIP approach was unable to improve upon its warm-start solution, and is therefore not included in any of the graphs. As can be observed from the largest instance, the LA heuristic finds significant improvements over the constructive heuristic. Furthermore, when focusing on the robustness of the heuristic, the LA solutions show only a moderate variance in solution quality over multiple runs; the longer the heuristic runs, the smaller the variance.

Figure 4 presents a progress-over-time graph for the LA Heuristic for various list lengths L (see Section 4.2). Choosing L small results in an aggressive conver-

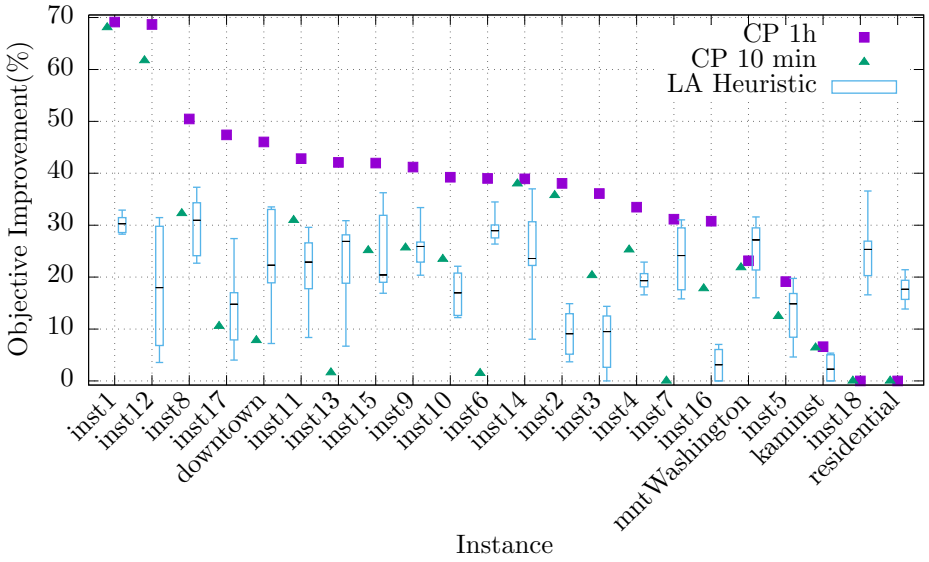


Fig. 2: Improvement over Constructive Heuristic: LA heuristic (8 iterations, 10 min runtime), CP (10 min resp. 1h runtime). Each of these methods is warm-started by the constructive solution.

gence, whereas higher values L allow a wider exploration of the search space at the cost of a slower convergence.

Finally, Figure 6 shows for the Residential instance the amount of plowing versus deadheading for every vehicle. The completion time of a vehicle schedule is obtained by summing these two values. As can be observed, the makespan of the schedule is dominated by the the completion time of the first vehicle. The capacity of this vehicle (1 ton salt) is significantly smaller than the capacity of the largest vehicle (20 ton). For such a large instance, the number of trips to a salt depot becomes significantly large, especially for smaller vehicles. Having a better approximation of the time required to travel to a depot would resolve this issue.

6 Conclusion

The constructive heuristic is capable of finding initial solutions of reasonable quality fast. The CP approach finds good solutions to instances up to a 1000 jobs, but does not scale well beyond that. The LA heuristic scales considerably better. A logical direction for further research would be to combine the LA heuristic and the CP approach in a Large Neighborhood Search. First, the LA heuristic is used to find a good global solution, after which the CP approach can be used to locally optimize small area's of the map in an iterative procedure.
Another research direction for this project involves online adaptations of the

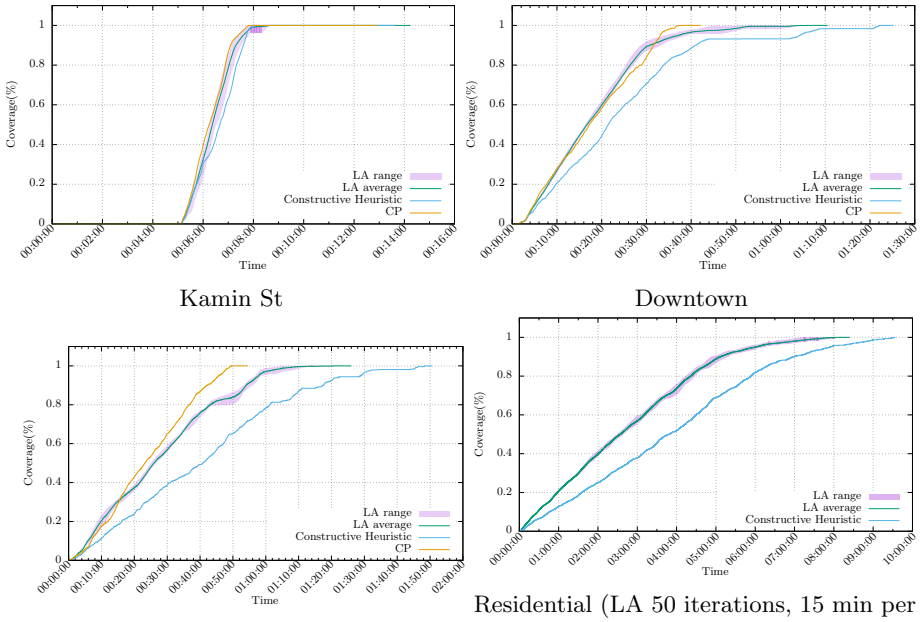


Fig. 3: Service over time

schedule. Unexpected events such as a blocked road, traffic congestion, emergency request etc, could necessitate modifications to the schedule. Again, the CP approach may be of use to 'repair' a small portion of the schedule, while leaving the remainder of the schedule intact.

Finally, from a model perspective, a number of additional features may be incorporated, including:

- Road priorities. The city assigns priorities to roads. In general, roads with high priorities should be serviced as fast as possible. This can be achieved by replacing the makespan objective by a weighted objective which minimizes the completion time per priority class.
- U-turns. Due to the size of the plows, having a large number of U-turns in a schedule is undesirable. As such, U-turns should be forbidden (hard-constrained) or penalized in the objective function.
- Road limitations. Some roads are too small or too steep to be plowed by the largest (and heaviest) vehicles. Similarly, in rural areas, the weight of large plows may exceed weight limitations on certain bridges. Consequently, a routing graph per vehicle category will be necessary. In addition, some plow jobs cannot be assigned to some of the heavier vehicles.

Road priorities are easily accounted for in the models presented, by assigning a priority class to each job and by using a weighted objective function which keeps track of the completion time of each priority class. Similarly, U-Turns can be penalized by increasing the setup time between a pair of jobs which would

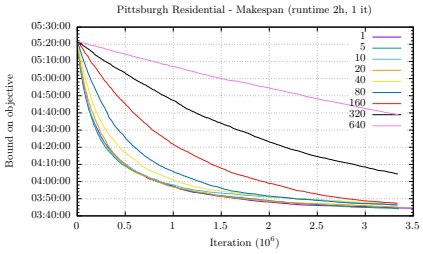


Fig. 4: Progress-over-Time for various list lengths

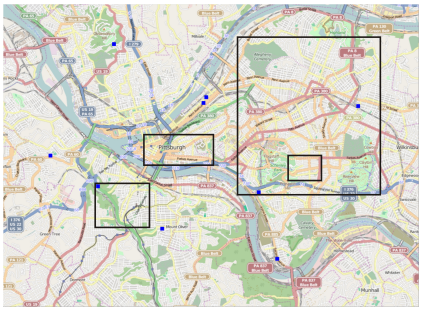
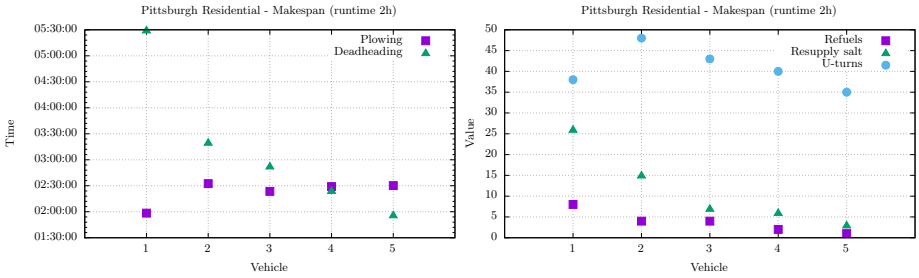


Fig. 5: Depots and names instances



Plowing vs Deadheading

Resources

Fig. 6: Solution details

require a u-turn if one is performed immediately after the other. In case of a forbidden U-Turn, the setup-time will be significantly larger, representing the detour the truck has to make to get back, e.g. the time it takes to drive around the block.

Bibliography

E. K. Burke and Y. Bykov, "The late acceptance hill-climbing heuristic," Department of Computing Science and Mathematics, University of Stirling, Tech. Rep., 2012. [Online]. Available: http://www.cs.stir.ac.uk/research/publications/techreps/pdf/TR192.pdf

H. A. Eiselt, M. Gendreau, and G. Laporte, "Arc routing problems, part i: The chinese postman problem," *Operations Research*, vol. 43, no. 2, pp. 231–242, 1995.

——, "Arc routing problems, part ii: The rural postman problem," *Operations Research*, vol. 43, no. 3, pp. 399–414, 1995.

Environmental Protection Agency, "Storm water management fact sheet, minimizing effects from highway deicing." Tech. Rep. EPA 832-F-99-016, 1999. [Online]. Available: http://water.epa.gov/scitech/wastetech/upload/2002_06_28_mtb_ice.pdf

D. Gupta, E. Tokar-Erdemir, D. Kuchera, A. K. Mannava, and W. Xiong, "Optimal workforce planning and shift scheduling for snow and ice removal." Center for Transportation Studies, University of Minnesota, Tech. Rep. Mn/DOT 2011-03, 2011. [Online]. Available: http://www.cts.umn.edu/Publications/ResearchReports

P. Laborie and J. Rogerie, "Reasoning with conditional time-intervals," in *FLAIRS Conference*, 2008, pp. 555–560.

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

K. Mei-Ko, "Graphic programming using odd or even points," *Chinese Math*, vol. 1, pp. 273–277, 1962.

N. Perrier, A. Langevin, and J. F. Campbell, "A survey of models and algorithms for winter road maintenance. part i: system design for spreading and plowing." *Computers & Operations Research*, vol. 33, pp. 209–238, 2006.

——, "A survey of models and algorithms for winter road maintenance. part ii: system design for snow disposal," *Computers & Operations Research*, vol. 33, no. 1, pp. 239 – 262, 2006.

——, "A survey of models and algorithms for winter road maintenance. part iii: Vehicle routing and depot location for spreading," *Computers & Operations Research*, vol. 34, no. 1, pp. 211 – 257, 2007.

——, "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*, vol. 34, no. 1, pp. 258 – 294, 2007.

N. Perrier, A. Langevin, and C.-A. Amaya, "Vehicle routing for urban snow plowing operations," *Transportation Science*, vol. 42, no. 1, pp. 44–56, 2008.

J. Rubin, P. E. Garder, C. E. Morris, K. L. Nichols, J. M. Peckenham, P. McKee, A. Stern, and T. O. Johnson, "Maine winter roads: Salt, safety, environment and cost," Margaret Chase Smith

Policy Center, University of Maine, Tech. Rep., 2010. [Online]. Available: http://umaine.edu/mcspolicycenter/files/2010/02/Winter-Road-Maint-Final.pdf

M. A. Salazar-Aguilar, A. Langevin, and G. Laporte, "Synchronized arc routing for snow plowing operations." *Computers & Operations Research*, vol. 39, no. 7, pp. 1432–1440, 2012.

T. Usman, L. Fu, and L. F. Miranda-Moreno, "Quantifying safety benefit of winter road maintenance: Accident frequency modeling," *Accident Analysis & Prevention*, vol. 42, no. 6, pp. 1878 – 1887, 2010.