A Short History of Arc Routing, in Honour of Leonhard Euler

Gilbert Laporte

Canada Research Chair in Distribution Management
HEC Montréal

Leonhard Euler (Basel, 1707; St.Petersburg, 1783)

- Most prolific mathematician ever: 25 books, 850 papers (800 pages a year from 1725 to 1783).
- 4500 letters and hundreds of manuscripts.
- Responsible for one quarter of the total output in mathematics, physics, mechanics, astronomy and navigation in the 18$^{\text{th}}$ century.
- Collected works: 70 volumes.
- Established the relation $e^{i\theta} = \cos\theta + i \sin\theta$.
- Introduced the symbol π in 1737.
- Contributed to the Königsberg bridges problem.

Reference: A.A. Assad, "Leonhard Euler: A Brief Appreciation", *Networks* 49, 190–198, 2007.

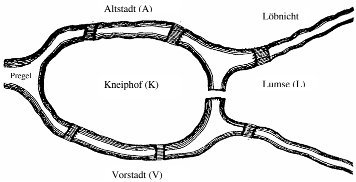
Outline

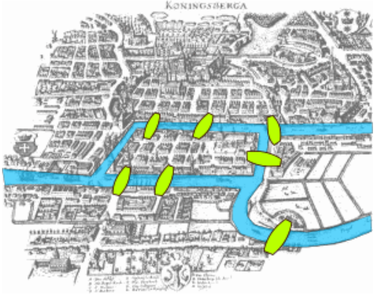
1. Eulerian graphs
2. The Chinese Postman Problem
3. The Rural Postman Problem
   - heuristics
   - exact algorithms
4. The Capacitated Arc Routing Problem
   - heuristics
   - exact algorithms

Eulerian Graphs

The bridges of Königsberg (Euler, 1736)

I. Gribkovskaia, Ø. Halskau, G. Laporte (2007), "The Bridges of Königsberg – A Historical Perspective", *Networks*, 49:199–203.

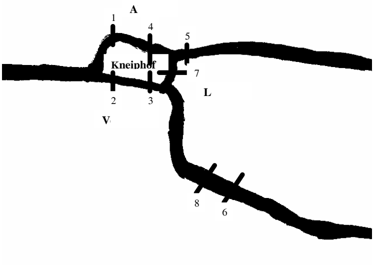




The Kaliningrad exclave

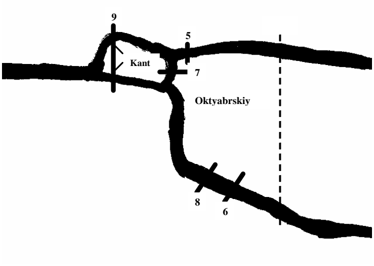
Aerial view of Kaliningrad

The bridges of Königsberg



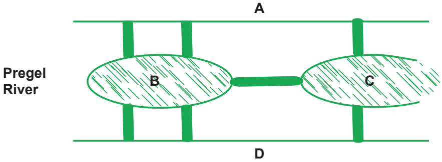
1. The Salesman's Bridge (1286, 1787, 1900)
2. The Green Bridge (1322, 1590, 1907)
3. The Slaughter Bridge (1377, 1886)
4. The Blacksmith's Bridge (1397, 1787, 1896)
5. The Timber Bridge (1404, 1904)
6. The High Bridge (1506, 1883, 1939)
7. The Honey Bridge (1542, 1882)
8. The Emperor's Bridge (1905)

The bridges of Kaliningrad



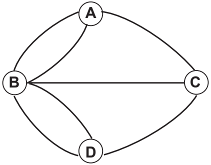
9. The Estacada (1972)
10. Bridge under construction

The Königsberg bridges problem (Euler, 1736)



Does there exist a closed traversal using each bridge exactly once?

Graph representation:



Is this graph unicursal (Eulerian)?

Euler: necessary conditions for the unicursality of an undirected graph

- Must be connected
- All vertices must have even degree

Undirected graphs (edges) (Euler, 1736)

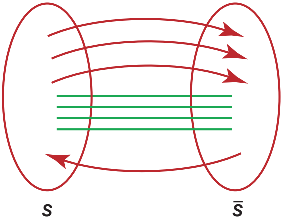
- Connectedness
- Even degrees

Directed graphs (arcs)

- Strong connectedness
- In-degree = out-degree (symmetry)

Mixed graphs (arcs and edges) (Ford and Fulkerson, 1962)

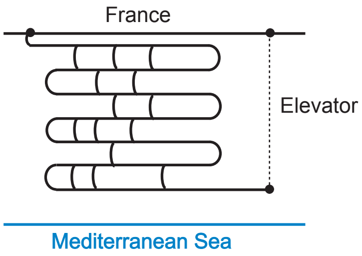
- Strong connectedness
- Even degrees (irrespective of directions)
- Balanced $\left[\begin{array}{l}\text{for every vertex partition } (S, \overline{S}), \\ \left|\text{edges } S - \overline{S}\right| \geq \left|(\text{arcs } S \to \overline{S}) - (\text{arcs } \overline{S} \to S)\right|\end{array}\right.$



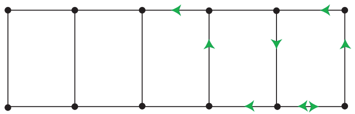
- Evenness and symmetry (in-degree = out-degree) imply that the graph is balanced

Famous non-Eulerian graphs

Monaco: Jardin exotique

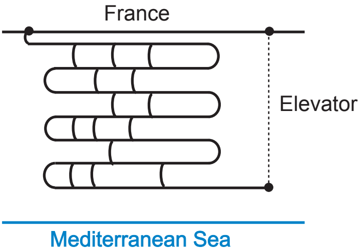


Your typical supermarket

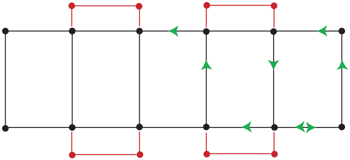


Famous non-Eulerian graphs

Monaco: Jardin exotique

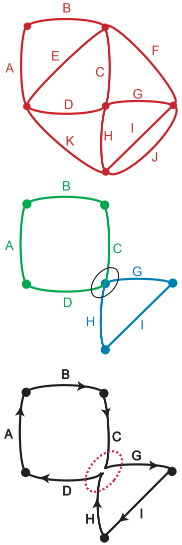


Your typical supermarket



Traversing a Eulerian graph (easy)

(See, e.g. Hierholzer, 1873)
End-pairing algorithm [described in Edmonds and Johnson, 1973]



The Chinese Postman Problem

The Chinese Postman Problem (Guan, 1962)

- Determine least cost traversal of all edges/arcs of a graph at least once: minimize deadheading
- Methodology:
  1. Determine least cost augmentation of the graph to make it Eulerian (i.e., replicate some of its edges/arcs)
  2. Apply end-pairing algorithm

Undirected case (Edmonds and Johnson, 1973)

- Solve matching problem on all odd-degree vertices

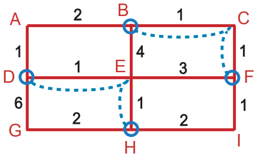
Directed case (Edmonds and Johnson, 1973; Orloff, 1974, Beltrami and Bodin, 1974)

- Solve transportation problem to balance vertices

Mixed case [NP-hard]

- Integer linear programming (branch-and-cut) (Grötschel and Win, 1992; Nobert and Picard, 1996; Corberán et al., 2000)

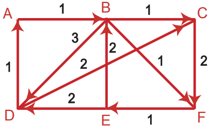
Undirected case



Matchings:

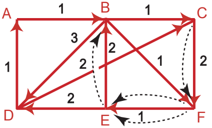
BD,FH:	$3 + 3 = 6$
BF,DH:	$2 + 2 = 4*$
BH,DF:	$5 + 4 = 9$

Directed case



Transportation problem

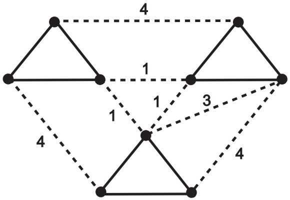
	B	E	
C	5	1 3	1
F	1 3	1	1
	1	1	



The Rural Postman Problem

Rural postman problem (Orloff, 1974)

$G = (V, E), R \subseteq E$: set of required edges
p connected components are induced by R

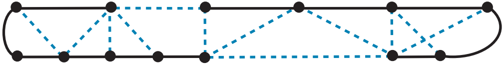


NP-hard: Lenstra and Rinnooy Kan (1976)

Frederickson's heuristic, 1979

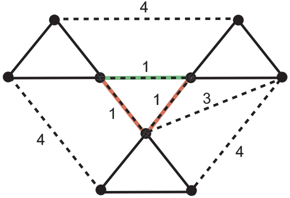
1. Shortest spanning tree over connected components
2. Matching of odd degree vertices

Worst-case performance ratio if triangle inequality is satisfied: 3/2



Rural postman problem (Orloff, 1974)

$G = (V, E), R \subseteq E$: set of required edges
p connected components are induced by R

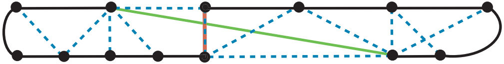


NP-hard: Lenstra and Rinnooy Kan (1976)

Frederickson's heuristic, 1979

1. Shortest spanning tree over connected components
2. Matching of odd degree vertices

Worst-case performance ratio if triangle inequality is satisfied: 3/2



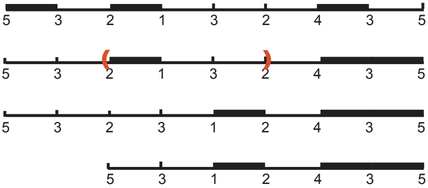
Good empirical performance of Frederickson's heuristic on random graphs

(Hertz, Laporte, Nanchen-Hugo, *INFORMS J. on Computing*, 1999)

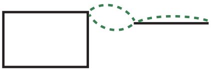
$\|V\|$	% deviation from optimum
20	3.36
30	1.69
40	4.02
50	3.98

Improvement heuristics (undirected graphs) (Hertz, Laporte, Nanchen-Hugo, *INFORMS Journal on Computing*, 1999)

Shorten (The lazy postman problem)



Add



+ call Shorten

Drop

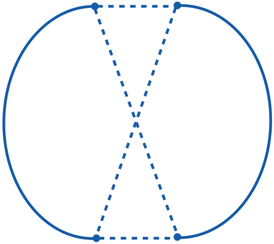
- Change status of a required edge to non required + call shorten

Make feasible

- if a required edge is missing, add it

2-opt

- As in TSP, try to improve a solution by
  - dropping 2 edges
  - reconnecting by shortest chains
  - calling shorten
  - making feasible
- Solution may not be feasible if some required edges were removed



Some computational results (random graphs)

$\lvert V \rvert$	Frederickson % deviation	2-opt % deviation	sec
20	3.36	0.36	0.02
30	1.69	0.00	0.14
40	4.02	0.00	0.70
50	3.98	0.00	1.26

Mittaz, 1999. Adaptation to the directed case.

Corberán, Martí, Romero (*Computers & Operations Research*, 2000)

- Heuristics based on flow + matching
- Tabu search
- For mixed graphs

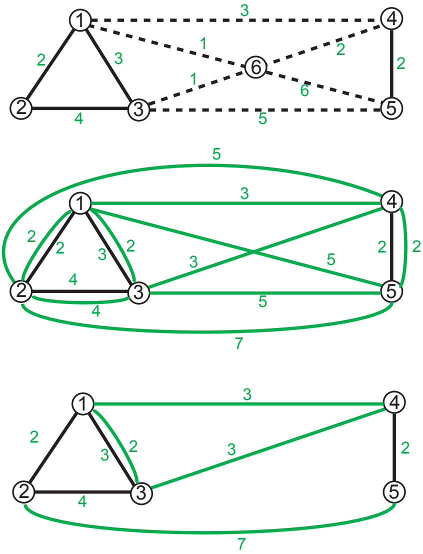
Exact algorithm for the Undirected Rural Postman Problem (Ghiani and Laporte, *Mathematical Programming*, 2000)

Graph simplification (Christofides, 1981)

- Add to $G_R = (V_R, R)$ an edge between each vertex pair of V_R using shortest path costs
- Delete any one of two parallel edges if they have same cost
- Delete $(i, j) \notin R$ if $c_{ik} + c_{kj} = c_{ij}$

Graph $G = (V, E) \quad R \subset E$

$V_R = \{1, 2, 3, 4, 5\}$



Formulations with deadheading variables

x_{ij}: number of times (i, j) is deadheaded.

* Never optimal to traverse an edge more than twice

Thus

$$
\begin{aligned}
x_{ij} &\le 1 \qquad \text{if } (i,j) \in R \\
x_{ij} &\le 2 \qquad \text{if } (i,j) \notin R
\end{aligned}
$$

Also, if $i, j \in V_R$ (same component), then $x_{ij} \le 1$ (Corberán and Sanchis, 1994).

$x_{ij} \le 1$

The Corberán and Sanchis Formulation (1984)

- p: number of connected components
- x_e: number of deadheadings of e
- $\delta(S)$: { edges with one extremity in S }
- v is R-even (R-odd) if it is incident to an even (odd) number of edges of R

$$\text{Minimize} \sum_{e \in E} c_e x_e$$

s.t.

$$\sum_{e \in \delta(v)} x_e = 0 \pmod 2 \qquad (v \in V_R \text{ is } R\text{-even})$$

$$\sum_{e \in \delta(v)} x_e = 1 \pmod 2 \qquad (v \in V_R \text{ is } R\text{-odd})$$

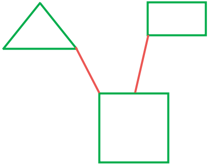
$$\sum_{e \in \delta(S)} x_e \geq 2 \; (S = \cup_{k \in P} V_k, P \subset \{1, \ldots, p\}, P \neq \emptyset)$$

$$0 \leq x_e \leq 1 \text{ or } 2 \text{ and integer} \qquad (e \in E)$$

Difficulties: 1) (mod 2) constraints ⇒ extra binary variables z
2) number of connectivity constraints
3) number of 0-1-2 variables

Ghiani and Laporte, 2000

All but $p - 1$ variables are 0–1.



In an optimal solution, only variables corresponding to shortest spanning tree over connected components need be 0–1–2.
For any such variable x_e, set

$$x_e = x'_e + x''_e$$

where x'_e, x''_e are 0–1

$\Rightarrow$ all variables of the problem are now 0–1

$\overline{E} = E \cup \{\text{duplicated edges}\}$

First self-contained formulation using edge variables only (Ghiani and Laporte, *Mathematical Programming*, 2000)

$$\text{Minimize} \sum_{e \in \overline{E}} c_e x_e$$

s.t.

(1) $$\sum_{e \in \delta(v) \setminus F} x_e \geq \sum_{e \in F} x_e - |F| + 1$$ (cocircuit inequalities)

$(v \in V, F \subseteq \delta(v), |F|$ odd if v is R-even, $|F|$ even if v is R-*odd*$)$

(2) $$\sum_{e \in \delta(S)} x_e \geq 2 \; (S = \cup_{i \in P} V_i, P \subset \{1, \ldots, p\}, P \neq \emptyset)$$

(3) $$x_e \in \{0, 1\} \qquad (e \in \overline{E})$$

- $\delta(S) = \{$ edges with one extremity in S $\}$
- v is R-even (R-odd) if it is incident to an even (odd) number of edges of R

Co-circuit inequalities (1)

$$(1) \qquad \sum_{e \in \delta(v) \setminus F} x_e \geq \sum_{e \in F} x_e - |F| + 1$$

$(v \in V, F \subseteq \delta(v), |F|$ odd if v is R-even,
$|F|$ even if v R-*odd*)

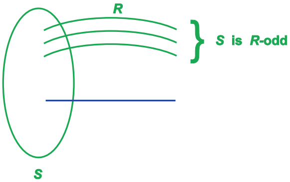
generalize to

$$(1') \qquad \sum_{e \in \delta(S) \setminus F} x_e \geq \sum_{e \in F} x_e - |F| + 1$$

$(F \subseteq \delta(S), |F|$ odd if S is R-even,
$|F|$ even if S is R-*odd*)

Special cases of (1′)

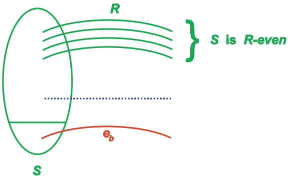
R-odd inequalities (1″) (Corberán, Sanchis, 1994)



at least one extra edge must be selected

$$\sum_{e \in \delta(S)} x_e \geq 1 \quad (S \text{ is } R\text{-odd}, F = \emptyset)$$

R-even inequalities (1‴) (Ghiani, Laporte, 2000)



at least one extra edge must be selected if $x_{e_b} = 1$

$$\sum_{e \in \delta(S) \setminus \{e_b\}} x_e \geq x_{e_b} \quad (S \text{ is } R\text{-even}, F = \{e_b\})$$

In practice, (1″) and (1‴) + (2), (3) are sufficient to obtain a feasible solution.

Polyhedral properties (Ghiani and Laporte, 2000)

All constraints of the model are facet defining.

Branch-and-cut algorithm

1. Upper bound: compute upper bound $\bar{z}$ on z^* (Frederickson).
2. First node of tree:
   Relaxed problem:
   - one connectivity constraint per component
   - one cocircuit inequality ($F = \emptyset$) for each R-odd vertex
3. Termination text: if list empty, stop. Otherwise select problem with least LB.
4. Solve subproblem. If undominated and feasible, update $\bar{z}$, and go to 3.
5. Add cuts (separation heuristics for violated constraints). If possible, to to 3. If not, go to 6.
6. Branch on a fractional variable. Insert subproblems in list, go to 3.

$$\begin{array}{ll} x' = 0 & x'' = 0 \\ x' = 1 & x'' = 1 \\ x' = 0 & x'' = 1 \end{array} \qquad x' + x'' = x$$

$\lvert V \rvert$	π	p	Succ	Root	Connect	R-odd	R-even	LB/z^*	Nodes	Seconds
	0.30	9.6	5	5	12.0	33.4	3.6	1.000	1	0.8
7 × 7	0.50	8.4	5	3	10.4	40.8	26.6	0.995	1.8	0.8
	0.70	6.0	5	4	8.0	39.2	28.8	0.996	3.0	1.2
	0.30	20.2	5	1	39.4	187.8	635.2	0.986	49.4	23.4
10 × 10	0.50	11.4	5	1	19.6	246.8	1399.2	0.997	33.8	56.4
	0.70	6.0	5	3	7.8	114.4	105.8	0.999	5.8	17.8
	0.30	26.8	5	1	55.6	435.4	1480.8	0.990	96.6	88.6
12 × 12	0.50	16.8	5	1	19.2	116.0	23.8	0.999	2.6	95.0
	0.70	5.0	5	3	8.6	379.6	680.2	0.998	27.4	122.6
	0.30	41.8	5	0	50.2	179.8	221.2	0.997	7.0	110.4
15 × 15	0.50	22.0	4	0	54.0	712.5	3684.7	0.995	113.5	580.7
	0.70	7.2	5	0	8.0	312.6	564.8	0.999	11.0	629.4
	0.30	55.8	5	1	189.0	1114.4	5587.6	0.997	199.8	820.8
17 × 17	0.50	26.4	5	0	33.0	441.8	969.8	0.998	21.4	964.6
	0.70	6.6	4	0	6.8	522.5	590.7	0.999	12.0	1368.0

Computational results for Type 3 graphs

$\|V\|$	π	p	Succ	Root	Connect	R-odd	R-even	LB/z^*	Nodes	Seconds
50	0.30	8.4	5	5	10.4	32.0	0.5	1.000	1.0	0.8
	0.50	8.0	5	3	11.2	55.0	21.4	0.999	4.2	1.6
	0.70	6.6	5	3	9.2	62.8	63.8	0.996	7.8	2.2
100	0.30	19.0	5	3	23.0	92.5	12.0	0.998	1.8	4.25
	0.50	14.8	5	3	18.4	122.6	66.2	0.999	5.8	9.0
	0.70	6.0	5	4	11.8	171.4	199.0	0.999	11.4	16.8
150	0.30	29.0	5	1	44.3	182.3	351.5	0.995	13.5	29.0
	0.50	19.6	5	2	26.2	226.6	144.0	0.996	8.2	50.2
	0.70	8.6	5	1	21.6	344.0	245.0	0.998	13.4	85.8
200	0.30	38.0	5	1	47.0	241.7	175.3	0.997	6.5	71.0
	0.50	22.0	5	2	30.5	289.3	310.3	0.998	8.5	181.8
	0.70	9.6	5	2	11.6	342.2	344.2	0.999	6.6	241.8
250	0.30	49.6	5	0	70.5	244.3	335.8	0.997	18.0	194.0
	0.50	30.2	5	2	34.6	295.0	111.6	0.998	7.0	420.0
	0.70	9.8	5	1	10.8	350.0	79.3	0.999	2.5	563.5
300	0.30	57.6	4	0	82.3	302.8	280.2	0.997	23.8	320.8
	0.50	32.4	5	0	52.1	324.8	420.5	0.998	20.2	410.2
	0.70	9.6	4	0	26.6	341.9	370.8	0.998	21.9	497.8

Comparison

1. Christofides et al., 1981
   Average of 1943 branch-and-bound nodes for $|V| \leq 84$
2. Corberán and Sanchis, 1984
   Visual branch-and-cut
3. Letchford, 1996
   $|V| = 50 : LB/z^* = 0.9972$
   Only the root node is solved; no branching implemented
4. Ghiani, Laporte, 2000
   $|V| = 50 : LB/z^* = 0.998$
   First full branch-and-cut algorithm

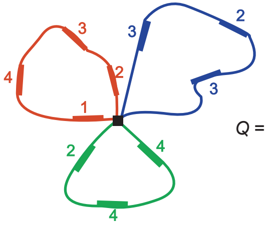
More recent results for the RPP

- Fernández, Meza, Garfinkel, Ortega (*Operations Research*, 2003): New formulation using flow variables
- Corberán, Romero, Sanchis (*Mathematical Programming*, 2003): Polyhedral analysis and lower bounds for the mixed general routing problem
- Blais and Laporte (*JORS*, 2003): Exact solution of the mixed general routing problem through graph transformations
- Corberán, Sanchis, Plana (*Networks*, 2007): Branch-and-cut algorithm for the windy general routing problem

The Capacitated Arc Routing Problem

The capacitated arc routing problem (CARP)

- Required edges have a demand q_{ij}
- Several vehicles of same capacity Q based at the depot



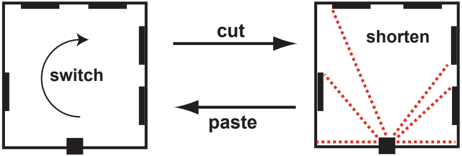
Literature review

- Introduced by Golden and Wong (1981)
- Several heuristics:

Path-Scanning (PS)	Golden et al., 1983
Augment-Merge (AM)	Golden et al., 1983
Construct-Strike (CS)	Christofides, 1973
Modified CS (MCS)	Pearn, 1989
Modified PS (MPS)	Pearn, 1989

CARPET: A tabu search heuristic for the CARP (undirected) (Hertz, Laporte, Mittaz, *Operations Research*, 2000)

1. 1st feasible solution
   - Solve RPP using Frederickson's (1979) heuristic
   - Cut solution into feasible routes
2. Post-Opt
   - Paste: merge routes (into a single route)
   - Switch: diversification strategies – reverse chains $(v, \ldots, v)$
   - Cut
   - Shorten: applied to each route
3. Tabu search
   - Move edges to neighbour routes
   - Infeasible routes are considered during the search



More recent heuristics for the undirected CARP:

- Hertz and Mittaz (*Transportation Science*, 2001): Variable neighbourhood search
- Beullens, Muyldermans, Cattrysse, Van Oudheusden (*EJOR*, 2003): Guided local search
- Doerner, Hartl, Maniezzo, Reimann (*Lecture Notes in Computer Science*, 2004): Ant colony optimization
- Lacomme, Prins, Ramdane-Cherif (*Annals of Operations Research*, 2004): Memetic algorithm
- Belenguer, Benavent, Lacomme, Prins (*Computers & Operations Research*, 2006): Lower bounds and heuristics for the mixed CARP
- Brandão and Eglese (*Computers & Operations Research*, 2008): Tabu search

Computational results on the DeArmon (1981) instances

Instance Number	$\|V\|$	$\|E\|$	PS	AM	CA	MCA	MPS	CARPET (T) F_0	$T=1$	$T=10$	CARPET	Best Known	$\underline{F}$	Seconds
1	12	22	**316**	326	331	323	**316**	337	323	323	**316**	316	316	17.1
2	12	26	367	267	418	345	355	345	345	345	**339**	339	339	28.0
3	12	22	289	316	313	**275**	283	289	**275**	**275**	**275**	275	275	0.4
4	11	19	320	290	350	**287**	292	330	**287**	**287**	**287**	287	287	0.5
5	13	26	417	383	475	386	401	414	406	**377**	**377**	377	377	30.3
6	12	22	316	324	356	315	319	323	315	**298**	**298**	298	298	4.6
7	12	22	357	**325**	355	**325**	**325**	**325**	**325**	**325**	**325**	325	325	0.0
10	27	46	416	356	407	366	380	398	382	358	352	348	344	330.6
11	27	51	355	339	364	346	357	350	332	328	317	311	303	292.2
12	12	25	302	302	364	**275**	281	283	283	**275**	**275**	275	275	8.4
13	22	45	424	443	501	406	424	427	417	409	**395**	395	395	12.4
14	13	23	560	573	655	645	566	560	516	506	**458**	458	448	111.8
15	10	28	592	560	560	**544**	551	564	556	548	**544**	544	536	13.1
16	7	21	102	102	112	102	**100**	104	102	**100**	**100**	100	100	2.6
17	7	21	**58**	**58**	**58**	**58**	**58**	**58**	**58**	**58**	**58**	58	58	0.0
18	8	28	131	131	149	**127**	131	131	129	**127**	**127**	127	127	9.2
19	8	28	93	**91**	**91**	**91**	93	**91**	**91**	**91**	**91**	91	91	0.0
20	9	36	168	170	174	**164**	167	174	172	**164**	**164**	164	164	1.5
21	11	11	57	63	63	63	**55**	63	57	**55**	**55**	55	55	1.1
22	11	22	125	123	125	123	123	125	125	123	**121**	121	121	51.5
23	11	33	168	158	165	**156**	163	164	158	**156**	**156**	156	156	6.1
24	11	44	207	204	204	**200**	202	206	204	202	**200**	200	200	18.3
25	11	55	241	237	237	**233**	244	239	239	237	235	233	233	186.3
Average Deviation			7.2%	5.6%	13.9%	3.9%	4.4%	6.6%	3.4%	1.4%	0.2%			
Worst Deviation			22.3%	25.1%	43.0%	40.8%	23.6%	22.3%	12.7%	10.5%	1.9%			
Number of Optima			2	3	1	12	5	3	5	13	18			
Number of Best			2	3	2	12	5	3	5	13	20			

A new Chinese Postman problem

An Adaptive Large Neighborhood Search for a Synchronized Arc Routing Problem

M. Angélica Salazar-Aguilar
André Langevin
Gilbert Laporte

CIRRELT

Outline

Synchronized Arc Routing Problem for Snow Plowing Operations

Given a network of streets and a fleet of snow plowing vehicles based at a depot, the SyARP consists of finding a set of routes such that all streets, some of which have multiple lanes, are plowed by using synchronized vehicles, and the completion time of the longest route is minimized.

- The street segments have one or two directions.
- The number of lanes goes from 1 to 3 in each direction.
- All lanes belonging to the same segment (in the same direction) must be plowed simultaneously.
- Deadheading is allowed.

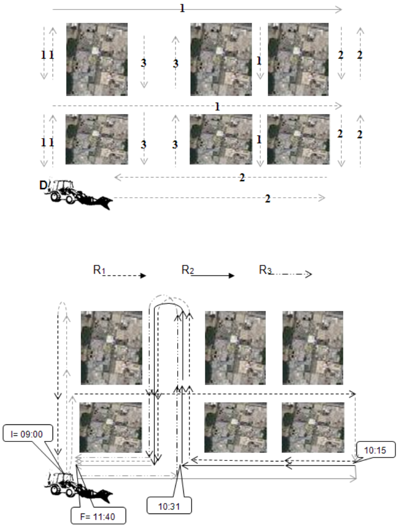


Figure 1: City map and synchronized routes for three vehicles.

Optimization Model

Decision Variables

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

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

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

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

Optimization Model

Parameters

t_{ij} is the servicing time of arc (i, j)
t'_{ij} is the traversing time of arc (i, j)
R is the set of available vehicles
e is the maximum number of arcs included in any route $r \in R$
n_{ij} is the number of lanes on arc (i, j)

Artificial vertex $0'$

This vertex represents the artifical depot, used as the end point of all routes. It is needed because any route may pass through the depot several times.

Optimization Model

Makespan Minimization

$$\text{Min} \quad z = \max_{r \in R, k \in K} \left\{ t^k_{00'r} \right\}$$

All vehicles start and end at a depot $0'$

$$x^1_{0'0r} + y^1_{0'0r} = 1 \qquad r \in R$$

$$\sum_{(0,j) \in A} x^2_{0jr} + y^2_{0jr} = 1 \qquad r \in R$$

$$\sum_{k \in K} x^k_{00'r} + y^k_{00'r} = 1 \qquad r \in R$$

Optimization Model

All lanes should be plowed

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

Time

$$(t^k_{ijr} + t_{ij}) y^k_{ijr} + (t^k_{ijr} + t'_{ij}) x^k_{ijr} + w^k_{ijr} = \sum_{(j,h) \in A \cup (0,0')} t^{k+1}_{jhr} (x^{k+1}_{jhr} + y^{k+1}_{jhr})$$

$$r \in R, (i,j) \in A; k \in K \setminus \{1, e\}$$

Optimization Model

At most one arc per position in each route

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

Synchronization of routes

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

$$(i,j) \in A, r \in R; k \in K \setminus \{1\}$$

Optimization Model

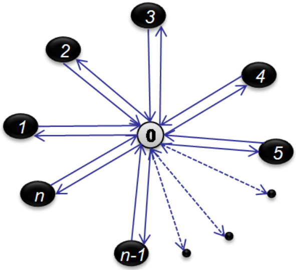
Routes Connectivity

$$\sum_{(i,j)\in A} (x_{ijr}^{k} + y_{ijr}^{k}) \leq \sum_{(j,h)\in A\cup(0,0')} (x_{jhr}^{k+1} + y_{jhr}^{k+1})$$

$$r \in R, j \in V; k \in K \setminus \{1, e\}$$

SyARP Complexity

Given an instance of the SyARP in which $\lambda = 1$, all arcs are incident to the depot and the servicing and traversing time on any arc are such that $t_{0i} = t'_{0i}$ and $t_{i0} = t'_{i0}$. The recognition problem consists of determining whether there exists a solution of cost at most equal to $\bar{z}$.



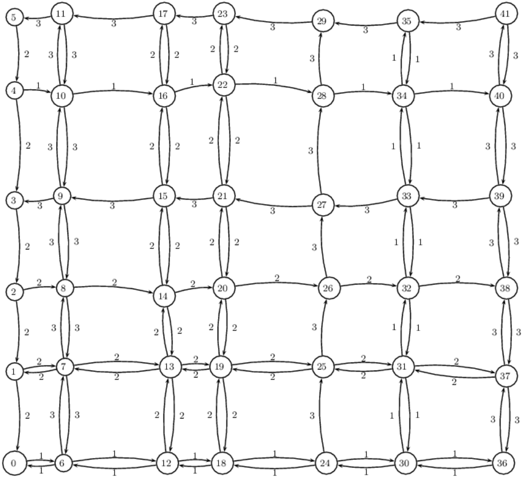
It is equivalent to the recognition version of a *Bin Packing Problem* with $|R|$ bins of capacity $\bar{z}$ and items of sizes $(t_{0i} + t_{i0})$, which is NP-complete. Then the SyARP is NP-hard.

Initial solutions

- Given a multigraph $G = (V, A)$, a set R of available vehicles, and a number λ.
- Let $\{C_1, ..., C_\lambda\}$ a partition of A.
- Let μ_l equal to l if $C_l \neq \emptyset$ and to 0 otherwise.
- Assume that $|R| \geq \mu_\lambda$.
- Since the objective function is to minimize the makespan, we try to service the different classes simultaneously. Therefore $|R|$ should be at least $\sum_{l=1}^{\lambda} \mu_l$. If it is not satisfied, we determine the largest α such that $\sum_{l=\lambda-\alpha}^{\lambda} \mu_l \leq |R|$ and we merge the classes $C_1, ..., C_{\lambda-\alpha}$ into a single class $C_{\lambda-\alpha}$. Otherwise, $\alpha = \lambda - 1$.
- We assign v_l vehicles to each class C_l such that v_l must be a multiple f_l of μ_l and $\sum_{l=\lambda-\alpha}^{\lambda} v_l = |R|$. As there are many ways to satisfy this equation, there also exists several possible fleet distributions $(f_{\lambda-\alpha}, ..., f_\lambda)$.
- For each fleet distribution $(f_{\lambda-\alpha}, ..., f_\lambda)$ we build a feasible solution of the SyARP.

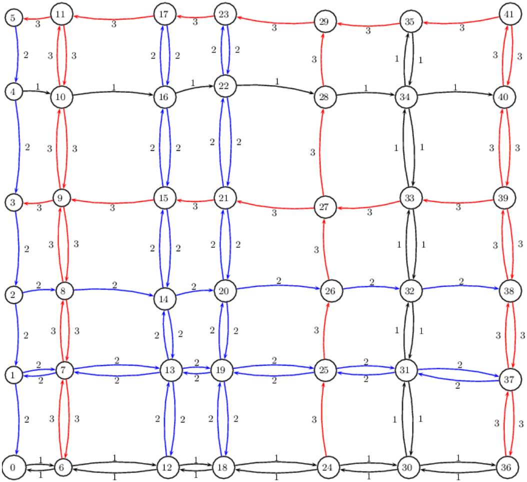
Initial solutions

- $G(V, A)$, $\lambda = 3$, and $|R| = 12$.



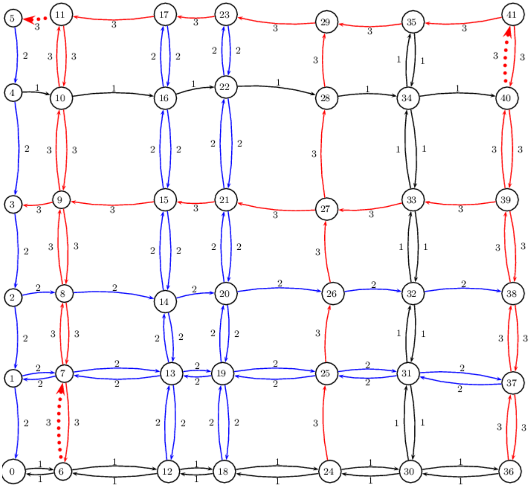
Initial solutions

- Arcs classification according to the number of lanes.



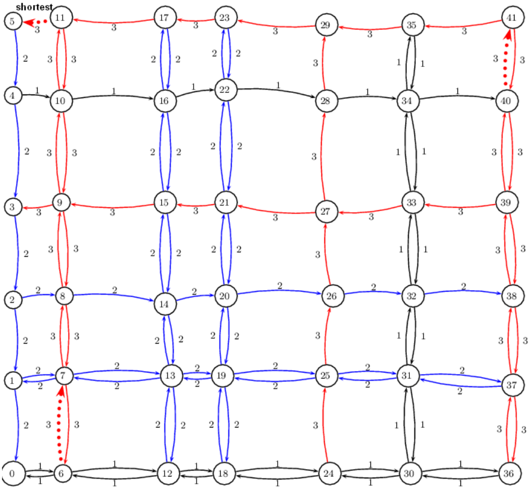
Initial solutions

- For each class C_l choose f_l seed arcs. Suppose $l = 3$ and $f_3 = 3$.



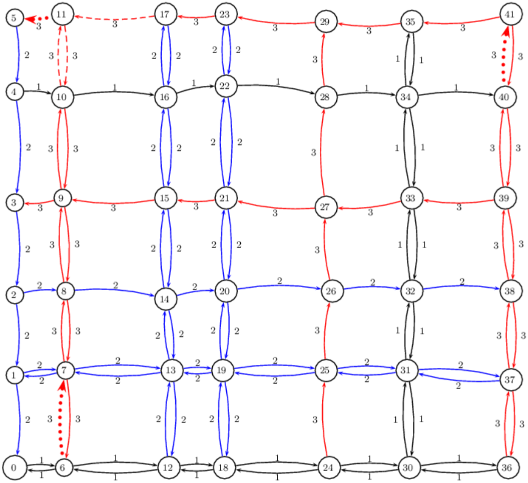
Initial solutions

- Routes extension by keeping route lengths balanced.



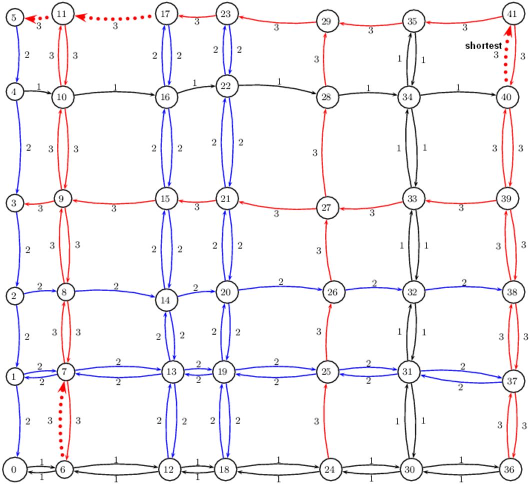
Initial solutions

- Assigning arcs to routes.



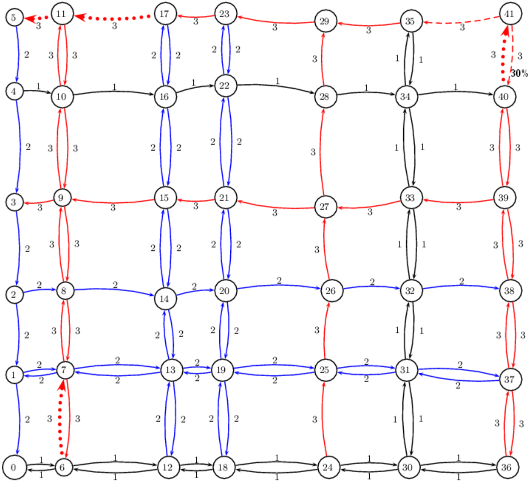
Initial solutions

- Assigning arcs to routes. All links are done by means of shortest paths.



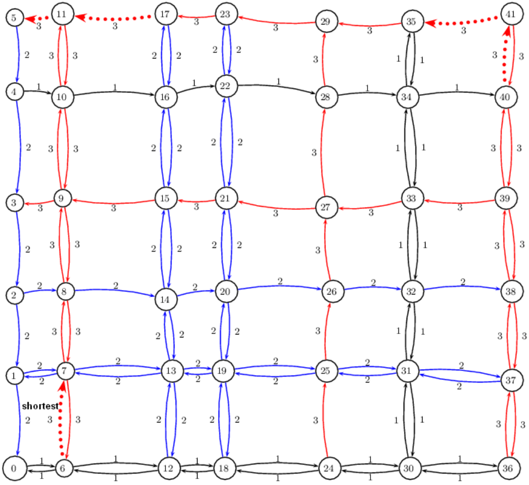
Initial solutions

- Assigning arcs to routes by minimizing the small circuits.



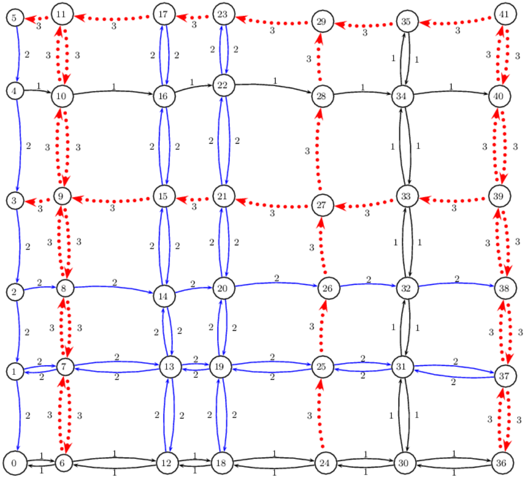
Initial solutions

- Assigning arcs to routes.



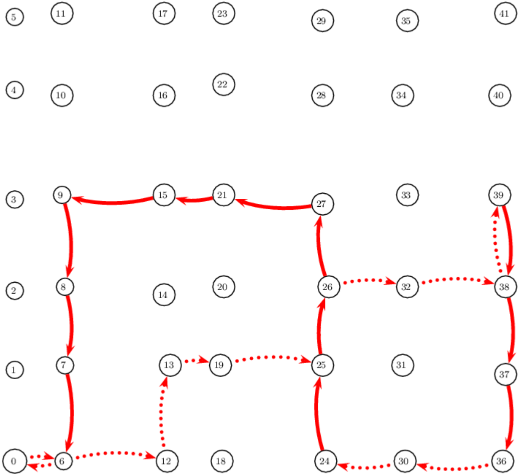
Initial solutions

- The assignment process finishes when all arcs $(i, j) \in C_l$ have been assigned.



Routes Ending

- When all arcs $(i, j) \in C_l$ have been assigned, the current routes are connected to the depot by means of shortest paths.



Algorithm 1 Improvement phase of the ALNS heuristic

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

Repair/Destroy Operators

Arcs Sequence Removal-Insertion

A sequence of arcs is removed from the longest route and is inserted in another route.

- N1: The insertion point is after the arc that has the shortest path from the initial arc of the sequence.
- N2: The insertion point is randomly chosen.

Repair/Destroy Operators

Interchange of Arc Status

- N3: An arc (l, m) serviced by the longest route is randomly chosen. If another route uses this arc just for traversal, the statuses of this arc in these routes are interchanged.
- N4: The route with the shortest ending time is chosen and an arc (i, j) which is just traversed by the route is randomly selected. Another route servicing the arc (i, j) is identified and an interchange of statuses in the arc (i, j) is carried out in both routes.

Repair/Destroy Operators

First Traversed-First Serviced

N5: this procedure is applied to each route $P_d \in P$ in such a way that the arcs that are both serviced and traversed by P_d are reordered in an attempt to reduce the makespan. Thus, each arc that is both serviced and traversed by P_d will be forced to be serviced the first time it is traversed in route P_d.

Instances tested

- The algorithm was coded in C++ and compiled on a 2592.574 MHz AMD Opteron(tm) processor 285 with 1GB of RAM under the Linux operating system.
- Three instance sets with 42, 180, and 300 vertices were generated on same grid shape.

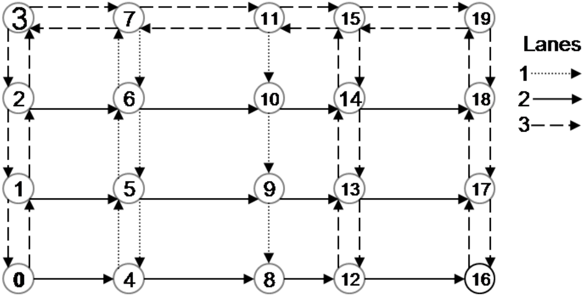


Figure 2: Instances shape.

Instances features

- Fifteen instances were generated for each set.
- The minimum number of arcs for each set was 113, 499, and 795, respectively.
- The instances have arcs with one, two or three lanes.
- The number of available vehicles for each set was 12, 18, and 25, respectively.

Table 1: Best found solutions reported by ALNS.

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

Table 2: Best found solutions reported by ALNS, instances set (180,18).

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

Table 3: Best found solutions reported by ALNS, instances set (300,25).

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

Instances (42,12)					
gap(%)	**N1**	**(N1,N2)**	**N\(N4,N5)**	**N\(N5)**	**N**
Min	3.00	3.24	0.00	0.00	0.00
Ave	7.23	7.24	3.80	2.97	**0.76**
Max	15.47	15.47	12.10	11.09	4.31
Instances (180,18)					
gap(%)	**N1**	**(N1,N2)**	**N\(N4,N5)**	**N\(N5)**	**N**
Min	1.70	1.70	0.00	0.00	0.00
Ave	9.17	9.27	2.07	2.22	**1.04**
Max	26.85	20.21	6.91	6.91	5.76
Instances (300,25)					
gap(%)	**N1**	**(N1,N2)**	**N\(N4,N5)**	**N\(N5)**	**N**
Min	0.19	0.19	0.00	0.00	0.00
Ave	9.79	9.86	2.31	3.05	**1.09**
Max	17.26	17.26	7.01	7.26	6.33

Table 4: Computation time for all instance sets.

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

Table 5: Average percent improvement of the improvement phase over the construction phase in our ALNS heuristic.

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

The average improvement ranges from 16.13% in the largest instance set (N300V25L3) to 46.78% in the smallest instance set (N42V12L3).

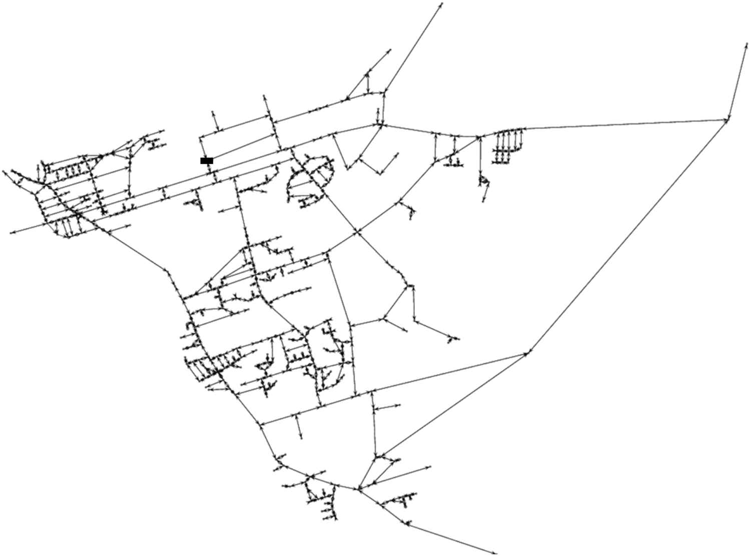


Figure 3: Case study: Dieppe, New Brunswick, Canada has approximately 144 km of roads to plow, representing 363 lane-km.

Dieppe instance

- The Dieppe street network contains 430 vertices and 1056 arcs which have one or two lanes.
- We have studied the case where there are eight available vehicles traveling and servicing at a speed of 48 and 12 km/h, respectively.

Results

Our ALNS heuristic was able to generate in 538 seconds a set of routes with a makespan of 3 hours and 28 minutes.

Synchronized Routes

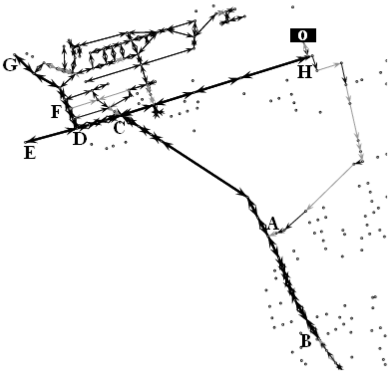


Figure 4: Synchronized vehicles for servicing two-lane streets.

Conclusions

1. We have introduced a synchronized arc routing problem for snow plowing operations.
2. A mixed integer non-linear model has been presented.
3. A solution procedure based on adaptive large neighborhood search metaheuristic was proposed.
4. Five destroy/repair operators were developed.
5. The performance of the proposed ALNS procedure was evaluated over large instances.
6. In addition, the performance of our ALNS procedure was evaluated on a real-world instance.