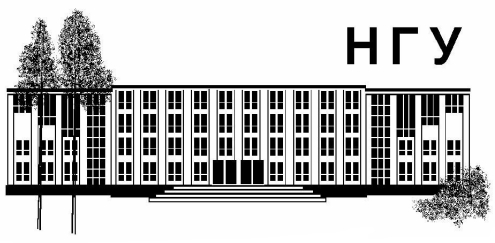
Fixed-parameter algorithms for serving links in transportation networks

René van Bevern

rvb@nsu.ru

Novosibirsk State University and Sobolev Institute of Mathematics



Novosibirsk, December 12th, 2016

1 Introduction

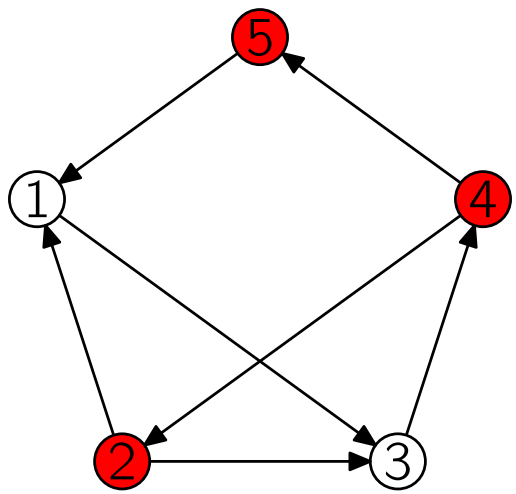
Find most profitable routes for a fleet of vehicles to serve a set of clients.

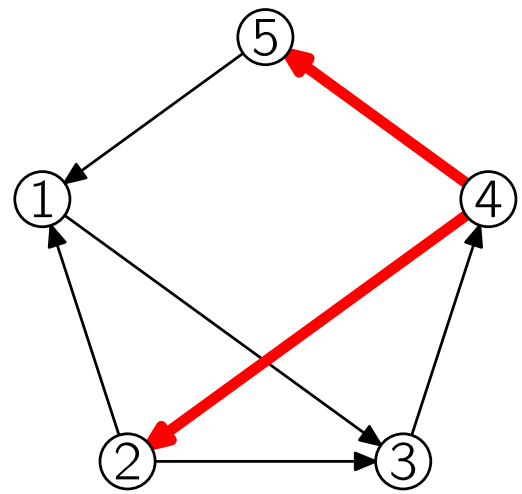
- ▷ transportation network = graph with arc lengths.
- ▷ clients = vertices or arcs of the graph.

Input: digraph $G = (V, A)$, arc lengths $c\colon A \to \mathbb{N}$, and a subset

$R \subseteq V$ of *required vertices (red)* **or** $R \subseteq A$ of *required arcs (red)*.

(Asymmetric) Subset TSP

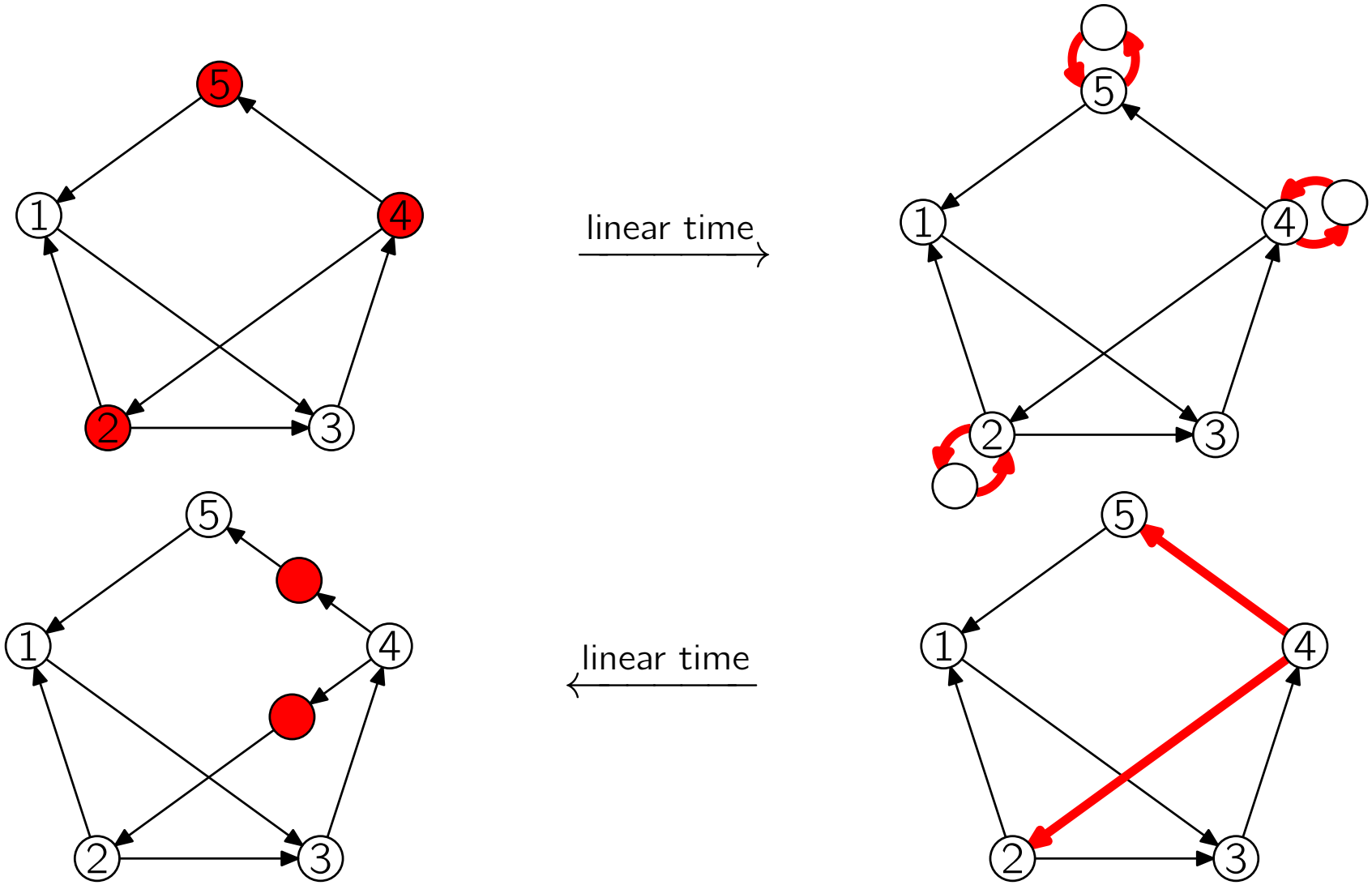




Rural Postman

Task: find shortest cycle containing all vertices or arcs of R, respectively.

Node and arc variants can be easily translated into each other



The node variant has a much longer history

Why study "arc routing" then?

When *all* clients along a road or the roads themselves have to be served, then the arc routing model yields much smaller networks.

- ▷ household waste collection, meter reading,
- ▷ street sweeping, street salting,
- ▷ inspection of pipe systems by remote-controlled robots.

Arc Routing problems are sometimes significantly easier:

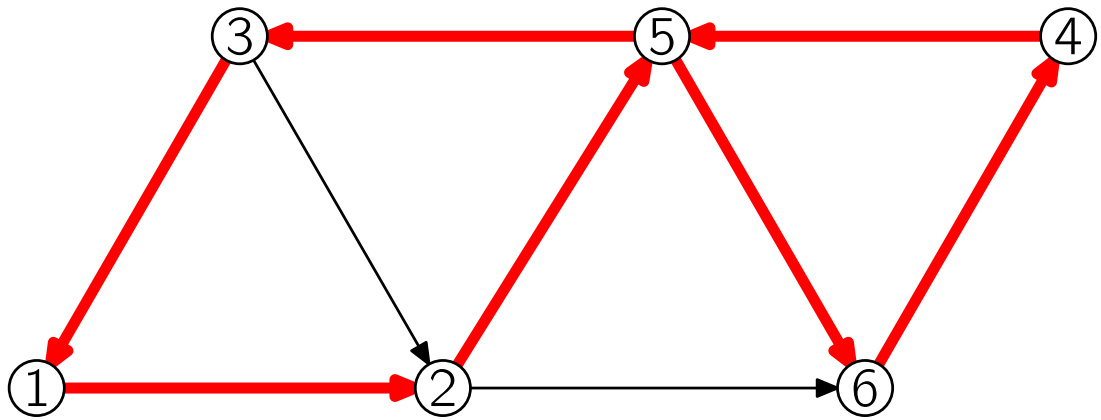
- ▷ Shortest cycle that visits all vertices $\rightsquigarrow$ NP-hard TSP problem.
- ▷ Shortest cycle that visits all arcs $\rightsquigarrow$ $O(n^3)$-time solvable.

Exploit the structure of the graph induced by the required arcs.

2 Eulerian demands

Definition 2.1. A connected graph is *Eulerian* if one of the following holds:

- ▷ It contains an *Eulerian cycle*, which visits each arc exactly once.
- ▷ Each vertex has the same indegree as outdegree, that is, is *balanced*.



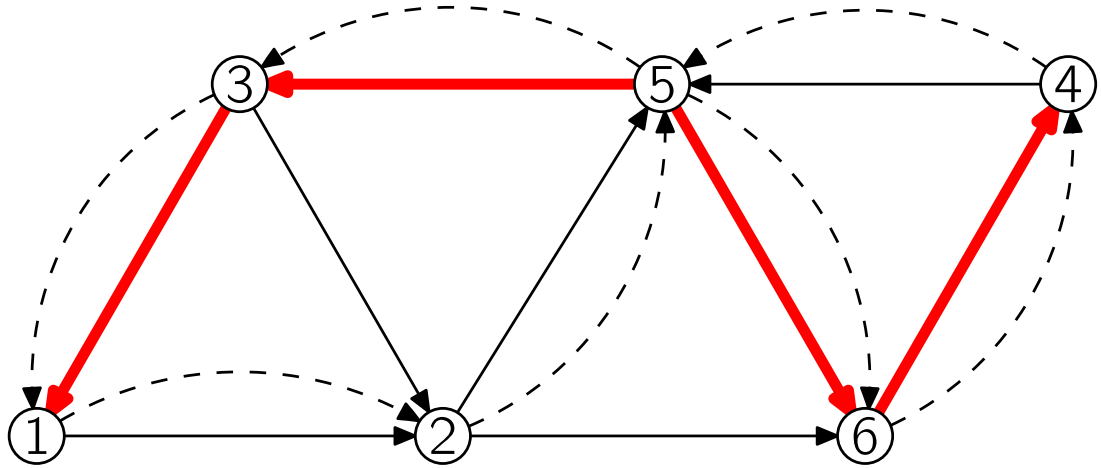
Consider subgraph $G[R]$ consisting only of the required arcs R.

- ▷ If it is Eulerian, then its Eulerian cycle is the optimal solution.

$\rightsquigarrow$ can be found in linear time.

3 Connected demands

Task: compute min-cost tour T (dashed arcs) visiting the set R of red arcs.



Observation 3.1. $G[T]$ is a min-cost Eulerian supermultigraph of $G[R]$ consisting of arcs of $G \rightsquigarrow$ compute this graph, its Euler tour will give T.

If $G[R]$ is connected: compute min-cost arc multiset R^* of G

- ▷ such that the vertices of $G[R \uplus R^*]$ are balanced.

$\rightsquigarrow$ $G[R \uplus R^*]$ is also connected and thus Eulerian.

Optimally balancing graphs

Goal: Given a set R of arcs in G, compute a min-cost multiset of arcs R^* of G such that all vertices of $G[R \uplus R^*]$ are balanced.

Definition 3.2. For a vertex v of G, let $b(v) := \text{indeg}_{G[R]}(v) - \text{outdeg}_{G[R]}(v)$.

Multiset R^* has to contain $b(v)$ more out-arcs than in-arcs of each v.

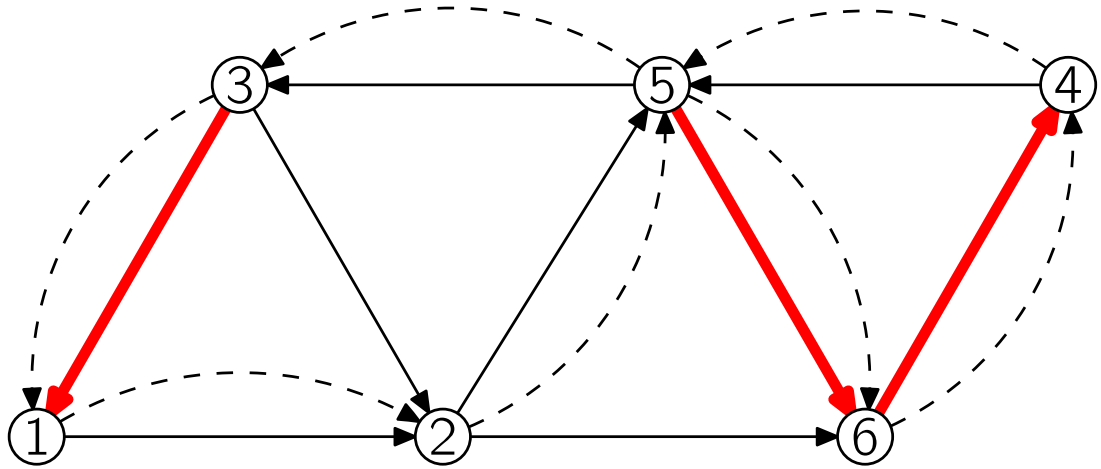
$\rightsquigarrow$ Min-Cost Flow problem, solvable in $O(n^3 \log n)$ time:

- ▷ *demand and supply* of vertex v are given by $b(v)$;
- ▷ cost of unit of flow = arc length.

Theorem 3.3. If the required arcs R induce a connected subgraph, then Rural Postman is solvable in $O(n^3 \log n)$ time.

4 Limited deadheading

Task: compute a min-cost tour T visiting all required arcs in R.



Or: compute min-cost arc multiset R^* such that $G[R \uplus R^*]$ is Eulerian.

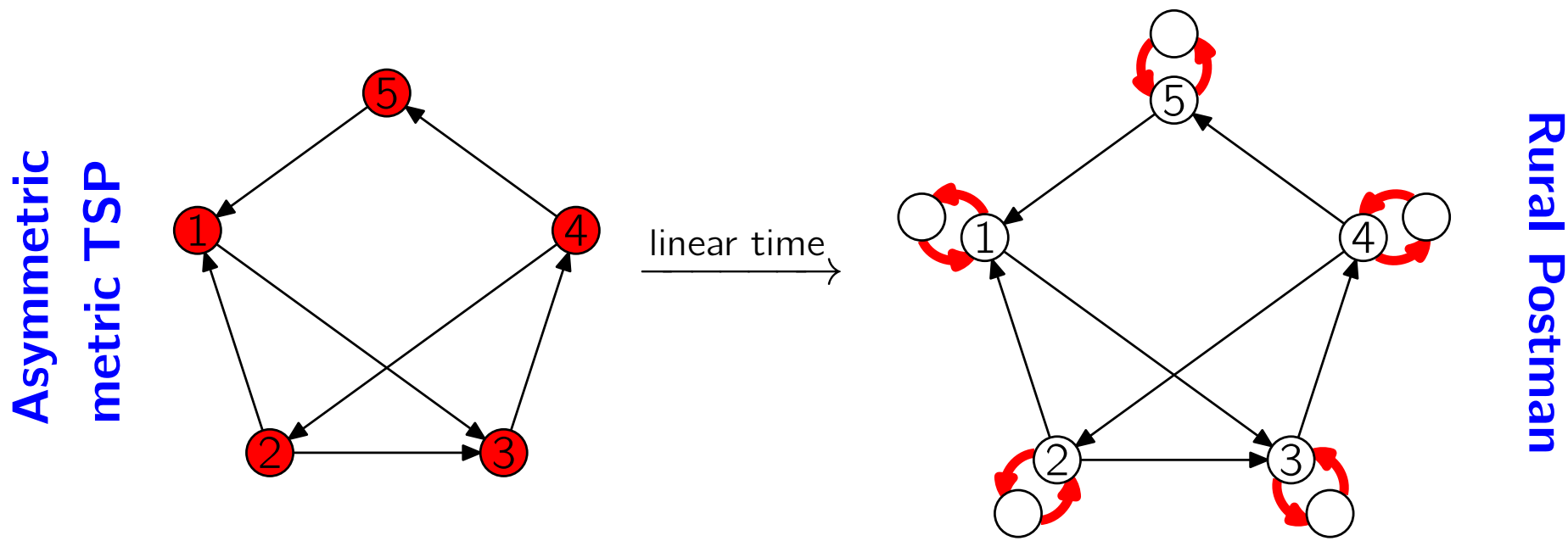
Observation 4.1. We always pay cost $c(R)$ $\rightsquigarrow$ can only minimize cost $c(R^*)$.

- ▷ $c(R^*)$ is the cost of *deadheading* = traversing without serving,
- ▷ $|R^*|$ is precisely the number of deadheading arcs.

Connecting is the hard part

Goal: Given a set R of arcs in G, compute a min-cost multiset of arcs R^* of G such that $G[R \uplus R^*]$ is Eulerian (connected and balanced):

- ▷ We have seen: *balancing* is easy using min-cost flows.
- ▷ *Connecting* is hard, even if all vertices in $G[R]$ are already balanced:



Rural Postman with few deadheading arcs

Proposition 4.2 (Dorn et al., SIDMA, 2013)**.** A min-cost multiset R^* with $|R^*| \leq k$ such that $G[R \uplus R^*]$ is Eulerian is computable in $O(4^k \cdot n^3)$ time.

Theorem 4.3. Rural Postman is solvable in $O(4^k \cdot n^3)$ time if we allow solutions with k deadheading arcs.

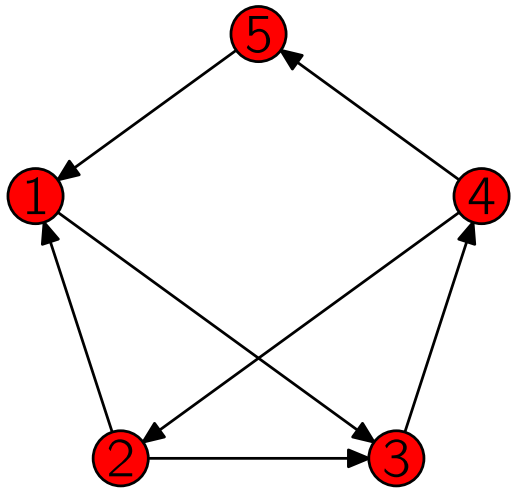
- ▷ cubic time for constant number of deadheading arcs,
- ▷ polynomial time if the number of deadheading arcs is $k \in O(\log n)$.

In other words,

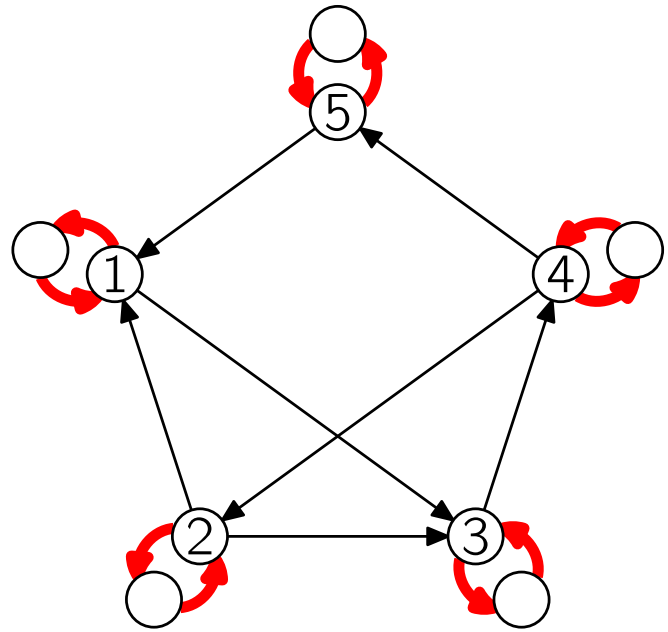
- ▷ Rural Postman is *fixed-parameter tractable*,
- ▷ when parameterized by the number k of deadheading arcs.

5 Few connected components

Asymmetric metric TSP (ATSP)



linear time



Rural Postman

Constant-factor approximability for ATSP is open since more than 30 years.

- ▷ There is a 1:1-correspondence between ATSP and Rural Postman tours.
- ▷ Rural Postman is at least as hard to approximate as ATSP.

The reduction creates many connected components consisting of red arcs.

How many connected components do realistic instances have?

Novosibirsk decomposes into

- ▷ ≈ 5 components for waste collection,
- ▷ 1 component for snow plowing.

Most benchmark instances:

- ▷ for generalizations of Rural Postman,
- ▷ from snow plowing, waste collection,
- ▷ have only one component.

We will now show a simple 2-approximation for Rural Postman with few components and generalize it to variants with multiple capacitated vehicles.

A simple 2-approximation for Rural Postman with C components

Goal: compute min-cost arc multiset R^* such that $G[R \uplus R^*]$ is Eulerian.

Approximation: select vertices $v_1, \ldots, v_C$, one of each component of $G[R]$.

- ▷ Compute min-cost multiset R_b such that $G[R \uplus R_b]$ is balanced.

  Works in $O(n^3 \log n)$ time. We have $c(R_b) \leq c(R^*)$.

- ▷ Compute min-cost cost tour R_c through $v_1, \ldots, v_C$.

  Works in $O(2^C C^2 + n^3)$ time via exact TSP. We have $c(R_c) \leq c(R^*)$.

Graph $G[R \uplus R_b \uplus R_c]$ is balanced and connected (Eulerian) and

$$c(R_b \uplus R_c) \leq 2c(R^*).$$

Theorem 5.1. Rural Postman is 2-approximable in $O(2^C C^2 + n^3 \log n)$ time.

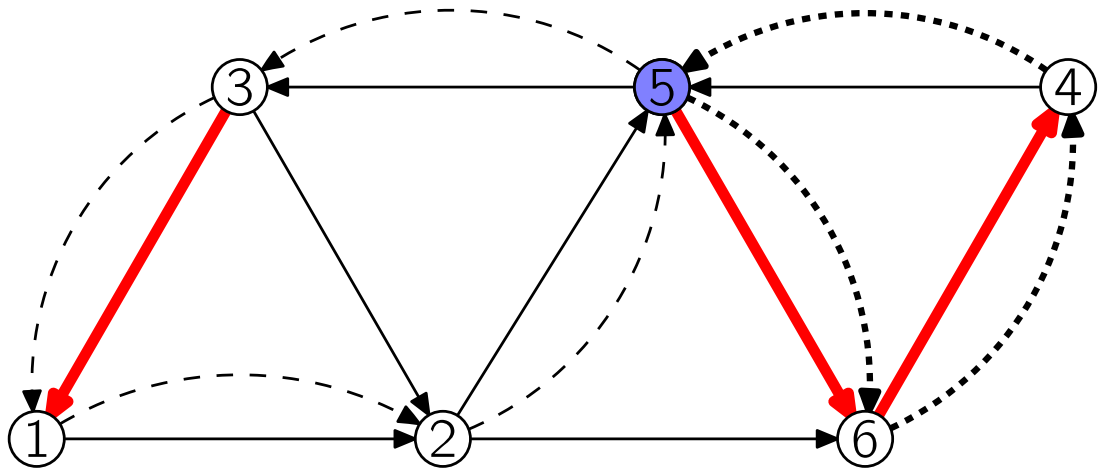
This can be used for the more general Capacitated Arc Routing problem...

6 Directed Capacitated Arc Routing (DCARP)

Input: directed graph $G = (V, A)$, depot $v_0 \in V$, travel costs $c\colon A \to \mathbb{N}$, demands $d\colon A \to \mathbb{N}$, and a vehicle capacity Q.

Task: find a set of vehicle routes of total minimum cost, each starting and ending in v_0, such that the demand of each arc is served by exactly one vehicle and each vehicle serves a total demand of at most Q.

Example 6.1. Red arcs have demand 1, capacity $Q = 2$, depot is violet.

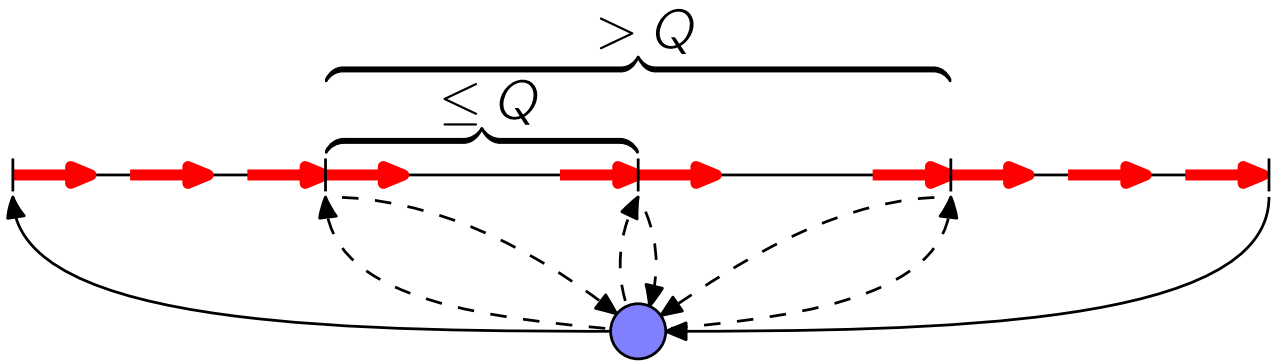


One vehicle takes dashed route, the other takes the dotted route.

Approximation for Directed CARP

Observation 6.2. Concatenating vehicle tours of optimal DCARP solution gives a Rural Postman tour visiting all positive-demand arcs and depot v_0.

1. Compute 2-approximate Rural Postman tour T (solid black) visiting depot v_0 and all positive-demand arcs (red).
2. Split it greedily into maximal subwalks of demand at most Q each.
3. Add a shortest path (dashed) from and to v_0 to each subwalk.



Theorem 6.3. DCARP is constant-factor approximable in $O(2^C C^2 + n^3 \log n)$ time or $O(\log C / \log \log C)$-approximable in polynomial time.

7 Experiments

Implemented and tested the $O(2^C C^2 \cdot n^3 \log n)$-time constant-factor approximation for Mixed CARP (directed and undirected edges).

Test instances

- ▷ well-known benchmark sets egl-large, lpr, and mval from the literature.

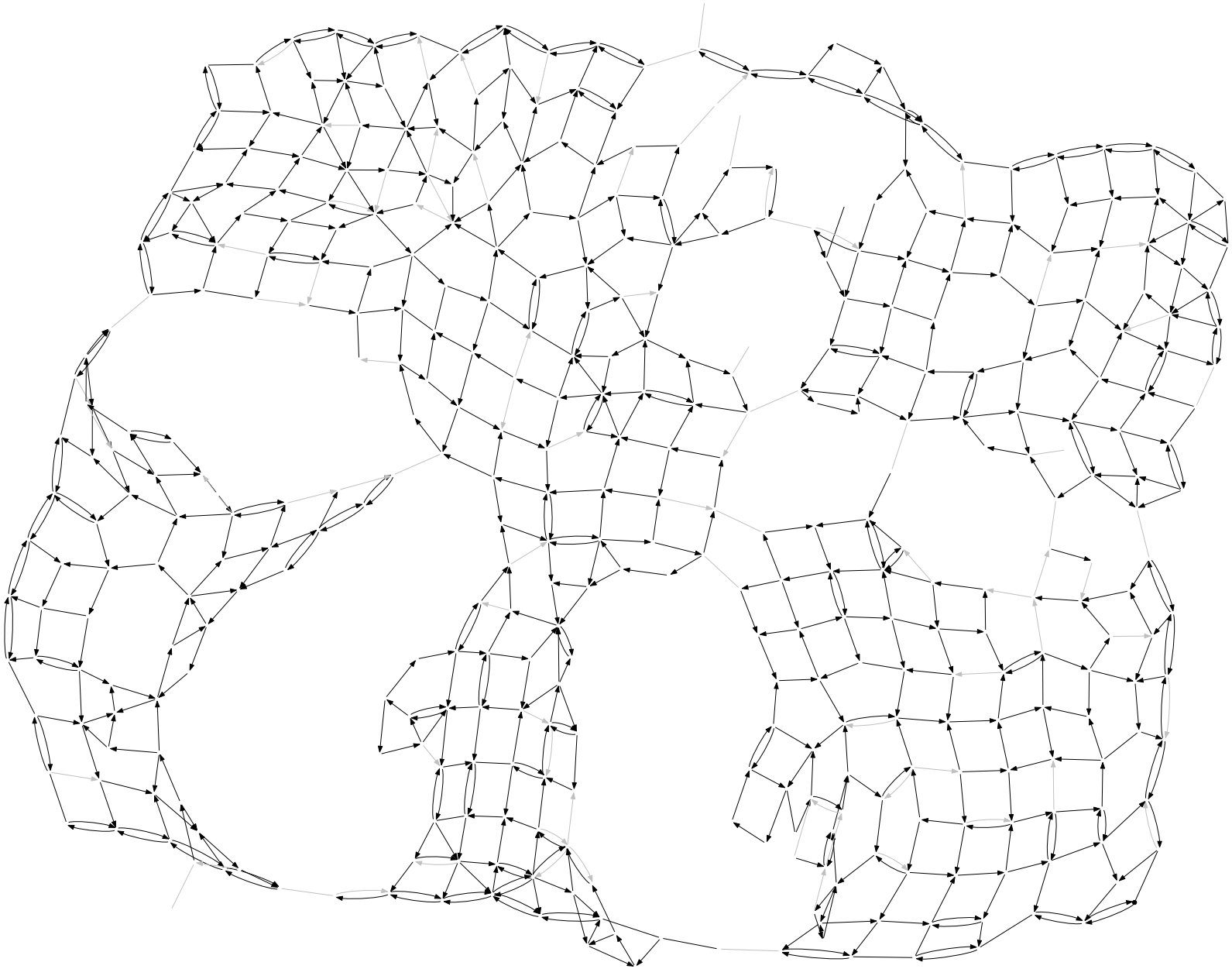
  In almost all of them the positive-demand arcs induce $C = 1$ component.

- ▷ a new "Ob" data set, which simulates cities divided by a river.

  Has $C \in [2, 5]$, which is still easy for a $O(2^C C^2 \cdot n^3 \log n)$-time algorithm.

Compared the cost of our solutions to

- ▷ lower bounds from integer linear programming (or relaxation),
- ▷ since optimal solutions for most instances are unknown.



Experimental results

Deviation from lower bound (ILP for "Normal", ILP relaxation for "Ob"):

Dataset	Normal min	Normal median	Normal max	Ob min	Ob median	Ob max
mval	0 %	8 %	19.2 %	10.8 %	23.3 %	35 %
lpr	0 %	0.3 %	0.4 %	0 %	0.34 %	0.5 %
egl-large	15.8 %	17.1 %	18.2 %	27.6 %	34 %	41 %

Our algorithm outperforms other polynomial-time algorithms

- ▷ on 15 of 34 mval instances ($\approx 44\,\%$), two solved optimally,
- ▷ on 7 of 15 lpr instances ($\approx 46.6\,\%$), two solved optimally,
- ▷ on all 10 egl-large instances (large real-world instances).

8 Conclusion

NP-hard arc routing problems are efficiently solvable in many cases:

- ▷ Rural Postman with connected demands,
- ▷ Rural Postman with few deadheading arcs,
- ▷ Approximate Directed/Mixed Capacitated Arc Routing with few connected components.

We solved the NP-hard cases using fixed-parameter algorithms,

- ▷ whose running time grows polynomially in the input size,
- ▷ exponentially only in small problem parameters.

The solution quality or the time for finding optimal solutions mostly depends on the number of connected components induced by arcs with positive demand.