How can arc routing techniques contribute to optimization of winter gritters routes

Subject:
Mathematics

Teodor Wójcik
Candidate M15-D-000703-0040

33 Copernicus LO Warsaw

Abstract

Aim of this paper is to explore applicability of graph theory in real life situations. More specific, arc routing techniques in modeling routes of winter gritters. Arc routing focuses on problems considering traversing graph edges. This paper presents various arc routing problems. Firstly basic theory of arc routing problems are introduced, then a problem corresponding to real life situation applicable to winter gritter path modeling problem is shown together with algorithm which produces optimal solution. Discussed algorithm is accompanied by simple example graphs that show steps of procedure. I also present my living district as a graph and apply final algorithm, which produces optimal path. Graphs and solutions are generated by a python program, which I coded using NetworkX graph library. To conclude I discuss vast applicability of aforementioned algorithms and possible improvement of solutions by considering additional factors that apply to winter gritting problem.

Word count: 145

Contents

1 Introduction

Development of technology helped ease our lifes and as one may think that technological advances benefited most from development of physics and chemistry but what I find more astonishing than the work of the materials is the way that despite great quantities our products seem to work together seamlessly just as cars and planes follow designated routes avoiding crashes. We may be unaware of it, but it is math that coordinates many aspects of our life. Graph theory enables us to model our surroundings, such as streets and crossings in language of mathematics. This gives opportunity to analyze and optimize many aspects of our life. One of these aspects is management of the flow of vehicles. Public services such as garbage collection, winter gritting and post offices operate not for benefit of individuals but rather for the well being of society and also most often have to deal with significant amount of request. That is why I find especially important to make their work more efficient. For the sake of this essay, I will concentrate on aspect of winter gritting as optimal route efficiency is for it particularly important. Due to the unpredictability of weather, winter gritters have to adapt to the changing conditions and operate fast despite their considerable sizes and weights. Preprocessed optimal routes are often essential for them.

Most intuitive way to represent our surroundings in language of graph theory would be identifying streets with edges and vertices with houses. It would then be a matter of Node Routing Problems (NRP), where every node has to be serviced. Winter gritters have to process streets, therefore it is preferred to represent vertices as intersections and edges as segments of streets. This leads to Arc Routing Problems, where arcs are in need of servicing.

Node Routing Problems deal with singular clients and routing an optimal servicing path, which is a most often case of goods pickup and delivery. Algorithms that has been invented for NRP allow to solve optimally complex and vast instances, but condition of today known ARP algorithms isn't that good. ARPs that closely mimic real life situations are NP-hard [2], no polynomial solutions are known, only integer linear programming methods are capable of providing optimal solutions, but as constraints grow exponential with size of graph they are limited only to smaller sized graph. Due to this fact this paper focuses on simplified ARP problems that are capable of delivering exact solutions in polynomial time. They operate in polynomial time thus aren't significantly limited by size of graph.

In this paper ARPs are defined as in [2] over a graph $G = (V, E \cup A)$, in which V is set of vertexes, E set of edges and A set of arcs. c_{ij} defines non-negative traversal cost of an edge between vertices (v_i, v_j), which is identified with length of an edge, so $c_{ij} = c_{ji}$. Degree d_i defines total number of arcs and edges that are incident to node v_i. When set A is empty, graph is called $undirected$. This paper considers undirected graphs only and definitions of problems will assume that A is empty. This simplification can be made because public servant's vehicles operate in best case scenario beyond traffic hours and are privileged to, having previously taken care of safety issues, drive wrong way.

2 Euler Tour Problem

2.1 Presentation

Historically first arc routing problem considered Konigsberg bridges. Formally problem states:

On a connected undirected graph $G = (V, E)$ find a closed tour that passes through every edge in E exactly once, or determine that no such tour exists.

In 1735 Leonhard Euler proved that such a tour could be defined only if every node in G has even degree. Euler work is considered to be first published paper in area of graph theory and topology [1] and aforementioned problem is thus known as Euler Tour Problem.

2.2 Algorithm Eulerian-cycle

INPUT: An Eulerian undirected graph $G = (V, E)$.
OUTPUT: An Eulerian tour T on G. [2]

Step 1.
Determine a tour T covering some edges in E. If T covers all edges in T then stop.

Step 2.
Consider any vertex v on T incident to an edge not on T. Form a second tour T' containing v and such that T and T' are edgedisjoint.

Step 3.
Let e_1, e_2 be two consecutive edges incident to v on T and let e_3, e_4 be two consecutive edges incident to v on T'. Merge T and T' into a tour T'', starting at v with e_1, following T until it meets v by way of e_2, then continue on T', starting with e_3 until v is met by way of e_4.

Step 4.
Set $T := T''$. If T covers E then stop, else return to Step 2.

3 Chinese Postman Problem

3.1 Presentation

In 1962 Chinese mathematician Kwan Mei-Ko formally stated Chine Postman Problem (CPP):

In a graph $G = (V, E)$ determine a tour of minimum cost that covers all edges in E.

If graph G was even, then applying ELURERIAN-CYCLE algorithm would produce a solution. In other case, edges should be added to graph G to create even graph G'. Eulerian Tour found in G' will correspond to a Chinese Postman Tour in G. To mantain minimum cost of the Chinese Postman tour added edges to G' must also be of minimum cost. To find these edges, perfect matching problem will be solved for odd vertices in G

3.2 Matching

Matching problems in graph theory deal with such transformations of graphs in which no common vertices exist. Matched vertex defines a vertex that is an endpoint of one of the edges in

the final matching. Particular example is *perfect matching*, defined as follows:

In a graph $G = (V, E)$ perfect matching is a set of pairwise non-adjacent edges covering in which no two edges share a common vertex and which covers all edges E

Algorithm used in this essay for defining perfect matching was developed by Jack Edmonds in 1961 and is called *blossom algorithm*. Due to its complexity details of blossom algorithm are beyond the scope of this paper, thus in calculations implementation from NetworkX algorithm library was used.

3.2.1 Algorithm Undirected Chinese Postman Problem

INPUT: An undirected graph $G = (V, E)$.
OUTPUT: A minimum cost chinese postman tour T on G.[2]

Step 1.
If G is even then determine a chinese postman tour T by means of the Eulerian-cycle algorithm, and stop.

Step 2.
Determine the set V_0 of odd degree vertices in G. Construct a complete graph G_0 with vertex set V_0. Define the cost of an edge (V_i, V_j) in G_0 as the length of the shortest chain SP_{ij} between V_i and V_j in G.

Step 3.
Find a minimum cost perfect matching in G_0 using blossom algorithm.

Step 4.
Set $G' := G$. For each edge (V_i, V_j) in the optimal matching, add to G' all edges that lie on SP_{ij}.

Step 5.
Determine an Eulerian tour T on G' by means of the Eulerian-cycle algorithm, and stop. T corresponds to an optimal chinese postman tour on G

3.3 Simple example graph visualization

To visualize Algorithm Undirected Chinese Postman Problem step by step transformations of graph $G1$ containing 5 nodes will be here presented. Numbers in in ovals refer to node indexes and numbers above edges refer to their weights.

3.3.1 Graph G_1

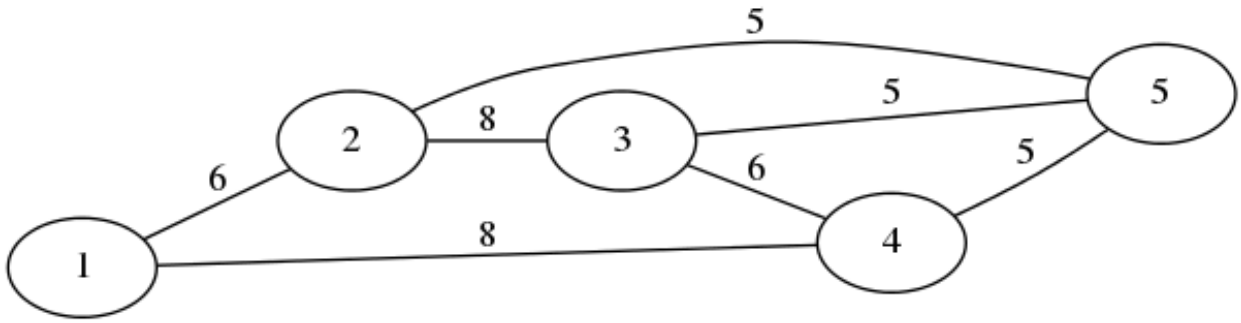

Figure 1: Base graph G_1

3.3.2 Construction of complete graph from odd only edges

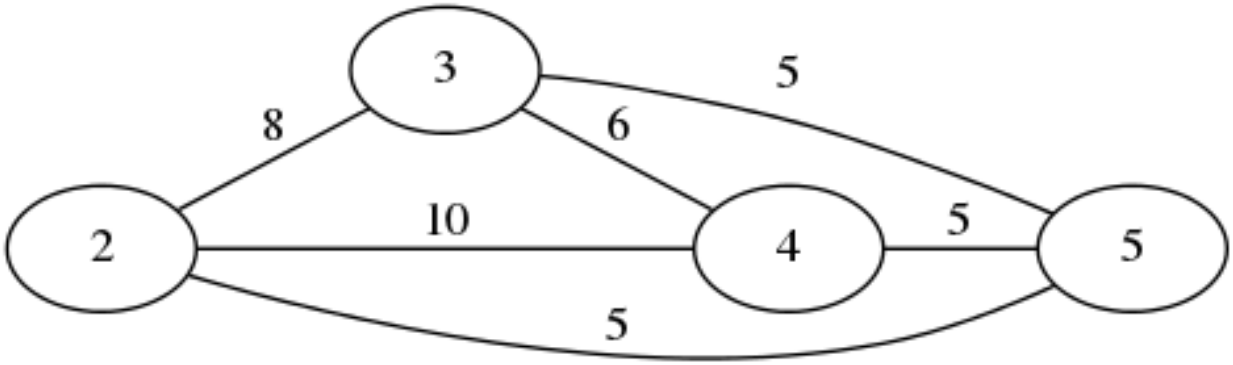

Figure 2: Complete graph G_2 constructed from odd only edges of graph G_1

3.3.3 Perfect matching using blossom algorithm

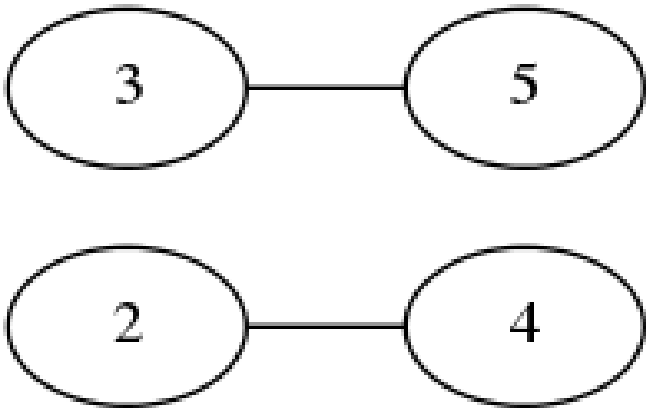

Figure 3: Perfect matching of graph G_2

3.3.4 Edges corresponding to shortest paths between nodes from perfect matching in G_1

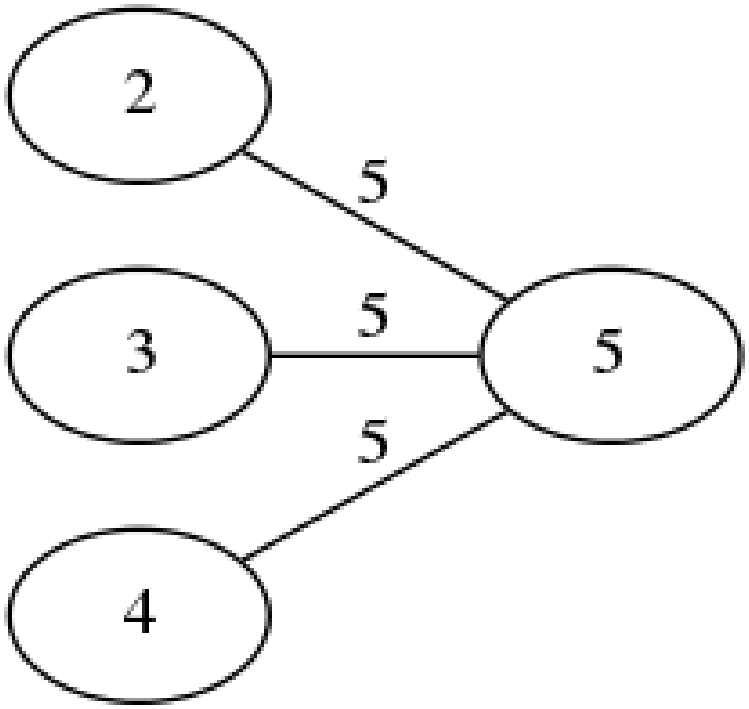

Figure 4: Graph representing extra edges to be added to make G_1 eulerian

3.3.5 Constructed eulerian graph

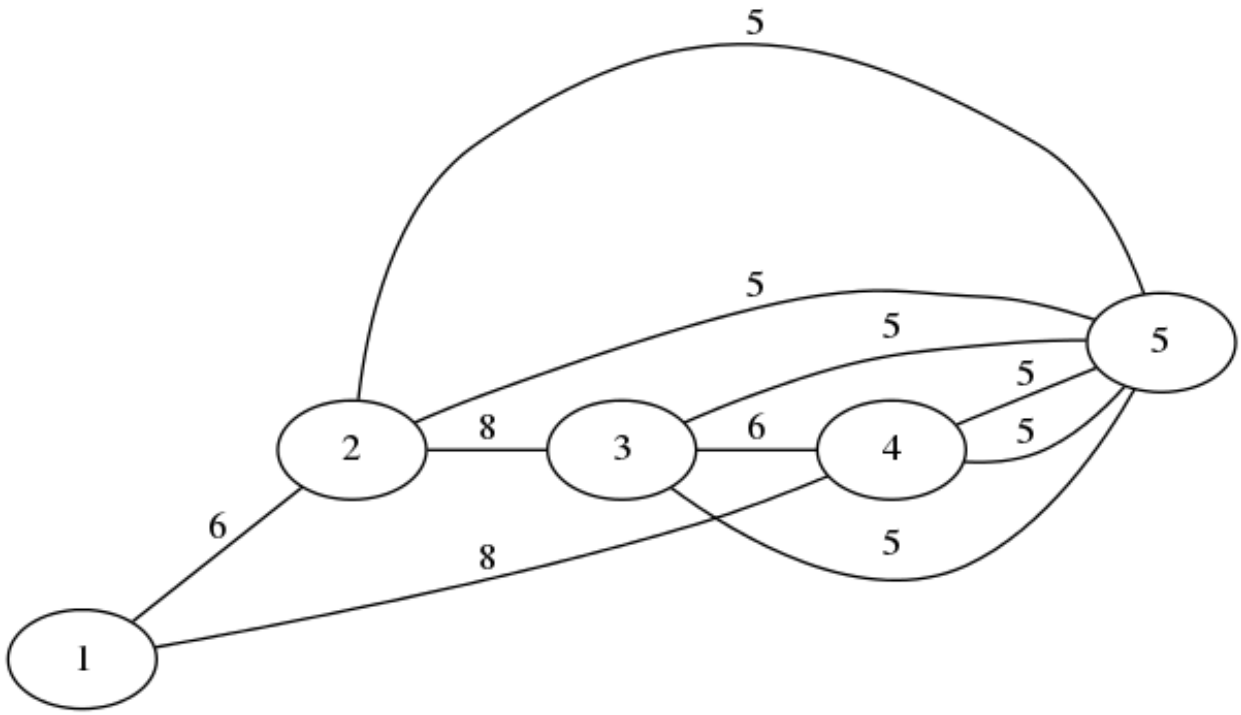

Figure 5: Resulting eulerian graph G_4 that contains added edges

3.3.6 Constructed eulerian graph

Resulted Chinese Postman Tour equals
$1 \rightarrow 4 \rightarrow 5 \rightarrow 3 \rightarrow 5 \rightarrow 2 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$
and its length is equal 58 units

4 Real life situation

4.1 Presentation

Winter gritters need to service certain set of streets. This paper introduced algorithm of that solves Chinese Postman Problem in which all edges of graph should be visited maintaining minimal cost of the tour. By representing streets that need to serviced by vehicles on a graph in which street segment correspond to edges and crossings to vertices we could apply Chinese Postman Problem Algorithm and obtain optimal path. To show direct application of this method I represented my neighborhood on a graph and applied Chinese Postman Problem Algorithm.

4.2 Data

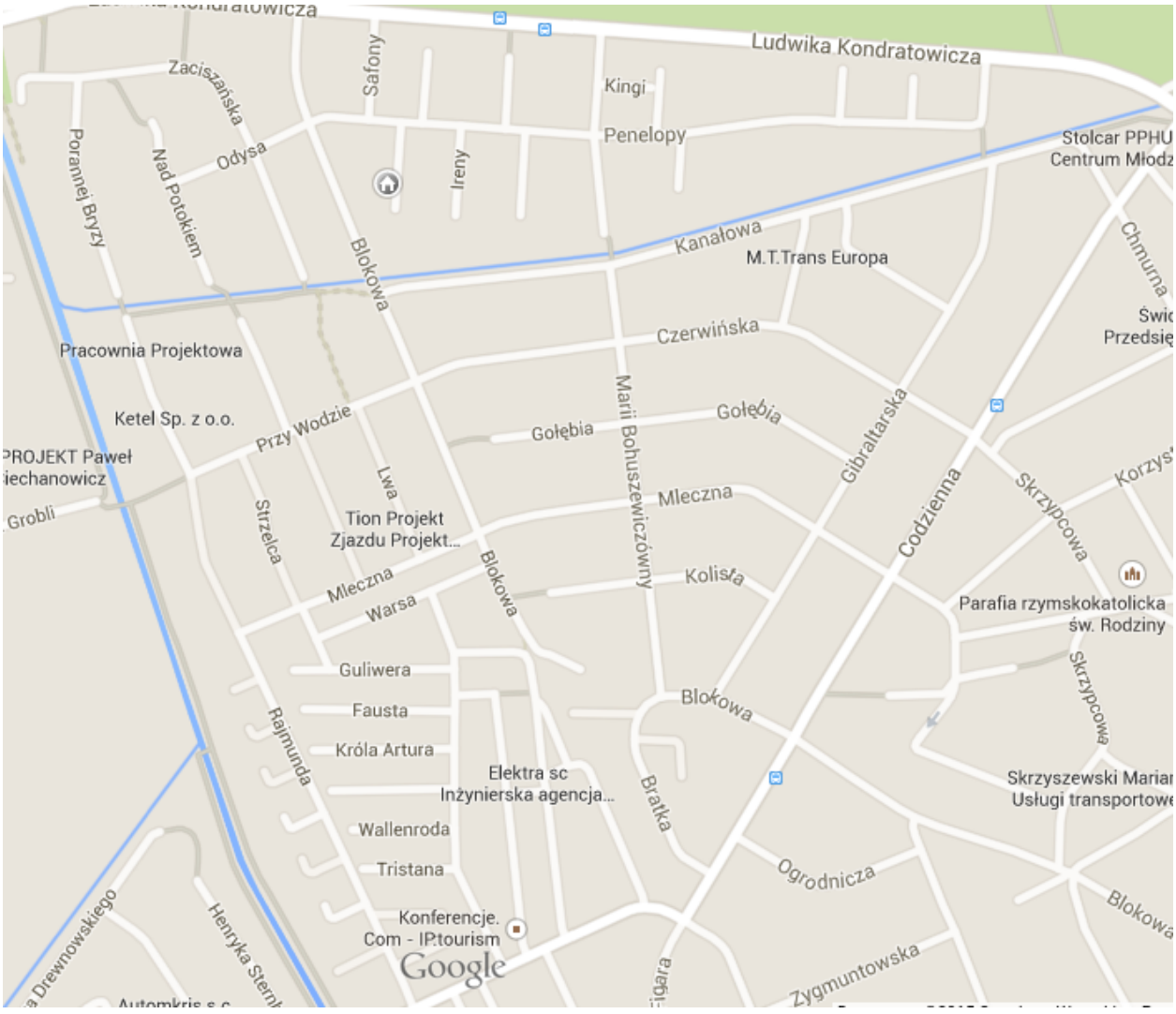

Figure 6: Map of my neighborhood

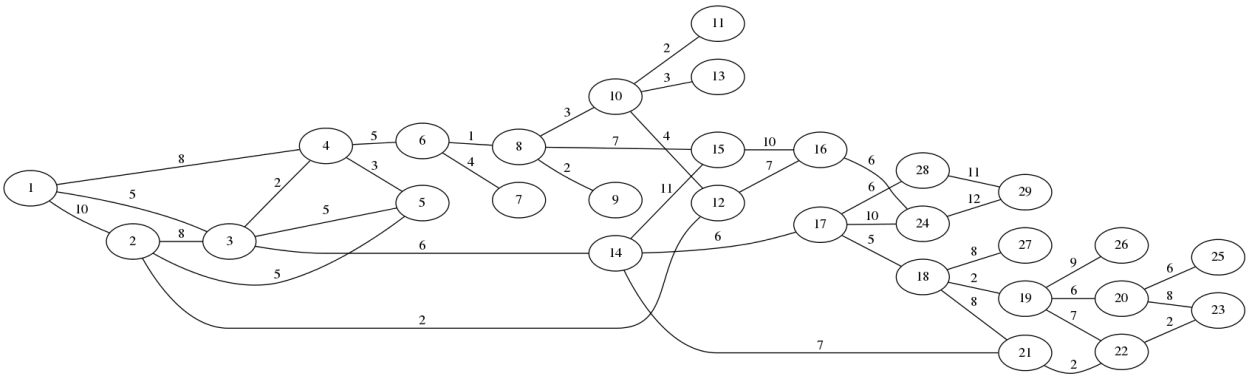

Figure 7: Base graph G_5 formed from the Figure 6.

4.2.1 Perfect matching using blossom algorithm

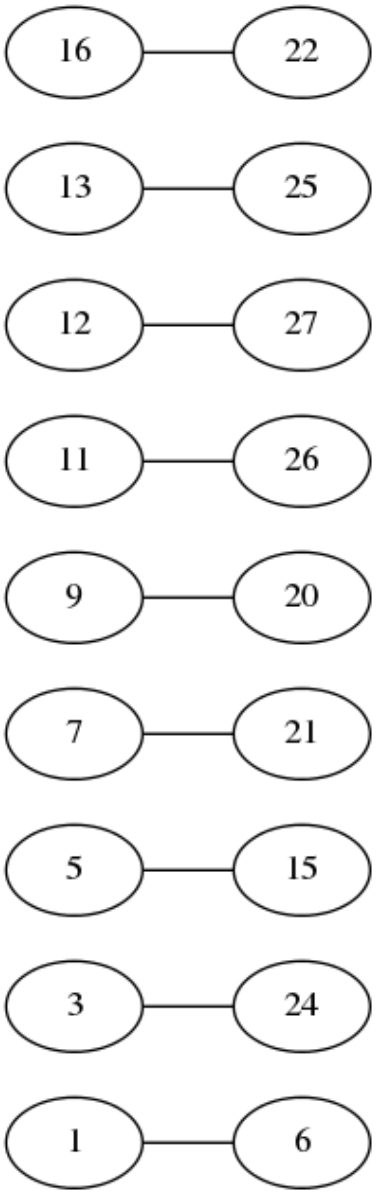

Figure 8: Perfect matching of graph G_6

4.2.2 Edges corresponding to shortest paths between nodes from perfect matching in $G5$

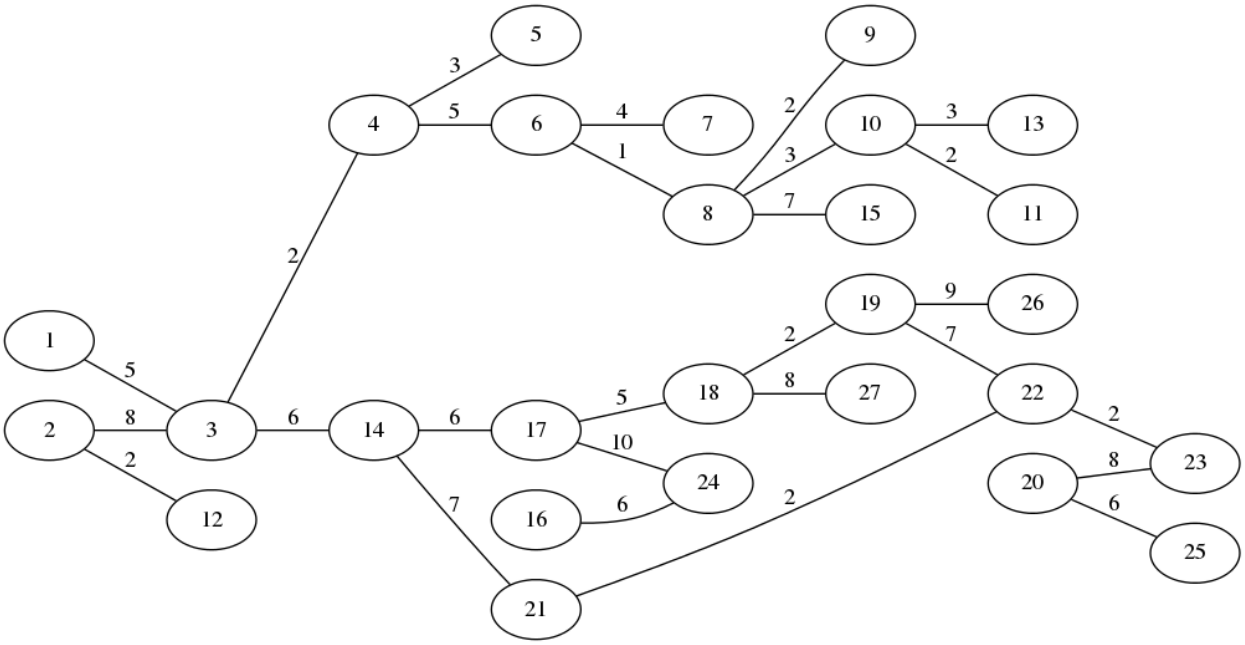

Figure 9: Graph representing extra edges to be added to make G_5 eulerian

4.2.3 Constructed eulerian graph

Figure 10: Resulting eulerian graph G_7 that contains added edges

4.2.4 Constructed Chinese Postman Tour

Resulted Chinese Postman Tour equals

$1 \to 4 \to 6 \to 7 \to 6 \to 4 \to 6 \to 4 \to 6 \to 4 \to 5 \to 4 \to 6 \to 8 \to 6 \to 8 \to 6 \to 8 \to 15 \to 14 \to 21 \to 14 \to 21 \to 22 \to 23 \to 22 \to 21 \to 22 \to 19 \to 26 \to 19 \to 22 \to 23 \to 20 \to 23 \to 20 \to 25 \to 20 \to 19 \to 18 \to 19 \to 18 \to 21 \to 14 \to 3 \to 14 \to 3 \to 14 \to 3 \to 14 \to 17 \to 14 \to 17 \to 18 \to 17 \to 18 \to 27 \to 18 \to 17 \to 28 \to 29 \to 24 \to 17 \to 24 \to 17 \to 14 \to 3 \to 4 \to 3 \to 4 \to 3 \to 4 \to 3 \to 5 \to 2 \to 12 \to 10 \to 13 \to 10 \to 11 \to 10 \to 8 \to 10 \to 8 \to 9 \to 8 \to 15 \to 16 \to 24 \to 16 \to 12 \to 2 \to 3 \to 1 \to 3 \to 2 \to 1$

and its length is equal 494 units.

5 Conclusion

By investigation Arc Routing Problems I showed that even simple algorithms can provide certain help in efficient modeling paths of winter gritters. Although representation of winter gritting by the means of Chinese Postman Problem allows to deduce optimal solution, certain additional factors could be investigated. Firstly, winter gritters have given capacity of salt, which taken into consideration extends ARP into Capacitated Arc Routing Problem. Secondly, Chinese Postman Problem considered only one vehicle. In real life situation, city disposes certain number of vehicles. Investigating these factors would allow to derive even more efficient and optimal solutions. During the course of the research for this paper I discovered that some cities already tried to tackle the problem of optimization of public services, that includes winter gritters, street sweepers and garbage collection, however many still aren't eager to change their old and known systems for far more innovative and efficient mathematical approach. I believe that this essay shows that implementing graph theory solutions in our everyday life problems isn't an overly complex procedure and may render almost immediate benefits.

References

[1] URL <https://math.dartmouth.edu/ euler/pages/E053.html>. Accessed: 2014-12-30.

[2] M. Dror. *Arc Routing Theory, Solutions and Applications*. Springer US, Boston, MA, 2000. ISBN 978-1-4613-7026-0.

[3] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part II: The rural postman problem. *Operations Research*, 43(3):399–414, jun 1995. doi: 10.1287/opre.43.3.399. URL <http://dx.doi.org/10.1287/opre.43.3.399>.

[4] J. Karskens. Mail delivery problem; route optimization with capacity constraints. URL <http://www.few.vu.nl/en/Images/researchpaper-karskens_tcm39-363659.pdf>. Accessed: 2014-12-30.

[5] W. Pearn and T. Wu. Algorithms for the rural postman problem. *Computers & Operations Research*, 22(8):819–828, oct 1995. doi: 10.1016/0305-0548(94)00070-o. URL <http://dx.doi.org/10.1016/0305-0548(94)00070-o>.

A Appendix

A.1 Code used to generate Undirected Chinese Postman Problem solutions

```
1  import networkx as nx
2  import pygraphviz as pgv # need pygraphviz or pydot for nx.to_agraph()
3
4  def Graph_draw (G_in, G_name):
5      default_attributes={'graph':{},'node':{},'edge':{}}
6      #default_attributes['graph']['label']='pygraphviz graph'
7      default_attributes['node']['shape']='box'
8
9      for u,v,d in G_in.edges(data=True):
10         d['label'] = d.get('weight','')
11     A = nx.to_agraph(G_in)
```

```
12|
13|    A.edge_attr['labelfontcolor']='dimgrey'
14|    A.edge_attr['labeldistance']='1'
15|    A.edge_attr['forcelabels']='true'
16|    #A.edge_attr['decorate']='true'
17|    #A.graph_attr['label']=G_name
18|    A.graph_attr['rankdir']='LR'
19|    #A.graph_attr['splines']='ortho'
20|    A.node_attr['shape']='oval'
21|
22|    A.layout(prog='dot')
23|    A.draw(G_name+'.png')
24|
25|    return
26|
27|G = nx.Graph()
28|Graph_draw(G, 'Base graph G1')
29|
30|odds = nx.Graph()
31|
32|for key, value in G.degree().iteritems():
33|    if value % 2 != 0:
34|        odds.add_node(key)
35|
36|Graph_draw(odds, 'Graph G2 constructed from odd only edges of graph G')
37|
38|length=nx.all_pairs_dijkstra_path_length(G)
39|for i in odds.nodes():
40|    for v in odds.nodes():
41|        if (v!=i):
42|            odds.add_edge(i,v,weight=length[i][v])
43|
44|Graph_draw(odds, 'Complete G2 graph constructed from odd only edges of graph G1')
45|
46|matches = nx.max_weight_matching(odds, maxcardinality=True)
47|perfectMatch = nx.Graph()
48|for node in odds:
49|    perfectMatch.add_node(node)
50|for key, value in matches.iteritems():
51|    perfectMatch.add_edge(key,value)
52|
53|
54|Graph_draw(perfectMatch, 'Perfect matching of graph G2')
55|
56|eGraph = nx.MultiGraph()
57|eGraph.add_nodes_from(G.nodes())
58|for i in G.edges():
59|    eGraph.add_edge(i[0],i[1],weight=G[i[0]][i[1]]['weight'])
60|
61|AGraph = nx.Graph()
62|
63|for key, value in perfectMatch.edges():
64|    q = nx.dijkstra_path(G,key,value)
65|    for i in range(len(q)):
66|        if(i!=(len(q)-1)):
67|            eGraph.add_edge(q[i],q[i+1],weight = G[q[i]][q[i+1]]['weight'])
68|            AGraph.add_edge(q[i],q[i+1],weight = G[q[i]][q[i+1]]['weight'])
69|
70|            #print q[i],q[i+1],G[q[i]][q[i+1]]['weight']
71|Graph_draw(AGraph, 'Graph representing extra edges to be added to make G1 eulerian')
72|
73|Graph_draw(eGraph, 'Resulting eulerian graph G4 that contains added edges')
```

References

[1] URL <https://math.dartmouth.edu/ euler/pages/E053.html>. Accessed: 2014-12-30.

[2] M. Dror. *Arc Routing Theory, Solutions and Applications*. Springer US, Boston, MA, 2000. ISBN 978-1-4613-7026-0.

[3] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part II: The rural postman problem. *Operations Research*, 43(3):399–414, jun 1995. doi: 10.1287/opre.43.3.399. URL <http://dx.doi.org/10.1287/opre.43.3.399>.

[4] J. Karskens. Mail delivery problem; route optimization with capacity constraints. URL <http://www.few.vu.nl/en/Images/researchpaper-karskens_tcm39-363659.pdf>. Accessed: 2014-12-30.

[5] W. Pearn and T. Wu. Algorithms for the rural postman problem. *Computers & Operations Research*, 22(8):819–828, oct 1995. doi: 10.1016/0305-0548(94)00070-o. URL <http://dx.doi.org/10.1016/0305-0548(94)00070-o>.