Problems and Algorithms for Common Urban Operations Research Problems (2019)

Prepared by:

Joseph Malkevitch
Department of Mathematics
York College (CUNY)
Jamaica, New York 11451

email:

malkevitch@york.cuny.edu

web page:

http://york.cuny.edu/~malk

Many urban mathematics problems can be formulated as graph theory problems. Examples include finding efficient routes for street sweepers, collecting back-up information about ATM transactions from ATM machine sites, finding the largest number of workers who can be assigned for tasks that they are qualified for, or synthesizing communications networks.

Problems of this kind serve to illustrate the design and complexity of algorithms and various topics in discrete mathematics. However, they can also often be used to review traditional skills topics, problem solving, and/or modeling issues. It is very useful for students to think about how problems that have similarities are the same and how they differ from each other. For example, how are the problems of delivering mail and collecting garbage similar and different?

Chinese Postman Problem

Example:

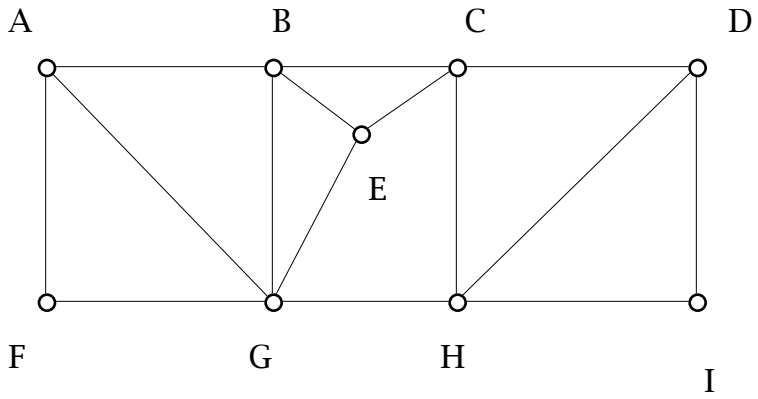


Figure 1

1. Starting at E, find a route which traverses each edge of this graph at least once and which returns to E, for which the number of repeated edges is a minimum. (What is this minimum number of repeated edges?) The intuition here is that all the edges, though they look as if they have different lengths, all have the same cost to "traverse."

2. What real world situations (e.g. mail delivery) might this serve as a model for?

3. Starting at E, find a route which traverses each edge of this graph at least once and which returns to E, for which the total cost of the tour is a minimum. (What is the minimum length tour?)

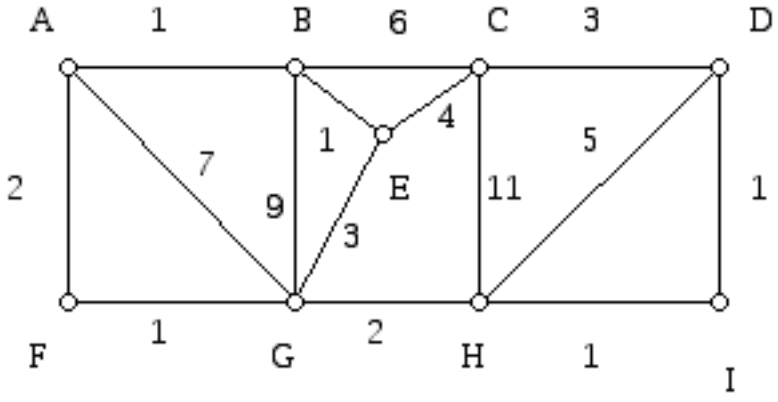


Figure 2

The weights on the edges of a graph may represent times, costs, efficiencies, etc.

Solving problems of this kind begins with Euler's traversability theorem. This result says that a connected graph has a tour (usually called an Eulerian Circuit) which starts at any vertex and returns to that vertex which visits each edge once and only once, if and only if G has an even number of edges which meet at each vertex. (The number of edges at a vertex is called the valence of the vertex or the degree of the vertex.)

Problem:

1. For the graph below, verify that the graph satisfies Euler's traversability theorem.

2. Write down two different Eulerian Circuits for this graph that start and end at A.

3. Write down two different Eulerian Circuits for this graph that start and end at B.

4. If a graph G has an Eulerian Circuit that starts and ends at vertex v, explain why it also has an Eulerian Circuit that starts and ends at any of its vertices.

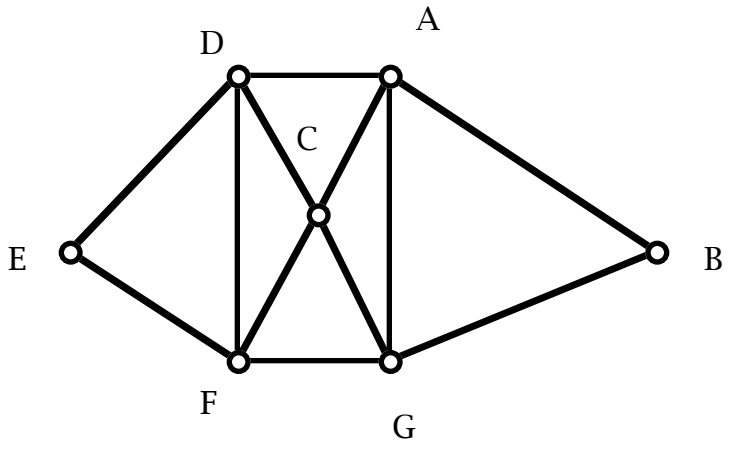


Figure 3

Chinese Postman Problem:

Given a connected graph with weights, find a tour of the edges that starts and ends at the same vertex, traverses each edge at least once, and for which the sum of the weights of the tour is as small as possible.

This problem is solvable in polynomial time using a complex algorithm

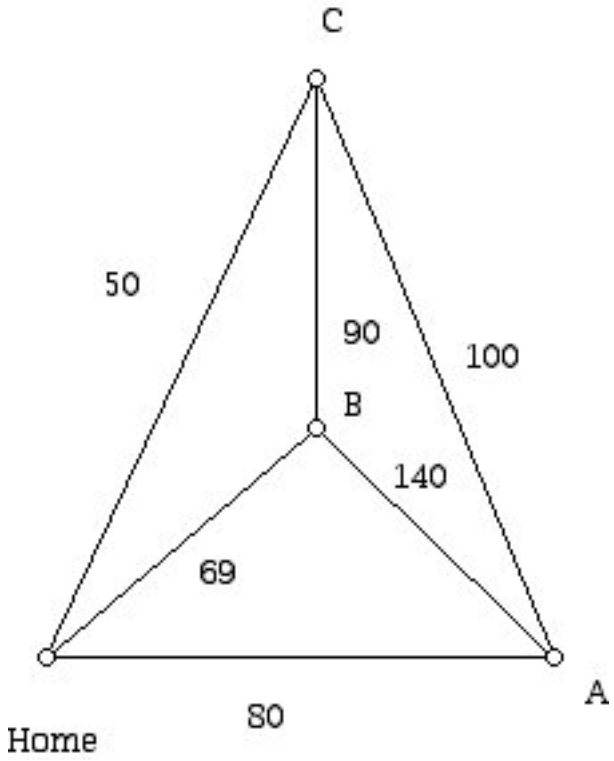
developed by Jack Edmonds and Ellis Johnson. The algorithm involves finding a minimum weight matching in a general graph. However, small problems can be solved by trial and error methods. To solve such problems it is useful to have the shortest paths in the graphs between vertices which are odd-valent. To find the shortest cost path between two vertices in a graph one can use Dijkstra's algorithm. This is the algorithm which is often used in conjunction with the hardware and software that is increasingly available in automobiles and which displays shortest time routes for driving between two locations.

It turns out that the Chinese Postman Problem for undirected graphs with weights and directed graphs (there is a line with an arrow on the edges) can be solved in polynomial time. Rather surprisingly, if the graph has some directed edges and some undirected edges (such graphs are called mixed graphs), the problem becomes NP-complete (see below).

Traveling Salesman Problem

Example:

1. What is the cheapest route that starts at home and visits each site once and only once?



2. If one employs "brute force" to find a cheapest route when there are n sites

to visit, how many routes must be examined?

Unfortunately, the TSP and its simple variants are NP-Hard problems. This means:

a. There is no known polynomial time algorithm for the problem

b. There is no proof that all algorithms to solve the problem require exponential time

There is a "complexity class" known as NP-complete problems which have the property that if any one of them could be shown to be solvable in polynomial time, then all of them could be so solved, and if any one of them could be shown to require exponential amounts of time than than all of the problems in the class would have this property. This problem of whether or not P = NP is one of the most important open problems in mathematics and computer science. There are hundreds of appealing problems known to be NP-complete many of them in the field of operations research.

However, there are "heuristic algorithms" which quickly yield solutions for the TSP that are often reasonably good, despite the fact that for most heuristics there are examples which show these algorithms in the worst case to be rather bad!

Heuristics:

a. Nearest neighbor

*visit next that vertex not already visited which is nearest (smallest weight) to where one is currently

b. Sorted edge lengths

*sort the edges in order of increasing cost and form a "circuit" by adding one edge at a time adding at each stage the cheapest edge choice

c. Clarke-Wright

This algorithm is more flexible than those above because it not only solves the TSP but also the vehicle routing problem. In this problem one has one or more trucks with fixed capacity C and one wants to visit sites to drop off goods so that the routing can be minimized in cost and subject to the constraint that the amount carried by a truck doesn't exceed its capacity.

Though Clarke-Wright typically does not yield optimal solutions, it often yields very good solutions and it can easily be adapted to other constraints beyond that of truck capacity.

The algorithm works by computing a list of "savings" from serving each site using a separate truck from the "home" site (or depot). It then merges tours in a greedy (e.g. the best choice available at that moment) way based on these "savings."

Minimum cost spanning tree

In many urban situations it is necessary to create a sewer pipe network, a communications network, or some other physical links between a collection of sites. These situations can be modeled with a connected graph where the vertices represent the sites, the edges are the potential links that might be built, and there are weights on the edges which represent the cost of putting in that edge as a link. Not all the edges need be present, because if we can get between a pair of vertices along a path joining a pair of vertices, this is good enough. In the communications situation this equivalent to saying that one can relay messages from one site to another using the links that are present. In graph theory terms, what is required here is to find a minimum cost spanning tree. A tree is a connected graph which has no "circuits." Spanning refers to the fact that the tree must include all of the vertices of the original graph.

This problem has a complex history, having first been solved by a Czech mathematician (Otakar Boruvka (1899-1995)) whose work did not become widely known because the article was published in Czech. The problem was independently solved by two Bell Laboratory mathematicians, Joseph Kruskal and Robert Prim.

Kruskal's Algorithm is very similar to the sorted edge cost method sometimes used to solve the TSP:

Add next to the selected links, the cheapest edge (ties can be broken arbitrarily) not already selected which does not create a circuit.

The algorithm terminates for an n vertex connected graph when n-1 edges have been chosen.

Prim's Algorithm, is similar to the nearest neighbor approach for the TSP.

Maximum cardinality matching

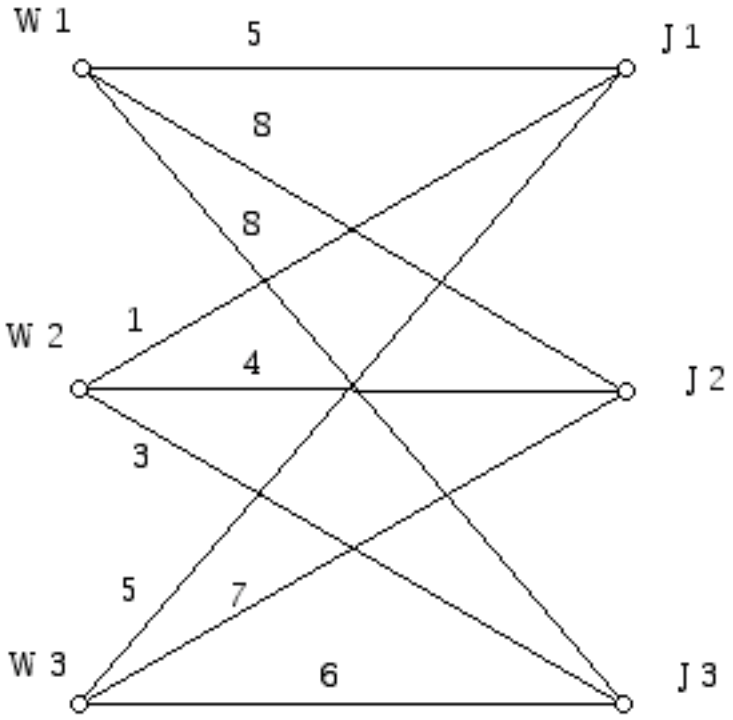
A matching in a graph is a disjoint collection of edges. A perfect matching is a collection of disjoint edges which includes all of the vertices of the graph. A necessary condition for a perfect matching is that the graph have an even number of vertices. Philip Hall's Theorem characterizes situations where there is a perfect matching but does not allow for the largest matching to computed in a time efficient way. The alternating path algorithm (graph theory based) due to König is an efficient (polynomial time) algorithm for finding a maximum cardinality (size) matching.

Minimum cost matching

A typical problem of this kind is to match workers to jobs where there is a time that each worker will be able to perform the job he/she is assigned to. The idea is to find an assignment where the total time to get the jobs done will be a minimum. This calls for finding a collection of edges which are disjoint from each other and includes (ideally) all of the vertices of a graph where the sum of the costs of the edges involved is as small as possible.

Example 1:

The diagram below shows 3 workers and the amount of time that it will take them to do each of three jobs. What assignment of workers to jobs will make it possible to complete all three jobs in as short a time as possible?



This problem can be solved by using a matrix based algorithm or graph theory algorithm based on the idea of "flows" in networks. The algorithm which solves this problem "directly" on the matrix version of the "data" shown in the graph above is called the Hungarian Method.

References:

1. COMAP, For All Practical Purposes, 10th Edition, W.H. Freeman, New York, 2016. (Earlier editions have very similar content.)

2. Wu, B and K. Chao, Spanning Trees and Optimization Problems, Chapman & Hall/CRC, Boca Raton, 2004.