Combinatorial Optimization

NP Problems II

Dias, Schiewe, Pattanaki *Instructor: Pattanaki*

§ Week VI §

Problem 1: TSP alternative

For the following graph, propose an algorithm to find a solution for TSP.

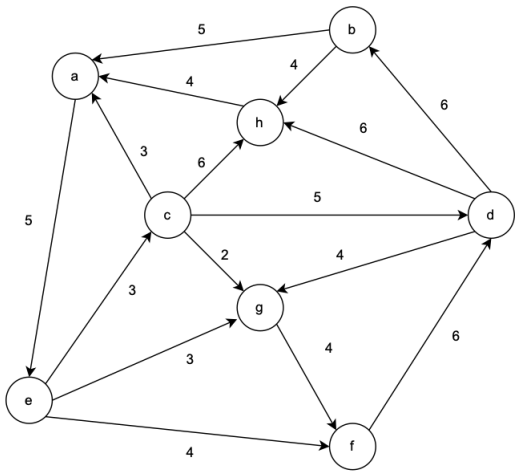

Figure 1: Direct, weighted graph.

Give examples of at least three subtours and their corresponding subtour constraint elimination.

Solution:

There are plethora of algorithms that could be used as possible **heuristics** in order to approximate a solution for TSP. A simple algorithm could an extension of either Dijkstra's algorithm or from minimum spanning tree. All those alternatives are valid.

An original approach could be as follows:

- For each node, choose the edge with the lowest code;
- If that edge leads to a node already visited, skip such node;
- Repeat the process until all nodes are visited.

Problem 2: Vertex Cycle Cover

Given a directed graph $G = (V, E)$, a vertex cycle cover is a subset of vertices such that every simple cycle in G passes through at least one of these vertices. For example, the graph shown in Fig 1 has a vertex cycle cover of size 2 (shaded).

Vertex Cycle Cover (VCC): Given a digraph G and an integer k, does G contain a vertex cycle cover of size at most k?

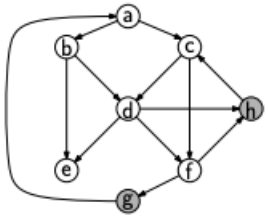

Figure 2: A graph and a vertex cycle cover consisting of $\{g, h\}$.

Is this problem NP? Provide a possible algorithm to solve such problem.

Solution:

This problem is more commonly referred to as the Feedback Vertex Set. A naive approach to showing that it is in NP-would be to guess the k vertices of V' that will constitute the VCC, and then verify that every simple cycle passes through at least one of these vertices. The problem is that there are exponentially many simple cycles, so you cannot enumerate them all.

You might wonder, couldn't we simply guess the cycle as well? If the guessed-at cycle passes through some vertex of V' then we output "no" and otherwise we output "yes." The problem is that this does not work according to the rules of nondeterministic computation. Recall that a nondeterministic computation succeeds of any sequence of guesses leads to an answer of "yes". Now, suppose that your graph does not have a VCC. The above program will guess some subset of k vertices and then may guess a simple cycle that does not pass through these vertices. It then answers "yes" which implies that the global answer is "yes." But the global answer should be "no". (The problem is that it checked only one cycle, but it would need to check them all before answering "yes".)

The trick is to realize that the problem can be restated in a manner that makes the verification process much simpler. We can equivalently define a VCC to be a subset of vertices such that, after removing these vertices, the graph is acyclic. (For example, in Fig. 1(a) the vertices g, h form a VCC, and in (b) their removal results in a DAG.) To see this, observe that if every simple cycle passes through some vertex of the VCC, then removing these vertices destroys all cycles. Conversely, if the removal of the vertices of the VCC results in an acyclic graph, then every cycle of the original graph must pass through at least one of these vertices. We can check that a digraph is acylic either by (1) running DFS and checking that there is no back-edge or (2) compute the strong components and check that every vertex is in its own strong component.

Problem 3: Cliques

When finding cliques, it is natural to look for vertices of high degree. Suppose, however, that you want to find cliques consisting of vertices of relatively low degree. We will show that even this problem is NP-complete.

Low-Degree Clique (LDC): Given a graph $G = (V, E)$ and an integer k, does G have a clique of size at least k consisting entirely of vertices who degree is not greater than the median vertex degree of the entire graph?

By the *median vertex degree*, we mean the median value of the degrees of all n vertices of the graph.

For example, the graph shown in Fig. 2 has median vertex degree of 3. There exists an LDC of size 3 (vertices b, c, e) since all these vertices have degree at most 3. Even though there is a clique of size 4 (vertices a, f, g, h) it is not an LDC since it contains (at least one) vertex of degree higher than 3.

Show that the LDC problem is NP.

Solution:

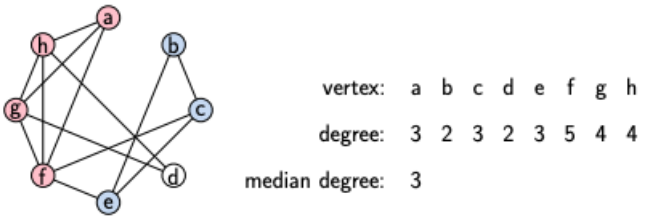

Figure 3: Low-Degree clique

$LDC \in NP$: Given a graph G and integer k, we guess k vertices of G. We then compute the degrees of all the vertices of G and compute the median of this set. (This can be done in $O(n + m)$ time by inspecting the adjacency lists of all the vertices and using a fast selection algorithm.) We check that every one of the guessed vertices has degree less than or equal to the median degree, and that each of these vertices is adjacent to all the others. If both of these are satisfied, we output "yes," and otherwise we output "no." If G has an LDC, then one of these guesses will succeed in identifying the LDC and we will output "yes." If not, none of the guesses will work, and we will output "no."

LDC is NP-hard: We will show that the standard clique problem is polynomially reducible to LDC (Clique $\leq$ p LDC). The idea is to artificially increase the median degree of G so that any valid clique in the original graph is an LDC in the modified graph. We want to do this without accidentally creating an LDC. Suppose that we are given an input graph G and integer k. Let n be the number of vertices in G, and let us make the (trivial) assumption that k $\geq$ 3. (Otherwise, the problem reduces to determining whether G has at least one edge.) We create a new graph G0 by making a copy of G and adding to this a complete $n \times n$ bipartite graph. The result is a graph with $3n$ vertices. The original vertices of G have degree at most $n - 1$, and the $2n$ newly added vertices all have degree n, so the median degree of G' is n. We output G' and k. Clearly, this can be done in polynomial time. (An alternative approach is to generate a very large clique in G, but some care is needed to avoid this large clique from providing a spurious LDC.) To establish correctness, we will show that G has a clique of size k if and only if G' has an LDC of size k.

$\rightarrow$ Suppose that G has a clique V' of size k. We assert that same vertices form an LDC within G'. The reason is that G0 has median degree n and each original vertex of G has degree at most $n-1$, so all the vertices of V' are LDC-eligible.

$\rightarrow$ Suppose that G' has an LDC V' of size k. The largest clique in any bipartite graph is of size 2, and by our assumption $k \geq 3$. Thus, the bipartite part of G' cannot contribute to the existence of a clique, which implies that V' is a clique within the original graph G.

Problem 4: Hamiltonian Path vs TSP

What is the main difference between Hamiltonian Cycle and Travelling Salesman Problem. Can we solve one using the other?

Solution:

Hamiltonian Cycle (or Path, for that matter) only concern with the existence of a path based on its cardinality and following the problem constraints. On the other hand, Travelling Salesman Problem concerns with those metrics as well as the cost of the chosen route.

Problem 5: Paths and Cycles

The graph shown below is the Petersen graph. Does it have a Hamilton cycle? Justify your answer. Does it have a Hamilton path? Justify your answer.

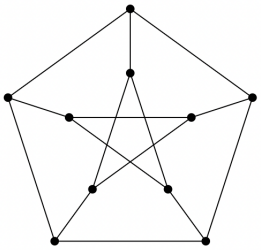

Figure 4: Peterson Graph

Solution:

A Hamiltonian cycle is a cycle that uses every vertex of a graph while a Hamiltonian graph is a graph that has a Hamiltonian cycle. In Peterson's graph, there exists Hamiltonian paths, but no Hamiltonian cycle.

Thus, the Petersen graph is not Hamiltonian. However, it is interesting to note that by deleting any vertex in the Petersen graph, it makes it Hamiltonian.