Applications of Graph Theory and Combinatorics in Computer Science

Dylan Galea

October 26, 2018

Contents

1 Introduction

2 The Travelling Salesman Problem

2.1 Basic Definitions and Results

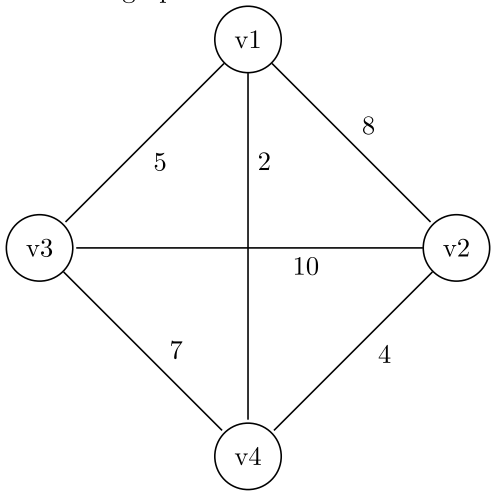
To define the Travelling Salesman Problem , first the concept of a Hamiltonian Cycle must be defined. This concept is defined in definition 2.1.1 below.

Definition 2.1.1 (Hamiltonian Cycle)**.** *Given a graph G(V,E), a Hamiltonian Cycle in G is a cycle in G such that $\forall$ v $\in$ V, v is in the cycle and is visited only once(except the first vertex because it needs to be visited again to complete the cycle) . A graph that contains a Hamiltonian cycle is called a Hamiltonian Graph [1].*

The Travelling Salesman problem can now be defined as shown in definition 2.1.2 below.

Definition 2.1.2 (Travelling Salesman Problem)**.** *Given a simple positively weighted graph G(V,E) such that $\forall$ v,w $\in$ V ,v $\neq$ w $\{v, w\} \in E$, the Travelling Salesman Problem is the task of finding a minimum weight Hamiltonian Cycle in G [2].*

To understand the Travelling Salesman Problem defined in definition 2.1.2 , the following example is constructed :
Consider the graph G below :



Then using definition 2.1.1 the distinct Hamiltonian Cycles in G are (Hamiltonian Cycles not using the exact same edges) :
1. (v1 v3 v4 v2 v1) with cost 24
2. (v1 v3 v2 v4 v1) with cost 21
3. (v1 v2 v3 v4 v1) with cost 27

Thus using definition 2.1.2, the answer for the Travelling Salesman Problem on this graph instance would be (v1 v3 v2 v4 v1), because it is the minimum weight Hamiltonian Cycle in G.

After illustrating the Travelling Salesman Problem, some basic results are needed to be prooved so that no assumptions are taken.In fact ,definition 2.1.2 suggests that in the Travelling Salesman Problem it is already known that the graph to be evaluated is Hamiltonian, otherwise a minimum weight Hamiltonian Cycle can never be found.This uncertainty can be tackled by proving that any complete graph on more than 3 vertices is Hamiltonian. This fact is proved in lemma 2.1.3 below.

Lemma 2.1.3. *For n $\geq$ 3, The complete graph on n vertices is Hamiltonian*

Proof. Let K_n be the complete graph on n $\geq$ 3 vertices labelled v1,v2,...,vn. Order all the vertices in the order v1,v2,...,vn with no repetitions of vertices.Then C=(v1 v2 ... vn v1) must be a cycle in K_n because because $\forall$ vi,vj $\in$ C, vi $\neq$ vj then $\{vi, vj\}$ $\in$ E(K_n). Also since $\forall$ v $\in$ V(K_n) ,v is a vertex in the cycle with only one occurence in C(except for v1 which has 2 occurences) then C must be a Hamiltonian Cycle in K_n . Thus K_n must be Hamiltonian. □

Since every K_n for n $\geq$ 3 vertices is Hamiltonian this suggests that in order to find the minimum weight Hamiltonian cycle one could compute every Hamiltonian cycle in K_n and then return the cycle of least cost. However, this is infeasible because this algorithm K_n generates $\frac{(n-1)!}{2}$ distinct Hamiltonian cycles(proved in lemma 2.1.4 below) . Thus the algorithm would have a time complexity O(n!) making it very infeasible to compute in reasonable time. [2]

Lemma 2.1.4. *For n $\geq$ 3 the complete graph on n vertices has $\frac{(n-1)!}{2}$ distinct Hamiltonian cycles*

Proof. Let K_n be the complete graph on n $\geq$ 3 vertices .Since E(K_n) contains an edge for every possible dinstinct vertices u,v $\in$ V(K_n) then every permutation of V(K_n) must represent a Hamiltonian Cycle in K_n and vice versa(1-1 correspondence). Now on n vertices there are n! possible number of permutations, thus we have n! Hamiltonian cycles due to the 1-1 correspondence. However, different permutations of V(K_n) may represent the same Hamiltonian Cycle in K_n, since the same edges would be used from E(K_n) but in a different order of the vertices . In fact, consider the Hamiltonian Cycle C represented by the permutation (v1 v2...vn). In terms of Hamiltonian cycles, the permutation (v1 v2 ...vn) is the same as the permutation (v2...vn v1) because the same edges in E(K_n) are used . Thus for each Hamiltonian cycle in K_n we can have n permutations representing the same cycle.However,for each of these n permutation representations , the reverse of each of the permutations represent the same Hamiltonian Cycle with the difference being that the cycle is traversed in reverse order, thus we have 2n permutations representing the same Hamiltonian Cycle. Thus the number of distinct permutations is $\frac{n!}{2n} = \frac{(n-1)!}{2}$. [3] □

Lemma 2.1.4 confirms the difficulty in writing an algorithm that executes in reasonable time, to solve the Travelling Salesman Problem for any number of cities .The reason is that in order to compute the minimum weight Hamiltonian Cycles , all algorithms that were constructed so far take superpolynomial time.

This leads to a discussion on the different classes of problems including NP-Completness in the next sub-section.

2.2 The classes P,NP and NPC

It is known that not all problems in the universe can be solved by algorithms even if a lot of time is dedicated to them . In fact Alan Turing proved this by proposing the Halting problem which is a problem that proveably cannot be solved by any computer. In addition to this class of problems there is the class of NP-Complete problems(NPC). For problems in this class, no polynomial time algorithm has yet been discovered , but at the same time no one was able to prove that such a polynomial time algorithm exists or not for them. In what follows, the important classes of problems will be discussed more formally, describing in the process how to prove that a problem is in NPC so that it can be proved that TSP is NP-Complete. [4]

The three most important classes of problems are called P,NP and NP-Complete (NPC). Firstly, problems belonging to P are problems that can be solved in polynomial time and thus are tractable. Such problems can be solved by algorithms that can complete their execution in a reasonible time. Secondly, the class NP consists of decision problems that can be verified in polynomial time . This means that given a certificate(answer) to a problem in NP , this certificate can be checked in polynomial time to verify if it is correct or not. Thirdly, a decision problem is in the class NPC if it is in NP and it is as hard as any problem in NP(this will be made more clear in the next paragraph when discussing how to compare problems' complexity using reductions). Thus when proving that a problem is NP-Complete a statement is being made as to how hard is that problem and thus that no efficient algorithm exists. It must be noted that according to the definitions , in order to be in the classes NP and NPC the problems must be decision problems. This is of great benefit because decision problems are much easier to reason about and all problems in the universe can be converted to a decision problem representing and vice versa.For example, TSP can be converted to the decision problem, "Is there a Hamiltonian cycle in K_n having length less than k?". [5]

Thus reasoning on whether a problem is as hard as any other problem, is as hard as conducting the same reasoning on their decision versions. Checking whether a decision problem is as hard as another decision problem can be done using reduction algorithms. Suppose that 2 decision problems A and B are given, were α is the input to the decision problem A and β is the input to decision problem B. Then a reduction algorithm transforms any input α of A into some input β of B such that the transformation takes polynomial time and the answers are the same. This means the answer for the decision problem A given α is 'yes' $\iff$ the answer for the decision problem B given β is 'yes'. Thus if a decision problem A is required to be solved in polynomial time and it is known that a decision problem B can be solved in polynomial time , then A can be solved in polynomial time if a reduction algorithm can transform every input of A to and input of B since transforming this instance would take polynomial time and runnning the algorithm on this transformed instance takes also polynomial time . Thus the total number of time is polynomial . Note that this could only be done if one problem can be transformed into an other , since it is only then that 'yes' result for one is a 'yes' result for the other. [5]

Read geek for geeks to join everything and explain as least as hard in np

3 Conclusion

References

[1] E. Weisstein, “Hamiltonian cycle,” Oct 2018. [Online]. Available: http://mathworld.wolfram.com/HamiltonianCycle.html

[2] “Travelling salesman problem — set 1 (naive and dynamic programming),” Sep 2018. [Online]. Available: https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/

[3] “How many hamiltonian cycles are there in a complete graph k_n $(n \geq 3)$ why?” Dec 2012. [Online]. Available: https://math.stackexchange.com/questions/249817/how-many-hamiltonian-cycles-are-there-in-a-complete-graph-k-n-n-geq-3-why

[4] “Np-completeness — set 1 (introduction),” Sep 2018. [Online]. Available: https://www.geeksforgeeks.org/np-completeness-set-1/

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 3rd ed. MIT PRESS.