A Note on Graph and Its Application

Okeyinka, A.E., Asani, E.O., Adegun, A.A., Adeniyi, A.E. & Ayoola, J.A.
Department of Computer Science
Landmark University
Omu-Aran, Nigeria
E-mails: okeyinka.aderemi; asani.emmanuel; adegun.adekanmi {@lmu.edu.ng}

ABSTRACT

Graphs could be used to represent several physical structures including communication networks, irrigation system, data organization, computational flow among others. The goal of this study is to report an aspect of what has been done so far in our group research activities in combinatorial algorithms. Hence, basic concepts and application areas of graph theory are presented. Furthermore, The Travelling Salesman Problem (TSP) is presented and a 6-city TSP is solved both by the use of a heuristic and exhaustive enumeration. The efforts expended in arriving at both solutions show that the use of heuristics is worthwhile. This explains why heuristics have been gaining increasing interest and scientific respectability from the computer science community.

Keywords: Graph, TSP, Heuristics, Exhaustive Enumeration, Vertices, Edges

ISTEAMS Cross-Border Conference Proceedings Paper Citation Format
Okeyinka, A.E., Asani, E.O., Adegun, A.A., Adeniyi, A.E. & Ayoola, J.A..(2017). A Note on Graphs & Its Applications. Proceedings of the 9th iSTEAMS Multidisciplinary Conference, University of Ghana, Legon, Accra Ghana. Pp 283-290

1. INTRODUCTION

In computer science there are formalisms that lead to an understanding of the concept of computables. These formalisms provide us with various models of computation such as Turing machine, Recursive functions, and Markov productions. Also in the study of computers and computations, the following basic questions occur at the very beginning:

- What problems can be solved by computer programs?
- What solutions can be effectively obtained in practice?

The theory of computability and computational complexity has provided a very general answer to both questions, in a form largely independent of specific assumptions about computers and computations. In particular the theory has shown that some well-defined, natural problems are inherently unsolvable by computer programs for any conceivable type of computer and programming system. Other problems can be solved by computer only in principle, they are shown to be so difficult, that solving them will always require computational resources exceeding all practical limits, for any type of computer and any kind of program formulation. We define an algorithm as a systematic method to process discrete symbolic information. However the classical theory of computation, dealing with discrete, denumerable sets, has been extended by Blum, Shub and Smale to the case of continuous domains. Specifically combinatorial algorithms are algorithms for solving combinatorial problems; and combinatorial problems are problem whose solution space explodes exponentially. Many of such problems are though solvable but impractical. Hence heuristics or approximate algorithms are usually used in solving them.

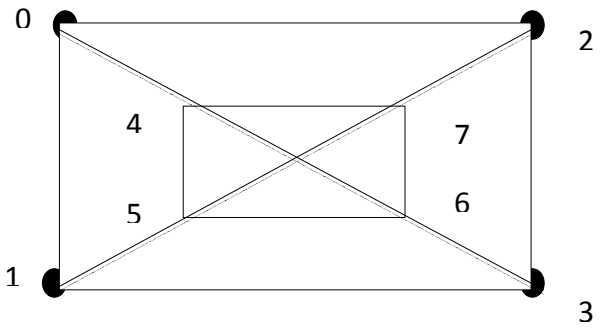
Combinatorial optimization consists of finding an optimal object from a finite set of objects. In many of such problems, exhaustive search is not feasible, it operates on the domain of those optimization problems in which the set of feasible solutions is discrete or can be reduced to discrete, and in which the goal is to find the best solution. Combinatorial optimization has important applications in several fields, including Artificial Intelligence, Software Engineering, Telecommunication

Engineering, Civil Engineering, Highway Engineering, Town and Regional Planning, Warfare Simulation, Irrigation Construction GIS and so forth.

1.1 Graph Theory

A graph is a pair (V, E) where V is a finite set of vertices (nodes, points), E is a finite set of edges (arcs, lines).

G =



$V = \{0,1,2,3,4,5,6,7\}$

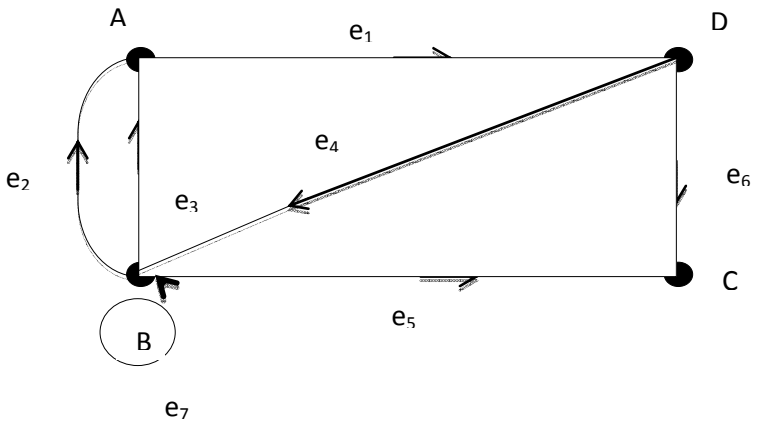
$$E = \left\{ \begin{matrix} \{0,4\},\{0,1\},\{0,2\},\{2,3\},\{2,6\}, \\ \{1,3\},\{1,5\},\{3,7\},\{4,5\},\{4,6\}, \\ \{4,7\},\{5,6\},\{5,7\},\{6,7\} \end{matrix} \right\}$$

1.2 Directed graph

A directed graph G is a graph where each edge e in G is assigned a direction or in other words, each edge is identified with an ordered pair (u,v) of nodes in G rather than an unordered pair [u,v].

Suppose G is a directed graph with a directed edge e = (u , v), then the following terminology is used:

- e begins at u and ends at v
- u is the origin or initial point of e, and v is the destination or terminal point of e
- u is a predecessor of v, and v is a successor or neighbor of u.
- u is adjacent to v, and v is adjacent to u.



1.3 Adjacency matrix

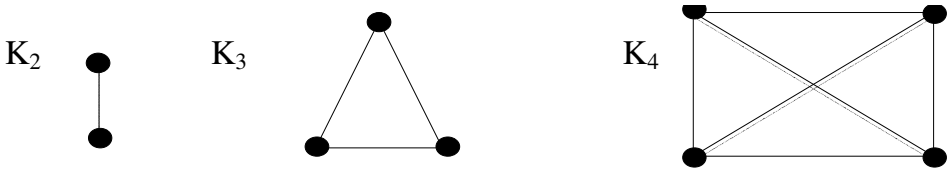
Suppose A is a simple directed graph with m nodes and suppose the nodes of A have been ordered and are called $v_1, v_2, \ldots, v_m$. Then the adjacency matrix $A = (a_{ij})$ of the graph G is the m x m matrix defined as follows:

$$a_{ij} = \begin{cases} 1 & \text{if } v_i \text{ is adjacent to } v_j, \text{ that is;} \\ & \text{if there is an edge } (v_i, v_j) \\ 0 & \text{otherwise} \end{cases}$$

The adjacency matrix A of the graph G does depend on the ordering of the nodes of G; that is, a different ordering of the nodes may result in a different adjacency matrix.

1.4 Complete Graph:

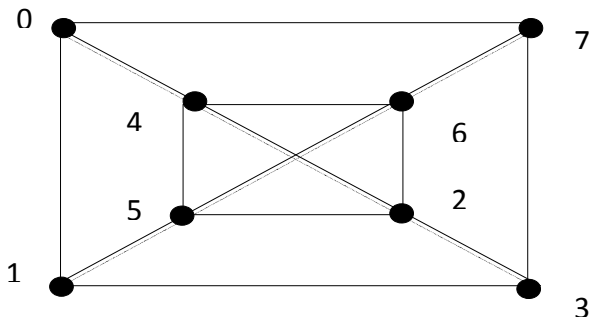
A graph G is said to be complete if every node v in G is adjacent to every other node v in G.



k_2, k_3 and k_4 are complete graphs.

1.4 Hamiltonian Cycle

A Hamiltonian cycle is a closed path that passes through each vertex once



The list [0, 1, 5, 4, 6, 7, 3, 2] describes a Hamiltonian cycle in the graph above

Proposition 1: A graph G is connected if and only if there is a simple path between any two nodes in G.

Proposition 2:

Let A be the adjacency matrix of a Graph G. then $a_k(i, j)$ the ij entry in the matrix A^k, gives the number of paths of length k from v_i to v_j.

Proposition 3:

Let A be the adjacency matrix and let P = (P_{ij}) be the path matrix of a digraph G. then P_{ij} = 1 if and only if there is a nonzero number in the ij entry of matrix $B_m = A + A^2 + A^3 + \ldots\ldots\ldots + A^m$

2. APPLICATION OF GRAPH THEORY

Graph theory finds relevance in solving many real life problems, including the following:

Mathematical Research, Electrical Engineering, Computer Programming, Computer Networking, Business Administration, Sociology, Economics, Marketing, and Telecommunication. In particular, many problems can be modeled with paths formed by traveling along the edges of a certain graph. For example, problems of efficiently planning routes for mail delivery, garbage pickup, snow removal, diagnostics in computer networks can be solved using model that involve paths in graph. Others are: cable laying to connect towns, network analysis in Geographic Information System (GIS), minimum-spanning trees, and so forth.

2.1 The Traveling Salesperson Problem (TSP)

The TSP asks for a Hamiltonian circuit of minimum length, where the length of a circuit is defined as the sum of the lengths of the edges in the circuit. The TSP is simple to characterize but difficult to solve in that we know of no efficient procedure for solving the problem. One of the heuristics used in solving TSP-based structures is Nearest Neighbor Heuristic.

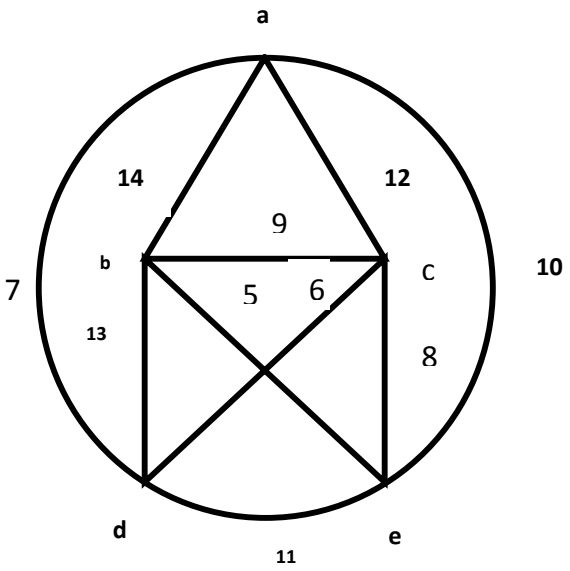
The Nearest Neighbor Heuristic is a procedure which gives a reasonably good result for the TSP and is stated as follows:

Step 1: Start with an arbitrarily chosen vertex, and find the vertex that is closest to the starting vertex to form an initial path of one edge. We shall augment this path in a vertex-by-vertex manner as described in step 2.

Step 2: Let x denote the latest vertex that was added to the path. Among all vertices that are not in the path, pick the one that is closest to x and add x and the vertex. Repeat this step until all vertices in G are included in the path.

Step 3: Form a circuit by adding the edge connecting the starting vertex and the last vertex added.

$G^1 =$



If we start from vertex a, in graph G^1 above, a vertex-by-vertex construction of a Hamiltonian circuit according to the Nearest-Neighbor Heuristic is 40. However, the total distance of the minimum hamiltonian circuit is 37. To get the minimum however we may have to do exhaustive enumeration which could be time-consuming for practical purposes.

3. HEURISTICS

A heuristic algorithm or heuristic for short is not an exact algorithm. It produces a feasible and hopefully very good, but not necessarily optimal solution. Heuristics are generally very fast.
We often use heuristics when an exact algorithm is not available or when the exact algorithm is computationally impractical.

The following are the desirable characteristics of heuristics,

i. They run in reasonable computational time
ii. Solutions generally are close to optimal
iii. The probability of anyone solution being very far from optimal is small
iv. Storage requirements are small.

Although there is no real format into which we can place heuristics, quite a few are based on either the sub-goals or the hill-climbing method. One general approach to the design of a heuristic is to list all the requirement of an exact solution and to divide these requirements into two classes A and B as follows,

A:

1. Those that are easy to satisfy
2. Those that are not so easy to satisfy.

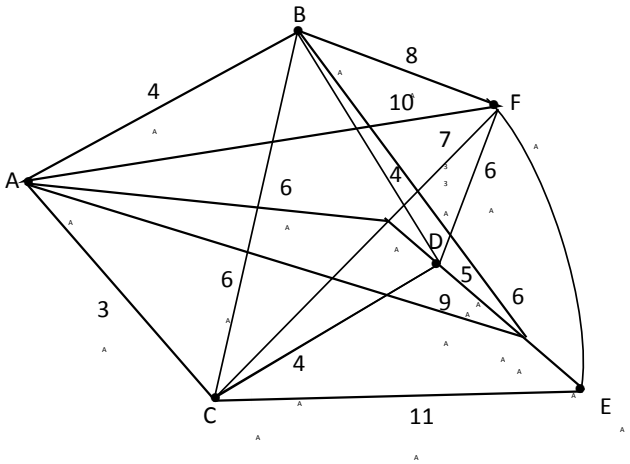
B:

1. Those which must be satisfied
2. Those which we would be willing to compromise.

The design objective then is to construct an algorithm that guarantees the requirements in class A but not necessarily those in class B. This does not mean to imply that no effort is made to satisfy the class B requirements; it merely means that no guarantee can be made. Often very good algorithms have to be considered as heuristics. For example, suppose we have designed a last algorithm which seems to work on all test problems, but we cannot prove that the algorithm is correct. Until such a proof is given, the algorithm must be considered a heuristic.

4. EXAMINING A 6-CITY TSP.

The 6-city network shown below is solved by the use of the Nearest Neighbour Heuristic as well as by exhaustive enumeration. A program was written to evaluate the exhaustive solutions. The results are shown on Table 1.



A 6-city Travelling Salesman Problem.

The preceeding figure can be represented in the following way as a table.

Table 1: Matrix representation of Graph of Salesman Tour.

Vertex	A	B	C	D	E	F
A	0					
B	4	0				
C	3	6	0			
D	6	4	4	0		
E	9	6	11	5	0	
F	10	8	7	6	5	0

The application of the heuristic gives ACDBEFA with cost 32 as the solution. Form the exhaustive enumeration however, 28 is the cost that is the shortest distance. This is therefore a case of trading-off optimality for computational efficiency. Many solution techniques are available for the travelling salesman problem. A large number of these solution techniques rely heavily on advanced results in integer linear programming, non-linear programming and dynamic programming.

The heuristics provide solutions that usually are within a few percent of the optimum. Thus for problems of realistic sizes, heuristics represent a practical solution approach.

ABDEFCA	28	AFDCBEA	41	AFCBDEA	41
ABEDFCA	31	AFDBECA	40	AFDECBA	42
ABEFCDA	32	AFEBDCA	33	AFDBCEA	46
ABEFDCA	28	AFEBCDA	37	AFECDBA	38
ABFEDCA	29	AFBEDCA	36	AFBDECA	41
ACBEFDA	32	AFBCDEA	42	AFBECDA	45
ACDFEBA	28	ACDBEFA	32	ACDEFBA	29
ACFEBDA	31	ACFDEBA	31	ACFEDBA	28
ADFEBCA	32	ADCFEBA	32	ADBEFCA	31
ABCDFEA	34	AFEBDCA	32	ABCDEFA	34
ABCFEDA	33	ABCEDFA	42	ABCEFDA	38
ABDCFEA	33	ABCFDEA	37	ABDCEFA	38
ABDFCEA	41	ABDECFA	41	ABDFECA	33
ABECFDA	40	ABEDCFA	36	ABECDFA	41
ABFECDA	38	ABFDECA	37	ABFDCEA	42
ACBDEFA	33	ABFCEDA	41	ABFCDEA	37
ACBFEDA	33	ACBDFEA	33	ACBEDFA	36
ACDEBFA	36	ACBFDEA	37	ACDBFEA	33
ACEDFBA	37	ACDFBEA	36	ACEDBFA	41
ACEFBDA	37	ACEBDFA	40	ACEBFDA	40
ACFBEDA	35	ACEFDBA	33	ACFDBEA	35
ADCBFEA	38	ACFBDEA	36	ADCBEFA	37
ADCFBEA	40	ADCEBFA	45	ADCEFBA	38
ADBECFA	44	ADBCEFA	42	ADBCFEA	37
ADEBCFA	40	ADBFECA	37	ADBFCEA	45
ADECFBA	41	ADEBFCA	35	ADECBFA	46
ADFBECA	40	ADEFCBA	33	ADEFBCA	33
ADFCEBA	40	ADFBCEA	46	ADFECBA	38
AECDFBA	42	ADFCBEA	40	AECDBFA	46
AECFBDA	45	AECBDFA	46	AECBFDA	46
AEDCFBA	37	AECFDBA	41	AEDCBFA	42
AEDFBCA	37	AEDBCFA	41	AEDBFCA	36
AEBDFCA	35	AEDFCBA	37	AEBDCFA	40
AEBFCDA	40	AEBCDFA	41	AEBCFDA	40
AEFDCBA	34	AEBFDCA	36	AEFDBCA	33
AEFCBDA	37	AEFBDCA	33	AEFBCDA	38
AFCDBEA	40	AEFCDBA	33	AFCDEBA	36
AFCBEDA	40	AFCEDBA	41	AFCEBDA	44

4. CONCLUSION

An introduction to theory and application of graph is presented in this study. Different application areas are highlighted and a common framework of modeling them, that is, the TSP is presented. The concept of heuristics and indeed its applicability is demonstrated in the Nearest Neighbour Heuristic. A 6-city TSP is proposed and solved by use of both heuristic and exhaustive enumeration. From this study, it is realized that the effort required to apply heuristic is by far less than that expended on exhaustive enumeration. Hence in the light of heavy computational complexity, a heuristic approach to optimization problems is preferred. Since whenever it seems hopeless to find optimal solutions, it is still possible to get reasonably good solutions.

BIBLIOGRAPHY

1. Bernstein, Ethan; Vazirani Umesh (1997); “Quantum Complexity Theory”. SIAM, Journal of Computing 26(5):1411.
2. M.R. Garey and D.S. Johnson (1979); “Computers and Intractability: A Guide to the Theory of NP-Completeness”. W.H. Freeman and Company.
3. H.W.L. Von, A.K. Lenstra (1993); The Development of the Number Field sieve, Lecture Notes in Math. 1554 (Springer-Verlag).
4. Simon D.R. (1994); “On the power of quantum computation”. Foundation of Computer Science, 199 Proceedings, 35th Annual Symposium 116-123.
5. Arjen K. Lenstra (2000); “Integer Factoring”, Design, Codes and Cryptography, 19 (2/3): 101-128.
6. Daniel J. Bernstein (2009); “Introduction to Post Quantum Cryptography, Springer, Berlin.
7. Kobayashi, H.; Gall, F.L. (2006); “Dihedral Hidden Subgroup Problem: A Survey”, Information and Media technologies 1(1): 178-185.
8. A bibliography maintained by Daniel J. Bernstein and Tanja Lange on Cryptography not known to be broken by quantum computing.
9. Robert J. McEliece; “A Public-Key cryptosystem based on algebraic coding theory” Jet Propulsion Laboratory, DSN progress Report 42-44, 114-116.
10. Quantum Algorithm Zoo – Stephen Jordan’s Homepage.
11. Quantum Computing with Molecules; articles in scientific American by Neil Gershenfield and Isaac L.Chuang.
12. Krishan K’ (2004); Computer networks and Computer Security.
13. Adam Smith (2008); Understanding Cryptography and Data Security.
14. C.L. Liu, (1986); Elements of Discrete mathematics McGraw-Hill Book Company.
15. E.V. Krishnamurthy (1983); Introductory Theory of Computer Science, Macmillan Press, London.
16. Seymeour Lipschutz (1986); Data Structures.
17. Fabrizio Lucio, Angelo Marzoillo, Paulo Serafini (1993), Mathematics of Computing, UNESCO Project.
18. Alfred V.A., John E.H., and Jeffrey D.U. (1974): The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, Massachusetts.