TITLE: ON EXERCISING EVERY LINK OF A WEB-SITE WITH MINIMAL EFFORT

AUTHOR: VAIDYANATHAN SIVARAMAN

AFFILIATION: SCHOOL OF COMPUTER SCIENCE AND ENGINEERING, ANNA UNIVERSITY

CONTACT ADDRESS: NO.49,VI BLOCK,CEG HOSTELS,
ANNA UNIVERSITY,
CHENNAI — 600025.

E-MAIL ID: vaidy_comp@yahoo.co.uk

ON EXERCISING EVERY LINK OF A WEB-SITE WITH MINIMAL EFFORT

ABSTRACT

In this paper, I consider the problem of traversing every link of a web-site. Since the number of pages and links runs to hundreds and thousands (respectively) in important web-sites, random traversal yields very poor results. I model the problem as a Directed Chinese Postman Problem and I propose an efficient polynomial time algorithm for solving the same. I also indicate numerous situations in which the concept comes into picture.

1. INTRODUCTION

A user of a web site, browsing it, may want to find the quickest route to explore an entire site; more seriously, an employee may be required to check every link of a site for correctness, say if it provides medical information. Since links are often descriptive, and require an understanding of their purpose, they must be checked manually to see whether they link to appropriate pages. However, on following a link, the human checker is now on another page.

2. MAPPING THE LINK-EXPLORATION PROBLEM TO DIRECTED CHINESE POSTMAN PROBLEM

This problem can be modeled as a directed Chinese postman problem. A vertex in the graph corresponds to page and a directed edge represents a link from one page to another. The edges are directed because a link has a direction associated with it. Suppose there is a link from page i to page j. This means that by following the link from page i the browser will go to page j. But the web-site may not have a link from page j to page i.

We construct a weighted digraph G where each edge represents a link, each vertex represents a page and the weight assigned to each edge represents the time it takes to follow the link. The aim is to find the optimal tour from a set of directed tours in which all arcs participate at least once .

3. ALGORITHM FOR SOLVING DIRECTED CHINESE POSTMAN PROBLEM

Here two cases are considered. The given digraph may be Eulerian or non-Eulerian. If the given graph is Eulerian the task is relatively simple. Otherwise, we have to convert the given non-Eulerian digraph into an Eulerian one in an optimal way. First the Eulerian case is considered.

CHARACTERISATION OF AN EULER DIGRAPH :

If all the vertices are balanced i.e in-degree = out-degree for all vertices, the digraph is Euler.

a) ALGORITHM FOR FINDING DIRECTED EULER TOUR IN AN EULER DIGRAPH :

Consider a digraph G in which all the vertices are balanced. Now we construct a walk starting at an arbitrary vertex v and going through the arcs of G such that no arc is traversed more than once. We continue tracing as far as possible. Since every vertex is balanced, we can exit from every vertex we enter; the tracing cannot stop at any vertex but v. And since v is also balanced, we shall eventually reach v when tracing comes to an end. If this closed walk h we just traced includes all the arcs of G, then h is the directed Euler tour we are looking at. If h does not include all the arcs of G, we remove from G all the arcs in h and obtain a subgraph h of G formed by the remaining arcs. Since both G and h have all their vertices in balanced condition, the vertices in h are also balanced. Moreover, h must touch h at least at one vertex a, because G is connected. Starting from a, we can again construct a new walk in graph h . Since all the vertices of h are balanced, this walk in h must terminate at vertex a; but this walk in h can be combined with h to form a new walk, which starts and ends at vertex v and has more arcs than h. This process can be repeated until we obtain a closed walk that traverses all the arcs of G.

b) FINDING THE OPTIMAL TOUR IN NON-EULERIAN SITUATIONS :

Suppose the given directed graph is not Eulerian. We duplicate some of the arcs so that the resulting super-digraph is Eulerian.

DUPLICATION OF ARCS
When we duplicate an arc going from vertex i to vertex j , we will include an additional arc from vertex i to vertex j and having the same weight as the original one. In the WWW context, it means that we are going through the link the second time.

OPTIMAL DUPLICATION OF ARCS
Of course, there are many ways in the arcs of a digraph can be duplicated so that the resulting super-digraph is Eulerian. But we are interested only in the optimal duplication of edges — duplicating edges in such a way that the sum of the weights of the arcs that have been duplicated is minimum.

ALGORITHM

STEP I

Find out the vertices that are unbalanced i.e for which in-degree is not equal to its out-degree. We will have two categories A and B. Those vertices which have in-degree greater than their out-degree are placed in category A . Those vertices which have their out-degree greater than their in-degree are placed in category B. Note that A and B are collections because if a vertex v has in-degree = 4 and out-degree = 2, then it will be placed in category A two times. Formally, a vertex is placed mod(in-degree — out-degree) times in its appropriate category.

STEP II

Using Floyds algorithm we will have to find out the shortest directed path between every pair of vertices a and b such that a is in A and b is in B.

STEP III

Now assign the vertices in A to the vertices in B in an optimal way. The cost of assigning a vertex a in A to a vertex b in B is the length of the shortest directed path from a to b (which has been computed in the previous step for all such pairs). The assignment problem can be solved by using the classic Hungarian Method.

STEP IV

Suppose in the optimal assignment , a is assigned to b. All the arcs in the shortest path from a to b are duplicated in the given digraph G. This procedure is repeated for all the assignments. The result is an Euler super-digraph S of the given digraph G.

Since the assignment is made optimally, optimal duplication of the arcs takes place.

STEP V

In the resulting digraph S (which is Euler) we find the Euler tour using the method used for Euler digraphs.

4. SIGNIFICANCE OF THE ALGORITHM IN THE WWW CONTEXT

Consider a website having hundreds of pages and thousands of links. When there is an urgent need to check whether all the links are working properly, random choosing of links may not result in checking all the links. For instance, the web site for Benjamin Franklin s House [40] had 66 pages and 1191 links at the time of writing. Its optimal CPT of 2248 links is excessive for a human to follow unaided; without following a CPT users would do even more work and be unable to guarantee thorough checking. Hence there is a need for an efficient algorithm.

The above algorithm makes sure we follow links second time only when the corresponding digraph is non-Eulerian. Also solving the resulting assignment problem makes sure that the checking is done in minimal time. The organized approach presented above makes sure that we do not check a link many times unnecessarily.

Suppose that an employee wants to check every link of a web-site for correctness because it provides some important medical information. It will be very difficult for him to do so in the absence of a disciplined approach. The above algorithm can be used in this context because it helps in finding the optimal way in which the checking can be done.

5. CONCLUSION

In the WWW context, there occur several instances in which there is a need to explore all the links of a website. The link-exploration problem has been carefully mapped on to directed chinese postman problem. This paper has developed a polynomial time algorithm for solving the directed Chinese postman problem. The proposed algorithm can be used to find out the order in which links are to be traversed. If a disciplined approach is not adopted, more work is expended and thorough checking is not guaranteed.

REFERENCES

1. R. K. Ahuja, K. Mehlhorn, J. B. Orlin & R. E. Tarjan, Faster Algorithms for the Shortest Path
Problem, *Journal of the ACM*, **37**(2):213—223, 1990.
2. R. K. Ahuja, T. L. Magnanti & J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*,
Prentice-Hall (Simon and Schuster), 1993.
3. S. Albers & M. R. Henzinger, *Exploring Unknown Environments*, Digital Systems Research Center,
SRC Technical Note 1997-014, 1997.
4. J. M. Aldous & R. J. Wilson, *Graphs and Applications*, The Open University, Springer Verlag, 2000.
5. D. Applegate, R. Bixby, V. Chvatal & W. Cook, On the Solution of Traveling Salesman
Problems, *Documenta Mathematica*, Extra Volume **ICM III**:645—656, 1998.
6. K. Arnold & J. Gosling, *The JavaTM Programming Language Second Edition*, 2nd. ed.,
Addison-Wesley, 1998.

7. A. A. Assad & B. L. Golden, Arc Routing Methods and Applications, in *Network Routing*, M. O.
Ball, T. L. Magnanti, C. L. Monma & G. L. Nemhauser, eds., 375—483, North Holland, 1995.