A COMPREHENSIVE PROBLEM FOR ALGORITHM AND PARADIGM VISUALIZATION*

Frank Hadlock *Robert Fly* *Brandon Malone*
Computer Science Department
Tennessee Technological University
Cookeville, TN USA

ABSTRACT

The web site for algorithm visualization described in a previous article [4] has been extended to include the multi-stage Chinese Postman problem in the context of grid graphs. The various stages are solved by algorithms, which illustrate paradigms featured by the site. Objectives of the site include aiding students in visualizing the behavior of these algorithms by actively involving them in manual input of graphs, networks and strings and in stepping through the algorithms, and to provide students with mental graphics model which can be used in problem solving. The visualizations employ a dynamic display of high-level pseudo code with an indicator of the current step being executed. The student controls execution and receives feedback from a 2D graphics illustration of the effects of the individual steps. The multi-stage Chinese Postman algorithm enables the user to step backwards as well as forward.

Keywords: algorithms, paradigms, graphs, Euler Circuits, visualization

1. INTRODUCTION

Algorithm textbooks (e.g. [1]) visualize successive intermediate runtime results. The algorithms visualizations intend to promote a fuller understanding of the algorithm's behavior. Previous studies [5, 6] indicate that students who have learned from passively watching a computer-based animation did not recall the algorithm as well as those with a textbook-based approach. Paper-based animations force students to correlate the sequential versions; textbooks typically try to reinforce learning with questions at the end of the section. Computer-based algorithm visualization offers the potential of using

graphics to display the current state of the algorithm and then reinforce learning by prompting the student for the effect of the next step in the algorithm. This extension to our web site will more actively involve students in studying algorithmic behavior and positively complement textbook training.

The web site also emphasizes the connection between high-level paradigms and specific algorithms, extending the "debugger paradigm" present in many integrated development environments. These debuggers prove helpful in understanding the effect of software at the code level. A The website supposes that high-level pseudo code derived from algorithmic paradigms will enable users to better understand the algorithms behavior, especially when a graphical display provides visual feedback to the user. For example, a backtracking algorithm for finding the longest path cycles through high-level steps to extend the path, backtracking when extension fails because the only neighbors are already on the path. The algorithm displays the input graph and the current search tree colored to indicate previously explored paths and the current path under consideration. The web site features five paradigms. The Postman Problem incorporates several algorithms into its solution, providing visualization opportunities for most of the paradigms.

1.1 The Postman Problem and Grid Graphs

The Postman Problem is to find a minimum length edge traversal of a graph, where an edge traversal includes each edge at least once. The problem [6] has many applications, including delivery of services to customers located on the edges of a service network [8-12]. A lower bound to the length of the edge traversal is the number of edges in the network. Euler's work on the Bridges of Konigsberg problem demonstrates that this lower bound can only be achieved in networks in which at most two of the vertices are of odd degree. Euler's proof constructs an Euler circuit for networks with all vertices of even degree. Since most service networks, particularly those with a grid structure, have more than two odd vertices, the problem grows to include adding edges corresponding to additional traversals to make the network even. In order to minimize the length of the traversal, one must add edges to render all odd vertices even. This Postman Problem has been extensively studied and can be broken down into the following stages:

1. Identify odd vertex.
2. Calculate distance between all pairs of odd vertices.
3. Match odd vertices to minimize the sum of the distances between matched odd vertices.
4. Duplicate the edges of the paths between matched pairs of odd vertices to produce an Eulerian multigraph, in which all vertices are even and multiple edges between vertices are permitted.
5. Construct an Euler path or circuit through the Eulerian multigraph.

1.2 Associated Paradigms

The web site features five standard paradigms, which provide a high level view of the approach taken by the algorithms featured on the site.

Each of the five paradigms applies to one of the stages of the Postman Problem added to the site.

- **Incremental**: applies when the algorithm computes an output from sequential input by combining the previous output with the next input. This simple paradigm forms the basis of many elementary tasks such as finding the sum or the maximum of a sequence of numbers. It pertains to the Postman Problem while building the list of odd vertices (stage 1) and finding a circuit in a network of even degree (stage 5).
- **Dynamic Programming**: applies when a manageable number of initial atomic problems can build up a solution. After entering the solutions to the atomic problems, the rest of the table is filled in bottom up fashion, as opposed to divide and conquer's top down recursive approach. This paradigm solves the all pairs shortest path problem and could be used as the solution to stage 2 of the Postman problem. This approach utilizes a distance table which initially includes a 1 in the ij^{th} entry if vertex i is connected by an edge to vertex j and infinity otherwise. The problem is decomposed on the allowable set of intermediate vertices in a shortest path, with the initial table being the solution to the atomic problem of no intermediate vertices. The solution of intermediate vertices zero **..** k-1 is combined or extended to the solution of intermediate vertices zero .. k by choosing the minimum of the distance from i to k plus the distance from k to j and the direct distance from i to j.
- **Greedy**: applies when a sequence of choices based on current conditions, without regard to the effect on final solution, can still optimize the solution. Dijkstra's single source shortest path algorithm utilizes the greedy paradigm since it grows a minimum distance tree by including the nontree vertex closest to the start or source vertex. The author's shortest path algorithm for grid graphs [2,3] also use the greedy paradigm. Although the dynamic programming solution to calculating the all pairs distances between odd vertices might be more efficient with respect to time, it does not lend itself to visualization. For this reason, the web site uses the greedy paradigm for stage 2 of the Postman problem, finding the distance to other odd vertices for each odd vertex. This approach readily lends itself to visualization. Since grid graphs tend to be sparse with the number of edges less than 4n (n = number of vertices) as opposed to general graphs which can have n(n-1)/2 edges, this approach is more space efficient. For example, the site supports grid graphs with up to 30 rows and columns which would require tables with around a million entries.
- **Backtracking**: applies when the output is a sequence over an ordered alphabet, and the sequence can be constructed subject to a constraint which governs partial as well as terminal solutions. If an objective function is involved, it is additive with respect to the terms of the sequence. Matching of vertices in an arbitrary graph can be accomplished in polynomial time using the blossom algorithm, where a blossom refers to an odd circuit. This problem can also be solved in a suboptimum fashion

by backtrack as follows. Name the odd vertices 1 .. 2n (there are always an even number of odd vertices). The matching is represented by a sequence of odd vertices, beginning with 1, where an odd indexed component is matched with its even indexed successor. Since an odd vertex is matched uniquely with another, the backtrack constraint is simply that a vertex cannot be included if it has occurred before. Since the odd vertices are named 1 .. 2n, they are ordered by their names and backtrack can try feasible alternatives systematically by their order. By adding the distances between odd indexed vertices and their even indexed successors, a matching sequence is assigned a cost which is an objective function to be minimized.

- **Divide and Conquer**: applies when the problem instance can be divided into smaller instances of the same problem type and an answer computed from answers to these instances, or the problem instance is atomic and the answer is immediate. Divide and conquer performs more efficiently when partitioning induces distinct sub-problems. Stage 5 of the Postman Problem involves finding the Euler circuit of the Eulerian multigraph constructed in stage 4. Finding the Euler circuit can be viewed as an incidental divide and conquer solution, rather than a deliberate divide and conquer solution. Atomic problems consist of incrementally finding a circuit by extending a path one edge at a time. Only the starting vertex has an odd number of edges not in a circuit. Consequently, the process terminates at the start vertex. The removal of this circuit from the graph leaves a network of even degree. Thus, this decomposition produces instances of the same problem which later combine to solve the original problem, which is a common characteristic of divide and conquer. In order to find a solution (circuit) to the remaining problem to combine with the previous solution, a new starting vertex is chosen as any vertex on the previous circuit which has an unused edge. This edge begins a solution (circuit) to the remaining problem, which combines with the previous solution in a splicing process: traverse the original circuit to the new starting vertex; follow the new circuit back to the new starting vertex; and finally complete the original circuit. This process continues until the circuit contains every network edge.

2. ALGORITHM VISUALIZATION

Visualization of the Postman Problem begins with manual entry of a grid graph. While other problems featuring general graphs involve clicking on both vertices to add an edge, the Postman section enables the user to input a complete grid by clicking on gridpoints at opposite corners of the desired grid. Below, the user has entered a 1x1 composed with a 2x1 grid.

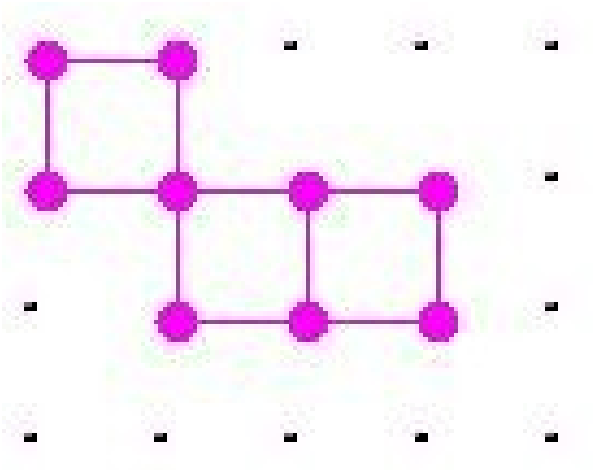

Figure 1a. Manual input of grid graph.

	Step
1.	Identify odd vertex.
2.	Calculate distance between all pairs of odd vertices.
3.	Match odd vertices to minimize the sum of the distances between matched odd vertices.
4.	Duplicate the edges of the paths between matched pairs of odd vertices.

Figure 1b. Pseudocode for stages 1 through 4 of the Postman Problem.

After entering a grid graph, the user can see the effects of stepping through each of the 1st stages of the Postman problem as shown below. Clicking the Step button highlights the next stage of the pseudocode and shows the effect in the graphics window.

Figures 2a and 2b below show the results of stage 1 and stage 4 computation for a more complex grid graph. Stages 2 and 3 are omitted because at the visualization site, they are illustrated with colored edges which does not show up in grey scale.

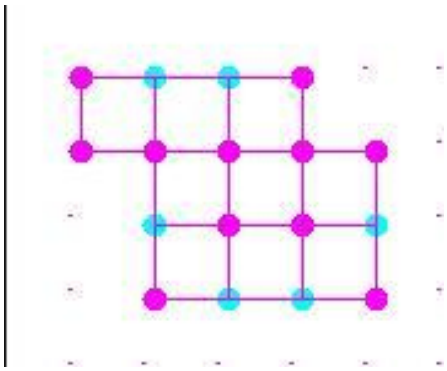

Figure 2a. First stage identification of odd vertices for a more complex grid graph.

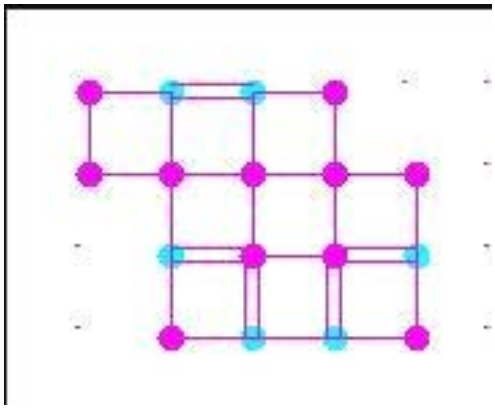

Figure 2b. Fourth stage duplication of shortest paths connecting matched odd vertices

The stage 5 Divide and Conquer algorithm uses the Incremental paradigm to solve the atomic problem of finding a circuit in an Eulerian multigraph by dividing the graph into two components: a circuit and a remaining Eulerian multigraph. The user selects the initial starting. The next starting vertex is picked as v in the first component with an unvisited neighbor. Splice is then called which recursively calls the FindPath for the remainder and composes the two solutions. Pseudocode for Splice (not shown) is displayed during execution.

```
Find Path(G)

    vertex v = any_vertex(G)
    path P = (P.start_vertex = v, P.end_vertex = v)
    Repeat
      test = Extend(P)
    Until not test
    While remaining_unvisited_vertices
        vertex v = any_vertex(P)
            And v_has_unvisited_neighbor
        P = Splice(P,v)
      End While
       Return P
  End
```

Figure 3a. Main pseudocode for stage 5 Postman algorithm.

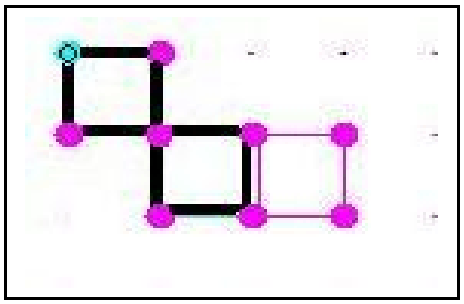

Figure 3b. Completion of one decomposition of main Divide and Conquer.

3. IMPLEMENTATION

The algorithms are implemented as Java applets. The project emphasized software reuse. For example, several of the algorithms use manual graph input and pseudo code tracking. To ensure a consistent look, graph display is another operation shared by various algorithms.

4. WEBSITE

The site employs an hierarchal organization, donating navigational divisions to paradigms, algorithms and questions. Each paradigm has its own page with applicability and high-level pseudo code. Each algorithm has a background page with problem statement and terminology, a pseudo code page with overlying paradigm, an analysis page with runtime and storage space analysis, a visualization page for viewing algorithmic behavior, and a references page with links to related work. The questions division will allow users to ask question about paradigms or algorithms, and to enter suggestions.

Many students find that the visualizations provide dynamic examples that complement texts very well. Because the visualizations allow the students to modify the conditions under which the algorithms operate, they instantly answer students' "What If…" questions. The url for the web site is www.csc.tntech.edu/~algviz/.

REFERENCES

[1] Cormen,T., C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms*, 2nd Ed., McGraw Hill, 2001.

[2] Hadlock,F., "A Shortest Path for Grid Graphs," *Journal of Network Theory*, 1977.

[3] Hadlock,F., "An Edit Distance Based Algorithm for String / Sequence Classification," Proceedings of the 39th Annual ACM Southeast Conference, March 2001.

[4] Hadlock,F., et al, "An Internet Based Algorithm Visualization System," *The Journal of Computing Sciences in Colleges*, Vol 20 Number 2 (December 2004) , pp 304-310.

[5] Hubscher-Younger,T. and N. Narayanan, "Constructive and Collaborative Learning of Algorithms," Proceedings SIGCSE 2003, pp 6-10.

[6] Hundhausen, C., S. Douglas and J. Stasko. "Meta Study of Algorithm Visualization Effectiveness," *Journal of Visual Languages and Computing*, 2002.

[7] Kuan (Kwan or Guˇan) Mei-Ko, "Graphic Programming Using Odd or Even Points," *Chinese Mathematics*, 1:273–277, 1962.

[8] Orloff, C. S., "A Fundamental Problem in Vehicle Routing," *Networks*, 4(1):35–64, 1974.

[9] Papadimitriou C. H., "On the Complexity of Edge Traversing," *Journal of the ACM*, 23:544–554, 1976.

[10] Pearn, W. L. and C. M. Liu, "Algorithms for the Rural Postman Problem." *Computers & Operations Research*, 22(8):819–828, 1995.

[11] Pevzner, P. A., H. Tang and M. Waterman, "An Eulerian path approach to DNA fragment assembly," *Proceedings of the National Academy of Sciences*, 98(17):9748–9753, 2001.

[12] Shen, Y. N. and F. Lombardi, "Graph Algorithms for Conformance Testing using the Rural Chinese Postman Tour," *SIAM Journal on Discrete Mathematics*, 9(4):511–528, 1996.