An application of Spanning Tree Algorithm to Municipal Solid Waste Management

J. Suresh Kumar[1], Satheesh E.N[2]
Post-Graduate Department of Mathematics,
N.S.S.Hindu College, Changanacherry,Kerala, India-686102

ABSTRACT: *Regarding the scheduling problem of the trucks which collects municipal solid waste, a major concern to the municipal administrators is how to effectively distribute collection vehicles and crews in the region. Vehicle Routing Problem (VRP) deals with the problem of allocating trucks to the waste collection sites with minimum cost. There were many studies and methods for the determination of the optimal vehicle routing for solid waste collection. Depending on the complexity of the model, exact and heuristic methods have been developed. One of the best-known approach to VRP is the "savings" algorithm of Clarke and Wright[4]. In this paper, we formulate a weighted-graph model of the VRP and analyze worst-case algorithmic complexity of the Clarke-Wright algorithm to determine optimal routes which allow minimizing total distance traversed, total rounds duration and financial costs including salary, fuel and vehicle operation. Also, some practical infeasibility of Clarke-Wright algorithm is pointed out and better alternate, simple graph-based algorithm for VRP is suggested.*

Key Words: *Solid Waste Management, Vehicle Routing Problem, Clarke-Wright Algorithm, Algorithmic Complexity, Graphs, Spanning Trees.*

1. Introduction

In the last two decades, several analyses of the environment condition conducted showthat one of the major ecological problems is the inadequate waste management. Due to the complexity of the waste management systems, the optimization proceduredepends on several factors such as theselection of waste handling technology, the selection of waste transfer stations, the selection of waste collection and transportation routes etc.There are many studies and methods for the determination of the optimal vehiclerouting for solid waste collection [1].

Depending on the complexity of the models, exact and heuristic methods have been developed.The application of the exact methods of problem solving is limited to some simple models,while more complex models are solved either by applying heuristic methods or bydividing the problem into several phasesand solved by using optimization techniques like Linear Programming, Integer Programming, Genetic Algorithms, Quadratic Programming, Goal Programming, Fuzzy Goal Programming, GISTechniques etc. [2,3].

One of the best-known approaches to Vehicle Routing Problem (VRP) was the "savings" algorithm of Clarke and Wright [4]. In this paper, we formulate a graph theoretical model of the VRP and analyze the worst-case algorithmic complexity and intractability of the Clarke-Wright algorithm. Also, some practical infeasibility of Clarke-Wright algorithm are pointed out and an alternate, efficient graph-based algorithm for VRP is suggested.

2. Graph-theory Formulation of Clarke-Wright Algorithm

In this section, we review the notion of 'graphs' in Mathematics and introduce some basic terminology required in this paper. We then discuss the Clarke-Wright algorithm and formulate its graph theoretical model.

2.1. Graph Preliminaries

In this section, we review the notion of graphs and some basic terms in Graph theory, that are used in this paper. For any terms not defined here, the reader may refer [8].

A graph,G, is a discrete mathematical structure consisting of a set,V, of objects (called vertices) and a set, E, of unordered pairs of vertices (called edges). If each edge has a weight (a number) associated to it, the graph is called a weighted graph. When we need to mention the vertex set and edge set of a graph it is denoted as G=(V, E). If the edge set of a graph is a collection of 'ordered pairs' of vertices, then it is called a 'directed graph'.

If {i, j} is an edge of a graph G, we say that i and j are adjacent to each other. Also i and j are called the end-points of that edge. If (i, j) is an edge of a directed graph G, we say that i is adjacent to j. If a vertex is not adjacent to any other vertex, it is called an isolated vertex.

Let G=(V, E) be a graph. Then H=(V′, E′) is called a subgraph of G, if $V' \subseteq V$ and $E' \subseteq E$. A sub-graph of G which contains all the vertices of G (that is,$V' = V$) is called a spanning subgraph of a graph, G.

A path $P = (v_0, e_1, v_1, e_2, \dots e_n, v_n)$ in a graph G is a traversal through the graph G, where $v_0, v_1, \dots, v_n$ are vertices and $e_1, e_2, \dots e_n$ are the edges such that$e_1 = \{v_0, v_1\}, e_2 = \{v_1, v_2\}, , \dots e_n = \{v_{n-1}, v_n\}$. We also say that the path P connects the two vertices v_0 andv_n.If$v_0 = v_n$, then it is called a cycle.

A graph G is said to be connected if every pair of vertices of G are connected by a path in G. If a

graph G is not connected, it is called a disconnected graph, which may look like several connected graphs put together. Each of these parts is called a component of the disconnected graph, G. Formally, a component of a graph G can be defined as the maximal connected subgraph of G. A graph G is connected if and only if g has exactly one component.

A connected graph without any cycles is called a tree. It is well known in Graph theory [8] that the number of edges of a tree is one less than the number of vertices of it. A spanning tree of a connected graph is defined as a sub-graph which itself is a tree.

Theorem: A connected graph G with p vertices and q edges is a tree if and only if p=q+1.

Thus, any choice of p-1 edges in a connected graph, G with p vertices forms a spanning tree of G and hence contains all its vertices. If G is a weighted, connected graph, the weight of a spanning tree, T, of G is defined as the sum of the weights of the edges of T. Among all the spanning trees of G, the spanning tree with the maximum weight is called a maximum weight spanning tree.

2.2. Clark-Wright Algorithm for Vehicle Routing Problem

Suppose that there is a depot, D and N waste-collection demand points. Suppose that initially the solution to the VRP consists of using N vehicles and dispatching one vehicle to each one of the N demand points. Now if we use a single vehicle to serve two points, i and j, on a single trip, then the distance traveled is reduced by an amount, $S_{(i,j)}$ (Figure. 2.1)

$$S_{(i,j)} = 2d(D,i) + 2d(D,j) - [d(D,i) + d(i,j) + d(D,j)] = d(D,i) + d(D,j) - d(i,j)$$

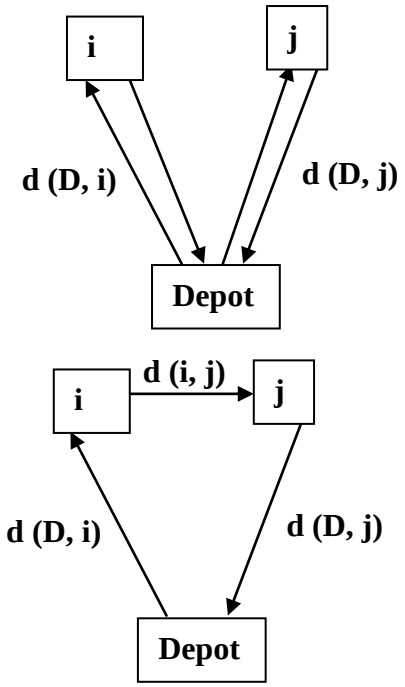

Figure 2.1.Savings concept in Clarke-Wright Algorithm

The quantity, S(i, j), is known as the "savings" resulting from combining two points *i* and *j* into a single tour. The larger *S*(*i, j*) is more desirable to combine *i* and *j* in a single tour. However, *i* and *j* cannot be combined if in doing so the resulting tour violates any constraints of the VRP.The algorithm can now be described as follows.

Step.1: Suppose that there are N demand points. Calculate the savings *S(i, j) = d(D, i) + d(D, j) − d(i, j)* for every pair *(i, j)* ofdemand points. There will be at most N(N-1)/2 such pairs.

Step.2: Rank the list of savings,*S(i, j),* in descending order of the magnitude. Process this"savings list" beginning with the firstentry in the list (the largest *S(i, j)*).

Step.3: For each of the savings,*S(i, j),* include the link (i, j) in a route if noroute constraints will be violated through the inclusion of (i, j) in a route, and

i) If neither i nor j have already been assigned to a route, initiate a new route including the points i,j and the link, (i,j).

ii) If exactly one of the two points (i or j) has already been included in an existing route and that point is not interior to that route (a point is *interiorto a route* if it is not adjacent to the depot D in the order of traversal of points), add the link (i, j) to that same route.

iii) If both i and j have already been included in two different existing routes and neither point is interior to its route, merge the two routes into a single one.Continue this process with the next entry in the list till the savings list S(i, j) is exhausted.

Step.4: Stop. The solution to the VRP consists of the routes created during Step 3. (Any points that have not been assigned to a route during Step 3 must each be served by a vehicle route that begins at the depot *D* visits the unassigned point and returns to *D*).

2.3. Graph Model of the Clark-Weight Algorithm

Suppose that there is a depot, D and N demand points for waste collection. Consider a graph G with the N demand points as its vertices. Its edges are defined as follows: join two vertices, *i* and *j* ,to an edge, {i, j}, if it does not violate any constraints of the VRP for including thatlink in a tour. For each edge, {i,j}, assign the savings quantity,S{i, j},as its weight. This results in a weighted graph with N vertices.Now, the algorithm can now be described as follows.

Step.1: Calculate the savings S{i, j} = d{D, i} + d{D, j}–d{i, j} for eachedge {i, j} of G.[There will be at most N(N-1)/2 such pairs]

Step.2: Rank the list of savings,S(i, j), and arrange them in descending order of magnitude.

Step.3: Process this "savings list" beginning with the first entry in the list (the largest S(i, j)).

For each of the savings, S(i, j), include the respective edge (i, j) in a route if no route constraints will be violated through the inclusion of (i, j) in a route, and

i) If neither i nor j have already been assigned to a route, initiate a new route including i, j and the edge {i, j}.

ii) If exactly one of the two vertices,i or j, has already been included in an existing route and that vertex is not interior to that route (a vertex is interior to a route if it is not adjacent to the depot D in the order of traversal of vertices), add the edge{i, j} to that same route.

iii) If both i and j have already been included in different existing routes and neither vertexis interior to its route, then merge the two routes into a single one by including the edge {i, j}.

Continue with the next entry in the list till the savings list S{i, j} is exhausted.

Step.4: Stop. The solution to the VRP consists of the routes created during Step 3. (Any vertex that is not assigned to a route during Step 3 must be served by a vehicle route that begins at the depot *D* visits the unassigned point and returns to *D*. Such vertices are isolated vertices of the graph, G)

3. Algorithmic Complexity of Clarke-Wright Algorithm

Let G has N vertices (demand points). The complexity of the Clarke-Wright Algorithm is calculated as follows: In step-1,we calculate the savings, S{i, j}, for every pair {i, j} of demand points and since there are at most N(N-1)/2 such pairs and 2 operations (one addition & one subtraction) is needed for the computation of S{i, j} = d{D, i} + d{D, j} – d{i, j}, the worst-case number of operations in this step is N(N-1).

In Step-2, we need to rank the savings, S(i, j), by listing them in descending order of magnitude. For this, we need to use some sorting algorithm. With Bubble sort, we need to compare each entry with all the remaining entries and interchange the two savings, if there is any mismatch. The number of operations (comparisons and exchanges for mismatches) required in this step is $\frac{2N(N-1)}{2} = N(N-1)/2$.

In Step-3, for each of the edges, we inspect the end points of the edge and take some decisions like including the edgein a route, starting a new route or merging two existing routes. This may take a maximum of $2\frac{N(N-1)}{2} = N(N-1)$operations.

Hence the total number of operations (worst case) of the algorithm is 3N(N-1). Although this algorithm is an efficient one, in the sense that it requires only a polynomial time, $3N^2$-3N3$N^2 - 3N = O(N^2)$, for its completion, this can be significant and sometimes unaffordable in the case of cities with large number of waste collection points. For example, a computer, which can perform a huge128 ~ 2^7 operations per second can do only 0.9×2^{13} operations per minute, can do only 0.9×2^{19} operations per hour and can do only 1.3×2^{23} operations per day.So, for cities with moderate size of 500 ~ 2^8collection demand points, the algorithm will take more than 2^{16}operations and even a computer, which can perform a huge 128 operations per second will take about one hour to produce the solution forproblems involving demand points of even such sizes.

4. A Graph-based Algorithm for VRP

In this section, we present a graph-based algorithm with only at most 2N(N-1) operations. Suppose that there is a depot, D and N demand points. Suppose that initially the solution to the VRP consists of using N vehicles and dispatching one vehicle to each one of the N demand points. Now if we use a single vehicle to serve two points, say *i* and *j*, on a single trip, the total distance traveled is reduced by the amount, $S_{(i,j)}$, where

$$S_{(i,j)} = 2d(D,i) + 2d(D,j) - [d(D,i) + d(i,j) + d(D,j)] = d(D,i) + d(D,j) - d(i,j)$$

Consider a graph G with the N demand points as its vertices. Its edges are defined as follows: join two vertices, *i* and *j*, into an edge, {i, j}, if it do not violate any constraints of the VRP for including that link in a tour. For each edge, {i, j}, assign the savings quantity, S{i, j}, as its weight. This will give us a weighted graph, G, with N vertices.

Without loss of generality, we assume that G is a connected graph with no isolated vertices, because if G has isolated vertices, any such vertex may be served by a vehicle route that begins at the depot *D* visits that vertex and returns to *D*. If G is not connected, we may apply the algorithm to each component of G.

Now, the algorithm can now be described as follows.

Step.1: Calculate the savings S{i, j} = d{D, i} + d{D, j} – d{i, j} for every pair {i, j} of (demand point) of the graph,G. There will be at most N(N-1)/2 such pairs.

Step.2: Find the edge with the largest savings in G and include it in a route. Remove this edge from G to obtain a graph G'or avoid this edge for consideration in future.

Step.3: If N-1 edges are selected for the tour, Stop. Otherwise, find the edge with the largest savings in the remaining graph, G'. Include it in the route and remove thatedge from G'. Continue this process of selecting edges and removing them from G' until N-1 edges are selected.

The solution to the VRP consists of the N-1 edges selected during Step 3.Thus, the solution of the VRP reduces to the graph-theoretic problem of finding the maximum weight spanning tree of a weighted, connected graph.

This graph based algorithm finds the solution by just finding the N-1 edges with the larger savings, while Clarke-Wright algorithm requires one torank all the edges in the descending order of magnitude and then inspecting all of them for possible inclusion in a route. Step-1 requires at most 2N(N-1) operations to find all the weights (savings) of the graph. Steps 2 and 3 require to find only the largest N-1 savings, which (in worst case) may require at most N(N-1) operations. Thus this algorithm finds the solution in at

most 2N(N-1) operations only. Although the computational complexity remains the same $O(N^2)$, it matters in the municipal cities like Bangalore and Mumbai, which have large number of waste collection points.

References

[1] Truitt, M., Liebman, J., Kruse, C., 1969. Simulation model of urban refuse collection. Journal of theSanitary Engineering Division, 289–298.

[2] Armstrong, J.M., Khan, A.M., 2004. Modeling urban transportation emissions: role of GIS. Computers,Environment and Urban Systems 28, 421–433.

[3] Lopez Alvarez, J.V., Aguilar Larrucea, M., Fernandez-Carrion Quero, S., Jimenez del Valle, A., 2008.Optimizing the collection of used paper from small businesses through GIS techniques: Legane´scase (Madrid, Spain). Waste Manage. 28, 282–293.

[4] Clarke, G. & Wright, J.W.: "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", *Operations Research*, Vol. 12, 1964, pp. 568-581.

[5] Vehicle Routing Efficiency-Comparison of districting method &Clarke-Wright method, American Journal of Agricultural Economics, Vol. 62, No. 3 (Aug., 1980), pp 534-536

[6] Danijelmarković, dragoslavjanošević, miomirjovanović, Vesnanikolić, Application method for optimization in solid Waste management system in the city of NIŠ, FactauniversitatisSeries: Mechanical Engineering vol. 8, no 1, 2010, pp. 63 – 76

[7] Alan Gibbons, Algorithmic Graph Theory, Cambridge University Press, 1985.

[8] Frank Harary, Graph Theory, Narosa Publishers, 1996.