M. Tech (CS) Dissertation Series

OPTIMAL SCHEDULING OF TRUCKS FOR THE CLEARANCE OF GARBAGE OF A CITY

A dissertation submitted in partial fulfillment of the requirements for the M. Tech (CS) degree of the Indian Statistical Institute

By

INDRANIL OJHA

Under the supervision of

Dr. Bimal Kr. Roy

1993

Indian Statistical Institute
203, B.T. Road, cal - 108

CERTIFICATE

This is to certify that the work described in the dissertation entitled 'Optimal Scheduling of Trucks for Garbage Clearance of a City', has been undertaken by Indranil Ojha under my guidance and supervision. The dissertation is found worthy of acceptance for the award of the degree of Master of Technology in Computer Science.

(Dr. Bimal Kumar Roy)

Indian Statistical Institute

Calcutta.

Dated : July ,1993.

Place : Calcutta

ACKNOWLEDGEMENT

It's a matter of great pleasure for me to express my deepest regard to Dr. Bimal Kumar Roy under whose supervision the project has been carried out. It was a sheer joy and excitement of learning things while working under his great optimism, all-out help and watchfull eyes lined with critical apprecitation.

This would remain incomplete without mentioning the name of Dr. K. Sikdar of my institute since all my knowledge regarding the algorithms, complexity as well as graph theory is due to him.

I must thank the staffs of CSSC the warm help they extended during the entire period. Finally, I thank my friend, Palash Sarkar for his all-time help and cooperation during carrying out this work.

Dated :

(INDRANIL OJHA)

Indian Statistical Institute

Calcutta.

CONTENTS

CHAPTER I : INTRODUCTION

The problem of constructing an optimal schedule of trucks for garbage clearance of a city is considered. The garbage is assumed to be accumulated along the streets of the city. The requirement is to minimize the total distance covered and also the number of trips. The optimasation problem of minimization of distance is shown to be NP-complete. It is also shown that there does not exist any k-absolute approximate algorithm for the above problem. An algorithm, however, has been found which yields a 1-relative approximate schedule for the case of two trips. This algorithm is generalised for the case of more than two trips using the divide and conquer strategy. This general algorithm is shown to require an optimal number of trips, along with an upper bound on the total distance covered. Further, this algorithm is modified to a heuristic, which is empirically seen to require a lesser amount of distance covered but at the cost of slight increase in the number of trips. However, there exists an upper bound on the number of trips required.

CHAPTER II : DESCRIPTION OF THE PROBLEM

Given the map of a city, length and garbage accumulation of each road, we are going to find out a schedule of trucks such that the total garbage is cleared in several trips of trucks each of which clears some amount of garbage without exceeding the truck capacity.

An optimal schedule is one that needs the minimum distance traversed by the trucks (i.e total length of all trips) as well as making the number of trips not too large.

We define a Network as a weighed graph (V,E) along with the garbage accumulation function g : E -> N and denote it by N(V,E,g). The length of edge e is denoted by d(e).

A Trip is defined as a network with the extra feature that the corresponding graph (V,E) is an eulerian multigraph with all vertices having positive degree connected with each other.

Now the scheduling problem, which we will call GARBAGE, is formally defined as follows.

Problem Definition :

Given a network $N(V,E,g)$, a distinguished node $v_0 \in V$, and an integer C. Find out a set of trips, which we will call trip-set and denote as $\tau = \{ T_1, T_2, \ldots, T_K \}$, where each trip T_i is of the form $N(V,E_i,g_i)$, such that the following conditions hold :

i) Degree of v_0 is positive.

ii) $g_i(e) = 0$ if $e \notin E_i$, $\forall i$.

iii) $\sum_e g_i(e) \leq C$, $\forall i$.

iv) $\sum_i g_i(e) = g(e)$, $\forall e \in E$.

v) $\sum_i d(T_i)$ is minimum where $d(T)$ is size of a trip T, considering repeatation of edges.

We denote the total amount of garbage as A and the minimum number of trips required as n_τ. Clearly

$$A = \sum_e g(e)$$

and $$n_\tau = \lceil A/C \rceil .$$

In general, the truck capacity is sufficiently larger than garbage accumulation on any road and hence we can assume that

$$g(e) < C/2, \quad \forall e \in E.$$

One immediate result of the assumption is that, $A < mC/2$, m being the number of edges in the graph. There are two more theoretical results given in the next pages.

RESULT - I :

If for some network $N(V,E,g)$, $g(e) > 0, \forall e \in E$ and $A \leq C$, then GARBAGE reduces to the Chinese Postman Problem.

PROOF :

Since $A \leq C$, only one trip is sufficient to clear the garbage. Also $g(e) > 0, \forall e \in E$ implies that each edge is to be traversed at least once.

Thus, we have to find out a cycle with, possibly, repeatations of edges and of minimum length among all such cycles where each edge is used at least once. This is simply the CPP.

The CPP has an $O(n^3)$ algorithm [by Edmonds J.].

Let us denote the optimal cost of GARBAGE as C_o and that of the CPP (on same graph) as C_{CPP}.

SULT - II :

If $g(e) > 0, \forall e \in E$, then

$$C_{CPP} \leq C_o \leq n_T . C_{CPP} .$$

Proof :

Let the optimal schedule contain k trips. Each of these k trips are cycles through the node v_o and hence concatenation of them is also a cycle. Since $g(e) > 0, \forall e \in E$, each edge is to be traversed, and this is a feasible CPP solution. But length of the minimum of all such feasible solutions of the CPP is C_{CPP}. Hence, $C_{CPP} \leq C_o$.

For the other part, we will see that there is always a schedule with cost $n_T.C_{CPP}$ and hence the optimal schedule does not cost more than this value.

Our above-mentioned schedule consists of n_T trips and in each trip, a truck travels through the CPP solution of the graph and clears garbage from roads at random only maintaining the condition that in all but last trip the truck clears as much garbage as possible, i.e., upto its capacity. Hence, at the last trip, the amount of garbage left out is maximum C which can be cleared out in the last trip.

The cost of above schedule is $n_T.C_{CPP}$. Hence the proof.

CHAPTER III : COMPUTATIONAL COMPLEXITY OF GARBAGE

During our effort to get an efficient algorithm that solves GAEBAGE optimally, we found it very difficult and we started thinking of its computational complexity. The following theorem is a result of that.

THEOREM I : The decision version of GARBAGE,(i.e. $GARBAGE_d$) is NP-complete.

PROOF : i) $GARBAGE_d \in NP$.

The decision version of GARBAGE is,

Given a network $N(V,E,g)$, a specified node $v_o \in V$, truck capacity C, and an integer b, does there exist a schedule of cost < b ?

The following polynomial time algorithm solves the problem nondeterministically.

STEP I : Guess any tour T in the graph.

STEP II : Assign function g' to the tour nondeterministically maintaining the constraints that

$g'(e) = 0$ if $e \in T$,

$0 \leq g'(e) \leq g(e)$, $\forall e \in E$,

and $g'(e) \leq C$.

STEP III : $\forall e \in E$ do $g(e) := g(e) - g'(e)$.

STEP IV : Include $<(V,T,g')>$ in the schedule.

STEP V : If $g(e) > 0$ for some $e \in E$ then go to STEP I.

STEP VI : If total cost of the schedule is not more than b, then conclude 'YES'.

ii) GARBAGE$_2$ is NP-hard.

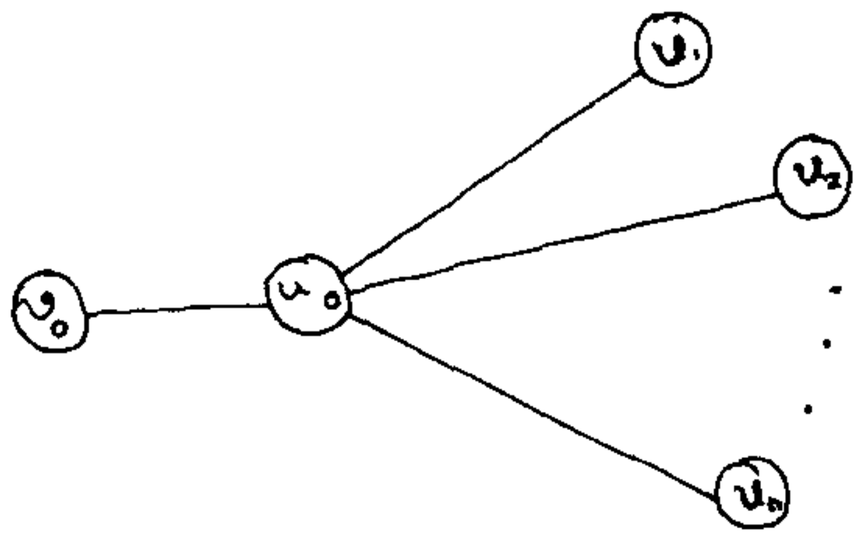
For this part, we will show that, PARTITION problem which is already proved to be NP-complete reduces to GARBAGE in polynomial time.

The PARTITION problem is defined as,

Given a set $S = \{a_1, a_2, \ldots, a_n\}$ of n integers such that $\sum_i a_i = 2T$ where T is an integer. Does there exist a set $R \subset S$ such that

$$\sum_{a_i \in R} a_i = \sum_{a_i \notin R} a_i = T \ ?$$

From an instance of PARTITION, we can produce an instance of GARBAGE in linear time as follows.



The network is a graph of the above figure, each edge being of unit length. The specified node is v_o .

$$g((v_o, u_o)) = 0$$

$$g((u_o, v_i)) = a_i$$

and capacity = T.

The question is,'does there exist a schedule of cost $\leq 2n+4$?'.

Now, we will show the equivalence of the two instances regarding their answers.

If PARTITION instance has an 'yes' answer then there is a set $R \subset S$ such that $\sum_{a_i \in R} a = \sum_{a_i \notin R} a = T$. Without loss of generality, we can assume that, $R = \{a_1, a_2, \ldots, a_k\}$ and $S = \{a_{k+1}, a_2, \ldots, a_n\}$. Think of a schedule with two trips

$$T_1 = < v_0, u_0, u_1, u_0, u_2, u_0, \ldots, u_0, u_k, u_0, v_0 >$$

and $$T_2 = < v_0, u_0, u_{k+1}, u_0, u_{k+2}, u_0, \ldots, u_0, u_n, u_0, v_0 >$$

Take $g_i(e) = g(e)$ if e occurs in T_i,

$= 0$ otherwise for $i = 1,2$.

$$\sum_e g_1(e) = \sum_{e \in T_1} g(e) = \sum_i g((u_0, u_i)), \text{ where } i \leq k.$$
$$= \sum_{i=1}^{k} a_i = T.$$

Similarly, $\sum_e g_2(e) = T$.

Hence the conditions are satisfied. Total cost = 2n+4.

Thus, GARBAGE instance also has an 'yes' answer.

On the other hand, if the GARBAGE instance has an 'yes' answer, then there exists a schedule of cost < 2n+4. Since A = 2T = 2C, there must be at least two trips. As each trip uses the edge (v_0, u_0) twice, and each edge (u_0, v_i) having a vertex v_i of degree one, is to be used twice in at least one trip, there can be no schedule of cost less than 2n+4. Hence the schedule has exactly two trips using (v_0, u_0) twice in each trip and each edge (u_0, v_i) is used twice in one of the trips and not used at all in the other. Garbage cleared in each trip must be exactly equal to T as there are only two trips and A = 2T.

Note that the two trips partition the set of edges $\{(u_0, v_i)\}$ in two parts. Let $R = \{a_i : (u_0, v_i)$ is cleared in first trip $\}$.

Clearly, $\sum_{a_i \in R} a_i$ = Total garbage cleared in the first trip.

$= T$.

Hence the answer to the PARTITION instance is also 'yes'.

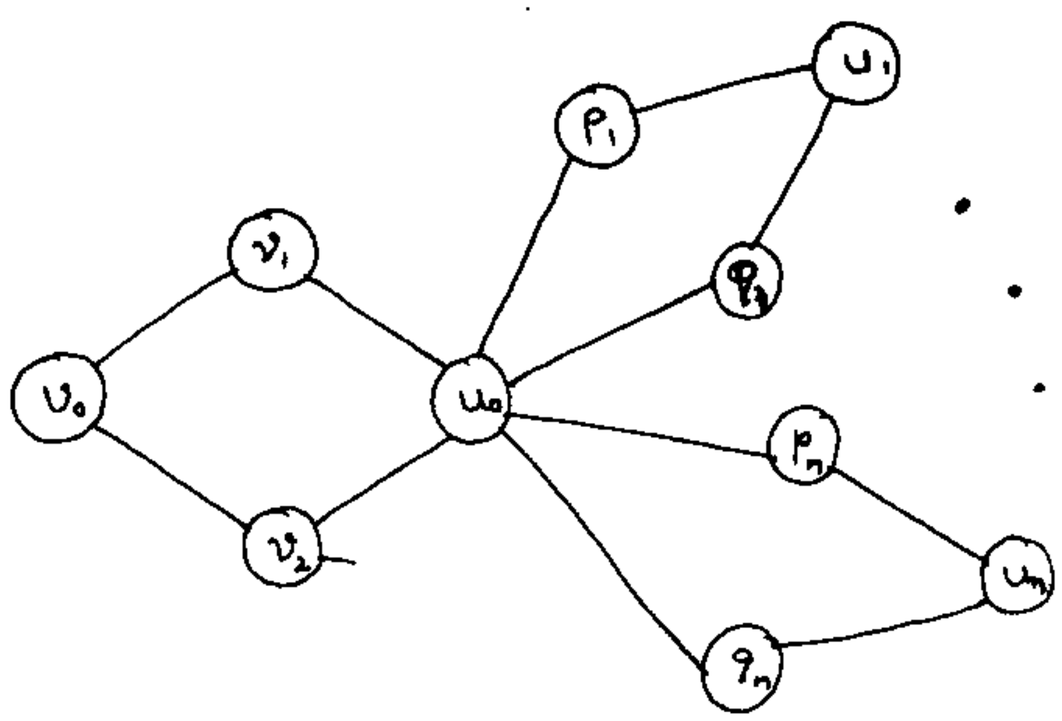
COROLLARY I : GARBAGE is NP-complete even if the underlying graph is acyclic.

COROLLARY II : GARBAGE is NP-complete even if the underlying graph is planar.

COROLLARY III : GARBAGE is NP-complete even if the underlying graph is eulerian.

COROLLARY IV : GARBAGE is NP-complete even if $A \leq 2C$.

The proof of the theorem directly proves corollaries I,II & IV as the GARBAGE instance obtained by reduction satisfies conditions in these corollaries. For the proof of corollary III, we will just give the method of reducing and the rest part of the proof is similar.



The eulerian graph is given above. Each edge is of unit length. The garbage-assignment is as follows :

$$g(v_0, v_1) = g(v_0, v_2) = g(v_1, u_0) = g(u_0, v_2) = 0$$

$$g(u_0, p_i) = g(u_0, q_i) = g(u_i, p_i) = g(u_i, q_i) = a_i.$$

and capacity = 4T.

and the question is "does there exists a solution of cost < 4n+8 7". The graph is eulerian and this instance can be proved to be equivalent to the PARTITION instance. Hence the corollary can be proved.

CHAPTER IV : APPROXIMATE ALGORITHMS

As the garbage has been found to be NP-complete,it is computationally difficult to find the optimal schedule exactly but we can go for approximate solutions by using polynomial time bounded approximate algorithms. There are two kinds of such algorithms. One is k-absolute approximate algorithms in which the error is always less than a predetermined constant k. The other is ϵ-approximate algorithms in which which the error is less than ϵ times the optimal value.

Let A be any algorithm for problem π . Let I be an instance of π . If we denote by $C_A(I)$ the cost of the solution returned by A and by C_0 the cost of the optimal solution then

$C_A(I) - C_0(I) = 0$, if A is an exact algorithm,

$C_A(I) - C_0(I) \leq k$, if A is a k-absolute approximate algorithm,

and $C_A(I) - C_0(I) \leq \epsilon . C_0$, if A is an ϵ-approximate algorithm.

By looking at GARBAGE, one can easily realise that there can be no k-absolute approximate algorithm for this problem. Still we give a brief proof of this fact.

THEOREM II : Unless P = [redacted], there exists no k-absolute algorithm for GARBAGE.

Proof : Let there exist a polynomial time algorithm A for GARBAGE such that

$C_A(I) - C_0(I) \leq k, \quad \forall\ I \in$ GARBAGE.

Then we can use this to solve any instance of GARBAGE exactly. On any instance I of GARBAGE, apply the algorithm of the next page.

ALGORITHM A :

STEP I : Get I' from I by multiplying lengths of all edges by (k+1);

STEP II : Apply A on I'.

STEP III : Return the resulting schedule. Optimal cost of I is cost of this schedule divided by (k+1) i.e.

$$C_0^*(I) = C_A(I')/(k+1).$$

The explanation of above is immidiate. Note that a schedule in I is also a schedule in I' and vice versa as the adjacency of the graph, the truck capacity and the garbage function are unchanged. Also the cost of a schedule in I' is k+1 times the cost of the same schedule in I. The sets of optimal schedules must be the same. Hence

$$C_0(I') = (k+1).C_0(I).$$

By definition of A,

$$C_A(I') \leq C_0(I') + k$$
$$= (k+1).C_0(I) + k.$$

As $C_A(I')$ is a multiple of (k+1),

$$C_A^*(I') = (k+1).C_0(I) = C_0(I').$$

Thus, the resulting schedule is an optimal schedule in I' implying that this is an optimal schedule in I also.

This indicates that there exists a polynomial time exact algorithm of GARBAGE, leading to the conclusion that P = NP.

Now it's the time to search for ϵ-approximate algorithm for GARBAGE. One such algorithm already follows from the proof of the following theorem for a very special class of instances of GARBAGE.

THEOREM III : There exists an ϵ-approximate algorithm for GARBAGE with $\epsilon = 1$ if $A \leq 2C$ and $g(e)>0$, $\forall e \in E$.

Proof : The following FILL & RETURN algorithm finds an approximate schedule for GARBAGE with relative error not exceeding 1.

FILL & RETURN ALGORITHM :

STEP I : Get the CPP solution of the graph. Let it be represented by the list of vertices $\langle v_0, v_1, v_2, \ldots, v_{m-1}, v_m \rangle$ where $v_0 = v_m$.

STEP II : For each i from 1 to m, define q(i) such that

$q(1) = 0$,

$q(i) = q(i-1)$, if the edge (v_{i-1}, v_i) has already occurred before this in the CPP solution, $i>1$.

$q(i) = q(i-1)+g((v_{i-1}, v_i))$, otherwise, $i>1$.

STEP III : Find out i such that $q(i-1)<C$ and $q(i)\geq C$. This is always posssible since q(i) is increasing from 0 to $A>C$.

STEP IV : Split the CPP solution in two paths p1 and p2, which are not necessarily simple, with

$p1 := \langle v_0, v_1, \ldots, v_{i-1}, v_i \rangle$

$p2 := \langle v_i, v_{i+1}, \ldots, v_{m-1}, v_m \rangle$

STEP V : If $q(i)>C$ then $p2 := \langle v_{i-1}, v_i \rangle .p2$ where the '.' means concatenation.

STEP VI : $g_1(e) := 0$, if e does not occur in p1.

$:= C-q(i-1)$, if $e=(v_{i-1}, v_i)$

$:= g(e)$, otherwise.

$g_2(e) := g(e)-g_1(e)$, $\forall e \in E$.

STEP VII : The two trips are T1 and T2 where

T1 := p1.p3, p3 is the shortest path from v_i to v_o.

T2 := p4.p2, p4 is the shortest path from v_o to the first vertex of p2.

The above algorithm gives a valid schedule as

$$0 \leq g_1(e) \leq g(e) < C, \qquad \forall e \in E.$$

$$0 < g_2(e) = g(e) - g_1(e) < C, \qquad \forall e \in E.$$

$$\sum_e g_1(e) = \sum_{j<i} g((v_{j-1}, v_j)) + C - q(i-1)$$

$$= q(i-1) + C - q(i-1)$$

$$= C.$$

$$\sum_e g_2(e) = \sum_e g(e) - C = A - C \leq C \text{ as } A \leq 2C.$$

Now to show the error margin, if we denote by C the cost of the schedule by the algorithm, then

C_A = length(T1) + length(T2)

= (length(p1)+length(p3)) + (length(p2)+length(p4))

< C_{CPP} + C_{CPP} = $2C_{CPP}$, since p3 and p4 are shortest paths.

But by RESULT II, $C_o > C_{CPP}$ and thus $C_A < 2C_o$.

An important aspect to be observed here is that the extra distance to be travelled by the trucks over the CPP solution is the total length of two shortest paths and, if needed, an edge (v_{i-1}, v_i). The maximum value is of this is $2.\max_u(\text{length}(sp(v_o,u))) + \max_e(d(e))$.

Let $\alpha = (2.\max_u(\text{length}(sp(v_o,u))) + \max_e(d(e)))/C_{CPP}$.

Then $(C_A - C_{CPP}) \leq \alpha . C_{CPP}$ which is a good error bound for dense graphs where the value of α is less.

CHAPTER V : ALGORITHM FOR CPP

The CHINESE POSTMAN PROBLEM is defined as the following.

A postman delivers mail along the streets of a place which is represented by a connected graph G. He must traverse each street at least once. He starts and ends at a particular vertex of the graph that represents the post office. What route should he follow in order to walk the shortest distance ?

The name of the problem is after the chinese mathematician Kwan who first solved the problem using the method of 'augmenting chains'. Later, Edmonds J. developed an $O(n^3)$ algm for this. The idea he used was that, as the tour is an eulerian walk, each vertex should be of even degree. Hence the duplicated edges form a set of paths between disjoint odd vertices. We have to search for such a set with shortest total length. This can be done by using the idea of weighted matching. Here follows Edmonds' algorithm for the CPP.

ALGORITHM :

STEP I : Identify the odd vertices. If there are none, go to step IV.

STEP II : Compute the shortest path between each pair of odd vertices.

STEP III : Partition the odd vertices into pairs so that the sum of lengths of shortest paths joining pairs is minimal, by solving the weighted matching problem over a complete graph whose vertices are the odd vertices of the network and weight of an edge is the shortest distance between the coresponding odd vertices.

STEP IV : The edges in the union of the shortest chains picked out in step III are the edges to be traversed twice. Use any efficient procedure to find out an eulerian walk in the graph when the edges have been duplicated.

The above algm takes $O(n^3)$ time as the best known algm for the weighted matching is of the order $O(n^3)$ [Papadimitriou & steiglitz].

CHAPTER VI : HEURISTICS

In this chapter, we are going to develop some heuristics for solving general GARBAGE instances. The algorithm given in the last section is an efficient tool to be used in such efforts. The obvious method that comes first in mind is the DIVIDE & CONQUER STRATEGY that uses this algorithm repeatedly on the given instance breaking it in smaller instances with less garbage accumulation till this value is below the truck capacity.

ALGORITHM DIVIDE & CONQUER :

INPUT : A network N(V,E,g), a vertex $v_o \in V$ and capacity C.

OUTPUT : A set of trips and corresponding garbage functions.

STEP I : If $A \leq C$, then return the CPP solution along with the garbage function identical with g and return.

STEP II : If $A \leq 2C$, use FILL & RETURN algorithm and return.

STEP III : Call the FILL & RETURN algorithm with N, v_o and f(C) as parameters (choice of function f is discussed later). Let the algorithm return trips T1 & T2 with g1 & g2 as garbage functions.

STEP IV : If $\sum_e g1(e) > C$ then call DIVIDE & CONQUER algm recursively on the graph of T1 with capacity C.

STEP V : If $\sum_e g2(e) > C$ then call DIVIDE & CONQUER algm recursively on the graph of T2 with capacity C.

STEP VI : Return all trips obtained in steps IV and V with their respective garbage functions.

In the above algorithm, the function f is still undefined. The choice of f matters a lot in reducing the total number of trips in the resulting schedule. One condition that is necessary in order to ensure that the FILL & RETURN algm runs well is that the input to FILL & RETURN must satisfy the property $n_T = 2$. For this, our choice of f should satisfy $A \leq 2f(C)$. One obvious choice of f is

$$f(C) = \lceil A/2 \rceil.$$

but this never guarantiqs the minimum number of trips.

The FILL & RETURN algorithm returns two trips, one of which clears a truckful amount of garbage and the other clears the rest. Also the DIVIDE & CONQUER algorithm terminates only when in each trip, the total garbage cleared is less than or equal to $f^k(C)$ where k is the maximum depth of recursion. Thus, the amount of garbage in all but possibly the last trip is exactly equal to $f^k(C)$ and that in the last trip is less than or equal to $f^k(C)$.

The value of k can be found out from the fact that k is the minimum integer for which $f^k(C) \leq C$. So, the number of trips in the output is $\lceil A/f^k(C) \rceil$.

To reduce this number, we have to increase $f^k(C)$ as much as possible not exceeding C. Hence, our aim is to choose f such that $f^k(C)$ becomes equal to C. One solution to this is

$$f(C) = kC \text{ where } k \text{ is the smallest integer with } kC \geq A/2.$$
$$= \lceil A/2C \rceil .C$$

for which the number of trips is exactly equal to n_o, the optimal one.

CLAIM : The DIVIDE & CONQUER algorithm gives a schedule with error not exceeding $(1+\alpha)^{\lceil \log_2 n_T \rceil} - 1$ if $g(e) > 0$, $\forall e \in E$, where

$$\alpha = (2.\max_u(\text{length}(sp(v,u))) + \max_e(d(e)))/C_{CPP}$$

if the choice of f is $f(C) = \lceil A/2C \rceil . C$.

PROOF : We will first prove that, if max depth of recursion is k, then then cost of the solution doesn't exceed $(1+\alpha)^k . C_{CPP}$.

Proof is by induction on k.

For k = 1, the FILL & RETURN algorithm is called only once and as we already know, cost of the soln $\leq (1+\alpha).C_{CPP}$.

Let k > 1. Let N be the input network, and N1 & N2 be the two trips at the first level. As N1 & N2 are eulerian multigraphs, their cpp-sizes are equal to their graph sizes. So

$$\text{cpp-size(N1)} + \text{cpp-size(N2)} \leq (1+\alpha).\text{cpp-size(N)}.$$

But the depth of recursion $\leq (k-1)$ for N1 and N2 and thus,

$$\text{cost(N1)} \leq (1+\alpha)^{k-1}.\text{cpp\_size(N1)}$$

and $$\text{cost(N2)} \leq (1+\alpha)^{k-1}.\text{cpp\_size(N2)}.$$

Now, cost of the soln = cost(N) = cost(N1) + cost(N2)

$$\leq (1+\alpha)^{k-1}.(\text{cpp\_size(N1)}+\text{cpp\_size(N2)})$$

$$< (1+\alpha)^{k}.\text{cpp\_size(N)}$$

$$< (1+\alpha)^{k}.C_{CPP}.$$

Thus the assertion holds.

To establish our claim, note that the maximum depth of recursion is the minimum integer k for which $f(C) \leq C$ and that is $\lceil \log_2 n_T \rceil$ for our choice of f. Thus,

$$\text{cost of soln} < (1+\alpha)^{\lceil \log_2 n_T \rceil}.C_{CPP}.$$

Now, using the fact that $g(e) > 0$, $\forall e \in E$, i.e., $C_{CPP} \leq C_o$, we establish our claim.

CHAPTER VII : AN IMMEDIATE MODIFICATION

The FILL & RETURN strategy finds out the vertex in which the the truck is just filled and thus the deviation of the cost of the soln from the cost of the CPP soln is more or less equal to twice the shortest distance of this vertex from v_0. The reason of the choice of this vertex is only that this divides the CPP tour in two parts each with a garbage accumalation less than capacity. A slight change in this strategy can improve the soln cost and yet maintaining the partition criteria.

Actually, any vertex v_i with q(i) lying between A-C and C divides the tour in two such parts for $A \leq 2C$. Why not choose one of them with the shortest distance ? One problem that arises is that there may not be any such vertex. In that case we can use the ordinary algorithm. Thus the modifications to be made in the algm FILL & RETURN are

STEP III′: Find maximum j and minimum i for which q(j) < A-C and q(i) > c. If j-i = 1 then go to STEP IV.

STEP IV′: Find out k such that $j < k < i$ and v_k is nearest from v_0.

STEP V′: Split the CPP tour in two paths

$$p1 := \langle v_0, v_1, \ldots, v_{k-1}, v_k \rangle$$

$$p2 := \langle v_k, v_{k+1}, \ldots, v_{m-1}, v_m \rangle$$

STEP VI′: $g_1(e) := 0$, if e does not occur in p1.

$:= g(e)$, otherwise.

$g_2(e) := g(e) - g_1(e)$, $\forall e \in E$.

STEP VII′: Go to STEP VII .

The above modification of the FILL & RETURN algm is sure to reduce the cost of the soln schedule but the number of trips is no longer optimal as there is no more guaranty of one of the two trips clearing truckful amount of garbage. But the number of trips never exceeds $2n_T$ as will be shown in the next section. So the modified version of FILL & RETURN algm is acceptable.

CHAPTER VIII : MINIMIZATION OF THE NUMBER OF TRIPS

In all discussions in the previous sections, we ignored the minimization of the number of trips, but mentioned that there is way so that this number can always be kept within some limit. Our algorithms give optimal number of trips, i.e., n_T , if we run them without making the modifications we mentioned in the last chapter. But even if we make the modifications, this number never exceeds $2n_T$, if we follow the algm in the proof of the following theorem.

THEOREM IV : For every feasible schedule with more then $2n_T$ trips, there exists a schedule with cost no more than that of the previous one yet having the number of trips $\leq 2n_T$.

PROOF : On the given schedule, apply the following algorithm.

ALGORITHM COMPACT :

Let S be a list and T a set of trips. Head(L) denotes the first element in the list L and Tail(L) denotes the rest part. '.' stands for the concatenation of two lists/trips as applicable.

STEP I : Let S := <>; T := trip_set;

STEP II : For each trip t do

if t clears garbage < C/2 then

{ S := S.<t>;

T := T \ {t}; }

```
STEP III :   While S contains more than one element do
                  {       t1 := head(S); S := tail(S);
                          t2 := head(S); S := tail(S);
                          t := t1.t2 (* concatenation of t1 & t2 *)
                          if garbage cleared by t is more than C/2
                          then    T := T U {t}
                          else    S := <t>.S;     }
STEP  IV :   If S is not empty then T := T U { head(S) };
             Return(T);
```

The above algorithm takes linear time to run and stops only when all the elements in the initial list S is scanned. But at that instance there can be at most one trip clearing less than or equal to C/2 garbage. Let k be the number of trips in the new schedule. Since total garbage cleared in all these trips is A, and there are at least (k - 1) trips clearing more than C/2 amount of garbage, we can say,

$$(k - 1).C/2 < A,$$

or, $$k - 1 < 2A/C,$$

or, $$k \leq \lceil 2A/C \rceil \leq 2\lceil A/C \rceil = 2n_T .$$

Also the cost of schedule is now equal to the cost of the initial schedule but after some processing as discussed in next section the cost may decrease as these trips are concatenations of two or more trips.

CHAPTER IX : FURTHER IMPROVEMENTS

PSEUDO PATHS :

The previous chapter used the fact that, the concatenation of two trips is again a trip if the sum of the amounts of garbage cleared by them doesn't exceed truck capacity. This concatenated trip may have used some path clearing no garbage. Let us call such paths as pseudo paths. The pseudo paths do nothing regarding the clearing of garbage and thier only necessity is in connecting two vertices so that a cycle results. Thus, it is intended that a pseudo path between two vertices must be the shortest path between them.

But the concatenation of two trips does not guaranty this and we have to process the trips in the resulting schedule in order to minimize the contribution of the pseudo paths. This can be done by using many clever methods. One such method is to scan through each of the trips and whenever a pseudo path is met, replace it by the shortest path between the two terminal nodes of the pseudo path. This algm can be implemented in polynomial time.

DECOMPOSITION :

The CPP tour of the given graph may have a number of occurences of the initial node v_0 . In such cases, we can decompose the tour in a number of tours each starting and ending at v_0. Then the algorithms can be applied to each of them seperately. But this may result further increase in the number of trips required. So, the COMPACT algorithm of the last chapter must be used after this.

REFERENCES

1. Combinatorial Optimization, Algorithm and Complexity.

   ----- Papadimitriou & Steiglitz.

2. Combinatorial Optimization --- Christfides et al.

3. Edmonds J. & E.L.Johnson, "Matching Euler Tours &

   The Chinese Postman". Math. Prog.,5(1973),88-124.