Chapter 11

Capacitated Arc Routing Problem with Vehicle-Site Dependencies: The Philadelphia Experience

John Sniezek
Lawrence Bodin
Laurence Levy
Michael Ball

11.1 Introduction

In residential solid waste collection, the vehicle fleet may consist of collection vehicles with varying capacity, size, and shape. Each set of identical vehicles is said to comprise a *vehicle class*. A vehicle from a vehicle class with a small capacity fills up quicker and requires more trips to a disposal facility (such as a landfill) than a vehicle from a vehicle class with a larger capacity. Each vehicle class can have a restriction on the streets that it can service and streets that it can traverse. Vehicles from the larger vehicle classes cannot traverse small alleys or bridges that can support only a specific weight. Some streets allow vehicles from a vehicle class to traverse but not service the street because the street is too narrow to conduct the service (for example, side-loading sanitation vehicles).

A *vehicle-site dependency* on a street is a constraint that prohibits a vehicle of a certain vehicle class from servicing or traversing the street because of some limitation. Thus, a vehicle-site dependency on a street reflects the ability of a vehicle from a vehicle class to service or travel a street. The *Capacitated Arc Routing Problem with Vehicle-Site Dependencies* (CARP-VSD) attempts to solve a Capacitated Arc Routing Problem (CARP) where there are vehicle-site dependencies. The CARP-VSD has received almost no attention in the literature, and Nag [9] is the only paper that we are familiar with that solves the node routing problem with vehicle-site dependencies.

The CARP-VSD is a generalization of the CARP. In the CARP, a connected directed network $G = (N, A)$ with node set N and arc set A, and a homogeneous vehicle fleet is specified. In G, some arcs require service with known service time, and all arcs can be traversed with known deadhead time. The CARP breaks the required arcs in G into partitions so that each partition contains about the same amount of time (service time plus deadhead time), and the arcs in each partition are sequenced so that the total additional deadhead time needed to form a continuous travel path through the arcs is minimized. For the CARP, any vehicle can be assigned to service any partition. For a review of the CARP and its solution techniques, see Assad and Golden [1], Eiselt, Gendreau, and Laporte [3, 4], Bodin et al. [2], Golden and Wong [5], Pearn [10], and Laporte [6].

The focus of this chapter is to develop effective approaches for solving the CARP-VSD. Since reasonable approaches exist for solving the CARP, and these approaches can be used for solving the travel-path-generation aspects in the CARP-VSD, this chapter emphasizes practical approaches for solving the partitioning aspect of the CARP-VSD. These partitioning approaches are integrated with the travel-path-generation procedures to derive an overall algorithm for solving the CARP-VSD. This algorithm is called the *Vehicle Decomposition Algorithm* (VDA). A fundamental assumption of the VDA is that if a vehicle from a vehicle class of specified capacity can service (or deadhead) a specific street segment, then all vehicles in vehicle classes of smaller capacity can also service (or deadhead) that street segment. The VDA has been successfully used by the Philadelphia Sanitation Department for solving its residential sanitation-vehicle scheduling problem with vehicle-site dependencies.

In section 11.2, we define the networks, assumptions, and goals of the CARP-VSD and the VDA. In section 11.3, we give a step-by-step description of the VDA and illustrate parts of the VDA with an example. In section 11.4, we present the results of using the VDA in Philadelphia. In section 11.5, we discuss future research directions.

11.2 Networks, Assumptions, and Goals of the CARP-VSD

In this section, the travel network, service network, assumptions, and goals of the CARP-VSD are defined. These elements of the CARP-VSD differ from the ordinary CARP due to the vehicle-site dependencies existing on the arcs, and these vehicle-site dependencies lead to a more complex network design. This more complex network design is accounted for in the design of the VDA algorithm.

11.2.1 Travel Network

The travel network $G = (N, A)$ is a directed network that represents the underlying street network over which the CARP-VSD is to be solved. Each node in G represents an intersection in the street network. Each arc in G represents one side of a street segment in the street network and is directed to indicate the travel direction on the street segment. The two directed arcs for each street segment in the underlying street network are referred to as counterpart arcs (see Levy [7]) and these counterpart arcs follow the direction of travel by side of the street. The travel network G is composed entirely of directed counterpart arcs; that is, every street segment in the underlying street network has two arcs defined in the travel network. If the street segment allows two-way traffic, then the two counterpart arcs in G are directed in opposite directions. If the street segment allows only one-way traffic, then the two counterpart arcs in G are directed in the same direction and in the direction of traffic. If the street segment allows only one-way traffic and is directed from f to g, then the two counterpart arcs, $a(f, g)$ and $a'(f, g)$, for the street segment can have different attributes.

The following attributes are defined for each arc $a(f, g)$ in the travel network:

$D(f, g)$: Deadhead travel time on arc $a(f, g)$.

$W(f, g)$: $= 1$ if $a(f, g)$ is a counterpart arc of a one-way street segment from f to g. In this case, $a(f, g)$ is replicated twice in the travel network, once for each side of the street segment. For notational purposes, these two counterpart arcs in G are represented as $a(f, g)$ and $a'(f, g)$.
$= 2$ if $a(f, g)$ is a counterpart arc of a street segment that allows two-way traffic. In this case, $a(f, g)$ and $a(g, f)$ exist in the travel network.

$M(f, g)$: $= 0$ if the street segment associated with arc $a(f, g)$ has to be serviced one side at a time.
$= 1$ if the street segment associated with arc $a(f, g)$ can be meandered or zigzagged. When a street segment can be meandered, both sides of the street segment are serviced with a single traversal of the street. It is assumed that if one vehicle class services the street segment associated with $a(f, g)$ as a meander, then all vehicle classes service the street segment as a meander. A rare exception to this assumption in residential sanitation collection is the following. One vehicle class can represent an automated side-loading vehicle that cannot meander and service a street segment while another vehicle class represents a rear-loading or front-loading vehicle in which meandering is possible. This case is not considered in this chapter.

$L(f, g)$: Length of arc $a(f, g)$. It is assumed that $L(f, g) = L(g, f)$.

$SC(f, g)$: Largest vehicle class that can service arc $a(f, g)$. If $SC(f, g) = s$, then vehicle classes $1, \dots, s$ can service arc $a(f, g)$. It is assumed that $SC(f, g) = SC(g, f)$.

$TC(f, g)$: Largest vehicle class that can travel along arc $a(f, g)$. If $TC(f, g) = t$, then vehicle classes $1, \dots, t$ can travel arc $a(f, g)$. It is assumed that $TC(f, g) = t \geq SC(f, g) = s$ and that $TC(f, g) = TC(g, f)$.

$S(f, g)$: Service time on arc $a(f, g)$. If $a(f, g)$ is to be serviced as a meander (i.e., $M(f, g) = 1$), then $S(f, g)$ is half the service time for the street segment associated with $a(f, g)$.

$Q(a(f, g))$: Quantity (volume or weight) of refuse to be picked up on arc $a(f, g)$. $Q(a(f, g)) = 0$ if arc $a(f, g)$ is not serviced. The notation $Q(a(f, g))$ rather than $Q(f, g)$ is used to define the volume on arc $a(f, g)$ to facilitate the definitions of the arc sets MA and SE in section 11.2.2.

11.2.2 Service Network

In the service network G_s, the arc set A from the travel network G is broken into four mutually disjoint sets representing the street segments that require different types of service and the street segments that can be traversed but do not require service. These four disjoint sets are as follows.

DA: Set of *deadhead arcs*. $DA = \{a(f, g)|Q(a(f, g)) = 0\}$. No service is required on $a(f, g)$ and $a(f, g)$ is used only for deadheading.

RA: Set of arcs that require service, cannot be meandered, and two-way traffic is allowed on the corresponding street segment. Thus, $RA = \{a(f, g)|Q(a(f, g)) > 0, M(f, g) = 0\}$. If $a(f, g) \in RA$, service of $a(f, g)$ must occur by traveling along $a(f, g)$ and not its counterpart arc.

MA: Set of arcs that require service, the service is a meander, and the street segment associated with each arc is a one-way street. $MA = \{a(f, g)|Q(a(f, g)) + Q(a'(f, g)) > 0, M(f, g) = 1, W(f, g) = 1\}$. Although $a(f, g)$ is replicated twice in A, once for each side of the street segment, arc $a(f, g)$ appears once in MA as a directed arc.

SE: Set of edges representing the street segments that allow two-way travel, require service, and the service must be carried out as a meander. $SE = \{e(f, g)|Q(a(f, g)) + Q(a(g, f)) > 0, M(f, g) = 1, W(f, g) = 2\}$. Both $a(f, g)$ and $a(g, f)$ are in A, but the edge, $e(f, g)$, appears once in SE representing both arcs in the travel network. The travel-path-generation algorithm in the VDA determines if edge $e(f, g)$ is serviced from node f to node g or from node g to node f.

The service network G_s is represented as the node set N, the arc sets DA, RA, and MA, and the edge set SE. The service network G_s can be denoted as $G_s = (N, DA \cup RA \cup MA \cup SE)$.

11.2.3 Vehicle Classes

The existence of vehicle classes in the vehicle fleet separates the CARP-VSD from the traditional CARP. Although the vehicle classes represent different types of vehicle, the vehicles are assumed to be homogeneous within each vehicle class. Attributes such as vehicle capacity and length of workday typically associated with a vehicle in the CARP can have different values for each vehicle class in the CARP-VSD.

The following attributes are defined for vehicle class k:

Q_k: Number of vehicles available from vehicle class k. For example, if a fleet has five 1-ton vehicles, three 5-ton vehicles, and four 10-ton vehicles, then $Q_1 = 5$, $Q_2 = 3$, and $Q_3 = 4$. The vehicle preference list (described below) further refines the notion of the number of vehicles in a vehicle class.

M_k: Capacity of the vehicles from vehicle class k. Vehicle capacity is the maximum volume or weight that a vehicle can hold. A vehicle route in residential sanitation collection can involve multiple trips to the disposal facility. A *trip* to the disposal facility is required whenever the vehicle capacity is reached, although going to the disposal facility before capacity is reached may reduce total deadhead time or distance on the route. Furthermore, it is assumed that a trip to the disposal facility is required at the end of the day even if the vehicle capacity has not been reached.

DT_k: Disposal time, i.e., the time it takes to empty a vehicle at the disposal facility. Disposal time is a function of vehicle class.

OT_k: Office time associated with a vehicle from vehicle class k. Office time is the time a crew is allocated at the depot at the beginning or the end of the day. At the office, the crew carries out mandated functions, such as filling the vehicle with gas, washing the vehicle, and showering.

11.2.4 Travel Network and Service Network for a Vehicle Class

In the VDA, each vehicle class has its own travel network and service network. These networks are not actually created in the VDA but are implied by indicator variables in the travel network G and service network G_s for the CARP-VSD.

For each arc in the travel network G, $TC(f, g)$ specifies the largest vehicle class that can travel on arc $a(f, g)$. Each vehicle class can have a different set of arcs that can be traversed by a vehicle from the vehicle class. The travel network for vehicle class i, $G^i = (N^i, A^i)$, is the network of arcs that can be traversed by the vehicles in vehicle class i. There is no guarantee that G^i is a connected network.

For each arc in the travel network G, $SC(f, g)$ specifies the largest vehicle class that can service arc $a(f, g)$. As mentioned, it is possible that a street segment can be traversed by a vehicle class but cannot be serviced by that vehicle class. Because of this vehicle-site dependency, the set of arcs that a vehicle can service is also a function of the vehicle class. The service network for vehicle class i is $G_s^i = (N^i, A^i = DA^i \cup RA^i \cup SA^i \cup SE^i)$.

The service network for vehicle class i, G_s^i, is created from the travel network for vehicle class i, G^i, in exactly the same way as the service network, G_s, is created from the travel network, G, as described in section 11.2.2. Arcs in A^i that can be serviced by vehicle class i are placed in RA^i, SA^i, or SE^i. Arcs in A^i that cannot be serviced by vehicle class i are placed in DA^i.

11.2.5 Vehicle Preference List

The *vehicle preference list* specifies, in decreasing order of preference, a vehicle class and the maximum number of vehicles that are available at that preference level. Certain vehicle classes may be more desirable due to operational costs, contractual obligations, or other reasons. A vehicle class may appear more than once in the vehicle preference list. An example of a vehicle preference list is as follows:

Vehicle class	Number available
3	3
1	11
2	4
1	4

In this vehicle preference list, the user most prefers the vehicles from vehicle class 3 because the user believes they are the most efficient, require only one trip to the disposal facility, hold the most volume, and are already owned by the organization servicing the area. The vehicle class 1 vehicles are split in the vehicle preference list because the organization owns 11 of these vehicles (no capital cost for these vehicles), but the organization can buy up to 4 additional vehicles (but these vehicles require a capital expenditure). The organization is also authorized to buy up to four vehicle class 2 vehicles. The four vehicle class 2 vehicles appear above the four vehicle class 1 vehicles because the user believes the vehicle class 2 vehicles are more efficient than the vehicle class 1 vehicles.

The VDA tries to use vehicles from the preferred vehicle classes, beginning at the top of the vehicle preference list. Less-desirable vehicle classes are used only as required.

11.2.6 Other Assumptions

As defined above, $G_s = (N, DA \cup RA \cup MA \cup SE)$ represents the service network. The VDA assumes that every arc in $RA \cup MA$ and every edge in SE can be serviced by the smallest vehicle class in the vehicle preference list.

The travel network G and the service network G_s are assumed to be connected networks, and the service network G_s^1 is assumed to be a strongly connected network. In this way, it is possible to service all arcs requiring service if there are enough vehicles from vehicle class 1 in the vehicle preference list. If G_s^1 is not strongly connected, then any service arcs not strongly connected to the depot and disposal facility will not be serviced by routes in the final solution.

A single depot exists where all vehicles, regardless of vehicle class, start and end their route. A single disposal facility exists where all vehicles are emptied. All vehicles, regardless of vehicle class, must return to the disposal facility at the end of the day to empty the vehicle before returning to the depot.

A *target route time*, TRT_k, is specified for each vehicle class k. This target route time can vary between vehicle classes and within a vehicle class. The target route time is the target length of time for a partition being serviced by a vehicle from a specified vehicle class. To model a difference in target route time within a vehicle class, the vehicle class is split into two or more vehicle classes with different target route times in the vehicle preference list, but the vehicle-site dependencies on the arcs are identical for each of these vehicle classes.

A soft constraint on the formation of the partitions is that the length for each partition p in vehicle class k lies in the interval $[Lower_p, Upper_p]$, where $Lower_p = TRT_k - A_k$, $Upper_p = TRT_k + B_k$, and A_k and B_k are generally around 15 minutes. However, we had one organization that set A_k and B_k equal to 3 minutes.

11.2.7 Goals and Constraints of the CARP-VSD

A desirable solution to the CARP-VSD, as well as most other CARP-type problems, has the following characteristics:

- Each street requiring service in the service network G_s is assigned to a partition as long as the service network G_s is connected.
- The route length for all routes in vehicle class k lies in the interval $[TRT_k - A_k, TRT_k + B_k]$. Such a solution is called a *balanced solution*. In the CARP-VSD, workload estimation in the VDA is more complicated because the VDA must decide how many vehicles of each vehicle class are needed to provide the service. Partitioning in the CARP-VSD using the VDA becomes more complex because a decision has to be made as to the vehicle class to assign to each partition that is grown (i.e. attach a vehicle class to each seed point created). These decisions on the fleet mix composition are

initially made before any partition is grown. However, the VDA is set up so that this estimate can be updated in subsequent iterations of the VDA. The vehicle preference list gives us an effective way for determining the fleet mix composition.

- The travel path associated with each partition minimizes nonproductive time (or deadhead time). In most arc routing problems, the deadhead time is a small percentage of the total time needed to service all the street segments that require service in the partition. As noted, in our implementation of the VDA for Philadelphia, we used the existing travel path generation procedures found in the RouteSmart software system[1] to determine the minimum deadhead travel time path for each partition, and we did not attempt to develop any new travel path generation procedure.

- The partitions interlace as little as possible. If the partitions do not interlace, the area of responsibility for each crew is better identified. In this way, it is easier to attribute any error in performing a service to the correct crew.

11.3 Vehicle Decomposition Algorithm (VDA)

The VDA solves the CARP-VSD by decomposing it into several smaller single vehicle class CARPs. Each of these smaller CARPs is solved by the partitioning procedures contained in RouteSmart. The solution to the smaller problems are then integrated together to form a final CARP-VSD solution. Once the partitions are formed, traditional travel-path-generation techniques contained in RouteSmart are used to find an approximately minimum deadhead time travel path for the arcs and edges in each partition.

The VDA consists of the following five steps:

- Step A. Create and verify vehicle class networks.

- Step B. Estimate total work and determine initial fleet mix.

- Step C. Partition the service network.

- Step D. Determine travel path and balance the partitions.

- Step E. Revise estimate of total work and adjust fleet mix.

The VDA initially carries out Steps A and B and then iterates between Steps C–E until it terminates. At that time, the routes and travel paths can be printed and plotted. Input to the VDA are the various quantities described in section 11.2, such as the travel and service networks, the vehicle preference list, and various parameters such as the number of partitions requested and the target route time for each partition.

11.3.1 Step A. Create and Verify Vehicle Class Networks

The vehicle class travel networks, $G^i = (N^i, A^i)$, are created for all vehicle classes from the travel network $G = (N, A)$, and each G^i is checked for connectivity. If G^i is not a strongly connected network, then the only arcs that can be serviced by a vehicle in vehicle

[1]Information on the RouteSmart software system can be obtained from the website www.routesmart.com.

class i are the arcs that are strongly connected to the depot and disposal facility. All the arcs in G^i not strongly connected to the depot and disposal facility are removed from G^i. The overall service network G_s and the vehicle class service networks G_s^i are then created.

11.3.2 Step B. Estimate Total Work and Determine Initial Fleet Mix

In Step B, an initial fleet mix estimate is determined. This fleet mix estimate is used as the fleet mix on the first iteration of the VDA.

Let $V = [V(i, j)]$ be a lower triangular matrix. $V(i, j)$ is the number of vehicles of vehicle class j needed to service all the arcs and edges in the service network whose largest feasible vehicle class is vehicle class i, $i \geq j$. Thus, $V(2, 1)$ is the number of vehicles of vehicle class 1 needed to service all the arcs and edges in the service network whose largest feasible vehicle class is vehicle class 2. The procedure that we developed for computing $V(i, j)$ is given in section 11.3.2.7, and $V(i, j)$ need not be integer. In the next sections, we describe the VDA procedure. A complete example is also given.

11.3.2.1 Pass 1 Through Step B

On the first pass ($k = 1$) through this procedure, $V(1, 1)$ is examined by searching the vehicle preference list and finding the first instance of the smallest vehicle class. Let $W1$ be the number of vehicles from the smallest vehicle class specified in that entry of the vehicle preference list. Let $U = \min(V(1, 1), W1)$ be the number of vehicles from the smallest vehicle class to assign to service the workload associated with $V(1, 1)$. If U is not an integer, then U is rounded up to its next largest integer.

Knowing U, $V(1, 1)$ is set equal to $V(1, 1) - U$ and $W1$ is set equal to $W1 - U$ in the appropriate entry of the vehicle preference list. Then, the following three cases can occur.

1. If $V(1, 1) > 0$, we continue to search down the vehicle preference list for the next occurrence of the smallest vehicle class since we have not, as yet, accounted for enough vehicles from vehicle class 1 to service the demand specified in $V(1, 1)$. If another occurrence is found, the procedure described here is repeated. If no other occurrence of a vehicle from the smallest vehicle class is found in the vehicle preference list, the VDA terminates since the problem is infeasible—there are not enough vehicles from vehicle class 1 available to service the demand.

2. If $V(1, 1) = 0$, all of the demand in the first row of V can be serviced by vehicles from vehicle class 1. The first pass through Step B is complete and the second pass through Step B can begin by setting $k = 2$ and executing the procedure in 11.3.2.2.

3. If $V(1, 1) < 0$, we have excess capacity for the first row of demand in V. $V(1, 1) < 0$ occurs if U is set to a noninteger $V(1, 1)$ above. Then U is rounded up to the next largest integer. This excess capacity is assumed to satisfy some of the demand in the second row of the V matrix. To accomplish this, the second row of V is updated as follows:

$$V(2, 1) = V(2, 1) - |V(1, 1)| \quad \text{and}$$
$$V(2, 2) = V(2, 2) - [|V(1, 1)| * V(2, 2)/V(2, 1)].$$

As an example, if we initially have $W1 = 3$, $V(1,1) = 1.1$, $V(2,1) = 3.2$, and $V(2,2) = 2.7$, then the following occurs on this first pass through Step B.

1. $U = 2$, $W1 = 3 - 2$, $V(1,1) = 1.1 - 2 = -0.9$.
2. Since $V(1,1) < 0$, $V(2,1) = 3.2 - 0.9 = 2.3$ and $V(2,2) = 2.7 - (0.9 * 2.7/3.2) = 2.7 - 0.759 = 1.941$

If $V(2,1) < 0$ as a result of the above accounting for excess capacity of a vehicle from vehicle class 1, then all the demand in row 2 of V is accounted for, row 3 of V would be updated in the same manner as described for row 2, and so on.

We have now finished with this first pass through Step B and can continue with the second pass through Step B by setting $k = 2$ and executing the following procedure.

11.3.2.2 Pass *k* Through Step B ($k > 1$)

On the kth pass through Step B, the kth row of V is examined and the vehicle preference list is scanned for the first occurrence of a vehicle class no greater than k. Let the following:

j:	The number of the vehicle class of the first occurrence of a vehicle class no greater than k in the vehicle preference list;
W_j:	The number of vehicles from vehicle class j that can be assigned to that workload W_j as specified in the vehicle preference list, and W_j is integer;
U:	The number of vehicles from vehicle class j to assign to service the workload associated with $V(k, j)$, i.e., $U = \min(V(k, j), W_j)$. If U is not integer, then U is rounded up to its next largest integer.

Knowing U, W_j is set equal to $W_j - U$ in the appropriate entry of the vehicle preference list and the kth row of V is updated as follows:

$$V(k,i) = V(k,i) - [U * V(k,i)/V(k,j)], \qquad i = 1, \ldots, k.$$

Then, the following three cases can occur:

1. If $V(k, j) > 0$, we continue to search down the vehicle preference list for the next occurrence of an available vehicle class no larger than k. If another occurrence is found, the procedure described in this section is repeated. If no occurrence of an available vehicle class no larger than k is found in the vehicle preference list, the VDA terminates since the problem is infeasible—there are not enough vehicles from vehicle class k or smaller available to service the demand.
2. If $V(k, j) = 0$, we have completed the kth pass through Step B and can continue with the next pass through Step B by setting $k = k + 1$ and repeating the procedure.
3. If $V(k, j) < 0$, we have excess capacity for the kth row of demand in V, and this excess capacity can be used to satisfy the demands in the remaining rows of V. To account for this excess capacity, the $k + 1$ row of V is updated by setting

$$V(k+1, i) = V(k+1, i) - [|V(k, j)| * V(k+1, i)/V(k+1, j)], \qquad i = 1, \ldots, k+1.$$

After updating row $k + 1$ of V, we have completed the kth pass through Step B and can continue with the next pass through Step B by setting $k = k + 1$.

11.3.2.3 Example

Assume there are three vehicle classes and the V matrix is

$$V = \begin{pmatrix} 4.70 & - & - \\ 4.10 & 2.20 & - \\ 9.10 & 5.00 & 3.10 \end{pmatrix}.$$

Further assume that the vehicle preference list is the following:

Vehicle Class	Number Available
3	2
1	7
2	4
1	4

11.3.2.4 First Pass Through Step B

Since $V(1, 1) = 4.7$ and $W1 = 7$, $U = \min(4.7, 7) = 4.7$, U is rounded to 5. Thus, five vehicles from vehicle class 1 are assigned to the fleet mix. Then, $W1 = 7 - 5 = 2$ and $V(1, 1) = 4.7 - 5.0 = -0.3$.

Since $V(1, 1) < 0$, there is excess capacity for the demand of row 1. Thus, we set

$$V(2, 1) = 4.1 - 0.3 = 3.8,$$
$$V(2, 2) = 2.2 - 0.3(2.2/4.1) = 2.04.$$

11.3.2.5 Second Pass Through Step B

The vehicle preference list entering pass 2 is

Vehicle Class	Number Available
3	2
1	2
2	4
1	4

and the updated V matrix is

$$V = \begin{pmatrix} - & - & - \\ 3.80 & 2.04 & - \\ 9.10 & 5.00 & 3.10 \end{pmatrix}.$$

Since $V(2, 1) = 3.8$ and $V(2, 2) = 2.04$, we scan the vehicle preference list and find vehicle class 1 as the first available feasible vehicle class. We set $W_2 = 2$ and $U =$

$\min(3.8, 2) = 2$, so we assign two more vehicles from vehicle class 1 to the fleet mix. Since there is still some demand in row 2 left over, the residual demands are updated as follows:

$$V(2,1) = 3.8 - 2 * (3.8/3.8) = 1.8,$$
$$V(2,2) = 2.04 - 2 * (2.04/3.8) = 0.97.$$

The updated vehicle preference list is

Vehicle Class	Number Available
3	2
1	0
2	4
1	4

and the updated V matrix is

$$V = \begin{pmatrix} - & - & - \\ 1.80 & 0.97 & - \\ 9.10 & 5.00 & 3.10 \end{pmatrix}.$$

Since $V(2,1) = 1.8$ and $V(2,2) = 0.97$, we scan the vehicle preference list and find vehicle class 2 as the first available feasible vehicle class. We set $W2 = 4$, $U = \min(0.97, 4) = 0.97$ and round U to 1. Thus, we assign one vehicle from vehicle class 2 to the fleet mix. $W2 = 4 - 1 = 3$ and $V(2,2) = 0.97 - 1 = -0.03$. Since there is excess capacity for the demand of row 2, we set

$$V(3,1) = 9.1 - 0.03 * (9.1/5.0) = 9.05,$$
$$V(3,2) = 5.0 - 0.03 * (5.0/5.0) = 4.97,$$
$$V(3,3) = 3.1 - 0.03 * (3.1/5.0) = 3.08.$$

All the demand for row 2 is satisfied.

11.3.2.6 Third Pass Through Step B

The vehicle preference list entering pass 3 is

Vehicle Class	Number Available
3	2
1	0
2	3
1	4

and the updated V matrix is

$$V = \begin{pmatrix} - & - & - \\ - & - & - \\ 9.05 & 4.97 & 3.08 \end{pmatrix}.$$

Since $V(3,1) = 9.05$, $V(3,2) = 4.97$, and $V(3,3) = 3.08$, we scan the vehicle preference list and find vehicle class 3 as the first available feasible vehicle class. We set

$W3 = 2$ and $U = \min(3.08, 2) = 2$, so we assign two vehicles from vehicle class 3 to the fleet mix. $W3 = 2 - 2 = 0$ and the third row of the V matrix is updated as follows:

$$V(3,1) = 9.05 - 2 * (9.05/3.08) = 3.17,$$
$$V(3,2) = 4.97 - 2 * (4.97/3.08) = 1.74,$$
$$V(3,3) = 3.08 - 2 * (3.08/3.08) = 1.08.$$

The updated vehicle preference list is

Vehicle Class	Number Available
3	0
1	0
2	4
1	4

and the updated V matrix is

$$V = \begin{pmatrix} - & - & - \\ - & - & - \\ 3.17 & 1.74 & 1.08 \end{pmatrix}.$$

Since $V(3,1) = 3.17$, $V(3,2) = 1.74$, and $V(3,3) = 1.08$, we scan the vehicle preference list and find vehicle class 2 as the first available feasible vehicle class. We set $W_2 = 3$ and $U = \min(1.74, 3) = 1.74$. U is then rounded to 2. Thus, two additional vehicles from vehicle class 2 are assigned to the fleet mix. Since $W_2 = 3 - 2 = 1$ and $V(3,2) = 1.74 - 2 = -0.26$, there is excess capacity for the demand of row 3. As all the demand for row 3 is satisfied, the initial fleet mix estimation is complete. The 0.26 excess vehicle capacity is part of the initial fleet mix.

Thus, the initial fleet mix estimate for the example is the following:

$$\text{Vehicle Class 1:} \quad 5 + 2 + 0 = 7,$$
$$\text{Vehicle Class 2:} \quad 0 + 1 + 2 = 3,$$
$$\text{Vehicle Class 3:} \quad 0 + 0 + 2 = 2.$$

11.3.2.7 Determination of $V(i, j)$ in the V Matrix

As stated previously, $V(i, j)$ is the number of vehicles from vehicle class j needed to service all the arcs and edges in the service network whose largest feasible vehicle class is vehicle class i, $i \geq j$. We now describe our workload estimation procedure for computing $V(i, j)$.

The components that go into computing the workload estimation for $V(i, j)$ are the following:

$K1$: Total volume on all arcs whose largest vehicle class that can service the arc is i; i.e., $K1 = \sum Q[a(f,g)]$ for all $\{a(f,g) | SC(f,g) = i\}$.

$K2$: Vehicle capacity for the vehicle class being analyzed; i.e., $K2 = Q_j$.

$K3$: Service time on all arcs whose largest vehicle class that can service the arc is i; i.e., $K3 = \sum S(f,g)$ for all $\{a(f,g) | SC(f,g) = i\}$.

$K4$: Estimate of the deadhead time between all service arcs whose largest vehicle class that can service the arc is i.

$K5$: Target route time for a partition from vehicle class j. $K5$ is specified by the user; i.e., $K5 = TRT_j$.

$K6a$: Office time for a partition from vehicle class j. $K6a$ is specified by the user; i.e., $K6a = OF_j$.

$K6b$: Disposal time for a vehicle from vehicle class j. $K6b$ is specified by the user; i.e., $K6b = DT_j$.

$K6c$: Time for a vehicle from vehicle class j to go from the disposal facility to the depot. $K6c$ is computed over the travel network and is known exactly.

$K6d$: Time for a vehicle from vehicle class j to go from a partition to the disposal facility. $K6d$ is computed over the travel network and is estimated as the average time from each arc $a(f, g)$ to the disposal facility where $SC(f, g) = i$.

$K6$: Total fixed overhead time. $K6$ is computed as follows: $K6 = K6a + K6b + K6c + K6d$. Since $K6d$ is an estimate, $K6$ is an estimate as well.

$K7$: Estimate of the time a vehicle from vehicle class j takes to make an additional trip to the disposal facility. $K7$ is the average round trip travel time from a partition to the disposal facility $(2 * K6d)$ plus the time at the disposal facility $(K6b)$. Thus, $K7 = 2 * K6d + K6b$. $K7$ is an estimate since $K6d$ is an estimate and a function of vehicle class.

Let X be the (integer) number of trips to the disposal facility required for each partition. Let $V(i, j) = \min_X(\max(PV(X), PT(X)))$, where

$PV(X)$: Minimum number of partitions needed to handle the total volume on the streets, which can be computed as $PV(X) = K1/(K2 * X)$.

$PT(X)$: Minimum number of partitions needed to handle all required time in the workload. $PT(X)$ is computed as

$$PT(X) = (K3 + K4)/(K5 - K6 - K7 * (X - 1)).$$

Since $K6$ is an estimate, $PT(X)$ is as estimate.

The problem is to compute the best value of X to use in the computation of $V(i, j)$. Initially, X is not known. However, it can be observed that as X increases, $PV(X)$ decreases and $PT(X)$ increases. A representation of $PV(X)$ and $PT(X)$ as a continuous function of X is shown in Figure 11.1.

To determine the value of X that will minimize $V(i, j)$, we set $PV(X) = PT(X)$ and solve for X, which may not be integer. We then compute the adjacent integer values of X that minimize $V(i, j)$.

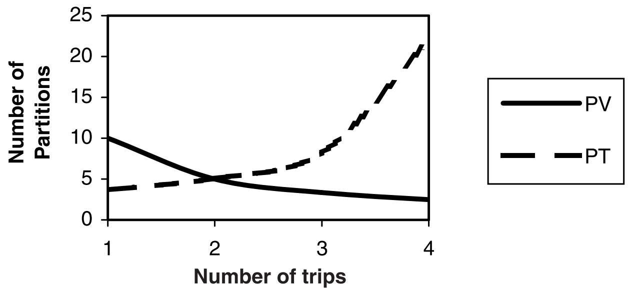

Figure 11.1. *Graph of $PV(X)$ and $PT(X)$.*

11.3.2.8 Example

This example illustrates the workload estimation procedure. Assume that the following are known:

$K1$: Total volume = 100 tons,

$K2$: Vehicle capacity per trip = 10 tons,

$K3$: Service time on arcs = 1200 minutes,

$K4$: Deadhead time between service arcs = 120 minutes,

$K5$: Route time = 480 minutes,

$K6a$: Office time = 30 minutes,

$K6b$: Disposal time = 60 minutes,

$K6c$: Time from the disposal facility to the depot = 10 minutes,

$K6d$: Time from a partition to disposal facility = 20 minutes,

$K6$: Fixed overhead = $K6a + K6b + K6c + K6d = 120$ minutes,

$K7$: Variable overhead = $2 * K6d + K6b = 100$ minutes,

Then $PV(X)$ and $PT(X)$ can be computed as follows:

$$PV(X) = K1/(K2 * X) = 100/(10 * X) = 10/X,$$
$$PT(X) = (K3 + K4)/(K5 - K6 - K7(X - 1)) = 1320/(460 - 100X).$$

Equating $PV(X)$ and $PT(X)$ gives X^*, the estimate of X:

$$X^* = (K1 * (K5 - K6 + K7))/((K1 * K7) + (K2 * (K3 + K4))) = 1.98.$$

Since X is 1.98, we compute $V(i, j) = \min(PV(1), PT(2))$, where $PV(1) = 10$ and $PT(2) = 5.08$. Therefore, $V(i, j) = 5.08$ and 5.08 is the estimate of the number of partitions needed to satisfy the workload.

11.3.3 Step C. Partition the Service Network

The partitioning algorithm in the VDA is significantly more complex than the partitioning algorithm used for solving the CARP with a homogeneous fleet. In the algorithms for the CARP, since all of the vehicles are identical, the partitioning step only has to assign all of the arcs in the service network to a partition. On the other hand, in the VDA, a decision has to be made about the vehicle class to service each partition, and all the required arcs and edges must be assigned to a partition.

In 11.2.6, the target route time for an entry in the vehicle preference list was defined to be the desired amount of work measured in time to a place in a partition and was allowed to vary between vehicle classes. As noted earlier, one of the constraints in the CARP-VSD is to form the partitions so that the total workload in each partition assigned to vehicle class k lies between $[TRT_k - A_k, TRT_k + B_k]$, although these bounds are assumed to be soft and, hence, can be violated.

The following five substeps of Step C form the partitions in the VDA. These five steps are iterated for each vehicle class in the vehicle fleet mix, beginning with the smallest available vehicle class in the fleet mix and ending with the largest available vehicle class in the fleet mix.

Step C1. A temporary service network (TSN) is created. The service arcs and edges in the TSN are those arcs and edges in $SA \cup SE$ that must be partitioned on this iteration and have not, as yet, been assigned to a partition.

Step C2. The TSN is partitioned as a CARP without vehicle-site dependencies using established CARP solution techniques. This partitioning is done using the available vehicles in terms of capacity that remain in the vehicle fleet mix determined in Step B and that have not as yet been assigned to a partition. This partitioning involves solving a nonhomogeneous CARP with no site dependencies. Sniezek [11] is conducting research into how to best accomplish this partitioning. For the Philadelphia study described in this chapter, this partitioning was accomplished using the existing CARP procedure for a nonhomogeneous fleet with no site dependencies that is available in the RouteSmart software system.

In this step, it is possible to have more vehicles from a vehicle class available in the fleet mix determined in Step B than the number of partitions that we want to create at this point in Step C. For example, the TSN for the smallest vehicle class may have only a small amount of workload that must be serviced by a vehicle from the smallest vehicle class. If the small vehicles were desirable, however, in the vehicle preference list, we may have several small vehicles in the fleet mix. However, to grow partitions for all or most of these small vehicles at this point in the algorithm would probably lead to inferior solutions containing imbalances in the workload and severe interlacing. The rule that we used for determining the number of partitions to grow at this time is equal to $\min(R1, R2)$, where

$R1$: $1.5*$ (minimum number of partitions needed to service the workload in the TSN). $R1$ is rounded up to the nearest integer.

$R2$: The number of vehicles of this vehicle class presently available in the fleet mix.

On the final iteration, the bounds on target route time can be violated to ensure that all arcs and edges in the service network G_s are assigned to a partition. The target route time has soft lower and upper bounds to ensure that all arcs and edges in G_s are assigned to partitions in the final solution.

Step C3. The service network G_s and all vehicle class service networks G_s^1, G_s^2, ..., G_s^i are implicitly updated based on the partitioning in Step C2. No service arcs or service edges in TSN assigned to a partition in Step C2 are service arcs or service edges in G_s^k, $k > i$ since Step C2 assigned arcs $a(f, g)$ only to partitions where $SC(f, g) \leq i$. Thus, the service networks G_s^k, $k > i$, need not be updated at this point.

Step C4. A second temporary service network TSN2 is now created. The service arcs and edges in TSN2 are the service arcs and edges in TSN that have already been assigned to partitions in Step C2 plus all remaining arcs and edges in $SA \cup SE$ that have, as yet, not been partitioned. In Step C4, the partitions grown in Step C2 are further expanded by assigning the arcs and edges in TSN2 until each partition grows to a specified percentage of the target route time of the vehicle class associated with each partition.

Step C5. If all of the arcs and edges requiring service have been assigned to a partition, the partitioning algorithm terminates. Otherwise, the service network G_s and all vehicle class service networks G_s^1, G_s^2, ..., G_s^k are updated (as in Step C3) based on the partitioning in Step C4 and the algorithm returns to Step C1.

11.3.3.1 Example

The example in Figure 11.2 illustrates the importance of partitioning the TSN first and the TSN2 later when dealing with a nonhomogeneous fleet of vehicles in the CARP-VSD. The graph in the figure has six nodes (a, b, c, d, e, f) and nine edges $(A, B, C, D, E, F, G, H, I)$. The volume on each edge is one unit, edges B, C, and D can be serviced by either a small or large vehicle, and the other edges can be serviced only by a small vehicle. Three cases are considered.

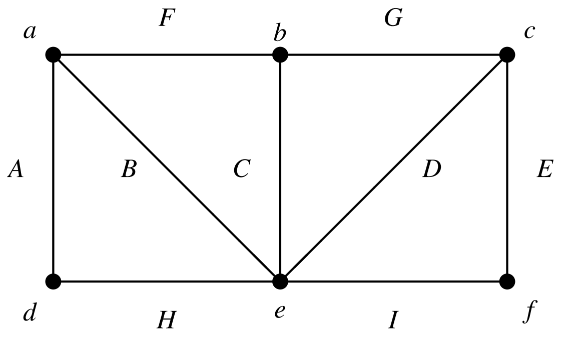

Figure 11.2.

11.3.3.2 Case 1: Homogeneous CARP

Assume that the fleet has three small vehicles and the site dependencies on the edges are disregarded. The partitions for these three vehicles are seeded at a, c, and e. The following partitions are found:

$$\text{Partition 1 (seed } a) = \{A, B, F\},$$
$$\text{Partition 2 (seed } c) = \{D, E, G\},$$
$$\text{Partition 3 (seed } e) = \{C, H, I\}.$$

11.3.3.3 Case 2: CARP-VSD (Using Only TSN2)

In this case, the fleet consists of three vehicles—two small vehicles from vehicle class 1 and one large vehicle from vehicle class 2. As noted above, B, C, and D are the only edges that can be serviced by vehicles from vehicle class 2. In this case, Steps C2 and C3 described in the partitioning algorithm are not used.

We first partition the two small vehicles over the entire service network (TSN2) instead of the TSN. If the partitions are seeded at a and c, then the following two partitions are created:

$$\text{Partition 1 (seed } a) = \{A, B, F\},$$
$$\text{Partition 2 (seed } c) = \{D, E, G\}.$$

These partitions are at capacity, and yet edges H and I, which must be serviced by vehicle class 1, have not been assigned to a partition.

On the next iteration through Step C, the TSN2 consists of edge C. Edges H and I are not in TSN2 since these edges cannot be serviced by the large vehicle. Thus, Partition 3 is the following:

$$\text{Partition 3 (seed } e) = \{C\}.$$

Thus, in the final solution for this case, H and I are not assigned to a partition and Partition 1 and Partition 2 are at capacity.

11.3.3.4 Case 3: CARP-VSD (Using TSN then TSN2)

In this case, the fleet composition is the same as in Case 2. However, we are going to do the partitioning by first forming the TSN and then forming the TSN2.

The initial TSN for the vehicles in the smallest vehicle class consists of edges A, E, F, G, H, and I. Edges B, C, and D (the edges that can be serviced by vehicles from vehicle class 2) are not in the TSN for the small vehicle class. Using a and c as the seed points for the partitioning, the partitions are as follows:

$$\text{Partition 1 (seed } a) = \{A, F, H\},$$
$$\text{Partition 2 (seed } c) = \{E, G, I\}.$$

The two partitions are at capacity and all edges that had to be serviced by vehicle class 1 are assigned to a vehicle from vehicle class 1.

The TSN2 for the smallest vehicle class consists of the entire network. Since the two partitions for the smallest vehicle class are at saturation, no new edges are assigned to these partitions.

The algorithm begins again for the vehicle in the larger vehicle class. The TSN consists of edges B, C, and D. The resulting partition is the following:

$$\text{Partition 3 (seed } e) = \{B, C, D\}.$$

11.3.3.5 Analysis

These three cases illustrate the following. The homogeneous solution (Case 1) gives the best set of partitions in terms of interlacing and covering all of the edges. When site dependencies are considered, Case 3 (TSN then TSN2) gives a better solution than Case 2 (TSN2 only) in the sense that all edges are assigned to partitions in Case 3 and some edges are not assigned to partitions in Case 2. The solution in Case 3 can be regarded as inferior to the solution in Case 1 because the Case 3 solution has more interlacing. However, the solution in Case 1 is not feasible to the vehicle-site dependency problem considered in Cases 2 and 3, where the fleet is made up of two small vehicles and one large vehicle, since there is at least one arc in each partition that must be serviced by a small vehicle only.

11.3.4 Step D. Determine Travel Path and Balance the Partitions

In Step D, there is no guarantee that the network of streets to be serviced in any partition found in Step C is connected. Thus, a *Rural Postman Problem* (RPP) is solved to find a minimum deadhead time travel path for each of the partitions. A solution to the RPP finds a continuous, minimum deadhead time travel path that covers all the arcs and edges in the partition that require service, begins and ends at the depot, and has the trips to the disposal facility integrated into the travel path. The appropriate vehicle class travel network is used in the solution of the RPP to determine the streets to deadhead when finding the minimum deadhead time travel path. Once the minimum deadhead time travel path is known, the actual time to traverse the route in the partition can be determined.

Solving the RPP can be extremely complex. The algorithm must take into account that the required street segments in a partition can form a disconnected network, some street segments are one-way, other street segments are two-way, some street segments must be serviced as a meander, and specified turn restrictions and street crossing difficulties must be considered.

If the time to traverse the route for any partition does not fall within the lower and upper bound of the target route time for the vehicle class assigned to that partition, then an automatic swapping procedure is employed to swap arcs and edges between partitions. The primary objective of this swapping is to improve the balance on the routes. In this swapping, vehicle-site dependency feasibility is maintained. When all swapping is completed, the RPP for each partition is solved again.

Let partition p be in vehicle class k. Then, in section 11.2.7 we defined a balanced solution as one in which the route length for each partition p lies in the interval, $[TRT_k - A_k, TRT_k + B_k] = [Lower_p, Upper_p]$. After solving the RPP for each partition p, if we have a balanced solution, then the results are stored and printed and the algorithm terminates.

If we do not have a balanced solution, a measure of *Solution Goodness*, SG, is computed as $SG = T1 + T2$, where

$T1$: $\sum(Time_p - Upper_p)^2$, where the sum is over all partitions p where $Time_p > Upper_p$.

$T2$: $\sum(Time_p - Lower_p)^2$, where the sum is over all partitions p where $Time_p < Lower_p$.

$Time_p$ is the total time to traverse the travel path in partition p. If this measure of solution goodness is better than that of any previous iteration, this solution is saved as the best solution found so far. A balanced solution is when the corresponding $SG = 0$.

The algorithm terminates when either a balanced solution is found or a specified number of iterations have been carried out. If a balanced solution is not found, then the best solution found so far is assumed to be the best solution for this problem. If the algorithm has not reached the specified number of iterations and we have not found a balanced solution, the algorithm continues with Step E.

11.3.5 Step E. Revise Estimate of Total Work and Adjust Fleet Mix

Steps B and E are similar, but the deadhead travel times and the actual travel paths for each partition are known in Step E. Thus, quantities that are estimated in Step B are known more accurately after carrying out Step E. This information is used to determine a possible new fleet mix. This revision in the workload estimation is based, in part, on the out-of-balance partitions that exist in the solution found in Step D. For example, if all vehicle class 1 partitions are over-saturated while all vehicle class 2 partitions have excess capacity, then a vehicle from vehicle class 1 may be assigned to a partition rather than the vehicle from vehicle class 2 that is already assigned to that partition. This swap assumes that the vehicle preference list has an available extra vehicle from vehicle class 1. After determining a revised fleet mix, the algorithm returns to Step C for a new iteration.

11.4 Implementation of the VDA in Philadelphia

In many major cities, residential sanitation routes generally are formed manually by supervisors and can be extremely out of balance. Some routes can require overtime while other routes require only 4 to 5 hours of work. Because these routes are severely out of balance and overtime has to be paid to some of the crews, management has viewed these routes as ineffective. Moreover, the crews have viewed these routes as inequitable and these inequities have lead to discontent and unhappiness. Balanced routes and schedules are generally viewed positively by both management and the crews. One of the objectives of the VDA is to develop routes that are balanced. Having routes that are balanced was a primary consideration in the acceptance of the routes generated by the VDA in Philadelphia.

The routing of the residential sanitation vehicles was part of a large-scale program that introduced Geographic Information Systems (GISs) to Philadelphia. Philadelphia wanted to increase the efficiency of its residential sanitation collection and needed a GIS-based computer system to carry out its routing. The RouteSmart system had as one of its components a set of algorithms for solving the CARP without site dependencies, and the RouteSmart system was implemented within a GIS. The VDA was developed and implemented within the RouteSmart system specifically to address the vehicle-site dependency issue. RouteSmart runs on both personal computers and UNIX platforms on various GISs. The Philadelphia Sanitation Department has RouteSmart running on a UNIX-based RS-6000 machine using ARC/INFO as the native GIS.

The version of RouteSmart in which the VDA is embedded has the following capabilities:

- The system can partition a street network of up to 20,000 street segments into as many as 100 efficient, compact partitions with up to 100 different vehicle classes. The partitions that RouteSmart generates generally have little overlap or interlacing. However, the RouteSmart solution for the CARP-VSD has more interlacing than if there were no vehicle-site dependencies.
- The system assigns a vehicle from the appropriate vehicle class to each partition.
- The system constructs a travel time path for each partition by solving an RPP over the street segments assigned to the partition.
- The system produces the desired route maps as well as a printed copy of the streets in each route and the line of travel path.
- The system creates a multicolor display of the partitions and the line of travel path for each partition.
- The system allows the user to manually swap street segments between partitions. Manual swapping allows local knowledge of the area to be incorporated into the final solution and is a critical aspect of implementation of the final solution. In many implementations in a variety of problem settings, we found that user-generated routes are more easily accepted and implemented than the computer-generated routes that do not allow user intervention.

The initial results of the VDA implemented inside RouteSmart were good. McCoy [8] reported that the city of Philadelphia was able to use 18 trucks to carry out the residential sanitation pickup in a trash district that previously required 23 trucks. These savings are similar to the results that other cities have found using the version of RouteSmart that use a homogeneous fleet. The city of Philadelphia is the first place where the VDA has been used. In Philadelphia, the target route time, TRT_k, for each vehicle class was assumed to be the same.

McCoy [8] pointed out that it is important to the efficiency of an operation to have accurate route maps and travel paths of a region and that RouteSmart generates route maps and travel paths. The importance of route maps and travel paths on a sanitation operation can be summarized as follows. With manually generated routes, the drivers may not receive route maps and travel paths of their region. This can cause missed collections and increased costs. In many sanitation operations, it is mandated that missed pickups be collected at the end of the day. In these cases, the crews servicing these missed collections are paid overtime and overtime can be a major expense. Furthermore, in some locations, it is a common occurrence for as many as 1 in 10 sanitation workers to be out sick each day. Thus, it becomes difficult for substitute drivers to efficiently drive their routes without route maps and travel paths.

11.5 Enhancements and Extensions

The use of the VDA in Philadelphia was regarded as a success. Despite this success, the development of the VDA is an ongoing effort. The following issues are being explored in the doctoral dissertation of Sniezek [11].

When determining the fleet mix to satisfy the workload, Step 2 of the VDA starts at the top of the vehicle preference list and takes the most preferred feasible vehicles available. We believe that this hierarchical approach can be improved on in determining a more desirable fleet mix. We are constructing a mathematical model that uses a weighted objective function to determine the most desirable fleet mix. In the model, we assign weights (dollar values) to the daily fixed costs and variable costs of the vehicles in each vehicle class. We believe that the fleet mix determined by this weighted objective cost function will yield a better solution than the hierarchical approach currently used. Components of fixed daily costs include salary (one-person crew versus two-person crew), depreciation of the vehicle, and daily operating expense (fuel consumption, service cost, etc.). Each of these components can vary by vehicle class. Components of variable daily costs include the tipping fee paid by the vehicle to the disposal facility every time the vehicle unloads at the disposal facility. The tipping fee is important because it can be expensive for a vehicle to be emptied at a disposal facility, and different vehicle classes can have a different number of trips to the disposal facility daily.

Algorithms for the traditional CARP grow all the partitions simultaneously since the entire service network is being partitioned at the same time. This is not the case with the VDA. The VDA creates the partitions that service streets whose site dependency says that they can only be serviced by the smallest vehicle class first. Then, the VDA creates the partitions that service streets whose site dependency says that they can only be serviced by the two smallest vehicle classes, and so forth. Some partitions for a vehicle for a vehicle class may not be created at the same time as other vehicles from that class. Creating too few partitions at a time may introduce too much deadheading. Creating too many partitions at a time may introduce too much overlap or interlacing. We are exploring other ideas for determining how many partitions to grow and the fleet mix on each iteration of Step 3 of the VDA.

Knowing the fleet mix on Step 3 of the VDA, the problem is where the seed points should be located and which vehicle class to assign to each seed point. More specifically, the following questions arise.

When concurrently forming partitions from more than one vehicle class, should the partitions from each of the vehicle classes be seeded near each other or far away from each other?

Should the location of the vehicle partitions from previous iterations and the vehicle classes assigned to those partitions influence the seeding of the vehicle partitions on the current iteration?

Should the larger vehicle partitions be seeded in high-volume areas since they can hold more volume, or should they be seeded farther from the disposal facility since they require fewer trips to the disposal facility?

Acknowledgments

We thank Dr. Arjang Assad and Dr. Bruce Golden for their ongoing guidance and insight related to vehicle-site dependencies. We also thank Professors Toth and Vigo and the referees of this chapter for their insightful comments.

Bibliography

[1] A.A. Assad and B.L. Golden. Arc routing methods and applications. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network Routing*, *Handbooks in Operations Research and Management Science* 8, North-Holland, Amsterdam, 1995, pp. 375–483.

[2] L.D. Bodin, B.L. Golden, A.A. Assad, and M. Ball. Routing and scheduling of vehicles and crews, the state of the art. *Computers and Operations Research*, 10:63–212, 1983.

[3] H.A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part I: The Chinese postman problem. *Operations Research*, 43:231–242, 1995.

[4] H.A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part II: The rural postman problem. *Operations Research*, 43:399–414, 1995.

[5] B.L. Golden and R.T. Wong. Capacitated arc routing problems. *Networks*, 11:305–315, 1981.

[6] G. Laporte. Vehicle routing. In M. Dell'Amico, F. Maffioli, and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*. Wiley, Chichester, UK, 1997.

[7] L. Levy. The Walking Line of Travel Problem: An Application of Arc Routing and Partitioning. Ph.D. thesis, University of Maryland, 1987.

[8] C. McCoy. High tech helps haul the trash. *The Philadelphia Inquirer*, July 10, 1995.

[9] B. Nag. Vehicle Routing in the Presence of Site/Vehicle Dependency Constraints. Ph.D. thesis, University of Maryland, 1987.

[10] W.L. Pearn. Augment-insert algorithms for the capacitated arc routing problem. *Computers and Operations Research*, 18:189–198, 1991.

[11] J. Sniezek. The Capacitated Arc Routing Problem with Vehicle/Site Dependencies: An Application of Arc Routing and Partitioning. Ph.D. thesis, University of Maryland, 1999.