Chapter 4

An improved initial solution algorithm

Although Ichoua et al. (2003) employ a random insertion heuristic to create initial solutions, Van Breedam (2001) introduces an initial solution parameter in his evaluation of improvement algorithms, and finds that, in most cases, a good initial solution results in significantly better final results. This thesis proposes the use of a savings route construction heuristic based on Joubert (2003)[1]. Solomon (1987) concludes that, from the five initial solution heuristics evaluated, the Sequential Insertion Heuristic (SIH) proved to be very successful, both in terms of the quality of the solution, as well as the computational time required to find the solution. Section 3.2.1 reviews a number of route construction heuristics.

4.1 A route construction heuristic

An overview of the initial solution algorithm proposed in this thesis is provided in Algorithm 4.1. Initializing the algorithm requires a distance matrix. When using benchmark data sets only customer coordinates are provided, and the Minkowski distances are calculated using (2.46). If a Geographical Information System (GIS) is used, the travel distances can be determined through a process referred to as *geocoding* and *route calibration.* The initial solution algorithm also requires a travel time matrix for all node pairs (i, j).

4.1.1 Time-dependent travel times

Congestion effects become critical when time windows are imposed by customers, because in routing the temporal issue is of greater concern than the spatial issue. Three valuable contributions that incorporate both time dependent travel time and time windows are Ahn

[1]A revised version of this chapter has been published by Joubert and Claasen (2006)

Algorithm 4.1: Initial solution heuristic

```
   Input: Customer data
   Input: Fleet data
 1 Initialize algorithm
 2 repeat Initialize tour
 3     Establish tour starting time
 4     Assign vehicle
 5     repeat Build tour
 6         Establish route start time
 7         Identify seed customer
 8         repeat Expand partial route
 9             Determine insertion criteria
10             Determine selection criteria
11             Insert node
12         until either all nodes are routed or no node identified for insertion
13         Determine multi route feasibility
14     until either all nodes are routed or route expansion infeasible
15 until either all nodes are routed or vehicles are depleted
16 Establish orphans
17 Report initial solution $s$
```

and Shin (1991), Fleischmann et al. (2004), and Ichoua et al. (2003).

Fleischmann et al. (2004) implement their routing algorithm when dynamic travel data is available through the Berlin traffic management system. Let:

$\tau_{ijk} \triangleq$ shortest travel time from node i to node j when the start time is in the time slot Z_k

with the day divided into K time slots $Z_k = [z_{k-1}, z_k], k \in \{1, 2, \ldots, K\}$. The planning horizon is denoted by the time interval $[z_0, z_K]$ which may coincide with the time window for the depot, becoming the time interval $[e_0, L_0^{\max}]$. The authors propose a smoothing of the travel time function with the introduction of

$\tau_{ij}(t) \triangleq$ travel time from node i to node j for the start time t at node i.

This is similar to the travel time proposed by Ichoua et al. (2003) where real traffic data is not accessible. A computationally efficient routine is introduced to acquire the travel time. A distance matrix $D = (d_{ij})$ is created for all $i, j \in \{1, 2, \ldots, n\}$ nodes. The planning horizon is also divided into K planning periods, while the edges are partitioned into C subsets $A = (A_c)_{1 \leq c \leq C}$ based on, for example, road type. To limit the number of speed values stored for each edge (i, j) for each time slot t, a travel speed v_{ct} is associated with each edge partition c for each time slot t. The dynamic travel time between nodes i and j can consequently be determined through Algorithm 4.2, if the travel start time at node i is denoted by $t_0 \in Z_k = [z_{k-1}, z_k]$.

Calculating the travel time matrix, however, is computationally expensive. Instead of calculating a travel time between each (i, j) pair for *each* time unit k in the scheduling period, Algorithm 4.3 introduces Time Window Compatibility (TWC) to only calculate travel time values for node pairs that have compatible time windows.

4.1.2 Time window compatibility

The introduction of the TWC concept assists in identifying, and eliminating, obvious infeasible nodes. This results in a more effective and robust route construction heuristic. The purpose of TWC is to determine the time overlap of all edges, or node combinations, (i, j), where $i, j \in \{0, 1, 2, \ldots, N\}$, and N the total number of nodes in the network. During the route construction phase, time window compatibility can be checked, and obvious infeasible nodes can be eliminated from the set of considered nodes. The Time Window Compatibility

Algorithm 4.2: Travel time calculation procedure

```
Input: Distance matrix $D = (d_{ij})$
Input: Travel speed matrix $V = (v_{ct})$
1  $t \leftarrow t_0$
2  $d \leftarrow d_{ij}$
3  $t' \leftarrow t + \frac{d}{v_{cZ_k}}$
4  while $t' > z_k$ do
5      $d \leftarrow d - v_{cZ_k}(z_k - t)$
6      $t \leftarrow z_k$
7      $t' \leftarrow t + \frac{d}{v_{cZ_k}}$
8      $k \leftarrow k + 1$
9  endw
10 $t_{ijt} = t' - t_0$
```

Algorithm 4.3: Incorporating time window compatibility with time dependent travel time

```
1  foreach *node pair* $(i, j)$ do
2      calculate $TWC_{ij}$
3      if $TWC_{ij} \neq -\infty$ then
4          foreach *time period* $k \in \{1, \ldots, K\}$ do
5              calculate $\tau_{ijk}$ using Algorithm 4.2
6          endfch
7      else
8          foreach *time period* $k \in \{1, \ldots, K\}$ do
9              $\tau_{ijk} \leftarrow \infty$
10         endfch
11     endif
12 endfch
```

Matrix (TWCM) is a non-symmetrical matrix as the sequence of two consecutive nodes, i and j, is critical. Let:

$N \triangleq$	be the total number of nodes
$e_i \triangleq$	be the earliest allowed arrival time at customer i, where $i = \{0, 1, \ldots, N\}$
$l_i \triangleq$	be the latest allowed arrival time at customer i, where $i = \{0, 1, \ldots, N\}$
$s_i \triangleq$	be the service time at node i, where $i = \{0, 1, \ldots, N\}$
$t_{ij} \triangleq$	be the travel time from node i to node j, where $i, j = \{0, 1, \ldots, N\}$
$a_j^{e_i} \triangleq$	be the actual arrival time at node j, given that node j is visited directly after node i, and that the actual arrival time at node i was e_i, where $i, j = \{0, 1, \ldots, N\}$
$a_j^{l_i} \triangleq$	be the actual arrival time at node j, given that node j is visited directly after node i, and that the actual arrival time at node i was l_i, where $i, j = \{0, 1, \ldots, N\}$
$TWC_{ij} \triangleq$	be the time window compatibility when node i is directly followed by node j

TWC_{ij} indicates the entry in row i, column j of the TWCM. Consider the following five scenarios that illustrate the calculation of time window compatibility. Each scenario assume customer j to be serviced directly after customer i, a service time of one hour, and a travel time of two hours from node i to node j.

Scenario 1: if $a_j^{e_i} > e_j$ and $a_j^{l_i} < l_j$, illustrated in Figure 4.1. Customer i specifies a time

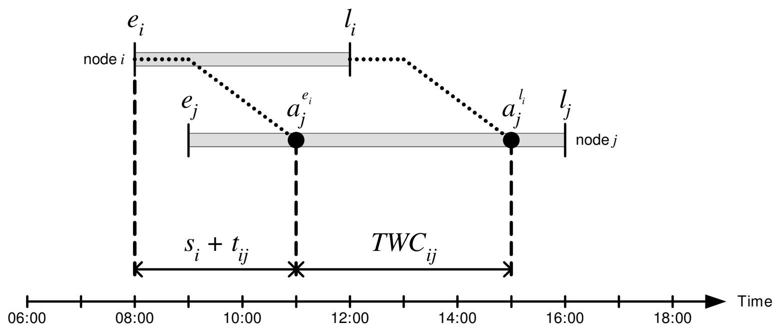

Figure 4.1: Time window compatibility scenario 1

window $[e_i, l_i]$ = [08:00,12:00], while customer j requires service during the time window $[e_j, l_j]$ = [09:00,16:00]. If service at customer i starts at the earliest allowed time, e_i, then the actual arrival time at customer j would be calculated as

$$a_j^{e_i} = e_i + s_i + t_{ij} \tag{4.1}$$

In this scenario $a_j^{e_i} = 11{:}00$. Similarly, $a_j^{l_i}$ would be the actual arrival time at customer j, given that the actual arrival time at customer i was l_i, and is calculated as

$$a_j^{l_i} = l_i + s_i + t_{ij} \tag{4.2}$$

The difference between $a_j^{e_i}$ and $a_j^{l_i}$ indicates the time window overlap between the two nodes. The time window compatibility is calculated as

$$TWC_{ij} = a_j^{l_i} - a_j^{e_i} \tag{4.3}$$

For this example, the time window compatibility is four hours (04:00).

Scenario 2: if $a_j^{e_i} > e_j$ and $a_j^{l_i} > l_j$, illustrated in Figure 4.2. Customer i specifies a time

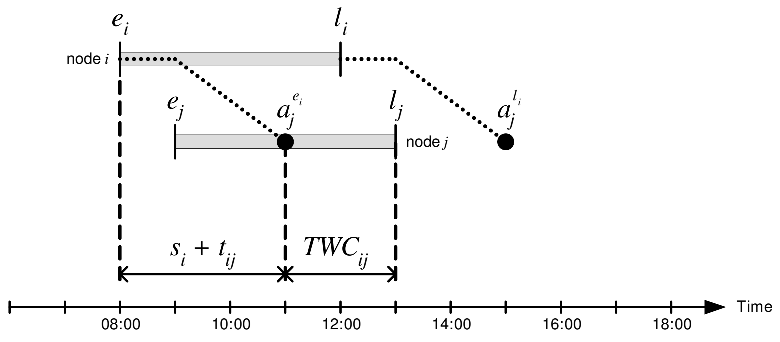

Figure 4.2: Time window compatibility scenario 2

window $[e_i, l_i]$ = [08:00,12:00], while customer j requires service during the time window $[e_j, l_j]$ = [09:00,13:00]. The calculations for $a_j^{e_i}$ and $a_j^{l_i}$ are similar to (4.1) and (4.2), respectively. The time windows of customer i and customer j only partly overlap, and the time window compatibility is calculated as

$$TWC_{ij} = l_j - a_j^{e_i} \tag{4.4}$$

For this example, the time window compatibility is two hours (02:00).

Scenario 3: if $a_j^{e_i} < e_j$ and $a_j^{l_i} < l_j$, illustrated in Figure 4.3. Customer i specifies a time window $[e_i, l_i]$ = [08:00,12:00], while customer j requires service during the time window $[e_j, l_j]$ = [12:00,16:00]. The calculations for $a_j^{e_i}$ and $a_j^{l_i}$ are similar to (4.1) and (4.2),

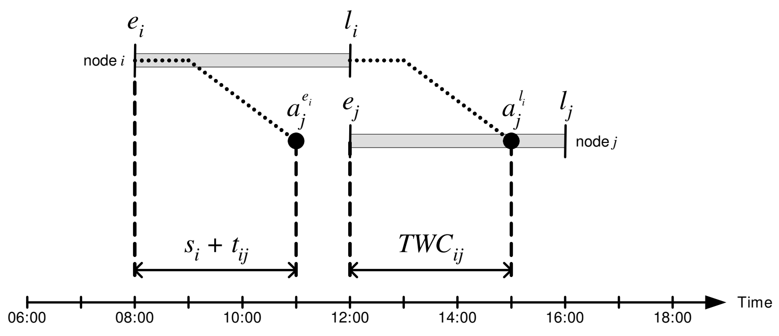

Figure 4.3: Time window compatibility scenario 3

respectively. The time windows of customer i and customer j only partly overlap, and the time window compatibility is calculated as

$$TWC_{ij} = a_j^{l_i} - e_j \tag{4.5}$$

For this example, the time window compatibility is three hours (03:00).

Scenario 4: if $a_j^{e_i}$ and $a_j^{l_i} < e_j$, illustrated in Figure 4.4. Customer i specifies a time

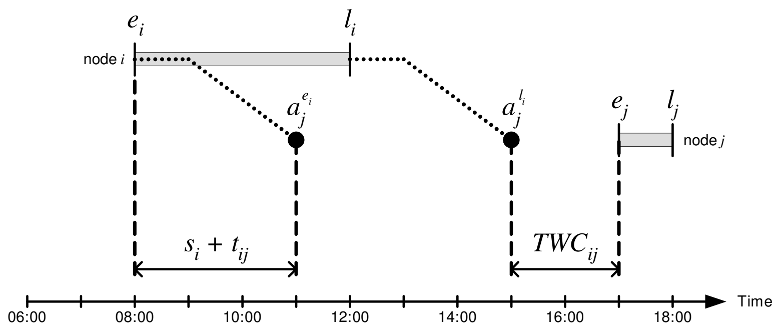

Figure 4.4: Time window compatibility scenario 4

window $[e_i, l_i]$ = [08:00,12:00], while customer j requires service during the time window $[e_j, l_j]$ = [17:00,18:00]. The calculations for $a_j^{e_i}$ and $a_j^{l_i}$ are similar to (4.1) and (4.2), respectively. The time windows of customer i and customer j do not overlap. Even if customer i is serviced as late as possible, l_i, a waiting time is incurred at customer j. The time window compatibility is calculated as

$$TWC_{ij} = a_j^{l_i} - e_j \tag{4.6}$$

For this example, the time window compatibility is negative two hours (-02:00). The significance of the negative time is that it is possible, in this case, to service customer j after customer i, although the waiting time is penalized.

Scenario 5: if $a_j^{e_i}$ and $a_j^{l_i} > l_j$, illustrated in Figure 4.5. Customer i specifies a time

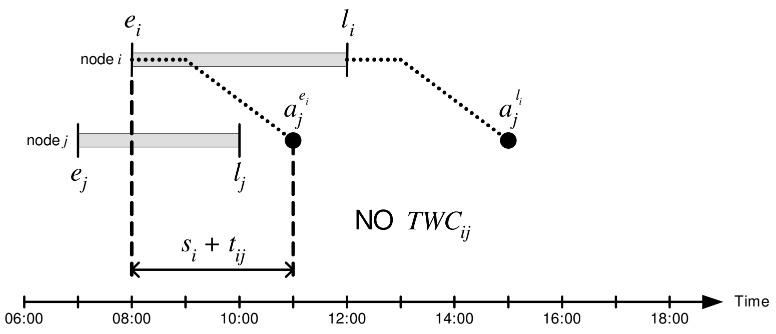

Figure 4.5: Time window compatibility scenario 5

window $[e_i, l_i]$ = [08:00,12:00], while customer j requires service during the time window $[e_j, l_j]$ = [07:00,11:00]. The calculations for $a_j^{e_i}$ and $a_j^{l_i}$ are similar to (4.1) and (4.2), respectively. Although the time windows of customer i and customer j partly overlap, it is impossible to service customer j, even if customer i is serviced as early as possible, e_i. Therefor, no time window compatibility exist.

A generalized equation is proposed that will address all five scenarios illustrated, and is given by (4.7).

$$TWC_{ij} = \begin{cases} \min\{a_j^{l_i}, l_j\} - \max\{a_j^{e_i}, e_j\} & \text{if } l_j - a_j^{e_i} > 0 \\ -\infty & \text{otherwise} \end{cases} \tag{4.7}$$

The higher the value, the better the compatibility of the two time windows considered. Therefore an incompatible time window is defined to have a compatibility of negative infinity.

Example. Consider the following example with five nodes geographical distributed around a depot in Figure 4.6. In the example, node c has indicated two possible time windows. To accommodate multiple time windows, the customer is artificially split and treated as two separate nodes, c^1 and c^2, respectively, each having a single time windows. The time windows for each customer, including the depot, as well as the service time at each node, are given in Table 4.2. The distance matrix, $\overline{D}$, is calculated using the rectangular distance between nodes. With the grid provided in Figure 4.6, the

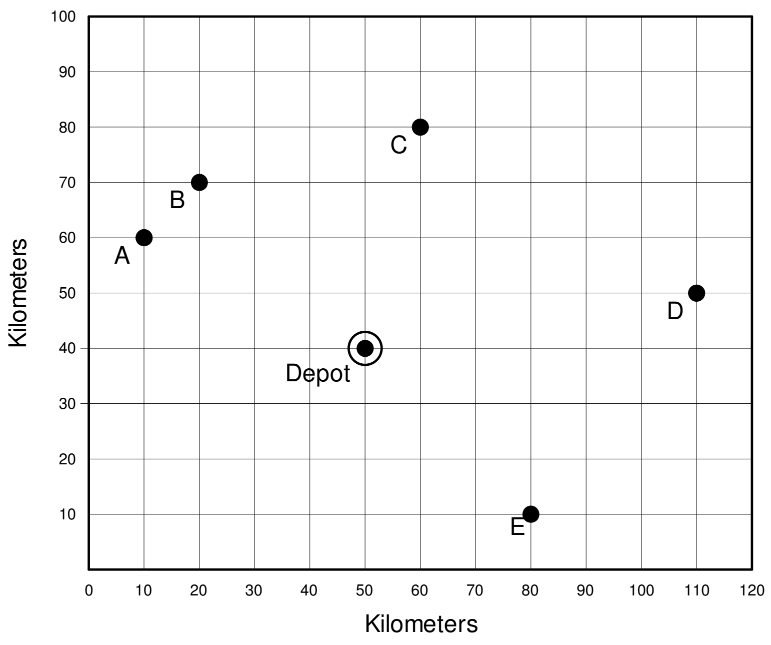

Figure 4.6: Geographical distribution of nodes around a depot

Table 4.2: Time windows and service times

Node	**Time window**	**Service time (in hours)**
(i)	$(e_i; l_i)$	s_i
Depot	07:00 – 18:00	0.00
a	08:00 – 12:00	0.50
b	11:00 – 13:00	0.25
c^1	08:00 – 09:00	0.25
c^2	15:00 – 17:00	0.25
d	08:00 – 12:00	0.50
e	10:00 – 15:00	0.25

distances can be obtained through inspection.

$$\overline{D} = \begin{bmatrix} 0 & 60 & 60 & 50 & 50 & 70 & 60 \\ 60 & 0 & 20 & 70 & 70 & 110 & 120 \\ 60 & 20 & 0 & 50 & 50 & 110 & 120 \\ 50 & 70 & 50 & 0 & 0 & 80 & 90 \\ 50 & 70 & 50 & 0 & 0 & 80 & 90 \\ 70 & 110 & 110 & 80 & 80 & 0 & 70 \\ 60 & 120 & 120 & 90 & 90 & 70 & 0 \end{bmatrix}$$

If the average speed is known, the time matrix, $\overline{T}$, can be calculated, but in the presence of time dependent travel time, the travel times are calculated using Algorithm 4.2. For illustrative purposes in this example only, $\overline{T}$ is given. Values are in hours.

$$\overline{T} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0.5 & 1 & 1 & 2 & 2 \\ 1 & 0.5 & 0 & 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 0 & 0 & 1.5 & 1.5 \\ 1 & 1 & 1 & 0 & 0 & 1.5 & 1.5 \\ 1 & 2 & 2 & 1.5 & 1.5 & 0 & 1 \\ 1 & 2 & 2 & 1.5 & 1.5 & 1 & 0 \end{bmatrix}$$

With the information at hand, the time window compatibility matrix can be calculated. For the given example,

$$\overline{TWCM} = \begin{bmatrix} 11 & 4 & 2 & 1 & 2 & 4 & 5 \\ 4 & 3.5 & 2 & -\infty & -1.5 & 1.5 & 4 \\ 2 & 0.25 & 1.75 & -\infty & -0.75 & -\infty & 1.75 \\ 1 & 1 & -0.75 & 0.75 & -5.75 & 1 & 0.75 \\ 1.75 & -\infty & -\infty & -\infty & 1.75 & -\infty & -\infty \\ 4 & 1.5 & 2 & -\infty & -1 & 3.5 & 3.5 \\ 5 & -\infty & 0.75 & -\infty & 1.75 & 0.75 & 4.75 \end{bmatrix}$$

4.2 Improving the initial solution heuristic

Initialization criteria in Algorithm 4.1 refer to the process of finding the *seed customer*: the first customer to be inserted into a new route. Joubert (2003) proposes the use of the TWC concept to identify seed customers. When looking at the TWCM example, it is clear that the

Table 4.3: Number of infeasible time window instances

	Number of infeasible time windows		
Node	**as origin**	**as destination**	**Total**
Depot	0	0	0
a	1	2	3
b	2	1	3
c^1	0	5	5
c^2	5	0	5
d	1	2	3
e	2	1	3

incompatibility is distinct for specific nodes. It is therefor possible to identify *incompatible* nodes. As opposed to the two most common initialization criteria, namely *customer with earliest deadline*, and *furthest customer*, as suggested by Dullaert et al. (2001), the author of this thesis proposes the use of the TWCM to identify seed nodes based on their time window compatibility. Table 4.3 indicates the number of instances where a node has an infeasible time window with another node, either as origin, or as destination. Both nodes c^1 and c^2 have five infeasible instances. The two artificial nodes are representing the same customer c. It can be concluded that customer c is the most incompatible node, and is identified as the seed customer. Ties are broken arbitrarily. Should two nodes have the same number of infeasible time window instances, either of the two customers could be selected as seed customer.

It may be possible to not have any infeasible time window instances. In such a scenario, a *total compatibility* value, denoted by C_a^{total}, can be determined for each node a, and is calculated using either (4.8) or (4.9),

$$C_a^{\text{total}} = \sum_{i=1, i\neq a}^{M} TWC_{ia} + \sum_{j=1, j\neq a}^{M} TWC_{aj} + TWC_{aa} \qquad \forall a \qquad (4.8)$$

$$C_a^{\text{total}} = \sum_{i=1}^{M} TWC_{ia} + \sum_{j=1}^{M} TWC_{aj} - TWC_{aa} \qquad \forall a \qquad (4.9)$$

where M refers to all the unrouted nodes, including all instances of those nodes that are split artificially. The customer with the lowest total compatibility is selected as seed customer.

Once the seed customer has been identified and inserted, the SIH algorithm considers, for all unrouted nodes, the insertion position that minimizes a weighted average of the additional distance and time needed to include a customer in the current partially constructed route. This second step is referred to as the *insertion criteria*. Note that the terms *nodes* and *customers* are used interchangeably. The insertion and selection criteria can be simplified using the example illustrated in Figure 4.7. The partially constructed route in the example

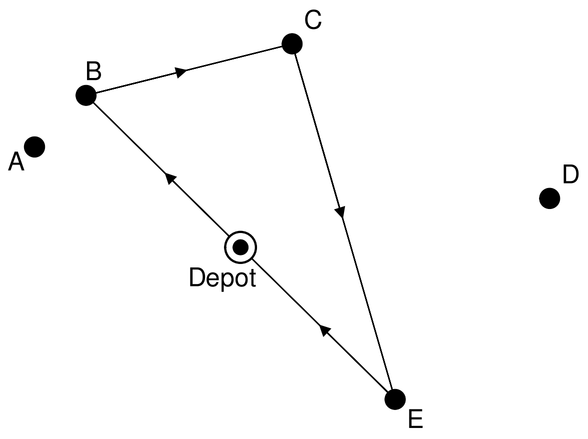

Figure 4.7: Sequential insertion of customers

consists of the depot and three routed nodes, namely B, C, and E. The route can be expressed as *Depot-B-C-E-Depot*. Nodes A and D are unrouted. The insertion criteria, denoted by $c_1(i,u,j)$, calculates the best position and associated cost, between two adjacent nodes i and j on the partial route, to insert a customer u, and is calculated for each of the unrouted nodes. Consider node A in the example. There are four edges where the node can be inserted, namely *Depot-B*, *B-C*, *C-E*, or *E-Depot*, as illustrated in Figure 4.8. Dullaert et al. (2001) extend Solomon's heuristic and determines $c_1(i,A,j)$ for the unrouted node A as

$$c_1(i,A,j) = \min_{p=\{1,2,\ldots,m\}} \left[c_1(i_{p-1},A,i_p)\right] \tag{4.10}$$

in which m represents the routed nodes in the partially constructed route. If the expressions are generalized for all unrouted nodes u, the insertion criteria is calculated as

$$c_1(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j) + \alpha_3 c_{13}(i,u,j) \tag{4.11}$$

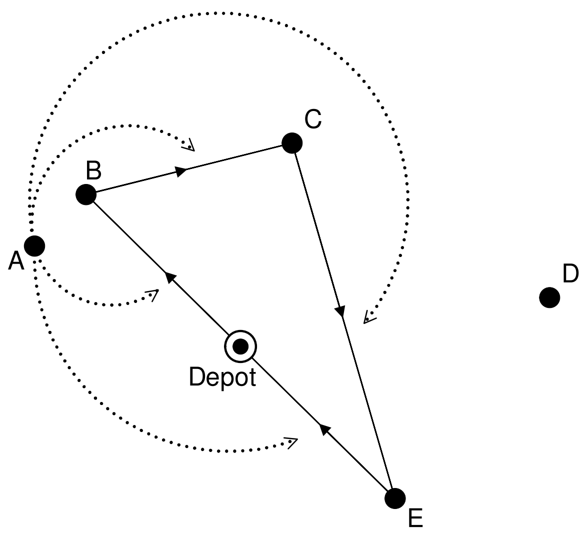

Figure 4.8: Selection criteria

with

$$c_{11}(i,u,j) = d_{iu} + d_{uj} - \mu d_{ij}, \mu \geq 0 \tag{4.12}$$

$$c_{12}(i,u,j) = a_j^{new} - a_j \tag{4.13}$$

$$c_{13}(i,u,j) = ACS,\ AOOS, \text{ or } AROS \tag{4.14}$$

With the extension to Solomon's heuristic, the weighting factors α_i need not add up to 1. The additional distance, and the additional time needed to serve customer u after customer i, but before customer j is denoted by $c_{11}(i,u,j)$ and $c_{12}(i,u,j)$, respectively. The new actual arrival time at node j is denoted by b_j^{new} in (4.13). The vehicle savings criteria, denoted by $c_{13}(i,u,j)$, considers any one of three parallel approaches to vehicle cost, where the savings concepts introduced by Golden et al. (1984) are adapted. Let:

$F(z) \triangleq$	the fixed cost of the smallest vehicle that can service a cumulative route demand of z
$F'(z) \triangleq$	the fixed cost of the largest vehicle whose capacity is less than or equal to z
$P(z) \triangleq$	the capacity of the smallest vehicle that can service a demand of z
$Q \triangleq$	be the load of the vehicle currently servicing the route
$\overline{Q} \triangleq$	be the maximum capacity of the vehicle currently servicing the route

$Q^{new} \triangleq$ be the new load of the vehicle after the customer has been inserted into the route

$\overline{Q}^{new} \triangleq$ be the (new) capacity of the vehicle after the customer has been inserted into the route

The Adapted Combined Savings (ACS) is defined as the difference between the fixed costs of the vehicles capable of transporting the load of the route after, and before, inserting customer u, and is calculated by (4.15).

$$ACS = F(Q^{new}) - F(Q) \tag{4.15}$$

The Adapted Optimistic Opportunity Savings (AOOS) extends the ACS by subtracting the fixed cost of the vehicle that can service the unused capacity, and is calculated by (4.16).

$$AOOS = [F(Q^{new}) - F(Q)] - F(\overline{Q}^{new} - Q^{new}) \tag{4.16}$$

The Adapted Realistic Opportunity Savings (AROS) takes the fixed cost of the largest vehicle smaller than or equal to the unused capacity, $F'(\overline{Q}^{new} - Q^{new})$, into account as an opportunity saving. It only does so if a larger vehicle is required to service the current route after a new customer has been inserted. AROS is calculated by (4.17).

$$AROS = [F(Q^{new}) - F(Q)] - \delta(\omega)F'\left(\overline{Q}^{new} - Q^{new}\right) \tag{4.17}$$

where

$$\delta(\omega) = \begin{cases} 1 & \text{if } Q + q_u > \overline{Q} \\ 0 & \text{otherwise.} \end{cases}$$

Any *one* of these savings criteria can be used as all three outperformed previous best published results for the initial solution (Dullaert et al., 2001). Once the best position for each unrouted node has been determined, as illustrated in Figure 4.9, the customer that is best according to the *selection criteria*, is selected — the third step in the SIH algorithm. The procedure can be expressed mathematically as

$$c_2(i, u^\star, j) \quad = \quad \max_u [c_2(i,u,j)], u \text{ unrouted and feasible} \tag{4.18}$$

$$c_2(i,u,j) \quad = \quad \lambda(d_{ou} + t_{ou}) + s_u + F(q_u) - c_1(i,u,j), \lambda \geq 0 \tag{4.19}$$

The best customer, $u^\star$, is then inserted into the partially created route between its specific nodes i and j. From Figure 4.9, consider node D to be the best node. After inserting D into

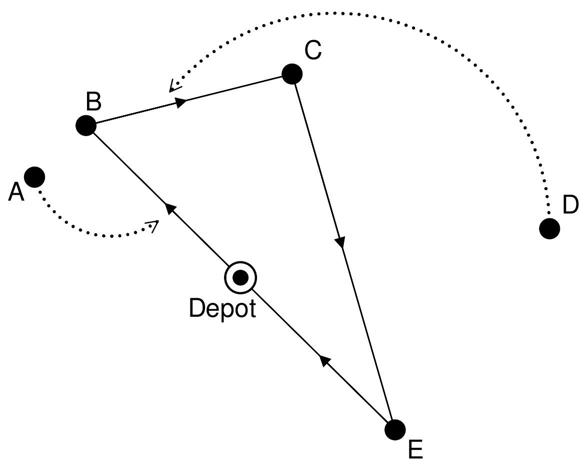

Figure 4.9: Best insertion position determined for each unrouted node

the current route, node A remains the only unrouted node, and the new route is illustrated in Figure 4.10, and can be expressed as *Depot-B-D-C-E-Depot*. The insertion process is

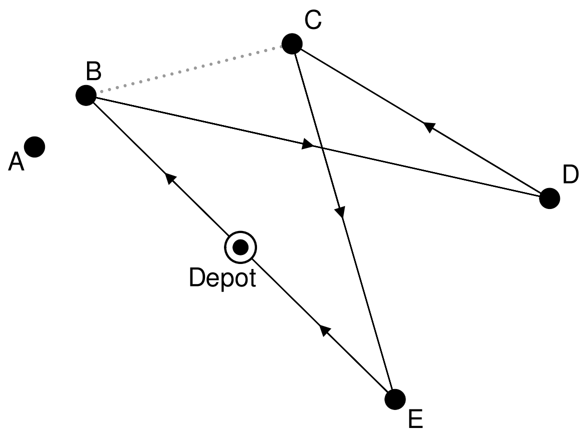

Figure 4.10: New route after inserting best customer

repeated until no remaining unrouted nodes have a feasible insertion place. A new route is then initialized and identified as the *current* route.

A shortcoming of Solomon's SIH 1987 is that it considers all unrouted nodes when calculating the insertion and selection criteria for each iteration. The fact that *all* unrouted nodes are considered makes it computationally expensive. The occurrence of obvious infeasible nodes in a partially constructed route becomes significant in the extended problem considered in this thesis. In each iteration, these criteria are calculated for each edge on the partially constructed route, irrespective of the compatibility of the time window of the node considered for insertion with the time windows of the two nodes forming the edge. For an improved case, consider the example where node u is considered for insertion between nodes i and j. As the TWCM is already calculated, it is possible to check the compatibility of node u with the routed nodes i and j. If either TWC_{iu} or TWC_{uj} is negative infinity ($-\infty$), indicating an incompatible time window, the insertion heuristic moves on and considers the next edge, without wasting computational effort on calculating the insertion and selection criteria. In the earlier example, eleven instances of infeasible time windows occur. If these instances are identified and eliminated, a computational saving in excess of 22% is achieved. The saving is calculated as the percentage of instances with time window incompatibilities of the total number of travel time instances.

4.3 Initial solutions

Solomon (1987) introduced 54 benchmark problems contained in six distinctive sets for the VRPTW, denoted by $c1$, $c2$, $r1$, $r2$, $rc1$, and $rc2$, each with 100 customer nodes. Each set highlights several factors that can affect the behavior of routing and scheduling heuristics. These factors include the geographical dispersion; the number of customers serviced by a vehicle, i.e. the relation between customer demand and vehicle capacity; and time window characteristics such as percentage of time-constrained customers, as well as the tightness and positioning of time windows.

The geographical data for the first group of problem sets are randomly generated using a uniform distribution (denote the corresponding problem sets by $r1$ and $r2$). The second group of sets are clustered (denote the corresponding problems sets by $c1$ and $c2$). A third semi-clustered group of sets have a combination of randomly distributed and clustered points (denote the corresponding problem sets by $rc1$ and $rc2$. Problem sets $r1$, $c1$, and $rc1$ have short scheduling horizons and along with vehicular capacities only allow a few customers to be serviced by a single vehicle. Problem sets $r2$, $c2$, and $rc2$ have long scheduling horizons, and when combined with large vehicular capacities, allows for a much higher number of customers being serviced by a single vehicle.

Homberger and Gehring (1999) extend the original problems to include problem sets having 200, 400, 600, and 1000 customer nodes. For illustrative purposes, Figure 4.11 shows the header of one of the Homberger and Gehring (1999) problem sets, as well as the first few customers. The depot is represented by customer '0'. The attributes for each customer include a customer number, coordinates, the demand, the earliest and latest allowed arrival, as well as the service time at each customer. The problem sets do unfortunately not accom-

```
c1_2_4


VEHICLE

NUMBER     CAPACITY

50           200


CUSTOMER

CUST NO.  XCOORD.    YCOORD.    DEMAND   READY TIME  DUE DATE SERVICE TIME


    0      70         70          0          0       1351           0
    1      33         78         20        750        809          90
    2      59         52         20          0       1240          90
    3      10        137         30          0       1172          90
    4       4         28         10          0       1183          90
    5      25         26         20        128        179          90
```

Figure 4.11: An excerpt of a problem set (Homberger, 2003)

modate a heterogeneous fleet, and the fleet structure proposed by Liu and Shen (1999b) is therefor used in this thesis — presented in Table 4.4 for each of the problem classes.

Time windows provided in the problem sets are *hard*, i.e. they allow neither early nor late arrivals. To create problem sets that will test the initial solution algorithm with *soft* time windows, a maximum lateness of $L^{\max} = 30$ time units is associated with each node, including the depot. Such time windows incur waiting time if arriving early, but allow late arrivals penalized at a unit cost of α.

Multiple scheduling is achieved through an elementary routine testing whether their is at least ρ time units between the return time of the current route and the end of the depot's time window. In this thesis the author uses an arbitrary value of $\rho = 60$ minutes.

Tables 4.5a through 4.5f show the results for 60 problem instances executed on an *Intel*® *Pentium*® *4* computer with a 3.6GHz processor (64Bit) and 3.25GB RAM.

Each table indicates the specific Homberger and Gehring (1999) problem instance from which the 100 customer data set as taken, the numbers of tours (vehicles) used in the initial solution, the total number of routes, the average time required to generate the initial solution, and the number of *orphans*. Orphans are customers from the data set that could not

Table 4.4: Heterogeneous fleet data (Liu and Shen, 1999a)

(a) Set $r1$

Type	**Capacity**	**Cost**
1	30	50
2	50	80
3	80	140
4	120	250
5	200	500

(b) Set $r2$

Type	**Capacity**	**Cost**
1	300	450
2	400	700
3	600	1200
4	1000	2500

(c) Set $c1$

Type	**Capacity**	**Cost**
1	100	300
2	200	800
3	300	1350

(d) Set $c2$

Type	**Capacity**	**Cost**
1	400	1000
2	500	1400
3	600	2000
4	700	2700

(e) Set $rc1$

Type	**Capacity**	**Cost**
1	40	60
2	80	150
3	150	300
4	200	450

(f) Set $rc2$

Type	**Capacity**	**Cost**
1	100	150
2	200	350
3	300	550
4	400	800
5	500	1100
6	1000	2500

Table 4.5a: Initial solution summary for the $c1$ problem class

Problem	**Tours**	**Routes**	**Time (sec)**	**Orphans**
$c1$_2_1	33	40	9	3
$c1$_2_2	27	30	14	1
$c1$_2_3	29	44	22	2
$c1$_2_4	19	19	30	1
$c1$_2_5	27	28	9	2
$c1$_2_6	28	37	12	2
$c1$_2_7	23	24	11	1
$c1$_2_8	23	23	14	0
$c1$_2_9	21	21	19	0
$c1$_210	19	20	22	0

Table 4.5b: Initial solution summary for the $c2$ problem class

Problem	**Tours**	**Routes**	**Time (sec)**	**Orphans**
$c2$_2_1	39	50	10	11
$c2$_2_2	29	39	15	8
$c2$_2_3	27	46	20	7
$c2$_2_4	17	17	34	6
$c2$_2_5	24	24	10	6
$c2$_2_6	25	25	14	2
$c2$_2_7	27	30	15	3
$c2$_2_8	25	25	14	1
$c2$_2_9	28	35	19	2
$c2$_210	23	24	20	0

Table 4.5c: Initial solution summary for the $r1$ problem class

Problem	**Tours**	**Routes**	**Time (sec)**	**Orphans**
$r1\_2\_1$	34	76	7	0
$r1\_2\_2$	37	71	6	0
$r1\_2\_3$	40	67	6	0
$r1\_2\_4$	59	70	6	0
$r1\_2\_5$	39	74	5	0
$r1\_2\_6$	42	69	6	0
$r1\_2\_7$	42	68	6	0
$r1\_2\_8$	57	70	7	0
$r1\_2\_9$	36	70	6	0
$r1\_210$	39	68	6	0

Table 4.5d: Initial solution summary for the $r2$ problem class

Problem	**Tours**	**Routes**	**Time (sec)**	**Orphans**
$r2\_2\_1$	13	21	21	1
$r2\_2\_2$	9	17	34	1
$r2\_2\_3$	6	7	64	0
$r2\_2\_4$	6	9	84	0
$r2\_2\_5$	9	12	28	0
$r2\_2\_6$	9	9	57	0
$r2\_2\_7$	8	10	74	0
$r2\_2\_8$	6	7	94	0
$r2\_2\_9$	9	11	41	0
$r2\_210$	10	11	41	0

Table 4.5e: Initial solution summary for the $rc1$ problem class

Problem	**Tours**	**Routes**	**Time (sec)**	**Orphans**
$rc1$_2_1	30	51	9	0
$rc1$_2_2	26	48	9	0
$rc1$_2_3	27	46	10	0
$rc1$_2_4	34	46	13	0
$rc1$_2_5	26	47	9	0
$rc1$_2_6	29	49	9	0
$rc1$_2_7	30	48	10	0
$rc1$_2_8	25	47	10	0
$rc1$_2_9	27	47	10	0
$rc1$_210	35	48	11	0

Table 4.5f: Initial solution summary for the $rc2$ problem class

Problem	**Tours**	**Routes**	**Time (sec)**	**Orphans**
$rc2$_2_1	13	26	16	0
$rc2$_2_2	11	26	23	0
$rc2$_2_3	11	24	31	1
$rc2$_2_4	11	18	31	0
$rc2$_2_5	12	20	19	0
$rc2$_2_6	12	18	18	0
$rc2$_2_7	9	18	21	0
$rc2$_2_8	11	18	22	0
$rc2$_2_9	13	18	21	0
$rc2$_210	13	19	25	0

feasibly be included in the initial solution. Ten iterations were used to calculate the average time values. Orphans are a result of the specific problem instance. The time dependent travel times that were calculated using randomly generated edge types, v_{cT}, may result in a situation whereby a customer can not be serviced within the time window of the depot, even if such customers are serviced by a dedicated vehicle.

A sample of an initial solution output file for the $r2\_2\_3$ problem set (see Table 4.5d) is provided in Appendix A. The initial solution indicates the algorithm's ability to generate more than one route per vehicle, and indicates the vehicle type assigned to the specific route. Each line represents a route, with each route starting and ending at the depot. Sequential numbers in each route represent the customers and the sequence in which customers are serviced. In the solution for the $r2\_2\_3$ problem all nodes are routed, and no orphans exist.

4.4 Conclusion

To establish an initial solution that addresses not only time windows, but also time dependent travel times and a heterogeneous fleet, requires a computational expensive routine. In this chapter the author introduced the concept of Time Window Compatibility (TWC) to ease the computational burden. The concept of TWC is also employed to identify seed customers as the most incompatible customer nodes.

Data sets from literature were adapted to create test problems for which the initial solution algorithm found solutions within seconds. The initial solutions generated in this chapter is used as inputs to the route improvement metaheuristics that are developed in Chapters 5 through 6.