2.3.2. *Pickup and Delivery*

We consider existing pickup and delivery problems to determine the similarity between it and multiple routes per vehicle. The Pickup and Delivery Problem with Time Windows (PDPTW) models the situation in which a fleet of vehicles must service a collection of transportation requests. Each request specifies a pickup and delivery location. The multiple routes per vehicle problem can be seen as a pickup from the depot and delivery to the customer. The route can stop several times at the depot to pickup goods for more customers. The depot must now also have a service time. While VRPTW is well studied, there is relatively less literature on PDPTW. Moreover, no one has developed comprehensive benchmark PDPTW instances that facilitate experimentation of new approaches.

Lau and Liang [35] presented a two-staged method to solve the pickup and delivery problem with time windows (PDPTW). In the first phase, they apply a novel construction heuristics to generate an initial solution. In the second phase, a tabu search method is proposed to improve the solution. In their model, they assume there is an unlimited number of vehicles and all vehicles have the same capacity. Lau and Liang implement a partitioned insertion heuristic, which is a hybrid heuristic combining the advantages of the standard insertion heuristic and sweep heuristic. The stops are inserted into the route as pairs, ensuring that a pickup stop is always on same route as the delivery route. They introduce three different neighbourhood moves, namely, Single Pair Insertion (SPI), Swap Pairs between Routes (SBR) and Within Routes Insertion (WRI).

The study of this method indicates that the VRPTW was adapted to work in pairs. Implementing the VRPTW with multiple routes per vehicle is less complex than the PDPTW. This thesis presents a similar approach as was presented by Lau et al [35]. From the results of Lau et al [35] study we conclude that some minor changes to the operators in our problem would be sufficient to solve the

additional constraint of allowing multiple routes per vehicle. Where the PDPTW needs to check for pairs, we will be force to check the affect of route alterations on subsequent routes.

2.3.3. VRP with Multiple use of vehicles

The Vehicle routing problem with multiple use of vehicles is a variant of the standard vehicle routing problem in which the same vehicle may be assigned to several routes during a given planning period. Taillard et al, [49] presented a tabu search heuristic for this problem.

One drawback of the standard VRP definition is that it implicitly assumes each vehicle is used only once over a planning period of duration M. For example, M could correspond to an eight-hour working day. In several contexts, once the vehicle routes have been designed, it may be possible to assign several of them to the same vehicle and thus use fewer vehicles. When m is given a priori and Q is relatively small, this will often be the only practical option. However, this possibility is not directly accounted for in the problem statement and more often than not, an efficient "packing" of the routes into working days will be hard to achieve. Designing routes with multiple uses of the vehicles is rather important in practice, but this problem (denoted by the abbreviation VRPM) has received very little attention in the Operational Research literature.

In recent years, several powerful tabu search algorithms have been proposed for the VRP. As a rule, these algorithms produce very good and sometimes optimal solutions. Rochat and Taillard presented an algorithm that allows diversification of the search process to take place by generating and combining promising solutions, not unlike what is done in genetic algorithms. More precisely, the route generation procedure first produces several good VRP solutions using tabu search. It then extracts single vehicle routes from this population of solutions,

and combines some of these routes to define a partial starting solution for another application of tabu search. This process is repeated a number of times and some of the vehicle routes generated are selected as candidates for the final VRP solution. Note that each application of tabu search has the effect of producing a full VRP solution starting from a limited set of routes and it may also modify these seed routes through the local search process.

Taillard et al [49] proposed a heuristic for the VRPM based on the algorithm of Rochat and Taillard. The proposed heuristic is made up of three parts. It first generates a large set of good vehicle routes satisfying the VRP constraints. It then makes a selection of a subset of these routes using an enumerative algorithm. Finally, it assembles the selected routes into feasible working days using several applications of a bin packing heuristic.

2.3.4. Heterogeneous Fleet

We considered work done on heterogeneous fleet for obvious reasons. The vehicle routing problem with a heterogeneous fleet of vehicles (VRPHE) is a major optimization problem. Indeed, most companies that have to deliver or collect goods own a heterogeneous fleet of vehicles. We will not consider composition of vehicles, although it is relevant to some of the problems in the industry.

The problem of composition of vehicles includes the additional problem of deciding which trailer goes with which vehicle. We solve this problem by building a vehicle set beforehand, and checking the vehicle capacity after routing. If the capacity is enough for the vehicle alone, the trailer is left at home and the total route cost is reduced.

The VRPHE has attracted much less attention than the VRP or VRPTW. This is mainly due to the fact that the VRPHE is much harder to solve than the classical VRP. Taillard [46] propose a heuristic column generation method for the VRPHE.

Taillard [46] defines the heterogeneous fleet as follows: In the heterogeneous problems, we have a set $\Psi = \{1, \ldots, K\}$ of different vehicle types. A vehicle of type $k \in \psi$ has a carrying capacity Q_k. The number of vehicles of type k available is n_k. The cost of the travel from customer i to j $(i, j = 0, \ldots, n)$ with a vehicle of type k is d_{ijk}. The use of one vehicle of type k implies a fixed cost f_k. Our implementation defines a fleet in a similar way.

A special case of VRPHE is the fleet size and mix vehicle routing problem (Golden et al., 1984 in Taillard [46]) also called the fleet size and composition VRP or the vehicle fleet mix (VFM, Salhi et al., 1992 in Taillard [46]). The goal of this problem is to determine a fleet of vehicles such that the sum of fixed costs and travel costs is minimized. This problem is a particular VRPHE for which :

1) The travel costs are the same for all vehicle types $(d_{ijk} = d_{ijk'}, k, k' \in \psi)$.

2) The number n_k of vehicles of each type is not limited $(n_k = \infty, k \in \psi)$.

We view this kind of problem as a strategic optimization and it will not be considered. Our problem is more concerned with the current situation at the depot, i.e. the fleet is already there, we cannot make major alterations on the fleet, but we must still try and optimise the vehicle use as best as we can. If results continuously show that a certain vehicle is not necessary, it can be considered to remove the vehicle from the system and determine if the algorithm still returns feasible solutions.

Another special case of the VRPHE is the VFM with variable unit running costs (VFMVRC, Salhi et al., 1992 in Taillard [46]). The VFMVRC is a particular VRPHE for which $(n_k = \infty, k \in \psi)$. Several papers on the VFM have been published. Golden et al. (1984) were among the first to address this problem. This problem's goal is similar to the VFM and is also strategic. We will not consider this implementation. Much less work has been done for the VRPHE. Let us quote the taboo searches of Semet and Taillard (1993) and Rochat and Semet (1994) (in Taillard [46]) for real-life problems including many other constraints.

"For homogeneous VRPs, many heuristic methods have been proposed. Among the most efficient ones, are the adaptive memory procedure (AMP) of Rochat and Taillard (1995) and the taboo search of Taillard (1993). This last method uses a local search mechanism based on the move of one customer from one tour to another or the exchange of two customers that belong to different tours. Since the vehicles are identical, it is easy to check the feasibility of a move and to evaluate its cost. For the VRPHE, the feasibility check or the evaluation of a move requires finding a new assignment of the vehicles to the new solution's tours. In Semet and Taillard (1993), several techniques have been proposed to simplify and accelerate the re-assignment of vehicles to tours. However, the re-assignment problem is very simple in the case of the VFM: each tour is performed with the cheapest vehicle type that is able to carry all the orders of the tours. This is certainly a reason that the VFM has been more studied than the VRPHE."

The above quote is a warning on the addition of heterogeneous fleet to our VRP, especially if we do not apply it in the sense of the VFM. We will show, however, that the methods used in our implementation are sufficient enough and effective in a reasonable time period.

Taillard presents a heuristic column generation method for solving the VRPHE. The column generation is based on the AMP of Taillard (1994), which uses an embedded taboo search. Taillard proposes to treat the VRPHE by solving a succession of homogeneous VRPs, since the solution methods for homogeneous VRPs are becoming more and more efficient. For each type of vehicle, they solve a homogeneous VRP (without limitation on the number of vehicles available) with an AMP. The tours of the homogeneous VRP solutions are then combined to produce a solution to the VRPHE.

The AMP first generates a set of good solutions using the taboo search. It then extracts single vehicle tours from this set of solutions, and combines some of these tours to define a partial starting solution for another application of taboo search. This process is repeated a number of times and the tours are memorized as candidates for the final VRPHE solution. Once the homogeneous VRPs are solved for each vehicle type, one has a set T of tours that have been memorized. The useless tours of T are removed: only one copy of each tour is kept in T; the dominated tours are eliminated (a tour is dominated if it is more expansive than another tour of T servicing the same customers). In the case of the VFM, the algorithm always produces a feasible solution if the iterative search used to solve the homogeneous VRP succeeds in finding feasible solutions.

In our objective, the proposed solution is not considered for the following reasons:

- In the case of the VFM, an unlimited number of vehicles exist to solve the problem. Whatever feasible tour is selected from the homogeneous solution list is possible, as the vehicle exist. In our instance, it might happen that the selected vehicle route cannot be used, as the number of routes for the type of vehicle already equals the number of vehicles

available. Another vehicle must be selected for this route, which might not result in the best solution.

- If we start to make alterations to the selection of vehicles, it might happen that the routes in the list for a specific vehicle on a specific stop are exhausted by the other vehicles. Al the routes, which included this stop, is removed from the possible route list. This can result in stops not being visited, because there is no vehicle available, or so it seems. We can build up a route, which consist of the unrouted stops to insure a feasible solution, but this will result in a solution that is not the best.

- As mentioned previously, we cannot guarantee that Taillard's method will result in the best solution. If we add to that the additional complexity of our problem, it can really get time consuming to rebuild the solution from a set of feasible homogeneous vehicle routes. This implies that the heuristic method applied on the homogeneous vehicle solution will be applied a few times. With the available computer power as well as the complexity of the data sets we work with, it will be more effective to implement the vehicle selection method into the heuristic. Taillard found that for problem instances involving very few vehicles, there was a higher probability that a run would not produce a good or even a feasible solution.

2.3.5. Time Windows

The Vehicle Routing Problem with Time Windows (VRPTW) is by far the most popular implementation of the VRP. Our problem implements various extensions on the original idea of a time window. The customers to be visited can have multiple time windows. The vehicles to be used will also have available time windows that will allow the user to schedule certain vehicles for long hauls where

necessary. There exist a wide variety of implementation methods for the VRPTW.

A Hybrid Search Based On Genetic Algorithms And Tabu Search For Vehicle Routing

Ombuki et al, [39] presented a hybrid search technique based on meta-heuristics for approximately solving the VRPTW. The approach is two phased; a global customer clustering phase based on genetic algorithms (GAs) and a post-optimization local search technique based on Tabu search (TS). They also devised a new crossover operator for the VRPTW and compare its performance with two well-known crossover operators for VRPTW and related problems. Computational experiments show that the GA is effective in setting the number of vehicles to be used while the Tabu search is better suited for reducing the total number of distance travelled by the vehicles. Through their simulations, they conclude that the hybrid search technique is more suitable for the multi-objective optimization for the VRPTW than applying either the GA or Tabu search independently. We definitely take this from their research and will also implement a hybrid approach.

In this paper a hybrid search technique is proposed which is suitable for multi-objective optimization. Their approach is two phased; a global customer clustering phase based on genetic algorithm and a post-optimization local search technique based on Tabu search. The objective function states that costs should be minimized. In this case the objective is to minimize the number of vehicles used and the distance travelled to meet the demand of all the customers while not exceeding capacity of the vehicle and the latest time for serving each customer. Thus this problem can be treated as a multi-objective optimization problem.

In the GA, each chromosome in the population pool is transformed into a cluster of routes. The chromosomes are then subjected to an iterative evolutionary

process until a minimum possible number of clusters are attained or the termination condition is met. The transformation process is achieved by the routing scheme whereas the evolutionary part is carried out like in ordinary GAs, that is, in each generation, genetic operations, crossover and selection are applied upon chromosomes. We represent each chromosome as sequence of cluster of routes. A route is composed of a sequence of nodes (customers). Each chromosome represents a possible solution for the VRPTW.

The following figure shows the performance of the genetic algorithm compared to that of the Tabu search technique. In the case of Figure 3, the main objective under scrutiny is how GA and Tabu search performs respectively in defining the final number of vehicles to be used to service the customers for the VRPTW. Likewise, Figure 4 demonstrates their performance when the main objective observation is to minimize distance travelled. The vertical axis in both figures shows the number of customers not served. The more customers served, the better. From Figure 3 we observe that GA performs better than the Tabu search in searching the "optimal" number of vehicles to service the customers. As the figure shows, the GA manages to employ a smaller number of vehicles and also to serve more customers than the Tabu search approach.

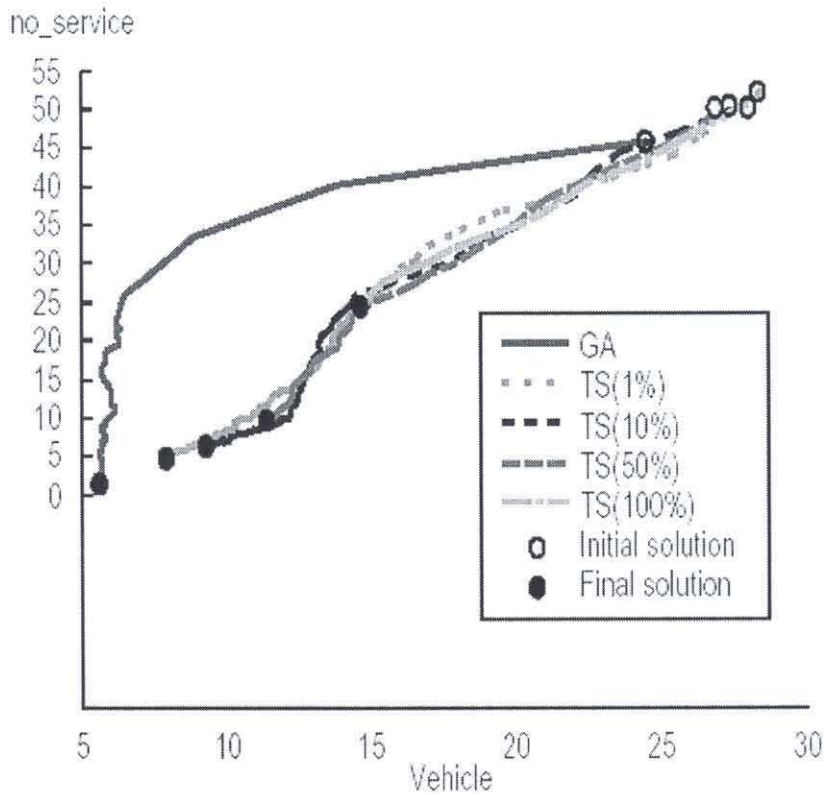

Figure 3: GA vs. Tabu Search for Minimizing Vehicles

On the other hand, Figure 4 depicts that the Tabu search outperforms the GA when it comes to minimizing the total distance travelled. Clearly, this is a case of conflicting objectives. In-order to reduce the travelled distance; one would need to increase the number of vehicles. On the other hand, to reduce the cost of employing more vehicles, one needs to increase the distance travelled per vehicle (which does not necessarily solve the problem as the cost of gas and other resources comes into play as well).

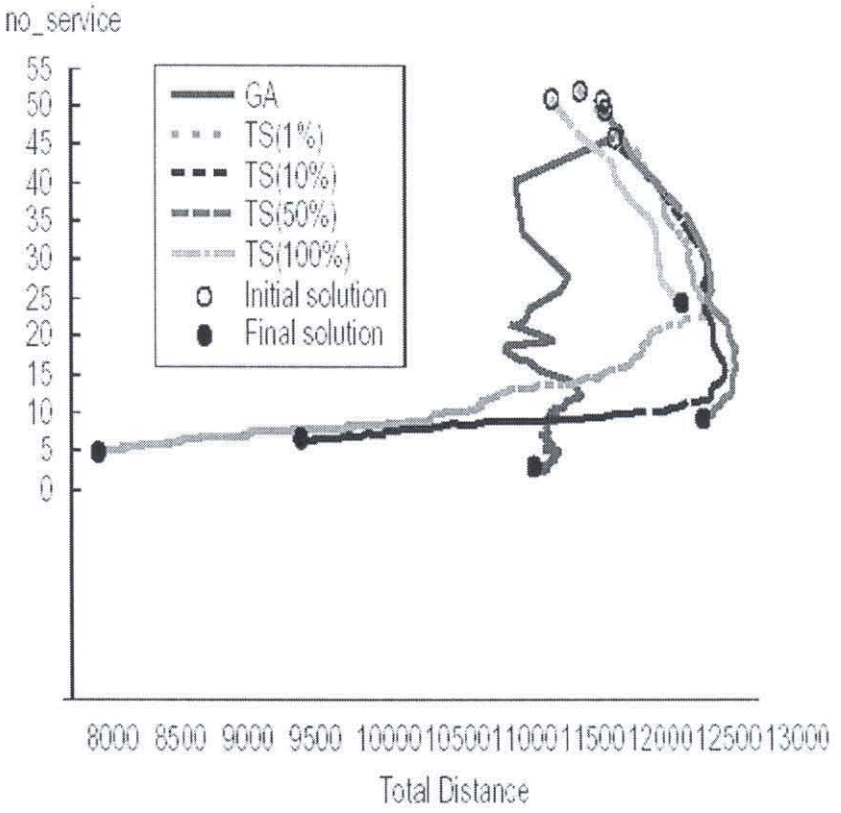

Figure 4: GA vs. Tabu Search for Minimizing Distance

In our problem we are not concerned about reducing vehicles as a main objective, although we would like to utilise a vehicle as good as possible. Instead of making use of GA for vehicle reduction, we implement methods to handle heterogeneous fleet, as well as multiple scheduling. The GA method in this implementation as a heuristic and not a meta-heuristic. What we are looking for is a method to handle the meta of our algorithm.

Although they do not specify the side constraints, except for the time windows, additional side constraints can be implemented and will affect the algorithm in testing for feasibility. We can expect similar results for our problem as in this instance.

A Network Flow-Based Tabu Search Heuristic for the VRP

Xu and Kelly [54] introduced a network flow model as a general local search strategy to solve the VRP. They used a straightforward model by relaxing the hard side constraints and introducing a dynamic penalty system, and efficiently update and frequently solve the network flow model to find the best customers to insert into new routes without the use of the generalized assignment problem. The penalty parameters are changed such that the feasibility of the search is controlled.

The network flow model implements Tabu Search restriction to prevent the method from getting trapped in local optima. TS restrictions with randomly generated tabu tenures are applied to them three neighbourhood moves: dropping a customer from its current route, inserting a customer into a different route and swapping two customers between routes. For the swap, in addition to Tabu restrictions on future swaps, the associated ejections and insertions are also subject to tabu restrictions. When a customer is moved to a new route, a tabu restriction that prevents its removal from that route is only activated when there are only a few customers (less than a pre-determined number) in the route.

From their implementation we conclude that each operation can have its own tabu tenure. Ideally we would like to set the tabu tenure during execution for each operation. We also conclude that the execution of an operation might result in tabu moves for other operations. We must identify the dependencies of operations beforehand.

Xu and Kelly [54] also implement an intensification strategy that we inherit from them based on advanced restart/recovery procedure. The set of best feasible solutions produced by the search are defined as elite solutions. A repository of elite solutions is maintained. Advanced restart is executed periodically during the late stages of the search. When restarting, the current solution is obtained from the repository and all tabu restrictions are released. This strategy is based on the assumption that there may exist short relinking paths in the search process from the restart points to new local or global optima. However, these paths may not be detected during prior search due to the tabu restrictions. The advanced restart/recovery strategy may find these paths and thereby lead the search to new local or global optima.

Vehicle Routing in Constraint Programming

De Backer and Furnon [18] consider constraint programming for solving VRPs. However, this raises many problems. Search in constraint programming is usually based on depth-first search. This means that the domains of each variable are monotonically reduced by propagation during the search. Although this approach can be useful for finding a first solution for the VRP, it is not practicable when an optimized solution is sought. This is the reason why much research has been devoted to the design and the implementation of local search techniques in the context of routing problems.

The paper presents basic principles for implementing local search techniques and meta-heuristics in constraint programming. These principles have been applied to Tabu Search. We consider the basic VRP with additional side constraints. Expressing such constraints as what we are considering, can be tedious, and yield problems with huge models, especially in the case of traditional linear programming (LP) models, or make programs solving VRP very complex and difficult to maintain.

In standard LP models, decision variables usually belong to a set of Boolean variables x_{ij}^{k} which take the value 1 if the vehicle k is used to travel from visit i to j. Therefore, these models use $O(mn_2)$ decision variables, where m is the number of vehicles and n is the number of visits to perform.

We implement the VRP with a number of variables that is linear (instead of quadratic) with respect to the number of visits. Each visit i is associated with two finite-domain variables $next_i$ and veh_i, representing respectively the possible visits following i and the vehicle serving visit i. This method allows us to quickly access the feasibility of a route by traversing only a part of the route depending on where the alteration took place.

De Backer and Furnon [18] devise a generic way of taking into account constraints on dimensions that can be a diverse as weight, time, or volume. They introduce the notion of a path constraint, which are similar to the way that we implement constraints on a route. Path constraints are able to propagate accumulated quantities such as time and weight along a vehicle tour.

We implement a similar method to test for feasibility of a route. Constraints are prioritised according to ease of calculation and importance on failing, e.g. to insert a node in a route, the vehicle capacity must be sufficient to accept the new node as well. It is quick to test the current capacity of the route plus the new load of the stop against the capacity of the vehicle.

Time Window Compatibility

Time Window Compatibility (TWC) refers to the compatibility of the time window(s) of one stop with regards to another. A good TWC figure indicates that the two nodes are likely to be inserted in sequence on the same route. In many cases two customers can be located next to each other, but their time

windows is not compatible. The trade-off between distance (i.e. cost) and time (i.e. customer delight) is an inherent part of the problem.

Insertion of stops in a heuristic fashion requires a selection process that result in a possible next stop. The TWC can assist us in ruling out infeasible stops from the start. We define the term neighbour for a stop. A neighbour is a stop that can be visited from the current stop. If we know that a stop is not a neighbour of the current stop, we do not even waste time of trying to implement that stop as a next stop. The neighbours of a stop are made up of all the time window compatible stops. We utilise the TWC principle as proposed by Van Schalkwyk [52], but we implement it in a different fashion. A discussion of the TWC follows and Chapter 3 will discuss the implementation of this concept in our solution.

The figure below illustrates a scenario where we evaluate the time adjacency of node *i* and node *j*. This scenario assumes that there will be a definite overlap in time windows between the two nodes. Other scenarios will subsequently be discussed.

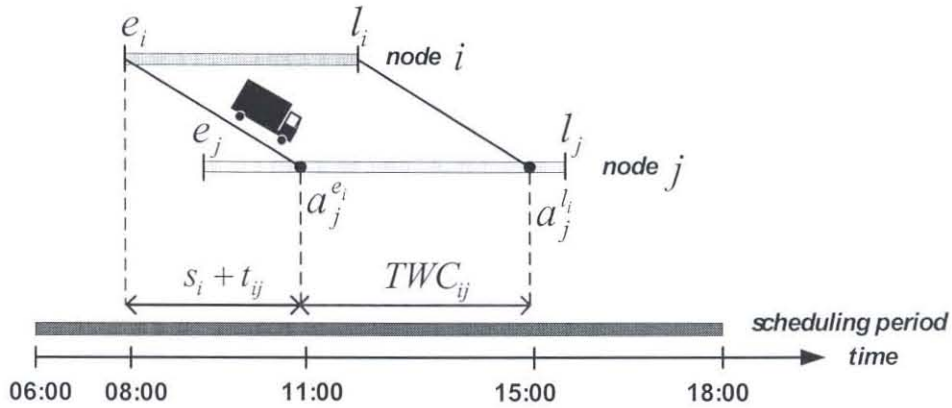

Figure 5: The basic TWC calculation - Scenario 0

Scenario 0: *IF* $a_j^{e_i} > e_j$ *AND* $a_j^{l_i} < l_j$

Customer i specified a time window (e_i, l_i) between 8:00 and 12:00, and customer j requires service between 9:00 and 16:00 (e_j, l_j). If serviced started at node i at e_i (the earliest feasible time), its arrival at j would be:

$$a_j^{e_i} = e_i + s_i + t_{ij}$$

In this scenario equals 11:00.

Similarly, al would be the arrival at j if service started at node i at the latest possible time (l_i):

$$a_j^{l_i} = l_i + s_i + t_{ij}$$

In this scenario equals 15:00.

The difference between $a_j^{e_i}$ and $a_j^{l_i}$ will yield the amount of time overlap between i and j:

$$TWC_{ij} = a_j^{l_i} - a_j^{e_i}$$

In this scenario it equals 4 hours. The significance of this value is that the bigger the overlap, the better we can insert the two nodes in sequence. This also ensures that the customer with a big overlap is routed first (more flexible).

A number of different scenarios will be illustrated in the following figures.

Scenario 1: If $a_j^{l_i} > l_j$

If the earliest arrival time at node j is inside the acceptable time window, but the latest arrival time is outside of the acceptable time window of node j, the two customers only partly overlap. The TWC_{ij} is then calculated by the following equation:

$$TWC_{ij} = l_j - a_j^{e_i}$$

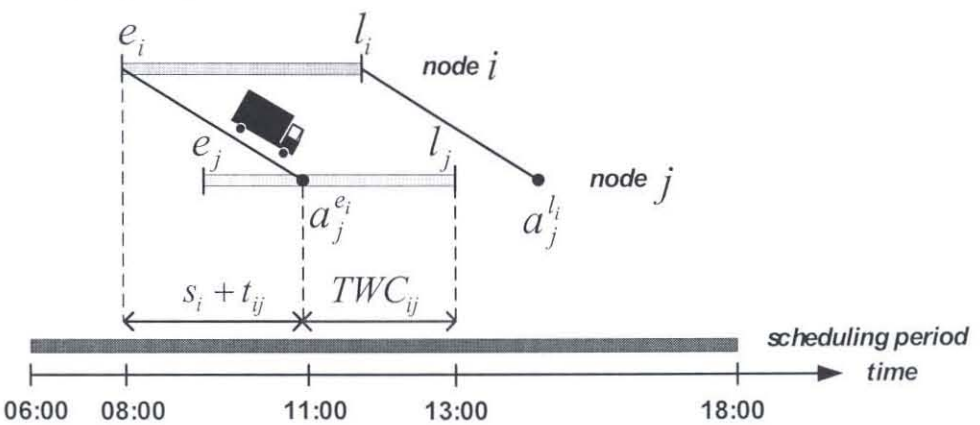

Figure 6: Scenario 1 TWC calculation

Scenario 2: If $a_j^{e_j} < e_j$

If the vehicle arrives at the earliest feasible time and this is before the acceptable time window of node j, and the arrival of the latest feasible time at node j is inside the acceptable time window, the two customers only partly overlap. The vehicle has to wait to service customer j. The TWC_{ij} is then calculated by the following equation:

$$TWC_{ij} = a_j^{l_j} - e_j$$

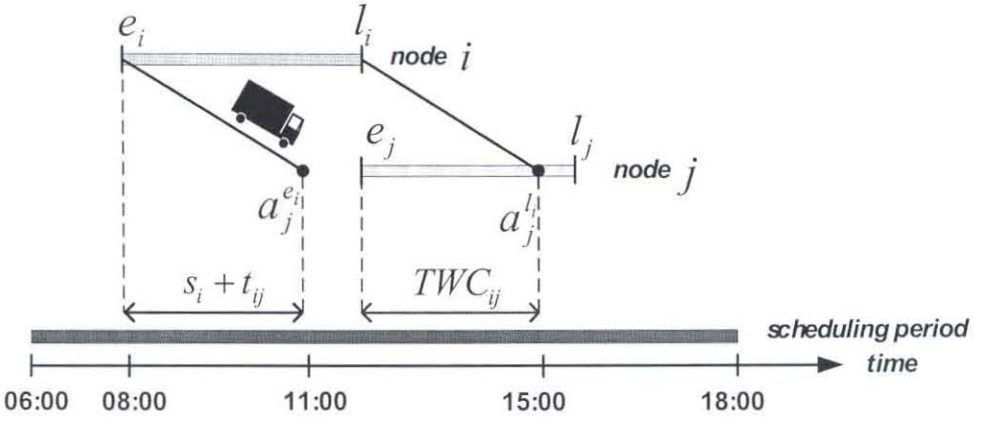

Figure 7: Scenario 2 TWC calculation

Scenario 3: If $a_j^{e_i} < e_j$ and $a_j^{l_i} < e_j$

If the latest arrival time at node j is earlier than the start of the acceptable time window at node j, the vehicle always waits at node j, irrespectable of the arrival time at node i. The arrival at j is always before its acceptable start time. This value will be negative, and calculated as follows:

$$TWC_{ij} = a_j^{e_i} - e_j$$

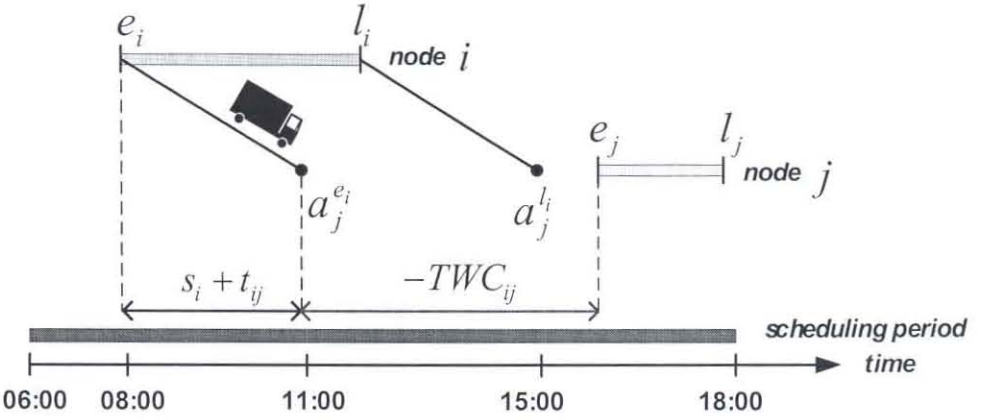

Figure 8: Scenario 3 TWC calculation

Scenario 4: If $a_e > l_j$ and $a_l > l_j$

If the arrival time at j is always bigger than the latest acceptable time at j, the node-combination is infeasible. The nodes forming part of this combination will typically be eliminated before starting the algorithm, as they can obviously not be included in the current route under construction.

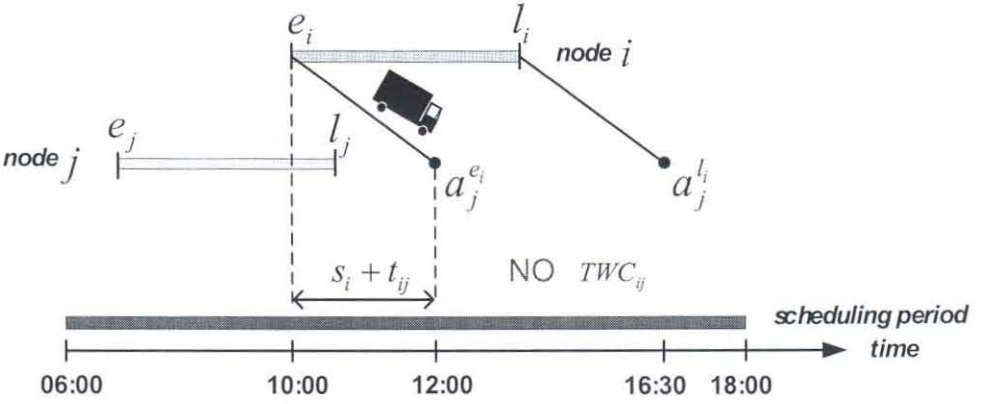

Figure 9: Scenario 4 - infeasible combination