

The Multi-shift Vehicle Routing Problem with Overtime

Yingtao Ren, Maged Dessouky, and Fernando Ordóñez

Daniel J. Epstein Department of Industrial and Systems Engineering
University of Southern California
3715 McClintock Ave, Los Angeles, CA 90089-0193

January 2010

Abstract:

In this paper, we study a new variant of the vehicle routing problem (VRP) with time windows, multi-shift, and overtime. In this problem, a limited fleet of vehicles is used repeatedly to serve demand over a planning horizon of several days. The vehicles usually take long trips and there are significant demands near shift changes. The problem is inspired by a routing problem in healthcare, where the vehicles continuously operate in shifts, and overtime is allowed. We study whether the tradeoff between overtime and other operational costs such as travel cost, regular driver usage, and cost of unmet demands can lead to a more efficient solution. We develop a shift dependent (SD) heuristic that takes overtime into account when constructing routes. We show that the SD algorithm has significant savings in total cost as well as the number of vehicles over constructing the routes independently in each shift, in particular when demands are clustered or non-uniform. Lower bounds are obtained by solving the LP relaxation of the MIP model with specialized cuts. The solution of the SD algorithm on the test problems is within 1.09-1.82 times the optimal solution depending on the time window width, with the smaller time windows providing the tighter bounds.

Keywords: Vehicle routing problem with time windows (VRPTW), multi-shift, overtime, insertion heuristic, tabu search

1. Introduction

The basic Vehicle Routing Problem (VRP) is concerned with finding a set of routes to serve a given number of customers, minimizing the total distance traveled. In this problem, the total demand of all the customers on a route must not exceed the vehicle capacity. If in addition each customer specifies a time window within which the customer must be visited, the problem is known as a VRP with time windows (VRPTW). There are many other variants of the VRP. Classical variants of the VRP aim at planning the routes of a fleet of vehicles for a single planning period (a day or a shift). The above VRP is however unrepresentative of some situations, in which companies have to route vehicles to satisfy demand for continuous operations, in many cases routing and scheduling in a 24 hour period divided into shifts. In such cases, a solution that forces all vehicles to return to the depot before the end of each shift can perform suboptimally, in particular in situations with high demand near shift changes or with long distances. If the demands were the same in each shift we can simply find an optimal solution (or a good solution) and repeat the same routes. These repeated routes could still be improved with overtime if before returning to the depot a vehicle turns out to be close to a demand of the next shift. In practice, however, the demand fluctuates causing repeated routes to be inefficient.

The objective of this paper is to introduce a VRP model that plans over several periods and allows routes to exceed shift lengths at a certain overtime cost, if that decision is economical. This work then investigates the impact of the tradeoff between overtime and other operational costs (e.g. travel cost, regular driver usage, and unmet demands) on the efficiency of the routing solutions. For example, if a customer in the next shift happens to be on the return trip to the depot for a vehicle of the current shift, the vehicle can serve it incurring a small overtime. This would reduce demands in the next shift, possibly resulting in less total travel distance and fewer drivers.

We consider a multi-shift VRP with overtime. The problem is inspired by a routing problem in a leading healthcare organization that operates about 240 medical office buildings in the Southern California region. The healthcare provider continuously routes medical samples, mails, x-rays, lab-specimens, documents, etc. between various medical facilities and a central lab for testing.

The medical facilities are located throughout Southern California, causing travel times between facilities to be on the order of the entire shift length in the worst case. The organization has about 70 vehicles to carry out the deliveries. Because of the random nature of customer demands in the healthcare industry, demands occur any time during a day, even during shift changes. In addition, most medical samples are perishable and must be processed within a certain period, so the demands have time windows. There are no capacity constraints because the items delivered are light and small. To serve these demands, vehicles in this problem travel through multiple urban areas and are operated on a 24/7 basis in three consecutive shifts in each day. The central lab includes the vehicle depot where all routes in a shift start and end. Overtime is allowed at a higher salary rate. Third party vehicles, such as taxis, are hired to serve the unmet demand.

There is limited research on VRP that plans multiple trips or over multiple periods. One VRP variant that considers dependency between routes is multi-trip VRP (MVRP) (Taillard et al., 1996; Brandão and Mercer, 1997, 1998; Petch and Salhi, 2003; Campbell and Savelsbergh, 2004; Azi et al., 2006; and Salhi and Petch, 2007). In the MVRP, vehicles can be assigned to more than one route within a time period. The MVRP is different from our problem because the multiple trips still occur in a single shift, and overtime is considered only for the last trip of a vehicle. Another VRP variant that considers dependency between periods is Periodic VRP (PVRP) (Angelelli and Speranza, 2002; Francis and Smilowitz, 2006; Hemmelmayr et al., 2009; and Alonso et al., 2008). The PVRP extends the classical VRP to a planning horizon of several days. Each customer requires a certain number of visits within this time horizon while there is some flexibility on the exact days of the visits. Hence, one has to choose the visit days for each customer and to solve a VRP for each day. In addition to deciding when a demand is serviced, the PVRP is different from our problem because the operations consider only one regular shift in a period (workday). In addition, overtime is not considered in the PVRP, and demands do not have time windows.

It is common in the production planning and scheduling literature to use overtime as an option for shift scheduling (Lagodimos and Mihiotis, 2006; Merzifonluoğlu et al., 2007). For instance, Lagodimos and Mihiotis (2006) show that effective use of overtime leads to workforce reductions and improved utilization in packing shops. However, only a few prior work has

considered overtime as an option for vehicle routing and scheduling. In the 1970s, transit systems generally used models to estimate the costs of bus systems. In these unit cost models, the estimated cost of a proposed timetable for a transit system was simply the sum of two cost factors: the number of buses and the total mileage. These cost models for analyzing bus systems were extended in Bodin et al. (1978, 1981) to include factors such as the number of bus drivers needed, overtime, maintenance costs, etc. Snizek and Bodin (2002) argue that only considering total travel time in the objective function is not enough in evaluating VRP solutions, especially for non-homogeneous fleets. Instead, they determine a Measure of Goodness, which is a weighted linear combination of many factors such as capital cost of a vehicle, salary cost of the driver, overtime cost, mileage cost, and the tipping cost of a sanitation vehicle at the disposal facility, to compare the various solutions generated. They use the cost models to generate and evaluate solutions to the Capacitated Arc Routing Problem with Vehicle-Site Dependencies (CARP-VSD). These cost models allow for the use of overtime in generating routes and analyzing solutions. These models confirm a long-term conjecture of the authors that using overtime can help to generate less expensive solutions to vehicle routing problems because of the savings in capital cost of the vehicles. Recently, Zäpfel and Bögl (2008) study a multi-period vehicle routing and crew scheduling problem with overtime and outsourcing options. The problem is different from our problem because the operation in their problem is not continuous. There are two break periods in each workday; second, the demands must be served in the same period as their occurrence; and third, the overtime constraint is imposed on a whole week basis, not on each individual shift.

The rest of this paper is organized as follows. In Section 2 we present a mixed integer programming (MIP) formulation of the problem. Section 3 describes two insertion heuristic algorithms to solve the problem. Section 4 describes an approach to obtain a lower bound of the problem by solving the relaxation of the MIP model. Section 5 presents the experimental results, which include comparison of the performance of the two algorithms, and comparison of the best solution with the lower bound. We conclude the paper and discuss future research in Section 6.

2 Problem Formulation

Assume we know in advance the demand for a planning horizon of P days. We consider three daily shifts of length L hours (e.g., the three shifts are 0:00am-8:00am, 8:00am-16:00pm, and 16:00pm-24:00pm if $L=8$). Let T denote the set of shifts, with $|T| = 3P$ and $T = \{1, 2, \dots, 3P\}$. For shift t , $t \in T$, the shift start time is $B_t = L(t-1)$, and the shift end time is $E_t = Lt$.

The set of demand nodes is denoted as D . A hard service time window $[e_i, l_i]$ is also associated with each demand node $i \in D$, where e_i and l_i represent the earliest and latest service start times, respectively. Let n denote the total number of demand nodes with $n = |D|$. We create $|T|+1$ copies of the depot represented by nodes $n+1, \dots, n+t, \dots, n+|T|+1$. Node $n+1$ represents the origin depot of shift 1, node $n+t$ represents the origin depot of shift t as well as the destination depot of shift $t-1$, $t \in \{2, \dots, T\}$, and node $n+|T|+1$ represents the destination depot of shift T . The problem can be defined on a directed graph $G=(V, E)$, where $V = D \cup \{n+1, \dots, n+t, \dots, n+|T|+1\}$. Each arc $(i, j) \in A$ has an associated travel time t_{ij} , and the travel cost is W per hour. Note that arcs $(n+t, n+t+1)$ are included in the graph to model situations in which a vehicle is not used during a shift.

Let K denote the set of vehicles. A vehicle can be reused by another driver in the next shift after it returns to the depot. A driver is ready to work at the start time of each shift. The working time of a driver is from the shift start time until he/she returns to the depot. At the end of each route in a shift, all vehicles should return to the home depot. Because there are a limited number of vehicles, the next driver assigned to a vehicle has to wait until the vehicle returns to start its route. For example, if a vehicle from shift $t-1$ returns at $B_t + 2$, its earliest start time for the next shift will be $B_t + 2$. For a fleet of 5 vehicles, if $L=8$ and the end times of the routes for shift 1 are $[5, 8, 9, 10, 6]$, then the earliest start times of the vehicles for shift 2 will be $[8, 8, 9, 10, 8]$.

We assume that the overtime limit is \bar{L} hours. That is, the working time of a driver cannot be longer than $L + \bar{L}$ hours. To deal with the overtime limit, we impose a time window $[E_{t-1}, E_{t-1} + \bar{L}]$ on the depot node $n+t$, $t \in \{2, \dots, T+1\}$. For node $n+1$, the time window is $[0, 0]$. It

is reasonable to assume $\bar{L} < L$, since overtime cannot be too long. In this case, the time windows of node $n+t$ and $n+t+1$ do not overlap. The regular salary rate is R per hour, and the overtime salary rate is S per hour. In some situations the use of overtime is inevitable. For example, if a demand occurs late in shift $t-1$ and $e_i + t_{i0} > E_{t-1}$, or a demand occurs early in a shift t and $B_t + t_{0i} > l_i$, then a vehicle in shift $t-1$ has to use overtime to meet such demands. Once a vehicle in a shift is used, it is assumed that the driver is paid for the entire shift even if the vehicle returns to the depot early.

The objective is to determine a set of routes and their scheduling to satisfy all demands and associated time windows with minimum total costs for the planning horizon. Total costs include travel cost (the product of total travel time and W), overtime cost (the product of total overtime and S), cost of drivers for regular time, and cost of unmet demands. We assume that each unmet demand will be served by a taxi at a transportation cost of A per hour.

The notation is summarized below:

(1) Sets and problem size parameters

P : Number of days in the planning horizon.

T : The set of shifts in the planning horizon, $T = \{1, 2, \dots, 3P\}$, and $|T| = 3P$.

D : The set of demand nodes.

V : $DU \{n+1, \dots, n+t, \dots, n+|T|+1\}$.

n : Total number of demand nodes in the planning horizon, $n = |D|$.

K : The set of vehicles, defined a priori or determined by the model.

(2) Time parameters

t_{ij} : Travel time from node i to node j .

L : Shift length (e.g. 8 hours).

\bar{L} : Overtime limit for each shift imposed by company policy.

B_t : Begin time of shift t , $B_t = L(t-1)$, $t \in T$.

E_t : End time of shift t , $E_t = Lt$, $t \in T$.

$[e_i, l_i]$: The service time window of node i , $i \in V$.

(3) Cost parameters

W : Traveling cost per hour (e.g., gas).

R : Regular salary rate for drivers.

F : Driver cost of using a vehicle in a shift, $F=LR$.

S : Overtime salary rate for drivers.

A : Cost per hour for using a taxi to serve each unmet demand.

(4) Decision variables:

$x_{ij}^k = 1$ if vehicle k travels from node i to node j , and 0 otherwise;

$y_t^k = 1$ if vehicle k is used in shift t , and 0 otherwise;

$u_i = 1$ if demand i is served by taxi, and 0 otherwise;

$w_i^k =$ the time at which node i is visited by vehicle k , $w_i^k \geq 0$.

The problem can be formulated as the following mixed integer program (MIP):

$$\text{Minimize } W \sum_{k \in K} \sum_{(i,j) \in E} t_{ij} x_{ij}^k + S \sum_{k \in K} \sum_{t \in T \setminus \{T+1\}} (w_{n+t+1}^k - E_t) + F \sum_{k \in K} \sum_{t \in T} y_t^k + A \sum_{i \in D} u_i (t_{0i} + t_{i0})$$

Subject to

$$\sum_{k \in K} \sum_{(i,j) \in E} x_{ij}^k + u_i = 1, \forall i \in D \quad (1)$$

$$\sum_{(n+t,i) \in E} x_{n+t,i}^k = 1, \forall t \in T, k \in K \quad (2)$$

$$\sum_{(i,n+t) \in E} x_{i,n+t}^k = 1, \forall t \in \{2, \dots, |T|+1\}, k \in K \quad (3)$$

$$\sum_{(j,i) \in E} x_{ji}^k - \sum_{(i,j) \in E} x_{ij}^k = 0, \forall i \in V \setminus \{n+1, n+|T|+1\}, k \in K \quad (4)$$

$$w_i^k + t_{ij} \leq w_j^k + M(1 - x_{ij}^k), \forall i \in V, j \in V, k \in K \quad (5)$$

$$e_i \leq w_i^k \leq l_i, \forall i \in D, k \in K \quad (6)$$

$$E_t \leq w_{n+t+1}^k \leq E_t + \bar{L}, \forall k \in K, t \in T \quad (7)$$

$$y_t^k = 1 - x_{n+t, n+t+1}^k, \forall k \in K, t \in T \quad (8)$$

In the objective function, the four cost terms are travel, overtime, regular driver usage, and taxi respectively. Constraints 1 ensure that each demand is served exactly once, either by internal vehicles or by a taxi. Constraints 2 and 3 ensure that every vehicle will visit all the depot nodes $n+t$, $t \in \{1, \dots, |T|+1\}$ in the planning horizon. Depot node $n+t$ must be visited before $n+t+1$ because their time windows are non-overlapping. The depot nodes divide demands visited by a vehicle into different shifts. All demands visited between node $n+t$ and $n+t+1$ are served in shift t . If no demand is visited between them for a vehicle, then this vehicle is not used and the route for shift t is empty. These constraints also ensure that every route in a shift starts from the origin depot and ends at the destination depot. Constraints 4 ensure that for all nodes except $n+1$ and $n+|T|+1$, the inflow equals to the outflow. Constraints 5 force consistencies of time variables, which are also subtour elimination constraints. M is an upper bound on $l_i + t_{ij}$, $\forall i \in V, (i, j) \in E$. Constraints 6 are time window constraints for the demand nodes, and constraints 7 are time windows constraints for the depot nodes. Constraints 8 calculate whether vehicle k is used in shift t or not. If $x_{n+t, n+t+1}^k = 1$, then no demand is served by vehicle k in shift t . Therefore, the vehicle is not used and $y_t^k = 0$. Otherwise, at least one customer is served by vehicle k , so $y_t^k = 1$.

3. Heuristic Algorithms

In this section, we first review recent relevant research on heuristics for vehicle routing problems. We then present the building blocks of our heuristics, followed by two heuristic algorithms, SI and SD, for solving our problem, and an improvement phase.

In the solution algorithm, we use an insertion-based heuristic to generate the initial solution and then use a tabu search algorithm to improve it. Insertion heuristics are effective solution methods for VRP. Recent works include Diana and Dessouky (2004), who present a parallel regret insertion heuristic to solve a large-scale dial-a-ride problem with time windows. The computational results show the effectiveness of this approach in terms of trading-off solution quality and computational times. Lu and Dessouky (2006) present an insertion-based construction heuristic to solve the multi-vehicle pickup and delivery problem with time windows.

This heuristic considers not only the classical incremental distance measure in the insertion evaluation criteria but also the cost of reducing the time window slack due to the insertion. Tabu search has also been applied to major variants of VRP, e.g. VRP with soft time windows (Taillard et al. 1997), VRP with split delivery (Archetti et al. 2006), and the pick-up and delivery problem (Bianchessi and Righini 2006). It is shown that tabu search generally yields very good results on a set of benchmark problems and some larger instances (Gendreau et al. 2002).

3.1 The Building Blocks

3.1.1 The Insertion Routine

First we describe the Insertion Routine (Algorithm 1 in the Appendix), which greedily inserts the customer with the cheapest cost in the routes. There are four associated costs (C_1 - C_4) in the algorithm. Similar to Lu and Dessouky (2006), we consider the cost of reducing the time window slack due to the insertion, which is denoted as C_1 . The costs C_2 , C_3 , and C_4 are respectively the overtime cost, the regular driver usage cost, and the cost of unmet demands. The insertion algorithm tries to insert as many customers as possible. The insertion cost of a new customer is the total weighted increase of C_1 , C_2 , and C_3 , while C_4 is calculated after the algorithm is finished. In terms of the weights of these costs, we assume C_4 is the highest; that is, we try to serve as many demands as possible with the internal vehicles. The weights of C_1 , C_2 and C_3 are W , S , and F respectively.

Whenever a new vehicle is used, a driver cost of $F=LR$ is incurred. Even if the vehicle leaves the depot after the shift start time, the salary is always F because the driver is available at the shift start time. If the working time is longer than L hours, there will be an overtime cost. After the routes are created, we calculate the cost of unmet demand and add it to the total cost. We assume that each unmet demand needs a separate taxi, so the cost for an unmet demand i is $A(t_{0i} + t_{i0})$. The total unmet demand cost is the sum of those costs.

The insertion algorithm is a parallel insertion, in which all routes are constructed simultaneously with initial seeds used to begin the route. If a pair of nodes i and j cannot be served by the same vehicle, i.e., $e_i + t_{ij} > l_j$ and $e_j + t_{ji} > l_i$, then the two nodes are called an *incompatible pair*. For

each shift, we construct a *compatibility graph* by taking the nodes to be served in the shift, and creating an edge between any incompatible pair. Then we solve a maximum clique problem approximately using a greedy algorithm on the graph (Lu and Dessouky, 2006). The seeds are the nodes that form the max clique. Let Z_t denote the set of seeds in shift t . Since each pair in Z_t is incompatible, every node must use a different vehicle. Thus the minimum number of vehicles used to serve the nodes in the shift is $|Z_t|$ if all nodes are served by internal vehicles. The role of the seeds is two fold. First, it helps to create balanced routes. Because of the high cost of a new route, without seeds, the insertion algorithm tends to fulfill a route and then initiates another when the first is full. Thus the last route usually has few customers. Second, using seeds to guide the construction of the routes avoids the late insertion of incompatible demand nodes, which can lead to higher costs or infeasibilities.

3.1.2 Intra-shift Tabu Search Algorithm

We use an intra-shift tabu search algorithm (Algorithm 2 in the Appendix) to improve the routes for each shift. This implementation of the tabu search considers the neighborhoods obtained from the standard 2-opt exchange move (Lin, 1965) and the λ -interchange move (Osman, 1993). The algorithm evaluates solutions based on the objective function, i.e. the sum of the travel cost, overtime cost, cost of unmet demands and driver cost. At each iteration the tabu search generates α_{\max} λ -interchange neighbors and β_{\max} 2-opt neighbors of the current solution. The tabu search at each iteration moves to the best neighbor, even if it is worse than the current best solution. The prevention of cycling is achieved by forbidding some moves for a given number of iterations θ . In our implementation, the number of tabu iterations is randomly generated in $(\theta_{\min}, \theta_{\max})$. For a λ -interchange move, feasible moves of a solution consider that up to 2 nodes are exchanged between two routes of the solution. The tabu status is overridden if the new solution is better than the best solution so far and the algorithm terminates if there is no improvement in N_{\max} iterations.

3.1.3 Inter-shift Tabu Search Algorithm

For algorithm SD, apart from the intra-shift tabu search algorithm (Algorithm 2), an inter-shift tabu search algorithm (Algorithm 3 in the Appendix) is implemented. There are two types of

moves that can possibly reduce the total cost. The first type is moving a node from the previous shift to a non-empty route in the current shift. Because of the high fixed route cost, the insertion algorithm (Algorithm 1) tries to insert as many as possible demands in the existing routes. As a result, some demands may be served with unnecessary high overtime cost. In such a move, if the decrease in overtime is larger than the increase in travel cost, then it is made. The second type of move is rescheduling a node from its current shift to the previous shift. The rationale is that increasing overtime might cause a larger saving in the travel cost or new driver cost.

3.2 Heuristics

3.2.1 Shift Independent (SI) Algorithm

The simplest way to tackle the multi-shift VRP with overtime is to treat each shift independently. Because of the limit in fleet size, we need to consider the availability of vehicles. That is, we can only start using a vehicle in the current shift after it returns to the depot from the previous shift. Specifically, for vehicle k in shift t , the earliest time it is available is $\max(w_{n+t}^k, E_{t-1})$.

In the SI algorithm (Algorithm 4 in the Appendix), we avoid using overtime as much as possible. If a demand can be served in either the current shift or previous shift, it is served in the current shift to avoid overtime. To do this, we first classify all demands D into different shifts D_t , $t \in T$, as shown in Figure 1. In particular, for demand i that occurs near shift changes, if $B_t + t_{0i} \leq l_i$ and $e_i + t_{0i} \leq B_t + \bar{L}$ (i.e., it can be served in either shift t or shift $t-1$), then we place the demand in D_t . Otherwise if $B_t + t_{0i} > l_i$, then it can only be served in shift $t-1$. So we place it in D_{t-1} . For example, suppose a demand i has time window $[B_t, B_t+2]$. If $t_{0i} \leq 2$ and $t_{0i} \leq \bar{L}$, then it can be served in shift t or shift $t-1$, and we place it in D_t . On the other hand, if $t_{0i} > 2$, then it has to be served in shift $t-1$ using overtime, and we place it in D_{t-1} .

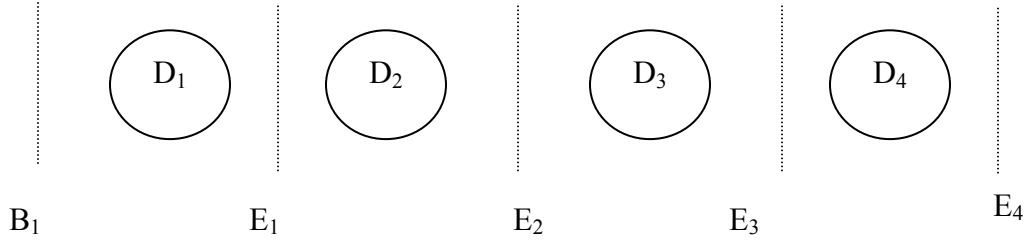


Figure 1: The classified demand nodes in SI algorithm

Each D_t is scheduled independently only considering the return time of the vehicles of the previous shift. As mentioned before, we have seeds for each shift. There are a lot of options to match the seeds of the current shift with the routes of the previous shift. One reasonable way is to match seeds with the routes that have the earliest available times for the next shift. In this way, the routes initiated by the seeds have more time available to serve the unscheduled demands, leading to a higher utilization level of the used vehicles. This is implemented by first ranking the routes in the previous shift in increasing order of w_{n+t}^k , and then inserting the $|Z_t|$ seeds in the first $|Z_t|$ routes. Because we consider each shift independently, only the intra-shift tabu search algorithm (Algorithm 2) is executed.

3.2.2 Shift Dependent (SD) Algorithm

In the SD algorithm (Algorithm 5 in the Appendix), as opposed to the SI algorithm, a demand can be served in either the current shift or the previous shift, as long as it is feasible. Both intra-shift tabu search algorithm (Algorithm 2) and inter-shift tabu search algorithm (Algorithm 3) are executed. The method to match seeds is also to place them in the routes with the earliest available time. It also classifies the demands into each shift, and then it schedules the demands shift by shift sequentially. However, the classification method is different. In SD, the classification differentiates demands that can be served in both shifts and those that can only be served in one shift. If two nodes cannot be served in the same shift, they are called *non-adjacent*. To identify all non-adjacent nodes, we classify all demands nodes into $2T-1$ disjoint sets. There are T sets called single shift (SS) demands, i.e. SS_t contains nodes that can only be served in shift t , $t \in T$. There are $T-1$ sets called inter-shift (IS) demands. That is, IS_t contains nodes that can be served either in shift t or shift $t+1$, $t \in \{1, \dots, |T|-1\}$. Starting from shift 1, for node i , if

$E_1 + t_{i0} > l_i$, then it can only be served in shift 1, and it is placed in SS_1 . If $E_1 + t_{i0} \leq l_i$ and $e_i + t_{i0} \leq E_1 + \bar{L}$, then it can be served either in shift 1 or shift 2, and it is placed in IS_1 . If $e_i + t_{i0} > E_1 + \bar{L}$ and $E_2 + t_{i0} > l_i$, then it can only be served in shift 2, and it is put in SS_2 . In this way, we can classify all demand nodes into SS or IS. Figure 2 illustrates the classified demand sets for $T=4$. We can see that $D_t = IS_{t-1} \cup SS_t$, $t \in \{2, \dots, |T|\}$. For shift t , SD first schedules demands in SS_t , and then schedules demands in IS_{t-1} .

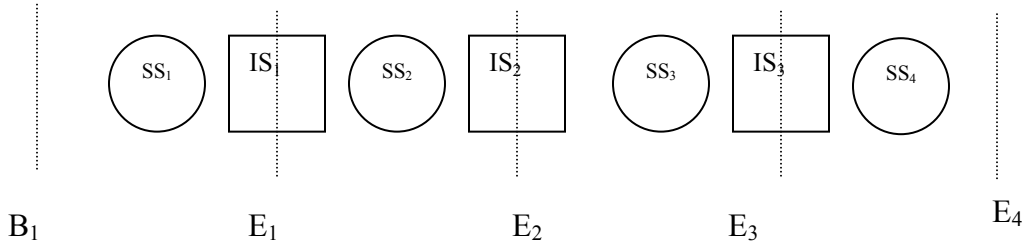


Figure 2: The classified demand nodes in SD algorithm

3.2.3 The Post Improvement (PI) Algorithm

This Post Improvement (PI) algorithm can reduce overtime by re-matching the routes in adjacent shifts without changing the travel time and sequence of nodes in each shift. The following is a simple example. Let r_t^k denote the route of vehicle k in shift t . Suppose for route r_{t-1}^k , $e_{t-1}^k = B_t + 1$, and for r_t^k , $e_t^k = B_{t+1} + 1$. In addition, assume that $w_i^k - e_i \geq 1$ for all $i \in r_t^k$. That is, all demands of r_t^k can be served 1 hour earlier. For route r_{t-1}^l , $e_{t-1}^l = B_t$, and for r_t^l , $e_t^l = B_{t+1} - 1$. In addition, assume $l_i - w_i^l \geq 1$ for all $i \in r_t^l$. That is, all demands of r_t^l can be served 1 hour later. Now we have an overtime of 2 hours for the four routes. But if we swap the second part of the routes of vehicles k and l (including all routes from t to $|T|$), we can obtain an overtime of only 1 hour for the four routes.

The PI is implemented by solving a series of minimum cost network flow problems, in particular, assignment problems. Because overtime cost in the current shift depends on the routes of the previous shifts, the algorithm is implemented sequentially from shift 1 to $|T|-1$ (Algorithm 6 in the Appendix). The sub-problem for every two consecutive shifts is a bipartite assignment

problem, which is solved to optimality using the Hungarian algorithm (Kuhn, 1955). In the sub-problem of shifts t and $t+1$, the nodes are the routes from 1 to t and the routes from t to $|T|$ of vehicle $k, \forall k \in K$. The arc cost c_{kl} is the overtime cost of matching the routes from 1 to t of vehicle k with the routes from $t+1$ to $|T|$ of vehicle $l, \forall k, l \in K$. The graph constructed is bipartite. We want to find the matching that has the minimum total overtime cost. We need to solve the sub-problem $|T|-1$ times. After SI and SD, PI is called to reduce the overtime cost. Then, the final routes are outputted and the associated costs in travel time, regular driver usage, overtime, and unmet demand are calculated.

4. Lower Bound

To estimate the performance of the SD algorithm relative to the optimal solution, we present a method to obtain a lower bound of the objective value. It is obtained by solving the LP relaxation of the MIP model of Section 2.1 with several types of cuts, which take into account the special structure of the problem. According to Cordeau et al. (2002), the LP relaxation of the VRPTW model often provides very loose lower bounds. To show this, the authors show a procedure to find near-optimal solutions in which time constraints are inactive. We need to add some cuts to increase the lower bound. In general the tighter the time window, the tighter the lower bound.

First of all, we restrict the definition of variables to only feasible ones. Recall that all demands can be classified into *SS* and *IS* (Figure 2). Although the classification is done only for SD, it exists for the data set, independently of any algorithm. Variables x_{ij}^k are only defined for pairs (i, j) of adjacent nodes; otherwise there are one or more depot nodes between them, thus $x_{ij}^k = 0$. In addition, x_{ij}^k are only defined for edges (i, j) for which it is feasible to visit j after i , i.e. $e_i + t_{ij} - l_j < 0$.

4.1 Minimum Number of Required Routes (MNRR)

In this section, we have four propositions to bound the minimum number of required routes. In each shift t , the minimum number of routes to be used is the size of the max clique in the compatibility graph induced by SS_t if there is no unmet demand. As before, the clique is obtained

by solving a maximum clique problem approximately using a greedy algorithm (Lu and Dessouky, 2006). Recall that Z_t is the set of nodes forming the clique obtained from SS_t .

Proposition 1: The number of routes used for each shift satisfies the following inequalities:

$$\sum_{k \in K} \sum_{i \in D} x_{n+t,i}^k \geq |Z_t| - \sum_{i \in Z_t} u_i, \forall t \in T;$$

$$\sum_{k \in K} \sum_{i \in D} x_{i,n+t}^k \geq |Z_t| - \sum_{i \in Z_t} u_i, \forall t \in \{2 \dots |T| + 1\}.$$

Proof: Since every pair of nodes in Z_t is incompatible, at least $|Z_t|$ vehicles have to be used in shift t if they are all served by the internal vehicles. If any one of the demands is served by a taxi, we need to decrease the size of the max clique by one. Therefore in shift t at least $|Z_t| - \sum_{i \in H_t} u_i$ vehicles have to travel from the origin depot node $n+t$ to a demand node, and similarly at least $|Z_t| - \sum_{i \in H_t} u_i$ vehicles have to return to the destination depot node $n+t+1$ from a demand node.

□

We extend the notion of incompatible pair to incompatible triple. If three nodes cannot be served by one vehicle, then they are called an *incompatible triple*. We just need to check whether all six possible permutations of the three nodes are infeasible to know whether they are incompatible or not. For example, for three nodes i, j and p , if $e_i + t_{ij} > l_j$ or $\max(e_i + t_{ij}, e_j) + t_{jp} > l_p$, then the permutation (i, j, p) is infeasible. Thus it takes $O(|SS_t|^3)$ computation time to identify all incompatible triples for shift t .

Proposition 2: Let IT_t be the set of all incompatible triples in SS_t , we have

$$\sum_{k \in K} \sum_{i \in D} x_{n+t,i}^k \geq 2 - \sum_{i \in P} u_i, \forall t \in T, P \in IT_t;$$

$$\sum_{k \in K} \sum_{i \in D} x_{i,n+t}^k \geq 2 - \sum_{i \in P} u_i, \forall t \in \{2 \dots |T| + 1\}, P \in IT_t.$$

Proof: For an incompatible triple P in SS_t , at least two vehicles are needed to serve the three nodes in P if all three nodes are served by internal vehicles. The route number is non-decreasing

if we serve more nodes by the two vehicles. Considering that some nodes might be served by taxi, we have Proposition 2. \square

Proposition 3: In the compatibility graph, suppose there exists four different nodes (i, p, q, l) in SS_t , such that node i is incident to nodes $p, q,$ and l , and (p, q, l) is an incompatible triple (Figure 3). Let TE_t be the set of all such tetrads in SS_t , we have:

$$\sum_{k \in K} \sum_{i \in D} x_{n+t,i}^k \geq 3 - \sum_{i \in P} u_i, \forall t \in T, P \in TE_t;$$

$$\sum_{k \in K} \sum_{i \in D} x_{i,n+t}^k \geq 3 - \sum_{i \in P} u_i, \forall t \in \{2 \dots |T| + 1\}, P \in TE_t.$$

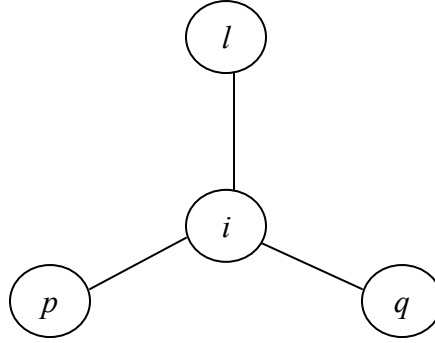


Figure 3: The node pattern for Proposition 3

Proof: Node i needs one vehicle k_1 , and nodes $p, q,$ and l , cannot be served by k_1 since they are incident to node i . Since (p, q, l) is an incompatible triple, they must be served by at least two other vehicles k_2 and k_3 . Therefore, at least three vehicles are needed if they are all served by internal vehicles. Considering the case of using taxi, we have *proposition 3*. \square

Proposition 4: Suppose there exists five different nodes (i, j, p, q, l) in SS_t , such that i is incident to p and j , and j is incident to q , and (l, p, j) and (l, i, q) are both incompatible triples (Figure 4). Let PE_t be the set of all such pentads in SS_t , we have:

$$\sum_{k \in K} \sum_{i \in D} x_{n+t,i}^k \geq 3 - \sum_{i \in P} u_i, \forall t \in T, P \in PE_t;$$

$$\sum_{k \in K} \sum_{i \in D} x_{i,n+t}^k \geq 3 - \sum_{i \in P} u_i, \forall t \in \{2 \dots |T| + 1\}, P \in PE_t.$$

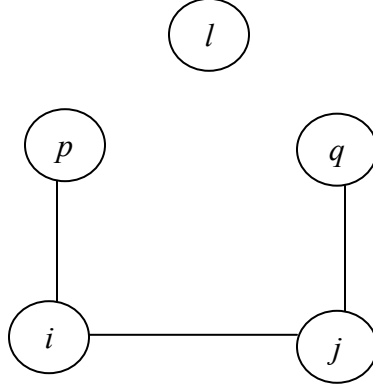


Figure 4: The node pattern for Proposition 4

Proof: First we assume that all nodes in SS_t are served by internal vehicles. Since nodes i and j are incident, at least two vehicles are needed. Suppose i and j are served by vehicles k_1 and k_2 respectively. Suppose two vehicles can serve all 5 nodes, then p must be served by k_2 , q must be served by k_1 , and l must be served by k_1 or k_2 . However, since both (l, p, j) and (l, i, q) are both incompatible triples, l cannot be served by either k_1 or k_2 . There is a contradiction. Thus there has to be a third vehicle k_3 to serve all of them. Considering the case of using taxi, we have *proposition 4*. \square

A straightforward implementation to find all pentads that satisfies proposition 4 is to check every pentad in SS_t . The computation time is $O(|SS_t|^4)$ for shift t . Similarly, the computation time of proposition 5 is $O(|SS_t|^5)$ for shift t .

4.2 No Incompatible Nodes Served by the Same Vehicle (NINSSV)

Proposition 5 (No Incompatible Pairs Served by the Same Vehicle): If nodes i and j is an incompatible pair, then the following inequalities are valid:

$$\sum_{(i,p) \in E} x_{ip}^k + \sum_{(j,p) \in E} x_{jp}^k \leq 1, \quad k \in K.$$

Proof: If nodes i and j are incompatible, then no vehicle k can serve both of them in a feasible solution. That is, $\sum_{(i,p) \in E} x_{ip}^k$ and $\sum_{(j,p) \in E} x_{jp}^k$ cannot be both 1. Therefore $\sum_{(i,p) \in E} x_{ip}^k + \sum_{(j,p) \in E} x_{jp}^k \leq 1$. \square

4.3 No Two-node Cycles (NTC)

Proposition 6 (No Two-node Cycles): The following inequalities are valid:

$$x_{ij}^k + x_{ji}^k \leq 1, \forall (i, j) \in E, k \in K.$$

Proof: Since there are no two-node cycles in a feasible solution, for any two nodes i and j , at most one of x_{ij}^k and x_{ji}^k can be 1. Thus $x_{ij}^k + x_{ji}^k \leq 1$. \square

4.4 Minimum Overtime Required (MOR)

Proposition 7 (Minimum Overtime Required): The following inequality is valid:

$$\sum_{k \in K} w_{n+t+1}^k \geq LK(t-1) + (e_i + t_{i0} - E_t)(1 - u_i), \forall t \in \{1 \dots T\}, i \in SS_t.$$

Proof: For node i in SS_t , $\forall t \in \{2 \dots T+1\}$, the minimum overtime of the node is $\max(e_i + t_{i0} - E_t, 0)(1 - u_i)$. That is, if $u_i = 0$, the earliest time the vehicle serving node i returns to the depot is $e_i + t_{i0}$, with a minimum overtime of $\max(e_i + t_{i0} - E_t, 0)$. If $u_i = 1$, then there is no overtime associated with this node. For each node i in SS_t , since the earliest start time of depot node $n+t$ is $B_t = L(t-1)$, we have

$$\sum_{k \in K} w_{n+t+1}^k \geq LK(t-1) + \max(e_i + t_{i0} - E_t, 0)(1 - u_i) \geq LK(t-1) + (e_i + t_{i0} - E_t)(1 - u_i). \square$$

Please note that to get a lower bound, these valid inequalities are all included from the beginning of the formulation. On the other hand, for each proposition defining the MNRR constraints, it is convenient to include only one constraint. The other one is redundant because if vehicle k in shift t is used, it must go from the origin depot node to a demand node and from a demand node to the destination node, i.e. $\sum_{i \in D} x_{n+t,i}^k = \sum_{i \in D} x_{i,n+t+1}^k = 1$. If vehicle k is not used, then

$$\sum_{i \in D} x_{n+t,i}^k = \sum_{i \in D} x_{i,n+t+1}^k = 0. \text{ Thus, } \sum_{k \in K} \sum_{i \in D} x_{n+t,i}^k = \sum_{k \in K} \sum_{i \in D} x_{i,n+t+1}^k.$$

5. Experimental Results

We randomly generate customers lying on a 2-dimensional Euclidean plane (a 200 minute*200 minute square) and then construct the travel time matrix. Problems generated with this procedure will have a symmetric travel time matrix, which obeys the triangular inequality. The depot is located at the center of the square, (100, 100).

We consider a time horizon of a week (5 working days), and each shift has 8 hours. There are a total of 15 shifts. Note that we evaluate the results in a steady state environment by solving a larger instance and removing several “warm-up” shifts and termination shifts. In the steady state, the customers that occur early in the current shift may be served by the vehicles of the previous shift, and the customers that occur late in the shift may be served by the next shift. The first shift and the last shift do not meet both requirements. It may take several shifts to enter or leave the steady state. Thus, we generate and solve an instance of 21 shifts, but only evaluate the middle 15 shifts (shifts 4-18). The earliest start time (e_i) is generated randomly in the interval (0, 120), and the latest start time (l_i) is $e_i + 2$ since we assume the base time window length is 2 hours. We also consider two problem classes where time window length is 4 hours.

We consider 4 problem classes: uniform demands (TW=2 hours), clustered uniform demands (TW=2 hours), clustered uniform demands (TW=4 hours), and clustered non-uniform demands (TW=4 hours). The problem classes are named according to the spatial (or geographical) distribution of customers and their arrival rate. Thus, “uniform demands” means uniform spatial distribution and uniform arrival rate; “clustered uniform demands” means clustered spatial distribution and uniform arrival rate; “clustered non-uniform demands” means clustered spatial distribution and non uniform arrival rate. For each problem class, we consider three uniform demand rates per shift: $Q=30, 60$ and 120 . For each case, the number of vehicles is chosen such that it is just enough to serve all the demands or almost all the demands for the SI algorithm. We generate 10 instances in the same manner and report the average results over the 10 instances. Note that the number of vehicles does not increase linearly with the demand rate because of the aggregate effect, i.e., each vehicle can serve more customers when the demand rate is high.

5.1 Parameter Setting and Tuning

The values of the cost parameters are set to real world costs. W is obtained by considering the fuel cost and fuel mileage of a standard shuttle vehicle. R is based on \$15/hour. Overtime salary rate is 1.5 times the regular salary rate. Taxi cost rate is based on a typical fare in Los Angeles County.

We do some experiments to tune the parameters for tabu search for the case of 30 demands/shift for uniform demands. Similar results are observed for other cases, and they are not reported for space consideration. We first show sensitivity results for N_{\max} , α_{\max} , and β_{\max} in Table 1. In the table, the columns “Travel”, “OT”, “Unmet Demand” and “Route” are travel cost, overtime cost, unmet demand cost, and route cost, respectively. “Total” is the total cost. “CPU Time” is the CPU time in seconds. We note that as we increase the parameters, we observe a smaller improvement in solution quality, however solution time increases steadily. The best solution is 14802 when $N_{\max}=2000$, $\alpha_{\max}=1000$, $\beta_{\max}=1000$, however the CPU time is more than an hour. When $N_{\max}=800$, $\alpha_{\max}=400$, $\beta_{\max}=400$, the solution is within 0.72% of the best solution, and the CPU time is only about 11 minutes. Hence, we use $N_{\max}=800$, $\alpha_{\max}=400$, $\beta_{\max}=400$ in later experiments.

Table 1: Parameter tuning for tabu search ($\theta_{\min}=10$, $\theta_{\max}=20$) based on 10 instances

N_{\max}	α_{\max}	β_{\max}	Travel	OT	Unmet Demand	Route	Total	CPU Time
100	50	50	6984	2092	16	6768	15861	18
200	100	100	6743	1880	16	6828	15467	53
400	200	200	6613	1768	33	6684	15098	186
800	400	400	6594	1615	16	6684	14909	675
1000	500	500	6565	1569	33	6684	14851	1027
2000	1000	1000	6555	1518	33	6696	14802	3882
3000	1500	1500	6573	1525	33	6696	14827	8627
4000	2000	2000	6554	1558	33	6708	14853	15525

We show sensitivity results for θ_{\min} and θ_{\max} in Table 2. We can see that the changes in CPU time are not significant and the best solution is obtained when $\theta_{\min}=10$ and $\theta_{\max}=20$ so we use these values in later experiments. When the tabu tenure is high, the solution converges faster, but the solution quality deteriorates.

Table 2: Sensitivity analysis of θ_{\min} and θ_{\max} ($N_{\max}=800$, $\alpha_{\max}=400$, $\beta_{\max}=400$) based on 10 instances

θ_{\min}	θ_{\max}	Travel	OT	Unmet Demand	Route	Total	CPU Time
5	10	6591	1545	33	6816	14985	741
10	20	6594	1615	16	6684	14909	675
20	40	6585	1687	33	6672	14977	656
40	80	6612	1771	33	6552	14968	630
80	160	6636	1784	29	6720	15169	616

The parameter values in the problem instances and algorithms are summarized below.

(1) Problem size and time parameters

P : 5 days

$|T|$: 15

L : 8 hours

\bar{L} : 4 hours

(2) Cost parameters

W : \$17.5/hour

R : \$15/hour

F : \$15*8=\$120

S : \$22.5/h (1.5 times of the regular salary rate)

A : \$40/hour

(3) The parameter values in the tabu search algorithms (Algorithms 2 and 3)

$$N_{\max} : 800$$

$$\alpha_{\max} : 400$$

$$\beta_{\max} : 400$$

$$\theta_{\min} : 10$$

$$\theta_{\max} : 20$$

All the experiments are performed on a Dell Precision 670 computer with a 3.2 GHz Intel Xeon Processor and 2 GB RAM running Red Hat Linux 9.0. The largest instance could be solved in about an hour of CPU time. For a typical instance of the problem class that takes the longest running time (the third case in Table 5 solved by SD), the total running time is 1 hour, the time used for constructing the routes is 16 minutes, and the time used for tabu search is 44 minutes. There are 45 tabu searches because SD algorithms calls 3 tabu searches in each iteration, so on average each tabu search takes 1 minute.

5.2. Uniform Demands

For uniform demands, we consider three cases: 30 demands/shift to be served by 7 vehicles, 60 demands/shift by 10 vehicles, and 120 demands/shift by 16 vehicles. The results are shown in Table 3, where the values are the ratios of SD/SI for the four cost measures, the total cost and the CPU time. In the table, “Route length” is the ratio of the average route length, where route length is defined as the sum of shift length L and overtime of the route. The ratio of route length r is an

indicator of overtime because for a problem instance, $r = \frac{L + \overline{OT}_{SD}}{L + \overline{OT}_{SI}}$, where \overline{OT}_{SD} is the average

overtime of the routes obtained by the SD algorithm, and \overline{OT}_{SI} is the average overtime of the

routes obtained by SI. Then we have $L + \overline{OT}_{SD} = r(L + \overline{OT}_{SI})$, i.e., $\overline{OT}_{SD} = r\overline{OT}_{SI} + (r-1)L$. It

can be seen that larger r corresponds to larger \overline{OT}_{SD} . In the column “CPU Time”, we give CPU times of both SD and SI, where the first value is for SD, and the second is for SI.

Table 3: The ratio of SD/SI for uniform demand

Cases	Travel	Route length	Unmet Demand	Route	Total	CPU Time SD/SI
450 customers, 7 vehicles	0.96	1.10	0.33	0.82	0.93	675/470
900 customers, 10 vehicles	0.97	1.11	0.38	0.79	0.93	1184/1034
1800 customers, 16 vehicles	0.96	1.11	0.50	0.78	0.93	2122/2133

From the results, we can see that SD outperforms SI in terms of total cost with a 7% saving. SD uses a little more overtime to get savings in travel cost, unmet demands, and cost of routes. The percentage savings in unmet demand is most significant, meaning that with the same number of vehicles, SD can serve more customers than SI. The savings in route cost are also significant (at least 18%). In terms of solution time, SI is faster than SD. But the CPU time of SD is not a problem since we are planning for a week.

5.3 Clustered Uniform Demands

We investigate the effect of geographical distribution of customers on the performance of the algorithms by considering clustered uniform demands. The method we use to generate clustered demands is similar to Sungur et al. (2008). We randomly generate customer locations in 5 different clusters with identical radius of 25. We center each cluster at a random location with a distance of 50 from the depot. The results are shown in Table 4. We consider the same demand quantity as before, but the fleet size is smaller because a vehicle can serve more customers for clustered demand.

Table 4: The ratio of SD/SI for clustered uniform demand

Cases	Travel	Route length	Unmet Demand	Route	Total	CPU Time SD/SI
450 customers, 5 vehicles	0.97	1.09	1.00	0.72	0.87	695/566
900 customers, 8 vehicles	0.98	1.11	1.00	0.70	0.88	1482/1199
1800 customers, 12 vehicles	0.99	1.15	1.00	0.69	0.90	2997/2229

From Table 4 we can see that the savings in total cost are larger than for uniform demands. The savings are about 10%. The solutions in general use more overtime than in the case of uniform demand, and the savings in route cost are larger. The reason is that in SD when the demands are clustered, the vehicles prefer to stay longer in a cluster to serve more customers of the next shift,

leading to more overtime cost and more reduction in the number of routes. However, the savings in travel cost are small. The reason is that in SD, there are savings from fewer trips from and to the depot; on the other hand, since there are fewer vehicles used, some demands are served with longer travel distance. The two effects cancel out and the total travel distance is almost the same. Note that the ratio of 1.00 of unmet demand means that there are no unmet demands for both algorithms. The solution time is longer than the previous problem class since there are more feasible moves when the demands are clustered.

5.4 Clustered Uniform Demands with Relaxed Time Windows

Now suppose that the demands are uniform and the time windows are all 4 hours. The results are shown in Table 5. The savings are even larger in this case, at about 22%. Compared with the smaller time windows, the routes use more overtime. The savings in route cost and travel cost are larger. The reason is that in SD, with longer time windows, the vehicles can serve more requests on their return trips to the depot, incurring more overtime cost but larger savings in travel distance and route cost. The solution time is longer than previous problem classes since there are more feasible moves with wider time windows.

Table 5: The ratio of SD/SI for clustered demand (time window 4 hours)

Cases	Travel	Route length	Unmet Demand	Route	Total	CPU Time SD/SI
450 customers, 5 vehicles	0.94	1.17	1.00	0.58	0.80	925/625
900 customers, 8 vehicles	0.93	1.13	1.00	0.55	0.76	1896/1195
1800 customers, 12 vehicles	0.96	1.16	1.00	0.53	0.77	3571/2185

5.5 Clustered Non-uniform Demands with Relaxed Time Windows

In the real world, clustered non-uniform demands are more likely to be observed. A new set of experiments is done for this case. The time windows are all 4 hours. Generally, we have the most demand in shift 2, fewer in shift 3, and the least in shift 1 for each day. Recall that shift 1 corresponds to the night shift, shift 2 corresponds to the day shift, and shift 3 corresponds to the evening shift. In particular, we assume that the demand rate is $0.25Q$ in shift 1, Q in shift 2, and $0.5Q$ in shift 3. For example, if the demand rate is 60/shift, we have 15, 60, and 30 demands in shifts 1, 2, and 3, respectively. In the three cases, the numbers of vehicles are 4, 7, and 11

respectively. We use fewer vehicles since the total demand quantity is less than in the previous cases. The results are shown in Table 6.

Table 6: The ratio of SD/SI for non-uniform clustered demand (time window 4 hours)

Cases	Travel	Route length	Unmet Demand	Route	Total	CPU Time SD/SI
260 customers, 4 vehicles	0.94	1.11	1.00	0.59	0.78	676/546
525 customers, 7 vehicles	0.91	1.13	1.00	0.54	0.73	1262/963
1050 customers, 11 vehicles	0.93	1.16	1.00	0.52	0.74	2621/1576

From Table 6, we can see that the savings are even larger with non-uniform demands and relaxed time windows. The reason is that in SD, demands in the shift with a lower demand rate tend to be served by vehicles of the previous shift using overtime, thus reducing substantially the number of routes in the low demand shifts. Compared with case 1, in bigger instances (cases 2 and 3), more overtime is used and larger savings in every other aspect are obtained. The CPU time is shorter than the previous problem class because there are fewer customers. Note that PI is applied at the end of both SD and SI. However, the improvement is very small (on average about 0.01%) due to the effectiveness of the tabu search. It only takes 1~3 seconds, which are included in the CPU time of SD and SI.

5.6. Effect of the Algorithms on the Number of Vehicles

In this section we study the effect of the algorithms on the minimum number of vehicles required to serve all customers. We only show the results for uniform demands. Similar results are obtained for other problem classes. Table 7 shows the minimum $|K|$ required to serve all customers for SD and SI. The data shown is also the average results over 10 instances. We can see that SD can save on the number of used vehicles. The average saving is about one vehicle. With a smaller fleet size, using SD results in less fixed vehicle and maintenance cost.

Table 7: Minimum $|K|$ required to serve all customers for uniform demand

Cases	SI	SD	SD/SI
450 customers	8.0	7.1	0.89
900 customers	10.7	9.6	0.90

1800 customers	16.5	15.4	0.93
----------------	------	------	-------------

5.7. Comparison of the Lower Bound with the Solution of SD

The CPLEX solver can only solve the case of 360 customers and 12 shifts for uniform demands. This problem (with cuts) has 104461 variables and 58055 constraints after presolve. For bigger instances, the memory usage is more than 2G and exceeds the memory of the computer. Using the CPLEX solver, the LP relaxation can be solved in at most 30 minutes of CPU time. We compare the lower bound with the solution of SD. The results are shown in Table 8. With all the cuts, the ratio of the solution of SD to the lower bound is between 1.09~1.82. Without adding the cuts described in Section 4, the ratio is 13.25~26.55, showing that the cuts are very effective. From column 4, we can see that the most important cut is the Minimum Number of Required Routes (MNRR) as described in Section 4.1. From the results, we can see that the tighter the time window, the tighter the lower bound. An important reason for the increase in the ratio is the size of the max clique which is quite small with wide time windows. Hence in the lower bound, the cut “minimum route number required” becomes loose, leading to much less route cost than in the algorithm.

Table 8: Comparison of SD and LB

Problem Classes	Problem size	SD/LB (no cut)	SD/LB (with cut MNRR)	SD/LB (with all cuts)
TW 60	360 customers, 10 vehicles	20.76	1.95	1.26
TW 60, non-uniform	208 customers, 10 vehicles	19.38	1.54	1.09
TW 60, clustered, non-uniform	208 customers, 7 vehicles	26.55	1.42	1.11
TW 120	360 customers, 7 vehicles	17.36	2.24	1.48
TW 120, non-uniform	208 customers, 7 vehicles	16.54	1.86	1.29
TW 120, clustered, non-uniform	208 customers, 5 vehicles	22.56	1.69	1.33
TW 240	360 customers, 5 vehicles	14.27	2.69	1.82
TW 240, non-uniform	208 customers, 5 vehicles	13.25	2.38	1.66
TW 240, clustered, non-uniform	208 customers, 4 vehicles	18.10	2.08	1.64

To show the performance of SD over the optimal solution and investigate whether a high ratio is due to a loose lower bound, we solve a set of small instances to optimality and report the ratios

of the solution of SD over the optimal solution, and over the lower bound. The results are shown in Table 9. Note that the problem size is the largest size the CPLEX can solve.

Table 9: Comparison of SD, optimal solution, and LB for small instances

Problem classes	Problem size	SD/OPT	SD/LB
TW 60	24 customers, 4 vehicles, 3 shifts	1.00	1.12
TW 60, non-uniform	14 customers, 3 vehicles, 3 shifts	1.00	1.10
TW 60, clustered, non-uniform	14 customers, 3 vehicles, 3 shifts	1.00	1.10
TW 120	24 customers, 3 vehicles, 3 shifts	1.01	1.25
TW 120, non-uniform	14 customers, 3 vehicles, 3 shifts	1.02	1.22
TW 120, clustered, non-uniform	14 customers, 3 vehicles, 3 shifts	1.00	1.16
TW 240	16 customers, 2 vehicles, 2 shifts	1.01	1.55
TW 240, non-uniform	14 customers, 3 vehicles, 3 shifts	1.01	1.30
TW 240, clustered, non-uniform	14 customers, 3 vehicles, 3 shifts	1.02	1.18

From Table 7, we can see that the ratios of SD/OPT are close to 1, meaning that the solution of SD is near optimal. On the other hand, the ratios of SD/LB are also high, meaning that high ratio is due to a loose lower bound. There are some differences in SD/LB between small instances and large instances because in small instances the MNRR cuts are generally much tighter.

6. Conclusions and future research

This research provides some practical insights for multi-shift VRP with overtime. We provide two algorithms: SI and SD. SI is based on scheduling each shift independently, while SD allows effective use of overtime. From the experimental results, we can see that SD can provide much better solutions than SI in terms of total cost. SD also uses fewer vehicles than SI to serve all customers. If we have geographically clustered demands, the savings are larger than in the case of uniformly distributed demands. The savings are even larger for wide time windows and non-uniform demand rates. We obtain a lower bound to the problem by solving the LP relaxation problem with cuts, which shows that the solution of SD is within 1.09~1.82 times the optimal solution on the test problems. We also show that the ratio of the SD solution to LB is high when the lower bound is loose.

The results can be generalized to other situations where inter-shift dependencies should be considered. There are a lot of interesting extensions for this problem. For example, in practice some companies have mixed shift lengths, i.e., there are shorter shifts such as 4 or 5 hour shifts as well as 8 hour shifts. We can study the routing problem with all these shifts. It is also useful to determine company policies, such as overtime limit or overtime salary rate to enhance operational effectiveness or reduce total cost.

References:

- Alonso F., Alvarez M.J., and Beasley J.E., A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. *Journal of the Operational Research Society*, 59, 963—976, 2008.
- Angelelli E. and Speranza M.G., The periodic vehicle routing problem with intermediate facilities. *European Journal of Operational Research*, 137, 233-247, 2002.
- Archetti C., Hertz A., and Speranza M., A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, 40, 64–73, 2006.
- Azi N., Gendreau M., and Potvin J., An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. *European Journal of Operational Research*, 178, 755–766, 2006.
- Bianchessi N. and Righini G., Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers and Operations Research*, 34, 578–594, 2006.
- Bodin, L., Rosenfield D., and Kydes A., UCOST, a micro approach to the transit planning problem. *Journal of Urban Analysis*, 15, 47–69, 1978.
- Bodin, L., Rosenfield D., and Kydes A., Scheduling and estimation techniques for transportation planning. *Computers and Operations Research*, 8, 25–38, 1981.
- Brandão J. and Mercer A., A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European Journal of Operational Research*, 100, 180–191, 1997.
- Brandão J. and Mercer A., The Multi-Trip Vehicle Routing Problem. *The Journal of the Operational Research Society*, 49, 799-805, 1998.
- Campbell A. and Savelsbergh M., Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation Science*, 38, 369–378, 2004.

- Cordeau J.-F., Desaulniers G., Desrosiers, J., Solomon M.M., and Soumis, F., The VRP with Time Windows, *The Vehicle Routing Problem*, Chapter 7, Paolo Toth and Daniele Vigo, eds., SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 157-193, 2002.
- Diana M. and Dessouky M.M., A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B: Methodological*, 38, 539-557, 2004.
- Francis P. and Smilowitz K., Modeling techniques for periodic vehicle routing problems. *Transportation Research Part B: Methodological*, 40, 872-884, 2006.
- Hemmelmayr, V.C., Doerner, K.F., Hartl, R.F., A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195, 791-802, 2009.
- Kuhn H.W., The Hungarian Method for the assignment problem. *Naval Research Logistic Quarterly*, 2, 83-97, 1955.
- Lagodimos A.G. and Mihiotis A.N., Overtime vs. regular shift planning decisions in packing shops. *International Journal of Production Economics*, 101, 246-258, 2006.
- Lin S., Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44, 2245-2269, 1965.
- Lu Q. and Dessouky M. M., A new insertion-based construction heuristic for solving the pickup and delivery problem with hard time windows. *European Journal of Operational Research*, 175, 672-687, 2006.
- Merzifonluoğlu Y., Geunes J., and Romeijn H.E., Integrated capacity, demand, and production planning with subcontracting and overtime options. *Naval Research Logistics*, 54(4), 433-447, 2007.
- Osman I.H., Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41, 421-451, 1993.
- Petch R. J. and Salhi S., A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics*, 133(1-3), 69-92, 2003.
- Salhi S. and Petch R. J., A GA based heuristic for the vehicle routing problem with multiple trips. *Journal of Mathematical Modeling and Algorithms*, 6(4), 591—613, 2007.
- Sniezek J. and Bodin L., Cost Models for Vehicle Routing Problems. *Proceedings of the 35th Hawaii International Conference on System Sciences*, 1403-1414, 2002.

- Sungur I., Ordóñez F., and Dessouky M. M., A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty. *IIE Transactions*, 40, 509–523, 2008.
- Taillard E., Badeau P., Gendreau M., Guertin F., and Potvin J., A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31, 170–186, 1997.
- Taillard E., Laporte G., and Gendreau M., Vehicle routing with multiple use of vehicles. *Journal of the Operational Research Society*, 47, 1065–1070, 1996.
- Zäpfel G., and Bögl M., Multi-period vehicle routing and crew scheduling with outsourcing options. *International Journal of Production Economics*, Article in Press, 2008.

Appendix:

List of Algorithms:

Algorithm 1: Insertion Routine

Algorithm 2: Intra-shift Tabu Search Algorithm

Algorithm 3: Inter-shift Tabu Search Algorithm

Algorithm 4: SI

Algorithm 5: SD

Algorithm 6: PI

Algorithm 1: Insertion Routine

***Require:** Set of unscheduled demands, travel time matrix, the initial routes*

Repeat

Calculate insertion cost of possible insertion positions for all the demands

Pick the demand with the cheapest insertion cost

Insert the demand in the cheapest position

Update the routes

Remove the inserted demand from the demands

***Until** no insertion is feasible, mark the infeasible demands as unmet demands*

***return** the resulting routes and unmet demands*

Algorithm 2: Intra-shift Tabu Search Algorithm

Require: The routes of shift t

repeat

Set the best solution as the current routes

Randomly choose two routes $R1$ and $R2$ of shift t

Generate α_{\max} neighbors from λ -interchange operator

Generate β_{\max} neighbors from 2-opt operator

Choose the neighbor with the least cost and make the move

if The cost of the current routes is less than the best solution **then**

Set the best solution as the current routes

end if

Randomly generate tabu tenure θ from a uniform distribution $U(\theta_{\min}, \theta_{\max})$

if The move is λ -interchange **then**

Make moving the exchanged nodes tabu for θ iterations

else

Make removing the new arcs tabu for θ iterations

end if

Until No improvement in N_{\max} iterations

return The best solution

Algorithm 3: Inter-shift Tabu Search Algorithm

Require: The routes of shift t and $t-1$

Repeat

Set the best solution as the current routes

Randomly choose two routes $R1$ and $R2$ from the solution

For $i=1$ to α_{\max} **do**

Select a node from shift $t-1$ in $R1$, and evaluate the cost of moving the node to a random position of the shift t in $R2$

and

Select a node from shift t of $R2$, and evaluate the cost of moving the node to a random position of shift $t-1$ in $R1$

End for

Choose the neighbor with the least cost and make the move

if The cost of the current routes is less than the best solution **then**

Set the best solution as the current routes

end if

Randomly generate tabu tenure θ from uniform distribution $U(\theta_{\min}, \theta_{\max})$

Make moving the moved nodes tabu for θ iterations

Until No improvement in N_{\max} iterations

return The best solution

Algorithm 4: SI

Require: The set of demands D , travel time matrix, T , K

Classify the demands D into D_t , $t \in T$

Calculate the seeds of D_t , $t \in T$

for shift $t=1$ to $|T|$ **do**

 Insert depot node $n+t$ into route k as the origin node of shift t , $k \in K$

if $t \geq 2$ **then**

 Rank the routes according to e_{t-1}^k

end if

for $i=1$ to $|S_t|$ **do**

 Insert the i th seed of S_t in route i

end for

 Remove seeds from D_t

 Insert depot node $n+t+1$ into each route as the destination of shift t

 Call Insertion Routine (Algorithm 1) to insert D_t into routes of shift t

 Call Intra-shift Tabu Search Algorithm (Algorithm 2) for shift t

end for

Execute PI

return The routes, unmet demands, and the total cost

Algorithm 5: SD

Require: The set of demands D , travel time matrix, T , K

Classify the demands D into SS and IS

Calculate the seeds of SS_t , $t \in T$

for shift $t=1$ to $|T|$ **do**

 Insert depot node $n+t$ into route k as the origin node of shift t , $k \in K$

if ($t \geq 2$) **then**

 Rank the routes according to e_{t-1}^k

end if

for $i=1$ to $|SS_t|$ **do**

 Insert the i th seed of SS_t in routes i

end for

 Remove seeds from SS_t

 Insert depot node $n+t$ into each route as the destination node of shift t

 Call Algorithm 1 to insert demands of SS_t into the routes of shift t

if $t \geq 2$ **then**

 Call Algorithm 1 to insert demands of IS_{t-1} into the routes of shift t or $t-1$

end if

 Call Intra-shift Tabu Search Algorithm (Algorithm 2) for shift $t-1$

 Call Intra-shift Tabu Search Algorithm (Algorithm 2) for shift t

 Call Inter-shift Tabu Search Algorithm (Algorithm 3) for shift $t-1$ and t

end for

Execute PI

return The routes, unmet demands and the total cost

Algorithm 6: PI

Require: The solution of SI or SD

for $t=1$ to $|T|-1$ **do**

 Calculate the cost matrix ($|K|*|K|$ matrix) of matching routes 1 to t of vehicle k with routes $t+1$ to $|T|$ of $l, \forall k, l \in K$

 Solve an assignment problem to optimality

 Re-match the routes according to the optimal solution

end for

Calculate the total cost

Return The final routes and the total cost