A multi-level variable neighborhood search heuristic for a practical vehicle routing and driver scheduling problem

Wen, Min; Krapper, Emil; Larsen, Jesper; Stidsen, Thomas Jacob Riis

Publication date:
2009

Document Version
Publisher's PDF, also known as Version of record

A multi-level variable neighborhood search heuristic for a practical vehicle routing and driver scheduling problem

Report 9.2009

DTU Management Engineering

Min Wen
Emil Krapper
Jesper Larsen
Thomas K. Stidsen
June 2009

DTU Management Engineering
Department of Management Engineering

A multi-level variable neighborhood search heuristic for a practical vehicle routing and driver scheduling problem

Min Wen **, Emil Krapper, Jesper Larsen, and Thomas K. Stidsen

Department of Management Engineering, Technical University of Denmark,
Produktionstorvet DTU-Building 426, DK-2800 Kongens Lyngby, Denmark

November 6, 2009

Abstract. This paper addresses an integrated vehicle routing and driver scheduling problem arising at the largest fresh meat producer in Denmark. The problem consists of a one-week planning horizon, heterogeneous vehicles, and drivers with predefined work regulations. These regulations include, among other things, predefined workdays, fixed starting time, maximum weekly working duration, break rule. The objective is to minimize the total delivery cost.

The real-life case study is first introduced and modelled as a mixed integer linear program. A multi-level variable neighborhood search heuristic is then proposed for the problem. At the first level, the problem size is reduced through an aggregation procedure. At the second level, the aggregated weekly planning problem is decomposed into daily planning problems, each of which is solved by a variable neighborhood search. At the last level, the solution of the aggregated problem is expanded to that of the original problem. The method is implemented and tested on real-life data consisting of up to 2000 orders per week. Computational results show that the aggregation procedure and the decomposition strategy are very effective in solving this large scale problem, and our solutions are superior to the industrial solutions given the constraints considered in this work.

Keywords: vehicle routing, driver scheduling, Variable Neighborhood Search, node aggregation.

1 Introduction

In this paper, we consider a *Multi-Period Vehicle Routing and Crew Scheduling Problem* (MPVRCSP) arising at the largest fresh meat producer in Denmark, Danish Crown. Danish Crown delivers the fresh meat from its distribution terminals to the supermarkets all over Denmark. The supermarkets place their orders with specified demand for different days of the week before the week starts. The distributor then makes a weekly delivery plan for the drivers and vehicles so that the orders are fulfilled, the drivers' working regulations are respected and the total travel cost is minimized.

** Corresponding author (mw@transport.dtu.dk)

Danish Crown is the largest Danish producer of fresh meat. It slaughters over 20 million pigs and 0.5 million pieces of livestock each year. Its pork production is the largest in Europe and the second largest in the world. Danish Crown is also responsible for the delivery of fresh meat to supermarkets all over Denmark every day. There are two distribution terminals in Kolding and Ringsted, operating independently. Here we consider the Kolding terminal, which is located in Jutland. It receives more than 2000 orders every week and delivers meat to over 800 Danish supermarkets across western Denmark. Figure 1 shows the locations of the customers and of the terminal. The total amount of meat delivered varies from day to day, ranging from 60 tons to 300 tons. As a result, the need for drivers varies daily as well. Danish Crown has a number of internal drivers who work on regular workdays with a fixed number of hours every week. For busier days, Danish Crown needs to hire external drivers to take the routes that can not be covered by the limited number of internal drivers. Different types of drivers are associated with different costs, the objective is to minimize the total cost of the delivery.

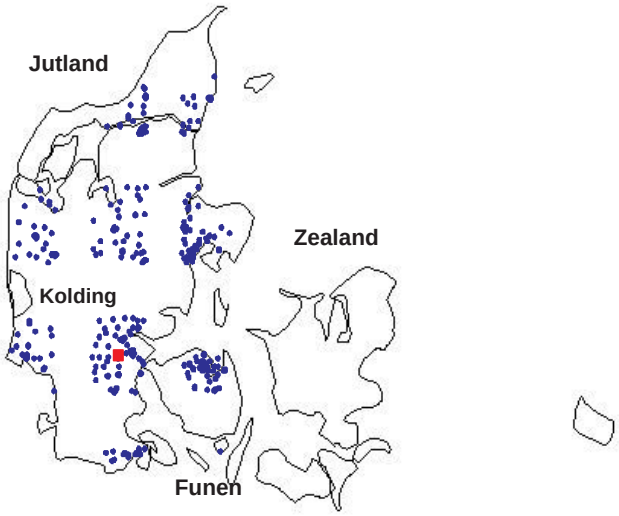

Figure 1. Locations of customers and depot (represented by a square) in the Danish Crown case study

The literature on the integrated Vehicle Routing and Crew Scheduling Problem (VRCSP) is rather limited. To our best knowledge, the most relevant work is that of Zaepfel and Boegl (2008). They addressed a weekly planning problem for postal companies, in which pickup tours and delivery tours must be decided for vehicles and drivers based on variable vehicle capacities and drivers' working regulations. Similar to our work, they also considered the driving rules and different types of drivers with different costs and working regulations. In their work, a solution framework was proposed, which consists of four parts: initialization, route generation, personnel assignment and solution evaluation. The framework was tested with two guiding metaheuristics, Tabu Search and Genetic Algorithm. The Tabu Search procedure was found to be competitive in solving their application. Another integrated vehicle routing and driver scheduling problem that has been studied in the literature is about the urban mass transit system, in which the buses and drivers are scheduled to serve a number of trips defined by a timetable (Huisman et al. (2005), Huisman and Wagelmans (2006), Freling et al. (2003) and Mesquita and Paias (2008)). Most of the solution methods for the VRCSP in the urban mass transit system are based on Column Generation and Lagrangian Relaxation. The integrated aircraft routing and crew scheduling problem is also relevant. This problem is solved by the Benders decomposition in Cordeau et al. (2001b), Mercier and Soumis (2007), Papadakos (2009), which decomposes the integrated problem into an aircraft routing problem and a crew pairing problem.

The vehicle routing part of our problem can be viewed as a Heterogeneous Vehicle Routing Problem with Time Windows (HVRPTW) in which a limited number of heterogeneous vehicles, characterized by different capacities, are available and the customers have a specified time window for service. The HVRPTW is usually solved by heuristics. The best known algorithms for this problem include: adaptive large neighborhood search (Pisinger and Ropke (2007)), variable neighborhood search (Paraskevopoulos et al. (2008), Imran et al. (2009)) and simulated annealing (Li et al. (2007), Braysy et al. (2008)). Choi and Tcha (2007) proposed an exact method based on column generation for the problem. For a recent literature review, see Baldacci et al. (2008).

The remainder of this paper is organized as follows. The MPVRCSP arising at the Danish Crown is defined in Section 2. In Section 3, the problem is formulated as a mixed integer linear program. A multi-level variable neighborhood search method is proposed in Section 4. Section 5 presents the computational results on the real-life data and conclusions follow in Section 6.

2 Problem Description

The problem is to determine routes for delivering fresh food to a set of supermarkets (or customers) every day over a one-week planning horizon. The routes are planned for a fleet of heterogeneous vehicles and a number of drivers with predefined working regulations.

A number of practical constraints need to be considered regarding the delivery. First of all, each customer orders a different amount of meat every day, measured in weight (Kg) and volume (pallets). Each vehicle has limited capacities both in weight and in volume. Secondly, each customer has a certain time window for receiving its order. These time windows are based on numerous factors such as working hours of the employees in the supermarket, city traffic etc. Lastly, certain special customers have requirements on the vehicle size. This is usually because of small roads or limited parking lot sizes. If an inappropriate vehicle type is used to serve such a customer, the driver usually needs to park some distance from the supermarket. This results in additional service time, which is proportional to the number of pallets ordered.

There are two kinds of drivers hired to carry out the delivery: internal and external. The internal drivers work on the predefined workdays and for no more than a maximum weekly working duration (37 hours) over a week. Both the internal and external drivers start from given starting times and finish before given latest ending times. The drivers cannot drive for more than 4.5 hours without a 45-minute break according to the EU driving legislation.

Several different types of costs are considered in this problem. We assume that the internal drivers have regular salaries according to their contracts. Hence only the fuel cost of the internal routes are considered, which depends on the distance travelled and the cost per kilometer. The external drivers are paid at a fixed price every hour, which covers both the salary for the driver and the vehicle cost. Therefore, the cost of the external routes is calculated by multiplying the route duration and the cost factor.

The objective of this problem can therefore be translated to minimize the fuel cost of the internal routes and the cost of the external routes over the planning horizon in such a way that each order must be served by one vehicle within its time window, vehicle capacities are not exceeded, each driver starts working at a predefined time and finishes before a given time on every workday, the internal drivers work for no more than a maximum weekly duration over the planning horizon, and the break rule regarding the driving legislation is respected.

This problem integrates vehicle routing and driver scheduling. The complexity of this problem can be characterized in many respects: the multiple periods, the heterogeneous vehicles, different types of drivers with different working regulations and the simultaneous planning of vehicles and drivers. Note that the orders on different days are fixed. The only constraint connecting the routes on different days is therefore the maximum 37 weekly hours for the internal drivers. This means that a certain driving schedule for an internal driver on one day will affect the maximum duration of the driver on the remaining days. Without this constraint, this weekly planning problem can be viewed as several independent daily planning problems, each of which considers the vehicle routing and crew scheduling problem on a single day, namely the daily

planning problem. This property is utilized in the solution method described in Section 4, and helps solve the large size problem effectively and efficiently.

3 Mathematical Formulation

This section presents a mixed integer linear programming formulation for the MPVRCSP. We denote the planning horizon by T and the set of drivers by D. The set of workdays for driver $l \in D$ is denoted by $T_l \subseteq T$. The start working time and latest ending time for driver $l \in D$ on day $t \in T$ are given by g_l^t and h_l^t, respectively. Let D_I and D_E denote the set of the internal and external drivers ($D = D_I \cup D_E$). For each internal driver $l \in D_I$, let H denote the maximum weekly working duration. We denote the maximum elapsed driving time without break by F and the duration of a break by G (according to the EU driver legislation).

Let K denote the set of vehicles. For each vehicle $k \in K$, let Q_k and P_k denote the capacity in weight and in volume, respectively. We assume the number of vehicles equals to the number of drivers. Denote the set of n customers (/ nodes) by $N = \{1, 2, ..., n\}$. Denote the depot by $\{0, n+1\}$. Each vehicle starts from $\{0\}$ and terminates at $\{n+1\}$. Each customer $i \in N$ specifies a set of days to be visited, denoted by $T_i \subseteq T$. On each day $t \in T_i$, customer $i \in N$ requests service with demand of q_i^t in weight and p_i^t in volume, service duration d_i^t and time window $[a_i, b_i]$. Note that, for the depot $i \in \{0, n+1\}$ on day t, we set $q_i^t = p_i^t = d_i^t = 0$. Denote the set of preferable vehicles for visiting customer i by K_i ($K_i \in K$) and the extra service time per pallet by e if a customer is not visited by a preferable vehicle. The travel time between customer i and j is given by c_{ij}. Denote the cost coefficients of the travel time of the internal drivers by A and the working duration of the external drivers by B.

We define binary variable x_{ijk}^t to be 1 if vehicle k travels from node i to j on day t, binary variable w_i^t to be 1 if customer i is not visited by a preferred vehicle on day t. Variable v_{ik}^t is the time that vehicle k visits node i on day t. Binary variable z_{ik}^t indicates whether vehicle k takes a break after serving customer i on day t. Variable u_{ik}^t is the elapsed driving time for vehicle k at customer i after the previous break on day t. Binary variable y_{lk}^t is set to 1 if vehicle k is assigned to driver l on day t. Variables r_l^t and s_l^t are the total working duration and the total travel time for driver l on day t, respectively.

This notation can be summarized as follows:

Notation:

Set:

T	The set of workdays in the planning horizon,
D_I	The set of internal drivers,
D_E	The set of external drivers,
D	The set of drivers $D = D_I \cup D_E$,
T_l	The set of workdays for driver $l \in D$,
K	The set of vehicles,
N	The set of customers,
N_0	The set of customers and depot $N_0 = \{0, n+1\} \cup N$,
K_i	The set of preferable vehicles for customer $i \in N$,
T_i	The set of days on which customer $i \in N$ orders,

Parameter:

Q_k	The weight capacity of vehicle $k \in K$,
P_k	The volume capacity of vehicle $k \in K$,
c_{ij}	The travel time from node $i \in N_0$ to node $j \in N_0$,
$[a_i, b_i]$	The earliest and the latest visit time at node $i \in N$,
d_i^t	The service time of node $i \in N_0$ on day $t \in T_i$,
q_i^t	The weight demand of node $i \in N_0$ on day $t \in T_i$,
p_i^t	The volume demand of node $i \in N_0$ on day $t \in T_i$,
e	The extra service time per pallet when a non-preferable vehicle is used,
$[g_l^t, h_l^t]$	The start time and the latest ending time of driver $l \in D$ on day $t \in T$,
H	The maximum working duration for each internal driver over the planning horizon,
F	The maximum elapsed driving time without break,
G	The duration of the break for drivers,
A	The cost factor on the total travel time of internal drivers,
B	The cost factor on the total working duration of the external drivers,

Variables:

x_{ijk}^t	Binary variable indicating whether vehicle $k \in K$ travels from node $i \in N_0$ to $j \in N_0$ on day $t \in T$,
w_i^t	Binary variable indicating whether customer $i \in N_0$ is visited by a non-preferable vehicle on day $t \in T$,
v_{ik}^t	The time at which vehicle $k \in K$ starts service at node $i \in N_0$ on day $t \in T$,
z_{ik}^t	Binary variable indicating whether vehicle $k \in K$ takes break after serving node $i \in N_0$ on day $t \in T$,
u_{ik}^t	The elapsed driving time of vehicle $k \in K$ at node $i \in N_0$ after the previous break on day $t \in T$,
y_{lk}^t	Binary variable indicating whether vehicle $k \in K$ is assigned to driver $l \in D$ on day $t \in T$,
r_l^t	The total working duration of driver $l \in D$ on day $t \in T$,
s_l^t	The total travel distance of driver $l \in D$ on day $t \in T$.

The mathematical formulation for this problem is presented as follows:

$$\min A \cdot \sum_{l \in D_I} \sum_{t \in T_l} s_l^t + B \cdot \sum_{l \in D_E} \sum_{t \in T_l} r_l^t \tag{1}$$

$$\sum_{k \in K} \sum_{j \in N_0} x_{ijk}^t = 1 \qquad \forall i \in N, t \in T_i \tag{2}$$

$$\sum_{k \in K \setminus K_i} \sum_{j \in N_0} x_{ijk}^t = w_i^t \qquad \forall i \in N, t \in T_i \tag{3}$$

$$\sum_{i \in N} \sum_{j \in N_0} q_i^t x_{ijk}^t \leq Q_k \qquad \forall k \in K, t \in T \tag{4}$$

$$\sum_{i \in N} \sum_{j \in N_0} p_i^t x_{ijk}^t \leq P_k \qquad \forall k \in K, t \in T \tag{5}$$

$$\sum_{j \in N_0} x_{0jk}^t = 1 \qquad \forall k \in K, t \in T \tag{6}$$

$$\sum_{i \in N_0} x_{ihk}^t - \sum_{j \in N_0} x_{hjk}^t = 0 \qquad \forall h \in N, k \in K, t \in T \tag{7}$$

$$\sum_{i \in N_0} x_{i,n+1,k}^t = 1 \qquad \forall k \in K, t \in T \tag{8}$$

$$u_{jk}^t \geq u_{ik}^t + c_{ij} - M(1 - x_{ijk}^t) - M z_{ik}^t \qquad \forall i, j \in N_0, k \in K, t \in T \tag{9}$$

$$u_{jk}^t \geq c_{ij} - M(1 - x_{ijk}^t) \qquad \forall i, j \in N, k \in K, t \in T \tag{10}$$

$$u_{ik}^t + \sum_{j \in N_0} c_{ij} x_{ijk}^t - F \leq M z_{ik}^t \qquad \forall i \in N_0, k \in K, t \in T \tag{11}$$

$$v_{jk}^t \geq v_{ik}^t + d_i^t + e \cdot p_i^t \cdot w_j^t + c_{ij} + G \cdot z_{ik}^t - M(1 - x_{ijk}^t) \qquad \forall i, j \in N_0, k \in K, t \in T \tag{12}$$

$$b_i \geq v_{ik}^t \geq a_i \qquad \forall i \in N, k \in K, t \in T_i \tag{13}$$

$$\sum_{k \in K} y_{lk}^t = 1 \qquad \forall l \in D, t \in T_l \tag{14}$$

$$\sum_{l \in D} y_{lk}^t = 1 \qquad \forall k \in K, t \in T \tag{15}$$

$$v_{0k}^t \geq \sum_{l \in D} (g_l^t \cdot y_{lk}^t) \qquad \forall k \in K, t \in T \tag{16}$$

$$v_{n+1,k}^t \leq \sum_{l \in D} (h_l^t \cdot y_{lk}^t) \qquad \forall k \in K, t \in T \tag{17}$$

$$s_l^t \geq \sum_{i \in N_0} \sum_{j \in N_0} c_{ij} x_{ijk}^t - M(1 - y_{lk}^t) \qquad \forall l \in D_I, k \in K, t \in T_l \tag{18}$$

$$r_l^t \geq v_{n+1,k}^t - g_l^t - M(1 - y_{lk}^t) \qquad \forall l \in D, k \in K, t \in T_l \tag{19}$$

$$\sum_{t \in T_l} r_l^t \leq H \qquad \forall l \in D_I \tag{20}$$

$$x_{ijk}^t, w_i^t, z_{ik}^t, y_{lk}^t \in \{0, 1\} \qquad \forall i, j \in N_0, l \in D, k \in K, t \in T \tag{21}$$

$$v_{ik}^t, u_{ik}^t, r_l^t, s_l^t \geq 0 \qquad \forall i, j \in N_0, l \in D, k \in K, t \in T \tag{22}$$

The objective function (1) minimizes weighted sum of the travel time of the internal drivers and the working duration of the external drivers over the planning horizon.

The constraints can generally be divided into two classes: one focuses on the vehicle routing (constraints (2-8) and (12-13)) and the remaining emphasizes the driver scheduling.

Constraints (2) state that each customer must be visited by one vehicle on each of its delivery days. Constraints (3) define whether each customer is visited by a preferable vehicle. Constraints (4—5) guarantee that the vehicle capacities are respected in both weight and volume. Constraints (6—8) ensure that each vehicle must start and terminate at the depot and that the flow is conserved at each customer on each day.

Constraints (9—10) define the elapsed driving time. More specifically, for the vehicle (k) travelling from customer i to j on day t, the elapsed driving time at j equals the elapsed driving time at i plus the driving time from i to j (i.e., $u^t_{jk} \geq u^t_{ik} + c_{ij}$) if the vehicle does not take a break at customer i (i.e., $z^t_{ik} = 0$); Otherwise, if the vehicle takes a break at customer i (i.e., $z^t_{ik} = 1$), the elapsed driving time at j will be constrained by (10) which make sure it is greater than or equal to the travel time between i and j (i.e., $u^t_{jk} \geq c_{ij}$). Constraints (11) guarantee that the elapsed driving time never exceeds an upper limit F by imposing a break at customer i (i.e., $z^t_{ik} = 1$) if driving from customer i to its successor results in a elapsed driving time greater than F.

Constraints (12) determine the time to start the service at each customer. If j is visited immediately after i, the time v^t_{jk} to start the service at j should be greater than or equal to the service starting time v^t_{ik} at i plus its service duration d^t_i, the extra service time $e \cdot p^t_i$ if i is visited by an inappropriate vehicle (i.e., $w^t_j = 1$), the travel time between the two customers c_{ij}, and the break time G if the driver takes a break after serving i (i.e., $z^t_{ik} = 1$). Constraints (13) make sure the services start within the customers' time window.

Constraints (14) assign each driver a route on each of his/her workday. Constraints (15) make sure each route on each day is assigned to exactly one driver. Constraints (16—17) ensure that the starting time and ending time of each route must lie between the start working time and latest ending time of the assigned driver. Constraints (18) calculate the total travel time for each internal driver. Constraints (19) define the working duration for each driver on every workday, which equals the time the driver returns to the depot minus the time he/she starts work. Constraints (20) make sure that the internal drivers work for no more than a maximum weekly working duration, referred to as 37 week-hour constraints. Constraints (21—22) define the binary and positive variables used in this formulation.

The formulation contains $O(|N|^2|K||T|)$ variables and $O(|N|^2|K||T|)$ constraints. Without Constraints (9-11) and (14-20), the problem can be reduced to a multi-period Heterogeneous Vehicle Routing Problem with Time Windows, which has already been proved to be NP-hard. Therefore, our problem is also NP-hard.

4 Multi-level Variable neighborhood search heuristic

We propose to solve this problem using a heuristic. Firstly, the problem is NP-hard and secondly we foreseen that the size of the problems that needs to be solved makes an exact approach prohibitive. The proposed method is named Multi-Level Variable Neighborhood Search heuristic (MLVNS) and illustrated in Figure 2.

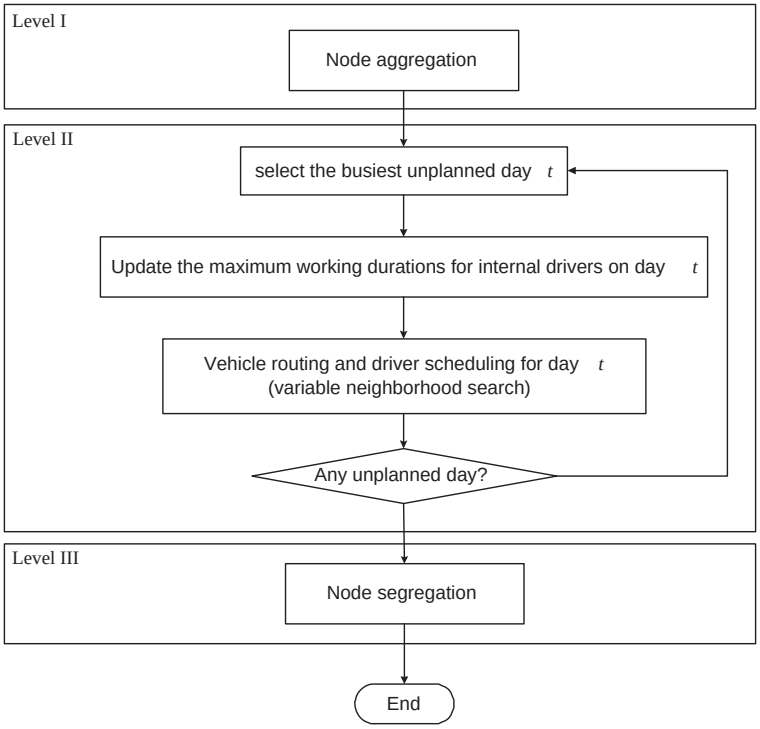

Figure 2. The flowchart of the MLVNS.

The MLVNS consists of three levels. The first level reduces the problem size through a node aggregation procedure. The second level constructs the solution to the aggregated problem. To reduce the computational overhead, we decompose the weekly planning problem into six daily planning problems, which are then solved sequentially in a given order. Before a specific daily problem is solved, the maximum daily duration of each internal driver is updated based on the 37 week-hour constraints and the workload that has been assigned to the driver on the previously planned days. Given the updated information on the internal drivers, the daily distribution plan is determined by means of a variable neighborhood search. At the last level, the solution of the aggregated problem is expanded to a solution for the original problem and the time to visit each customer is determined.

In the remainder of this section, the aggregation procedure is described in Section 4.1. How to update the maximum daily durations for the internal drivers is described in detail in Section 4.2. The variable neighborhood search that is applied to solve the daily planning problem is presented in Section 4.3. The overall method is summarized in Section 4.4.

4.1 Aggregation procedure

The basic idea of the aggregation procedure is to reduce the problem size by combining several nodes (customers) to a single supernode. The nodes to be aggregated are selected by analyzing their time windows, demands, and the travel times between them. Intuitively, it is preferable to visit supermarkets located close to each other, if possible, by the same vehicle in order to minimize the total travel distance. We hence treat such supermarkets as one supernode in order to reduce the size of the planning problem.

Our aggregation procedure is an iterative process and focuses on pairs of customers at each iteration, as shown in Algorithm 1. If two nodes i and j are close enough to each other (i.e., $c_{ij} \leq \rho$), have sufficient overlap in time windows (i.e., $min\{b_i, b_j\} - max\{a_i, a_j\} > \delta$) and the total amount of their orders is no more than κ_1 and κ_2 in weight and volume, they are allowed to be aggregated. In each iteration, the pair of nodes that satisfies the aggregation condition and has the minimum distance is selected to form a supernode, which replaces the two nodes and is treated as a new basic node available for further aggregation with other basic nodes or supernodes. Suppose at a certain aggregation stage, a supernode h is obtained, which contains a sequence of basic nodes, denoted as $\{h_1, ..., h_f\}$. The first node h_1 is called entry point and the last node h_f the exit point. The entry point and the exit point are used to update the distance between h and other nodes/supernodes. The demand of the aggregated node is defined by $q_h = \sum_{i \in \{1,...,f\}} q_{h_i}$ and $p_h = \sum_{i \in \{1,...,f\}} p_{h_i}$. The internal distance of h is calculated as $c_h = \sum_{i \in \{1,...,f-1\}} c_{h_i,h_{i+1}}$. For simplicity, the earliest start time to serve h is set to the maximum starting times of the nodes included in h, i.e., $a_h = max_{i \in \{1,...,f\}} a_{h_i}$. Since certain customers have special requirements on the vehicle size, we define the internal duration of h visited by vehicle k by d_h^k, which is the sum of total travel time, total service time and total additional service time caused by using vehicle k. The internal duration d_h of the supernode is set to $d_h = max_{k \in K} d_h^k$ to ensure the feasibility of the solution. The latest visit time of h is defined as $b_h = min_{i \in \{1,...,f\}}(b_{h_i} - d_h)$.

Without loss of generality, a basic node is also viewed as a special supernode consisting of only the single basic node. Therefore, in the preprocessing, we convert each basic node in the node set N' to a supernode. In each iteration, the best pair of nodes $(i^\star, j^\star)$ are selected and aggregated to a supernode h by using the approach mentioned above. We then replace node $i^\star$ and $j^\star$ in N' by the supernode h. The aggregation procedure stops when no more nodes can be aggregated.

Algorithm 1 : Level I (Aggregation procedure)

Input: the set of nodes N
Output: the set of Node N' after aggregation
$N' \leftarrow$Preprocessing(N)
repeat
  $(i^\star, j^\star) \leftarrow \emptyset$
  $minDist \leftarrow \infty$
  for $(i, j) \in N'$ **do**
    if ($c_{ij} < \rho$)& ($min\{b_i, b_j\} - max\{a_i, a_j\} > \delta$)& ($q_i + q_j < \kappa_1$)& ($p_i + p_j < \kappa_2$) **then**
      if $c_{ij} < minDist$ **then**
        $(i^\star, j^\star) \leftarrow (i, j)$
        $minDist \leftarrow c_{ij}$
      end if
    end if
  end for
  if $(i^\star, j^\star) \neq \emptyset$ **then**
    $h \leftarrow$ Aggregate($i^\star, j^\star$)
    $N' \leftarrow N' \setminus \{i^\star, j^\star\}$
    $N' \leftarrow N' \cup \{h\}$
  end if
until $(i^\star, j^\star) = \emptyset$
return N'

The parameters ρ, δ, κ_1 and κ_2 control the degree of aggregation. Increasing the values of ρ, κ_1 and κ_2, or decreasing the value of δ results in more aggregation of nodes. Generally, aggressive aggregation leads to a problem with small size and quick convergence. However, it also narrows down the feasible region, and may decrease the solution quality. The effects of the aggregation on solution quality and computational time are investigated in Section 5.

4.2 Updating driver duration

In order to accelerate the algorithm, we decompose the weekly planning problem into several daily problems and solve these daily problems sequentially. When decomposing, we only need to consider how to distribute the 37 weekly hours to each workday of the internal drivers. To respect this constraint, a maximum daily duration is imposed for each internal driver on each workday. There are several ways to determine this maximum daily duration.

A simple way is to evenly distribute the 37 hours to each workday, namely an even allocation strategy. This can be achieved by setting maximum daily duration M_l to 37 hours divided by the number of workdays for each internal driver l on each workday. In this case the internal drivers will never be assigned for more than 37 working hours over the week. However, since this simple strategy fails to take the significant variation of daily workload into account, some internal drivers might be idle on days with lower demand, while a lot of external drivers have to be hired for busier days.

In order to overcome this, we propose another strategy which adaptively determines the maximum daily duration before each daily plan is made, namely an adaptive allocation strategy. We first sort the days according to the number of orders and plan the busy days ahead of the quieter days. For a specific day t, if internal driver l works on day t (i.e., $t \in T_l$), we determine his/her unplanned work duration W_l by subtracting the total work duration already assigned to driver l on the previous planned days from the 37 hours and determine the number of unplanned workdays U_l for driver l. If day t is the last workday to be planned for l (i.e., $U_l = 1$), M_l is set to W_l so that the 37 week-hour constraints are respected. Otherwise, if $U_l > 1$, W_l is set to $W_l/U_l + \Theta$, where W_l/U_l is the average daily workload for the remaining unplanned days, and Θ is a user defined parameter. An appropriate value of Θ gives a degree of flexibility in the plan and leads to a good utilization of internal drivers on busy days since the daily problems are solved in descending order of workload. The adaptive allocation strategy is summarized in Algorithm 2 and a comparison of the two strategies is conducted in Section 5.

Algorithm 2 : Level II (Update daily work duration for internal drivers for day t)

Input: The planning day t; The set of routes $R = \{R_1, \ldots, R_{|T|}\}$
Output: The maximum daily work duration $M = \{M_1, \ldots, M_{|D_I|}\}$ for day t
for $l = 1, \ldots, |D_I|$ **do**
  $U_l \leftarrow$ GetTotalWorkDays(l)
  $W_l \leftarrow H$
  for $i \in T \setminus \{t\}$ **do**
    if ($R_i \neq \emptyset$) &($i \in T_l$) **then**
      $\sigma \leftarrow$ GetPlannedDailyWorkDuration(R_i, l)
      $W_l \leftarrow W_l - \sigma$
      $U_l \leftarrow U_l - 1$
    end if
  end for
  if $U_l > 1$ **then**
    $M_l \leftarrow W_l/U_l + \Theta$
  else
    $M_l \leftarrow W_l/U_l$
  end if
end for

4.3 Variable Neighborhood Search

The VNS was first introduced by Mladenovic and Hansen (1997) to "exploit systematically the idea of neighborhood change, both in the descent to local minima and in the escape from the valleys which contains them" (Hansen and Mladenovic (2001), Hansen and Mladenovic (2005)). During the past decade, this method has been successfully applied to a wide range of rich vehicle routing problems (Paraskevopoulos et al. (2008), Imran et al. (2009), Hemmelmayr et al. (2009)).

In this work we also develop a VNS to solve the daily planning problem which is an integrated vehicle routing and driver scheduling problem. The proposed VNS consists of three components: initialization, a shaking phase, and a local search. An initial solution is constructed and improved iteratively. In each iteration, one of five large neighborhoods is first exploited in order to diversify the search, referred to as shaking phase, and a local search is then applied in order to find the local optima. These components and the overall framework of the VNS are detailed below.

Initialization Our initial solution is generated by means of a sweep heuristic, as shown in Algorithm 3. We first assign each vehicle a random driver and sort the nodes in an ascending order of the angle they make with the depot and an arbitrary radius. The nodes are then assigned to the vehicles sequentially.

For each unrouted node, it is assigned to the vehicle considered currently if the vehicle capacities and the corresponding driver's duration are not exceeded or if the vehicle is the last available vehicle. Otherwise, the node is assigned to a new vacant vehicle.

Algorithm 3 : Level II (Initialization of VNS)

1: Sort the customers in an ascending order of the angle they make with the depot and an arbitrary radius $\{1, ..., n\}$
2: Assign each vehicle a random driver
3: Set the first vehicle k:= 1.
4: **for** $i = 1, ..., n$ **do**
5: **if** insertion of i to k results in violation of capacities or duration constraints **then**
6: $k \leftarrow min\{k+1, |\mathrm{K}|\}$.
7: **end if**
8: Insert i to k so as to minimize the total travel time of k.
9: **end for**

Local Search The local search in our VNS is performed by the Unified Tabu Search Algorithm (UTSA) (Cordeau et al. (2001a)). The UTSA allows intermediate infeasible solutions during the search by means of a penalized objective $f(s,t) = c(s,t) + \alpha p(s,t) + \beta q(s,t) + \gamma d(s,t) + \xi w(s,t)$, where $c(s,t)$ is the delivery cost on day t, $p(s,t) = \sum_{k\in K}(\sum_{i\in N}\sum_{j\in N_0} p_i x^t_{ijk} - P_k)^+$ and $q(s,t) = \sum_{k\in K}(\sum_{i\in N}\sum_{j\in N_0} q_i x^t_{ijk} - Q_k)^+$ are the total violations of the capacities in weight and in volume on day t, $d(s,t) = \sum_{k\in K}(v^t_{n+1,k} - \sum_{l\in D} h^t_l \cdot y^t_{lk})^+$ is the total violation of the daily duration of all the drivers on day t, and $w(s,t) = \sum_{i\in N}(v^t_{ik} - b_i)^+$ is the total violation of the service time window on day t, where $(x)^+ = max\{0, x\}$. The coefficients α, β, γ and ξ are positive self-adjusting penalties. A simple insertion is employed to improve the solution iteratively, which transfers customers from their original routes to other routes. The UTSA stops when the solution is not improved for a given number of iterations φ.

Shaking phase Five large neighborhoods are proposed for the shaking phase. The first three are based on the Ruin and Recreate Approach (RRA) (Schrimpf et al. (2000), Pisinger and Ropke (2007)). The basic idea of the RRA is to diversify the search by removing a number of bad customers from the current solution according to a removal scheme and then reinsert them into the routes again based on a reinsertion scheme. All these three neighborhoods use the same reinsertion scheme, i.e., regret heuristic (Potvin and Rousseau (1993), Ropke and Pisinger (2006)), but different removal schemes.

The first neighborhood uses a worst removal heuristic which selects a certain percentage (θ) of customers with the largest removal costs (Ropke and Pisinger (2006)). The removal cost of a customer is defined to

be the change in the solution value when it is removed from the route. This neighborhood is named the Worst Removal Neighborhood and denoted by WRN. The second neighborhood removes the customers covered by the external driver with the shortest working duration, namely the Driver Removal Neighborhood (DRN). The neighborhood helps not only to minimize the cost caused by using the external drivers but also reduce the number of vehicles used. The third neighborhood, Overlap Removal Neighborhood (ORN), removes all the customers in those routes that have the largest overlapping areas. The area of a route is defined as the area of the smallest rectangle that covers the depot and all the customers on that route. The areas of routes may overlap with each other. We define the overlapping area of each route to be the sum of its overlapping areas with all the other routes. In the ORN, we sort all the routes in a descending order of the overlapping area and remove the customers on the first λ routes. Since most of the customers have wide time windows, reducing the overlapping areas of routes may lead to a better solution. Such an example is illustrated in Figure 3, where, for simplicity, we consider two vehicles with capacity 4 and seven customers with unit demand. The solution before reducing the overlapping area is shown in (a). The area of each route in the solution and the overlapping area are defined in (b). The solution after reducing the overlapping area is depicted in (c) and the overlapping area is given in (d). As we can see from this small example, reducing the overlapping area leads to a better solution with smaller travel distance.

After the removal of customers, a regret heuristic, as detailed in Algorithm 4, is applied to reinsert the removed customer into the routes.

Algorithm 4 : Level II (Regret heuristic for the WRN, DRN and ORN in VNS)

1: N_{Rem} is the set of nodes to be inserted into solution s
2: **while** $N_{Rem} \neq \emptyset$ **do**
3: **for** $i \in N_{Rem}$ **do**
4: $bestIC_i \leftarrow$ CalculateBestInsertionCost(i, s)
5: $secondIC_i \leftarrow$ CalculateSecondBestInsertionCost(i, s)
6: **end for**
7: $i^\star \leftarrow \arg\max_{i \in N_{Rem}} (secondIC_i - bestIC_i)$
8: $s \leftarrow$ InsertCustomer($i^\star$, s)
9: $N_{Rem} \leftarrow N_{Rem} \setminus \{i^\star\}$
10: **end while**

The other two neighborhoods are constructed by a swap move. The fourth neighborhood, Swap Driver Neighborhood (SDN), swaps the drivers to find a good match between the drivers and the routes in terms of starting time and ending time. Similarly, the last neighborhood, Swap Truck Neighborhood (STN), swaps

vehicles, as shown in Figure 4. In the SDN (/STN), all possible pairs of drivers (/vehicles) are tried and the pair that leads to the minimum objective value is selected and applied.

To sum up, the five neighborhoods proposed for the shaking phase fall into two categories. The first three, WRN, DRN, and ORN, emphasize the construction of good routes, whereas the other two, SDN and STN, focus on assigning the right vehicles and right drivers to the routes. A sensitivity analysis on the effects of these neighborhoods is conducted in Section 5.

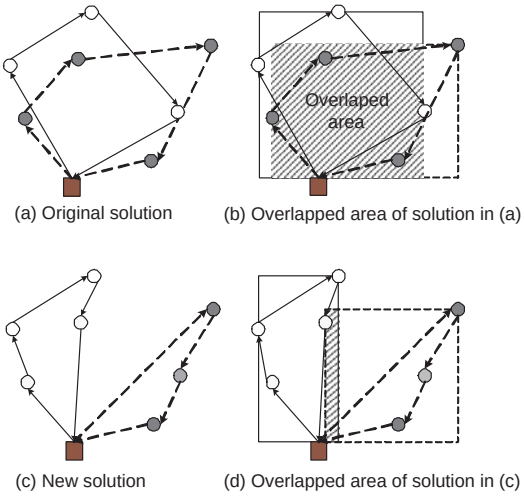

Figure 3. An example of two solutions with different overlapping areas

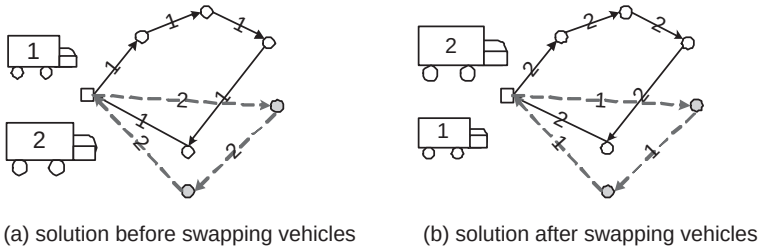

Figure 4. An example of swapping vehicles

VNS framework The overall framework of the VNS is given in Algorithm 5. Set $L = \{WRN, DRN, ORN, SDN, STN\}$ denotes the set of five large neighborhoods used in the shaking phase. Set L' denotes the set of available neighborhoods during the search procedure. The VNS starts with the initial solution given by

the sweep heuristic and improves the solution iteratively until the stop criteria reached. In each iteration, it exploits a neighborhood selected from L' and updates the current solution with a solution from the selected neighborhood. The UTSA is then applied on the neighboring solution and it stops when the best solution found so far has not been improved within φ iterations. There are two possible values, φ_1 and φ_2 ($\varphi_1 < \varphi_2$), for parameter φ depending on whether the UTSA is supposed to search thoroughly (i.e., $\varphi = \varphi_2$) or not (i.e., $\varphi = \varphi_1$). If the best solution found by the UTSA ($s^\star_{UTSA}$) is better than the best solution found in the previous VNS iteration ($s^\star$), $s^\star$ is updated by $s^\star_{UTSA}$ and the boolean parameter $improved$ is set to $True$, meaning that the best solution is improved in the current VNS iteration. Otherwise, $improved$ is set to $False$. The VNS stops once a certain time limit τ has been reached.

We now describe how the VNS selects the value of φ from $\{\varphi_1, \varphi_1\}$ and how it adaptively selects a neighborhood in the shaking phase at each iteration. The parameter φ is initialized by the small value φ_1, and L' by L. If the best solution ($s^\star$) is updated in the previous iteration (i.e., $improved = True$), the same neighborhood used in the previous iteration is applied again in the current iteration. Otherwise, if $improved = False$, the neighborhood used in the previous iteration is removed from the set of potential neighborhoods L' and another neighborhood from L' takes over. If the removal leads to an empty L', which means the best solution has not been improved by the last five iterations, we set φ to be the large value φ_2 so that the UTSA will search thoroughly in future, and reset L' to be L so that all neighborhoods become available again. As soon as the best solution is updated, φ is set back to φ_1 and L' back to L. When selecting a neighborhood from L', we first consider the SDN. If the SDN is not in L', the neighborhood that has not been used for the longest time is selected. The reason of giving the SDN a higher preference is that the assignment of the right drivers to the routes is found to be very crucial due to the various starting times of the drivers, as we will show in Section 5. Besides, the three neighborhoods, WRN, DRN and ORN, as well as the UTSA all emphasize on the route optimization, therefore a higher selection probability of the SDN balances the optimization efforts on all aspects of the problem.

4.4 Overall method

The overall MLVNS is summarized in Algorithm 6. Line (3—6), line (7—13) and line (14—16) correspond to Level I, II and III, respectively.

5 Computational Results

In this section we present the computational experiments on the real-life data provided by Danish Crown. Our method was programmed in C# and executed on a Pentium 2.66GHz machine and two GB of memory.

Algorithm 5 : Level II (VNS framework)

Input: The planning day t; The set of customers N_t to be planned on day t; The maximum duration M for internal drivers on day t.
Output: The route plan R_t for day t.
$improved \leftarrow False$
$L' \leftarrow L$
$s \leftarrow$ SweepHeuristic(N_t)
while $CPUTime \leq \tau$ **do**
  if $improved$ **then**
    $\varphi \leftarrow \varphi_1$
    $L' \leftarrow L$
  else
    $L' \leftarrow L' \setminus \{currentL\}$
    if $L' = \emptyset$ **then**
      $\varphi \leftarrow \varphi_2$
      $L' \leftarrow L$
    end if
    if $SDN \in L'$ **then**
      $currentL \leftarrow SDN$
    else
      $currentL \leftarrow$ FindLongestUnused(L')
    end if
  end if
  $s \leftarrow$ ApplyLNS($currentL, s, M$)
  $(s, s^{\star}_{UTSA}) \leftarrow$ TabuSearch(s, M, φ)
  if $s^{\star}_{UTSA}$ is better than $s^{\star}$ **then**
    $s^{\star} \leftarrow s^{\star}_{UTSA}$
    $improved \leftarrow True$
  else
    $improved \leftarrow False$
  end if
end while
$R_t \leftarrow s^{\star}$
return R_t

Algorithm 6 : Multi-level variable neighborhood search heuristic

Input: The set of nodes $N = \{N_1, \ldots, N_{|T|}\}$
Output: The set of routes $R = \{R_1, ..., R_{|T|}\}$
for $t = 1, ..., |T|$ **do**
    $N_t \leftarrow$ AggregationProcedure(N_t) // see Algorithm 1
    $R_t \leftarrow \emptyset$
end for
$daysPlanned \leftarrow 0$
while $daysPlanned < |T|$ **do**
    $t \leftarrow$ FindBusiestUnplannedDay(R)
    $M \leftarrow$ UpdateMaxWorkDuration(t, R) // see Algorithm 2
    $R_t \leftarrow$ VNS(N_t, M) // see Algorithm 5
    $daysPlanned \leftarrow daysPlanned + 1$
end while
for $t = 1, ..., |T|$ **do**
    $R_t \leftarrow$ Expand(R_t)
end for
return R

We first describe the data and parameters used in our tests and then present a sensitivity analysis of the parameters as well as a comparison between our solutions and Danish Crown's solutions.

5.1 Data and parameters

There are data sets for four weeks, each of which consists of six workdays. As an example, Table 1 shows the total number of orders and the total demand by volume and weight for each workday from 29 / 09 / 2008 to 04 / 10 / 2008. The length of time window (TW) in this week ranges from 1 hour to 24 hours and the histogram of the TW length is shown in Figure 5. Approximately 40% of the orders have an 8-hour TW, most of which have [0.00, 8.00]. Roughly 35% of the orders have 2- to 4- hour time windows in the early morning, such as [6.00 8.00], [7.00 10.00] and [6.00 10.00]. Around 18% of the customers do not have any restriction on visiting time and can be visited at any time during the day. This is because Danish Crown has the electronic keys to access these supermarkets. The vehicle information is provided in Table 2, including the sizes, the capacities and the numbers of the vehicles. Approximately 10% of the supermarkets have requirements on the vehicle size. Danish Crown uses approximately 11 internal drivers and at most 14 external drivers every day. Euclidean distances are used in our tests and we assume the vehicle speed is 60km / hour.

The value of each parameter used in the algorithm is set based on the preliminary tests. The minimum length (δ) for the overlap required between two time windows in order for two nodes to be aggregated is set to 60 minutes. The capacity parameters κ_1 and κ_2 are set according to the smallest vehicle, i.e., 7000 (KG) and 18 (pallets) respectively. The parameter Θ used in the adaptive allocation strategy in solving the daily problems is set to 60 minutes. The parameter θ in the WRN is set to 10%, i.e., 10% of customers are removed and reinserted again. The parameter λ in the ORN is set to 2, meaning that customers in two routes are removed. The iteration number φ_1 and φ_2 for the stop criteria in the TS are set to 350 and 1500, respectively.

		Total demand	
Date	Number of orders	(Pallet)	(Kg)
29 / 09 / 2008	279	329.5	84263.5
30 / 09 / 2008	381	439.5	125118.6
01 / 10 / 2008	365	399.0	124740.5
02 / 10 / 2008	364	434.5	124740.5
03 / 10 / 2008	397	577.0	170938.1
04 / 10 / 2008	360	483.0	144057.5

Table 1. Orders from 29 / 09 / 2008 to 04 / 10 / 2008

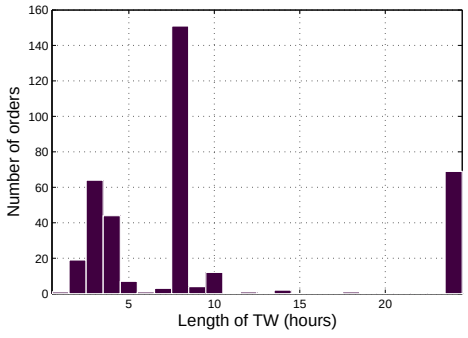

Figure 5. The TW length of the customers

Type	Number of vehicles	Capacity (Pallet)	(Kg)
Big	9	33	14000
Medium	2	27	10000
Small	14	18	7000

Table 2. Vehicle Resource

5.2 Sensitivity Analysis

The purpose of this section is to assess the behavior of the proposed heuristic and analyze the sensitivity of the parameters. The analysis can be classified into two categories. The first one focuses on the algorithm performance on daily problems. We tested the algorithm on six daily instances, including both busy days and easy days, and examined three aspects of the algorithm: the effectiveness of the node aggregation procedure, the effectiveness of using two alternative values for φ in the UTSA in the VNS, and the effects of the five large neighborhoods in the shaking phase of the VNS. The second group of tests evaluated the performance of the algorithm on solving weekly problems and provided the following results: a comparison of two work duration allocation strategies for decomposition and a comparison of the different number of the special supermarkets that have requirements on vehicle size.

Effectiveness of the aggregation procedure

As mentioned in Section 4, before the solution is constructed, the problem size is first reduced through a node aggregation procedure in which pairs of nodes with a distance less than or equal to ρ are considered to be aggregated into a single supernode. We tested the algorithm with different values of ρ on six daily instances. Figure 6 illustrates the convergence of the proposed heuristic for four values of ρ, 0, 2, 4 and 6. When ρ equals 0, no aggregation is done. When ρ equals 2, 4 or 6, the problem size is reduced by approximately 25%, 35% and 50%, respectively. Table 3 shows the detailed results. Column '*Index*' is the test descriptor and column '*Time*' is the running time in minute for each test, ranging from 3 to 36 minutes. For each ρ, the column '*Average solution value*' reports the average solution value $\bar{z}_t^\rho$ on the six daily instances in test t. The column '*Conv.(%)*' shows the relative difference in the average solution value between test $t-1$ and t, calculated as $\frac{\bar{z}_t^\rho - \bar{z}_{t-1}^\rho}{\bar{z}_{t-1}^\rho} \cdot 100$, which also indicates the speed of convergence with different values of ρ. The column '*Gap(%)*' provides the percentage gap between solution value of the aggregated problem ($\rho' = \{2, 4, 6\}$) and that of the original problem ($\rho = 0$), calculated as $\frac{\bar{z}_t^{\rho'} - \bar{z}_t^\rho}{\bar{z}_t^\rho} \cdot 100$. These gaps show how the solution value is influenced by different level of aggregation, depicted in Figure 7.

As seen from Table 3 and Figure 6, a higher value of ρ yields a faster convergence to the solution since more of the feasible region is cut by the aggressive aggregation. With a short running time, the aggregated problem leads to a better solution due to an intelligent search in a smaller feasible region. For example, $\rho = 6$ provides better results than $\rho = 0$ when the running time is shorter than 20 minutes. However, given an enough computation time, the aggregated problem is not competitive to the original problem any more in terms of solution quality. For instance, the solution value provided by $\rho = 0$ is consistently better than that provided by $\rho = 6$ when the running time is larger than 20 minutes. A good trade-off between the running time and solution quality is obtained with $\rho = 2$.

	ρ	0			2			4			6		
		Average			Average			Average			Average		
	Time	solution	Gap	Conv.	solution	Gap	Conv.	solution	Gap	Conv.	solution	Gap	Conv.
Index	(minute)	value	(%)	(%)	value	(%)	(%)	value	(%)	(%)	value	(%)	(%)
1	3	32531	0		29284	-10.0		28109	-13.6		28180	-13.4	
2	6	29774	0	-8.5	27407	-7.9	-6.4	26269	-11.8	-6.5	27046	-9.2	-4.0
3	9	28350	0	-4.8	26421	-6.8	-3.6	25633	-9.6	-2.4	26686	-5.9	-1.3
4	12	27305	0	-3.7	25663	-6.0	-2.9	25496	-6.6	-0.5	26334	-3.6	-1.3
5	15	26783	0	-1.9	24990	-6.7	-2.6	25187	-6.0	-1.2	26117	-2.5	-0.8
6	18	26228	0	-2.1	24723	-5.7	-1.1	25046	-4.5	-0.6	25944	-1.1	-0.7
7	21	25586	0	-2.4	24442	-4.5	-1.1	24992	-2.3	-0.2	25796	0.8	-0.6
8	24	25341	0	-1.0	24241	-4.3	-0.8	24872	-1.9	-0.5	25764	1.7	-0.1
9	27	25109	0	-0.9	24086	-4.1	-0.6	24791	-1.3	-0.3	25675	2.3	-0.3
10	30	24911	0	-0.8	24032	-3.5	-0.2	24704	-0.8	-0.4	25629	2.9	-0.2
11	33	24643	0	-1.1	23924	-2.9	-0.4	24563	-0.3	-0.6	25570	3.8	-0.2
12	36	24449	0	-0.8	23833	-2.5	-0.4	24561	0.5	0.0	25529	4.4	-0.2

Table 3. Average solution value of using different levels of aggregation

Effectiveness of using two alternative values for φ

The parameter φ in the stop criteria of UTSA is self-switched between two user defined values φ_1 and φ_2 ($\varphi_1 < \varphi_2$). We compare the performance of using two φ values (i.e., $\varphi = \{\varphi_1, \varphi_2\}$) with that of using solely one φ value (i.e., $\varphi = \varphi_1$ or $\varphi = \varphi_2$). Ten random runs on the six daily instances with different running times are performed. Given a running time t, we denote the average solution value with $\varphi = \varphi_1$ by $\bar{z}_{\varphi_1}(t)$, the average solution value with $\varphi = \varphi_2$ by $\bar{z}_{\varphi_2}(t)$, and the average solution value with $\varphi = \{\varphi_1, \varphi_2\}$ by $\bar{z}_{\{\varphi_1,\varphi_2\}}(t)$. Figure 8 shows the percentage gap between $\bar{z}_{\varphi_1}(t)$ (and $\bar{z}_{\varphi_2}(t)$) and $\bar{z}_{\{\varphi_1,\varphi_2\}}(t)$ as a function of running time t. These gaps can be calculated as $f_{\varphi_1}(t) = \frac{\bar{z}_{\varphi_1}(t) - \bar{z}_{\{\varphi_1,\varphi_2\}}(t)}{\bar{z}_{\{\varphi_1,\varphi_2\}}(t)} \cdot 100$ (and $f_{\varphi_2}(t) = \frac{\bar{z}_{\varphi_2}(t) - \bar{z}_{\{\varphi_1,\varphi_2\}}(t)}{\bar{z}_{\{\varphi_1,\varphi_2\}}(t)} \cdot 100$). The

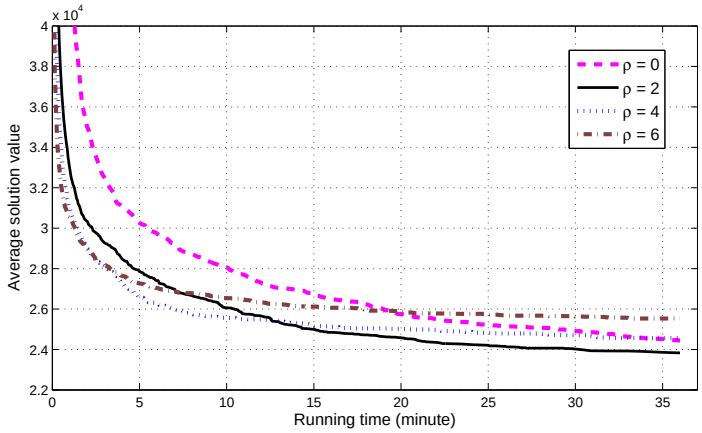

Figure 6. Average solution value as a function of running time using different values of ρ

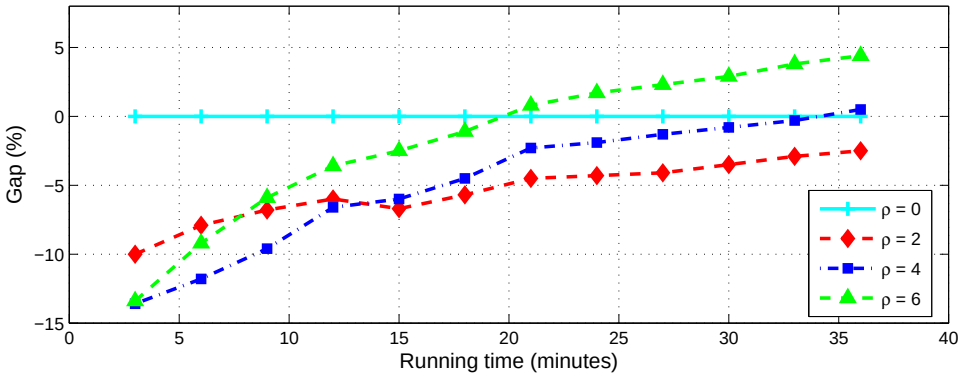

Figure 7. Solution gap in percentage between the aggregated problems and the original problem

results show that, given an enough computational time (more than 13 minutes), the solution of using two values of φ is consistently better than that of using solely one value. The improvement is approximately 2%.

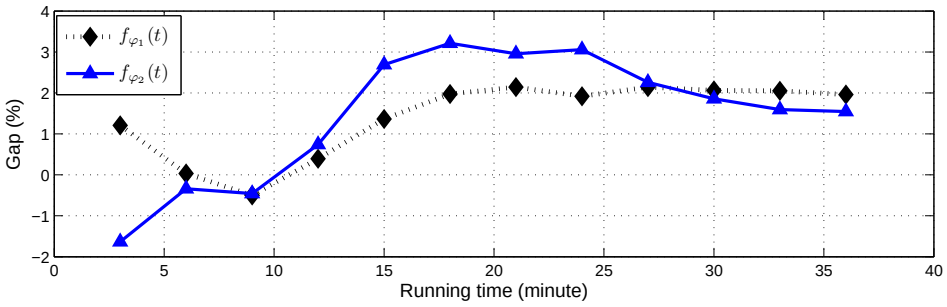

Figure 8. Solution gap in percentage between using two φ values and using solely one φ value

Effect of the five large neighborhoods in the shaking phase in the VNS

We proposed five neighborhoods in the shaking phase of the VNS, including the WRN, DRN, ORN, SDN and STN. We evaluated the contribution of each neighborhood in this section and show the effect of combining the five neighborhoods. In Table 4, column '*Index*' is the test descriptor and column '*Time*' is the running time in minute for each test, ranging from 3 to 36 minutes. For each neighborhood setting L, column '*Average solution value*' reports the average solution value $\bar{z}_t^L$ on the six daily instances in test t. Column '*Gap(%)*' presents the percentage gap in the average solution value between using one neighborhood L_1 and using five neighborhoods L, calculated as $\frac{\bar{z}_t^{L_1} - \bar{z}_t^L}{\bar{z}_t^L} \cdot 100$. Row '*Average*' provides the overall average value of each column. Figure 9 shows gaps as a function of running time.

From Table 4 and Figure 9, we can see that, among all the five neighborhoods, the SDN is the most effective one. This is the reason why we give SDN the highest selection probability in the shaking phase as mentioned in Section 4. The heuristic with all the five neighborhoods outperforms the heuristic with any single neighborhood by 0.6% to 3% given an enough computational time.

Effectiveness of the adaptive allocation strategy

We compared the two allocation strategies, even allocation strategy and adaptive allocation strategy, used to distribute the 37 weekly hours to each workday for the internal drivers. The average solution values on the four weekly instances are provided in column '*Average solution value*' in Table 5. The column '*Gap(%)*' shows the percentage gap between the solution values using the two strategies. Row '*Average*' shows the overall

	L	All		WRN		DRN		ORN		SDN		STN	
Index	Time (minute)	Average solution value	Gap (%)	Average Solution value	Gap (%)	Average Solution value	Gap (%)	Average Solution value	Gap (%)	Average Solution value	Gap (%)	Average Solution value	Gap (%)
1	3	29284	0	29597	1.1	29731	1.5	29731	1.5	29351	0.2	29925	2.2
2	6	27407	0	27733	1.2	27897	1.8	27792	1.4	27751	1.3	28734	4.8
3	9	26421	0	26542	0.5	27010	2.2	27064	2.4	26812	1.5	27603	4.5
4	12	25663	0	26094	1.7	26169	2.0	26580	3.6	26252	2.3	26664	3.9
5	15	24990	0	25762	3.1	25694	2.8	26060	4.3	25504	2.1	26008	4.1
6	18	24723	0	25565	3.4	25447	2.9	25427	2.8	25194	1.9	25561	3.4
7	21	24442	0	25393	3.9	25225	3.2	25291	3.5	24714	1.1	25169	3.0
8	24	24241	0	25063	3.4	24874	2.6	25036	3.3	24395	0.6	24890	2.7
9	27	24086	0	24848	3.2	24667	2.4	24819	3.0	24284	0.8	24672	2.4
10	30	24032	0	24654	2.6	24582	2.3	24612	2.4	24182	0.6	24467	1.8
11	33	23924	0	24548	2.6	24543	2.6	24521	2.5	24120	0.8	24238	1.3
12	36	23833	0	24420	2.5	24504	2.8	24463	2.6	24003	0.7	24182	1.5
Average		**25253.8**	**0.0**	**25851.6**	**2.4**	**25861.9**	**2.4**	**25949.7**	**2.8**	**25546.8**	**1.2**	**26009.4**	**3.0**

Table 4. Average solution values of using different large neighborhood settings

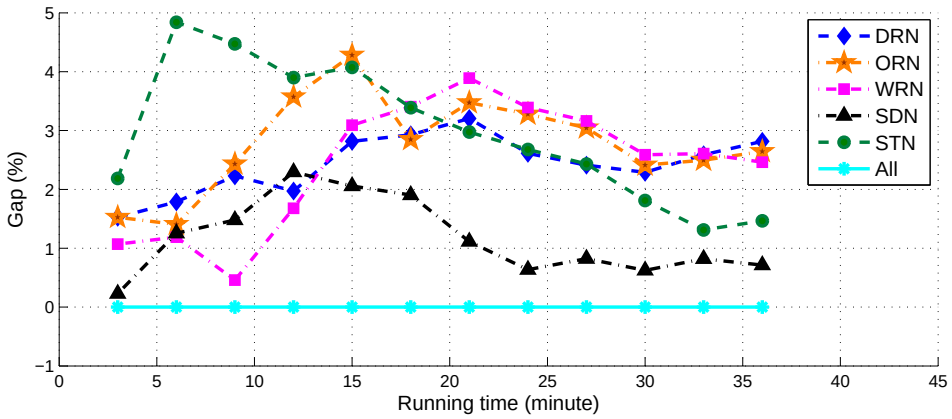

Figure 9. Solution gap in percentage between using different large neighborhood settings

average value of each column. For all the tests, the adaptive allocation strategy consistently performs better than the even allocation strategy and improves the solution by 4.5% on average.

Time	Even allocation strategy	Adaptive allocation strategy	
(minute)	Average solution value	Average solution value	Gap(%)
10	135104	129201	-4.4
14	128561	124412	-3.2
18	124790	119656	-4.1
22	122670	117304	-4.4
26	121410	115453	-4.9
31	120073	113849	-5.2
36	119289	113011	-5.3
Average	**124557**	**118984**	**-4.5**

Table 5. Average solution values with even allocation strategy and adaptive allocation strategy

Effect of the supermarkets that have requirements on vehicle size

In real life approximately 10% of the supermarkets have requirements on vehicle size, referred to as special supermarkets. To analyze the influence of these special supermarkets, in addition to the real-life case supplied by Danish Crown, we created two additional cases. In the first case we assume there is no special supermarket and the corresponding results are shown in column '*0% of supermarkets*'. In the second case, we randomly added vehicle size requirements to an additional 10% of the supermarkets from the real-life case so that altogether 20% of supermarkets were special. The test results are presented in Table 6. For each test, the average solution value, total distance, total duration and number of vehicles used are provided in column '*Average solution value*', '*Average total distance*', '*Average total duration*' and '*Number of vehicles*', respectively. Row '*Average*' shows the overall average value of each column. As we can see from the table, the number of special supermarkets has a large effect on the solution values. As the proportion of special supermarkets increases from 0% to 20%, the solution value, total travel distance and total route duration are increased by 21%, 5.7%, and 8.8%, respectively. More vehicles are required when there are a large number of special supermarkets.

5.3 Comparison with industrial solution

Danish Crown also provided the routes they planned and executed for the four weekly instances. However, the only accessible information about their real-life plan is the list of customers served in every route on

Special	0% of supermarkets				10% of supermarkets				20% of supermarkets			
	Average solution	Average total	Averate total	Number of	Average solution	Average total	Averate total	Number of	Average solution	Average total	Averate total	Number of
Data set	value	distance	duration	vehicles	value	distance	duration	vehicles	value	distance	duration	vehicles
Week1	129571	32188	62341	19	143770	32882	65633	19	159865	33601	68586	19.3
Week2	90066	28315	55953	16	98332	29260	58563	16.2	117294	30756	61683	17
Week3	91734	28293	56190	16.2	94817	28831	57986	16.2	103418	29617	59727	16.5
Week4	108419	29858	57751	18	115125	30394	60067	18.2	127513	31445	62613	18.7
Average	**104947**	**29663**	**58058**	**17.3**	**113011**	**30341**	**60562**	**17.4**	**127022**	**31354**	**63152**	**17.9**

Table 6. Average solution values with different number of special supermarkets

every day. The exact order in which and the time at which each customer is visited are not available. We therefore calculated a TSP lower bound on the travel distance for each route using Concorde (Appelgate et al. (2003)). These lower bounds are provided in column '*LB on travel distance*' in Table 7. The first column gives the names of the data sets and the second column corresponds to the index of days in each week. The daily solutions as well as the summarized weekly solutions are provided. We also tested our algorithm on the same instances. The average solution value and the average travel distance on ten random runs are presented in column '*Average solution value*' and column '*Average travel distance*'. The numbers of vehicles used in the two solutions are provided in columns '*Number of vehicles*'. Column '*Gap(%)*' shows the percentage gap between travel distance ($\bar{z}$) by our method and the lower bound ($z_{\bar{LB}}$) on the travel distance of the industrial solution, calculated as $\frac{\bar{z}-\bar{z}_{LB}}{\bar{z}_{LB}} \cdot 100$. The results show that, our solution is superior to the industrial solution in terms of both the total travel distance and the number of vehicles used.

It also needs to be stressed that the TSP lower bound is a very poor lower bound on the travel distance since a lot of constraints are not considered in the TSP, such as the time windows, the working regulation and so on. Therefore, the actual difference between the two solutions is likely to be larger.

6 Conclusion

We have addressed a planning problem with integrated vehicle routing and driver scheduling which arises from a practical problem of Danish Crown. In this problem, a routing plan, consisting of six days in a week, has to be made for a fleet of heterogeneous vehicles to deliver the fresh meat to the supermarkets according to their demands and preferences, such as the visiting time and the preferable vehicle sizes. The route plan also needs to comply with the drivers' working regulations, such as the fixed workdays, the fixed starting time and latest ending time, the maximum weekly working duration, break rule and so on. The objective

is to minimize the total delivery cost. We have presented a mixed integer linear programming formulation for the problem and a multi-level variable neighborhood search based heuristic for solving it. The first level of the proposed heuristic effectively reduces the problem size through a node aggregation procedure based on the locations, demands, and time windows of the nodes. The second level decomposes the aggregated weekly planning problem into six daily problems by wisely distributing the internal drivers' weekly workload to each workday and solves the daily problems sequentially by means of a variable neighborhood search. Two aspects of our VNS were proved to be very effective: the combination of five large neighborhoods in the shaking phase and the alternative usage of a short-term and long-term searching in the local search. At the last level, the solution of the aggregated problem is expanded to the solution of the original problem. The heuristic was implemented and tested on real-life data. Our solution is superior to the industrial solution in terms of the total travel distance and number of vehicles used.

Acknowledgments

This work was financially supported by the FoodDTU project. This support is gratefully acknowledged. Thanks are also due to Jacob Vesterdorf at Danish Crown for providing problem information and the real-life data.

		MLVNS solution			Danish Crown solution		
Data set	Day	Average solution value	Average travel distance	Number of vehicles	LB on travel distance	Number of vehicles	Gap (%)
Week1	0	13532	4365	15.0	5809	20	-24.9
	1	21839	5397	16.7	6383	23	-15.4
	2	23859	5518	17.0	6389	23	-13.6
	3	22176	5349	17.0	6252	23	-14.4
	4	29805	6018	19.0	6592	24	-8.7
	5	32559	6235	19.0	6374	23	-2.2
Total		**143770**	**32882**	**19**	**37799**	**24**	**-13.0**
Week2	0	12498	4032	15.0	6136	20	-34.3
	1	16122	4888	15.8	6801	23	-28.1
	2	17726	5404	15.5	6392	23	-15.5
	3	15152	4647	15.0	5960	23	-22.0
	4	20550	5280	16.0	6142	24	-14.0
	5	16284	5008	15.5	6068	23	-17.5
Total		**98332**	**29260**	**16**	**37500**	**24**	**-22.0**
Week3	0	12630	4074	14.8	5920	20	-31.2
	1	17246	4973	15.8	6586	23	-24.5
	2	16650	5259	15.2	6924	23	-24.0
	3	14881	4767	15.0	6720	23	-29.1
	4	19302	5208	16.0	6862	24	-24.1
	5	14107	4551	15.0	6292	23	-27.7
Total		**94817**	**28831**	**16**	**39304**	**24**	**-26.6**
Week4	0	13056	4211	15.0	5966	20	-29.4
	1	14722	4699	15.2	6514	23	-27.9
	2	20010	5372	16.0	6417	23	-16.3
	3	14960	4673	15.0	6262	23	-25.4
	4	28930	5833	18.0	6839	24	-14.7
	5	23448	5605	17.0	7080	23	-20.8
Total		**115125**	**30394**	**18**	**39078**	**24**	**-22.2**

Table 7. Comparison between the Danish Crown solution and MLVNS solution

Bibliography

Appelgate, D., Bixby, R., Chvátal, V., and Cook, W. (2003). Concorde tsp solver. available online at http://www.tsp.gatech.edu/concorde.

Baldacci, R., Battarra, M., and Vigo, D. (2008). Routing a heterogeneous fleet of vehicles. In Golden, B., Raghavan, S., and Wasil, E., editors, *The vehicle routing problem: Latest advances and new challenges*. Springer.

Braysy, O., Dullaert, W., Hasle, G., Mester, D., and Gendreau, M. (2008). An effective multirestart deterministic annealing metaheuristic for the fleet size and mix vehicle-routing problem with time windows. *Transportation Science*, 42(3):371–386.

Choi, E. and Tcha, D.-W. (2007). A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34(7):2080–2095.

Cordeau, J., Laporte, G., and Mercier, A. (2001a). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8):928–936.

Cordeau, J., Stojkovic, G., Soumis, F., and Desrosiers, J. (2001b). Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, 35(4):375–388.

Freling, R., Huisman, D., and Wagelmans, A. (2003). Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling*, 6(1):63–85.

Hansen, P. and Mladenovic, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467.

Hansen, P. and Mladenovic, N. (2005). Variable neighborhood search. In Burke, E. and Kendall, G., editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Springer.

Hemmelmayr, V. C., Doerner, K. F., and Hartl, R. F. (2009). A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791–802.

Huisman, D., Freling, R., and Wagelmans, A. (2005). Multiple-depot integrated vehicle and crew scheduling. *Transportation Science*, 39(4):491–502.

Huisman, D. and Wagelmans, A. (2006). A solution approach for dynamic vehicle and crew scheduling. *European Journal of Operational Research*, 172(2):453–471.

Imran, A., Salhi, S., and Wassan, N. A. (2009). A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem. *European Journal of Operational Research*, 197(2):509–518.

Li, F., Golden, B., and Wasil, E. (2007). A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34(9):2734–2742.

Mercier, A. and Soumis, F. (2007). An integrated aircraft routing, crew scheduling and flight retiming model. *Computers & Operations Research*, 34(8):2251–2265.

Mesquita, M. and Paias, A. (2008). Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem. *Computers & Operations Research*, 35(5):1562–1575.

Mladenovic, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Papadakos, N. (2009). Integrated airline scheduling. *Computers & Operations Research*, 36(1, Sp. Iss. SI):176–195.

Paraskevopoulos, D. C., Repoussis, P. P., Tarantilis, C. D., Ioannou, G., and Prastacos, G. P. (2008). A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows. *Journal of Heuristics*, 14(5):425–455.

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435.

Potvin, J. and Rousseau, J. (1993). A parallel route building algorithm for the vehicle-routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., and Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171.

Zaepfel, G. and Boegl, M. (2008). Multi-period vehicle routing and crew scheduling with outsourcing options. *International Journal of Production Economics*, 113(2):980–996.

This paper addresses an integrated vehicle routing and driver scheduling problem arising at the largest fresh meat producer in Denmark. The problem consists of a one-week planning horizon, heterogeneous vehicles, and drivers with predefined work regulations. These regulations include, among other things, predefined workdays, fixed starting time, maximum weekly working duration, break rule. The objective is to minimize the total delivery cost.

The real-life case study is first introduced and modelled as a mixed integer linear program. A multi-level variable neighborhood search heuristic is then proposed for the problem. At the first level, the problem size is reduced through an aggregation procedure. At the second level, the aggregated weekly planning problem is decomposed into daily planning problems, each of which is solved by a variable neighborhood search. At the last level, the solution of the aggregated problem is expanded to that of the original problem. The method is implemented and tested on real-life data consisting of up to 2000 orders per week. Computational results show that the aggregation procedure and the decomposition strategy are very effective in solving this large scale problem, and our solutions are superior to the industrial solutions given the constraints considered in this work.

ISBN 978-87-90855-54-3

DTU Management Engineering
Department of Management Engineering
Technical University of Denmark

Produktionstorvet
Building 424
2800 Kongens Lyngby
Tel. 45 25 48 00
Fax 45 93 34 35

www.man.dtu.dk