A template based adaptive large neighborhood search for the consistent vehicle routing problem

Attila A. Kovacs[(1)], Sophie N. Parragh[(1,2)], Richard F. Hartl[(1)]

(1) Department of Business Administration, University of Vienna,
Bruenner Strasse 72, 1210 Vienna, Austria
{attila.kovacs, sophie.parragh, richard.hartl}@univie.ac.at

(2) INESC Porto / IBM Center for Advanced Studies Portugal
Campus da FEUP, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

Abstract

The importance of customer satisfaction was identified by many industries as a key factor of competitive advantage. So, for companies in the small package shipping industry it can be reasonable to increase the service quality even at the expense of transportation cost in order to gain customer loyalty. These companies noticed that customer satisfaction can be increased by providing consistent service in the form of visiting customers with the same driver at approximately the same time of the day over a certain time period. Motivated by this real world problem, the consistent vehicle routing problem (ConVRP) combines traditional vehicle routing constraints with the requirements for service consistency. This paper presents a fast solution method called template based adaptive large neighborhood search for the described problem. Compared to state-of-the-art heuristics, the developed algorithm is highly competitive on the available benchmark instances. Additionally, new test instances are provided. These seem to be more challenging due to the variation of different model parameters and consequently help to identify interesting effects. Finally, a relaxed variant of the original ConVRP is presented. In this variant, the departure times from the depot can be delayed to adjust the service times of the customers. Experiments show that allowing later departure times considerably improves the solution quality under tight consistency requirements.

Keywords: vehicle routing, periodic distribution problems, large neighborhood search, metaheuristics

1 Introduction

The consistent vehicle routing problem (ConVRP), as defined in Groër et al. [7], is an extension of the periodic vehicle routing problem. It is used to model services like the ones performed by companies in the small package shipping industry. This problem category involves the construction of routes over a given time period, e.g., several days, such that customer demands are met. However, a simple delivery is nowadays not enough for companies to differentiate themselves in a competitive environment. Many companies in the above mentioned industry started to focus on customer satisfaction to increase customer loyalty and to obtain competitive

This is the peer reviewed version of the following article:
Kovacs, A. A., Parragh, S. N., Hartl, R. F. (2014). A templatebased adaptive large neighborhood search for the consistent vehicle routing problem. Networks, 63(1), 60-81.
which has been published in final form at http://dx.doi.org/10.1002/net.21522

advantage. From the customers' point of view, an important criterion for high quality service is consistency. The ConVRP considers traditional vehicle routing constraints and also accounts for this additional requirement. In this context, consistency is modeled in two ways. First, in order to form a stronger relationship between the supplier and the customer, a customer can only be assigned to one driver. Second, to enable the customers to prepare themselves for a delivery, they have to be serviced at about the same time of the day. Consistency requirements link the days in the planning period together. Therefore, it is not possible to divide the multi-period problem into several single-period problems.

In contrast to fleet-focused vehicle routing problems that have been studied extensively, the ConVRP is the first VRP variant that focuses primarily on customer satisfaction [7]. Campbell and Thomas [1] emphasize the importance of on-time delivery for package delivery companies to maintain competitiveness. They introduce the probabilistic traveling salesman problem with deadlines to model stochastic deliveries that have to be executed until a determined time. An overview of operative issues in the small package shipping industry is given in Wong [23]. After describing a typical day of a service provider who performs local delivery and pickup tasks, the author highlights the characteristics that distinguish this problem from standard vehicle routing problems.

The ConVRP itself is introduced by Groër et al. [7]. Motivated by a real world problem that has to be dealt with by companies in the small package shipping industry, the authors combine traditional VRP constraints with the requirements for service consistency. Their solution approach for the ConVRP is based on a template concept, in which a set of template routes is resolved to obtain the daily routes for all days in the planning period. High quality template routes are generated by applying a record-to-record travel algorithm [11]. Tarantilis et al. [22] solve the ConVRP by applying a template-based tabu search algorithm. The algorithm is divided into two stages. In the first stage high quality templates are built. The daily routing plans that are derived from the templates are post-optimized in the second stage.

Sungur et al. [21] address the courier delivery problem (CDP) in which time and driver consistent services are considered. Customer requests and service times are stochastic. The CDP integrates (soft) time windows for customer visits but driver consistency is not restricted to one driver per customer. The problem is solved by a master and daily scheduler heuristic (MADS). Similar to the template concept, MADS generates a master plan that can be converted into daily schedules with few modifications.

The bi-objective time-consistent vehicle routing problem (TCVRP) is introduced by Feillet et al. [5]. Their work is motivated by the need to provide time consistent transportation for handicapped persons through a given time period. In contrast to the ConVRP, where the total travel time is minimized while bounding the difference between the earliest and the latest arrival time for each customer, the TCVRP integrates a second objective. It involves minimizing the maximum number of time classes. Two services for the same customer i on day α and β may be assigned to the same time class if their arrival times $a_{i\alpha}$ and $a_{i\beta}$ fulfill $|a_{i\alpha} - a_{i\beta}| \leq L$. L represents the maximum arrival time difference within a time class. Driver consistency is not considered during the solution finding process, but a post-processing phase is performed to minimize the average number of drivers per customer. The TCVRP is solved by large neighborhood search (LNS) [19].

The effect of incorporating different driver management strategies into the periodic vehicle routing problem (PVRP) is investigated in [20]. The authors use a tabu search heuristic (TS) to minimize an aggregated objective function. This function is composed of the total travel distance plus a measure that represents the cost or benefit of the different driver assignment strategies. Time consistency is not considered.

Consistency in the context of inventory routing problems (IRP) is examined by Coelho et al. [3]. The classical IRP model is extended by six problem-specific consistency features and the problem variants are solved by adaptive large neighborhood search (ALNS).

ALNS, as introduced by Ropke and Pisinger [17] is an extension of LNS [19] and it is also related to the ruin and recreate principle applied by Schrimpf et al. [18]. ALNS has been applied successfully to solve different variants of the vehicle routing problem [13, 16, 17], the technician and task scheduling problem [4] and the service technician routing and scheduling problem [9]. For a survey on large neighborhood search see [14].

The contribution of this paper is twofold. First, we present a new solution method for the ConVRP called template based adaptive large neighborhood search (TALNS). It combines the ALNS by Ropke and Pisinger [17] with the template concept presented in Groër et al. [7]. Second, we introduce a relaxed variant of the ConVRP in which later departures from the depot than the earliest possible starting times are permitted. It is demonstrated that this slight relaxation of the starting times leads to improved time consistency, while the travel times remain almost unchanged. To test our algorithm, we use available data sets from the literature and show that it is highly competitive. On the basis of existing benchmark instances we design new data sets in which time consistency is modeled in a more realistic way. We provide heuristic solutions for these new data sets.

2 Problem definition

The ConVRP is defined on a complete directed graph $G = (N, A)$, where $N = \{0, 1, ..., n\}$ is the set of customers and the depot $\{0\}$. $A = \{(i, j) \mid i, j \in N, i \neq j\}$ is the set of arcs. Customers are visited on routes traversed by a homogeneous fleet of vehicles in the set $K = \{1, ..., m\}$. There is a sufficient amount of vehicles available (i.e., $m = n$). Each vehicle $k \in K$ with given capacity Q is located at the depot from where it departs at time 0 and where it must return to before time T. The planning horizon involves $|D|$ days where D is the set of days. On each day $d \in D$, each customer $i \in N\backslash\{0\}$ has a predetermined demand q_{id} that must be delivered and a service time s_{id}. We use auxiliary parameters w_{id} equal to 1 if customer i requires service on day d ($q_{id} > 0$) and equal to 0, otherwise. Each arc $(i, j) \in A$ is associated with travel time t_{ij}. We assume that the travel time matrix complies with the triangle inequality. Consistency requirements dictate that each customer $i \in N\backslash\{0\}$ must be assigned to the same vehicle k over the entire planning period. Furthermore, the difference between the earliest and the latest arrival time at each customer over all days $\in D$ must be within the maximum arrival time difference L. Vehicle idling to reduce the arrival time difference is not allowed.

The model uses the following binary variables:

$$x_{ijkd} = \begin{cases} 1, \text{if arc } (i,j) \text{ is traversed by vehicle } k \text{ on day } d, \\ 0, \text{otherwise;} \end{cases}$$

$$y_{ikd} = \begin{cases} 1, \text{if customer } i \text{ is assigned to vehicle } k \text{ on day } d, \\ 0, \text{otherwise;} \end{cases}$$

The continuous variables a_{ikd} denote the arrival time at customer i by vehicle k on day d. Given this notation, the ConVRP can be formulated in the following model. The only modification to the original MIP of [7] is the extension of the a_{id} variables with the index k to better comply with our applications.

$$Minimize \quad \sum_{d\in D}\sum_{k\in K}\sum_{i\in N}\sum_{j\in N} t_{ij}x_{ijkd} \tag{1}$$

subject to:

$$y_{0kd} = 1 \qquad \forall\ k \in K, d \in D, \tag{2}$$

$$a_{0kd} = 0 \qquad \forall\, k \in K, d \in D, \tag{3}$$

$$\sum_{k \in K} y_{ikd} = w_{id} \qquad \forall\, i \in N \backslash \{0\}, d \in D, \tag{4}$$

$$\sum_{i \in N \backslash \{0\}} q_{id} y_{ikd} \leq Q \qquad \forall\, k \in K, d \in D, \tag{5}$$

$$\sum_{i \in N} x_{ijkd} = \sum_{i \in N} x_{jikd} = y_{jkd} \qquad \forall\, j \in N, k \in K, d \in D, \tag{6}$$

$$w_{i\alpha} + w_{i\beta} - 2 \leq y_{ik\alpha} - y_{ik\beta} \qquad \forall\, i \in N \backslash \{0\}, k \in K, \alpha, \beta \in D, \alpha \neq \beta, \tag{7}$$

$$a_{ikd} + x_{ijkd}(s_{id} + t_{ij}) - (1 - x_{ijkd})T \leq a_{jkd} \qquad \forall\, i \in N, j \in N \backslash \{0\}, k \in K, d \in D, \tag{8}$$

$$a_{ikd} + x_{ijkd}(s_{id} + t_{ij}) + (1 - x_{ijkd})T \geq a_{jkd} \qquad \forall\, i \in N, j \in N \backslash \{0\}, k \in K, d \in D, \tag{9}$$

$$a_{ikd} + w_{id}(s_{id} + t_{i0}) \leq T w_{id} \qquad \forall\, i \in N \backslash \{0\}, k \in K, d \in D, \tag{10}$$

$$L - T(w_{i\alpha} + w_{i\beta} - 2) \geq a_{ik\alpha} - a_{ik\beta} \qquad \forall\, i \in N \backslash \{0\}, k \in K, \alpha, \beta \in D, \alpha \neq \beta, \tag{11}$$

$$x_{ijkd} \in \{0, 1\} \qquad \forall\, i, j \in N, k \in K, d \in D, \tag{12}$$

$$y_{ikd} \in \{0, 1\} \qquad \forall\, i \in N, k \in K, d \in D, \tag{13}$$

$$a_{ikd} \geq 0 \qquad \forall\, i \in N, k \in K, d \in D. \tag{14}$$

The objective function (1) minimizes the total travel time. Inequalities (2) and (3) define that each route starts from the depot at time 0. Constraints (4) guarantee that each customer is serviced on each day he requires a service and inequalities (5) make sure that the vehicle capacity is not exceeded. Constraints (6) ensure that all assigned customers have exactly one predecessor and one successor. Driver consistency is guaranteed in (7). Inequalities (8) and (9) set the arrival times at the customers on the condition that vehicle idling to improve time consistency is not allowed. Inequalities (8) also prevent sub-tours. The completion of the routes within the maximum travel time is enforced by inequalities (10). Constraints (11) ensure that the arrival time difference for each customer is not greater than L.

3 Solution framework

We propose a template based adaptive large neighborhood search (TALNS) to solve the Con-VRP. Given an initial solution, ALNS integrates several destroy methods to repeatedly remove parts of a solution and several repair methods to rebuild the partial solution [14]. In contrast to standard vehicle routing problems, we do not apply ALNS to the actual routing plan but to the template on the basis of which the routing plan is generated. In the following, we first describe the template concept and then the different design elements of the proposed TALNS.

3.1 The template concept

Due to the consistency requirements, the daily schedules are linked by the customers who require service on multiple days. To handle these interdependencies, we use the solution approach suggested by Groër et al. [7]. It relies on the generation of template routes from which the actual daily routes are derived. The template contains all customers who are relevant for consistency. These are the customers who require service on two or more days. We refer to them as frequent customers and collect them in set N_f. To obtain the daily routes, the template routes are resolved as follows. First, excessive customers are removed. Excessive customers are those, who are considered in the template but do not require service on that particular day. Second, customers who only request service on one day during the planning horizon are inserted on the corresponding day. This process is illustrated in the following example: assume a set of customers who must be served over a three days planning horizon as given in Table 1. The

Customer	1	2	3	4	5	6	7	8
Day 1	1	0	1	0	0	1	0	1
Day 2	1	1	0	1	0	1	1	0
Day 3	0	1	0	0	1	1	1	1
Number of services	2	2	1	1	1	3	2	2
Template	1	1	0	0	0	1	1	1

Table 1: Demand pattern

columns give the customers from 1 to 8 and the first three rows define whether a customer requests a service (value 1) or not (value 0) on the corresponding day. The last but one row sums the number of services over the entire planning horizon and the last row defines the customers who are considered in the template. All customers who are visited at least twice are part of the template (value 1). The remaining ones are ignored (value 0). Services are performed by vehicles that can visit at most three customers a day. A possible solution for the example is shown in Figure 1. In the template, all frequent customers are served by two routes. Based on the template routes the daily schedules are derived. On the first day, customers 2 and 7 are removed and customer 3 is inserted. The same approach is continued to obtain the remaining daily schedules.

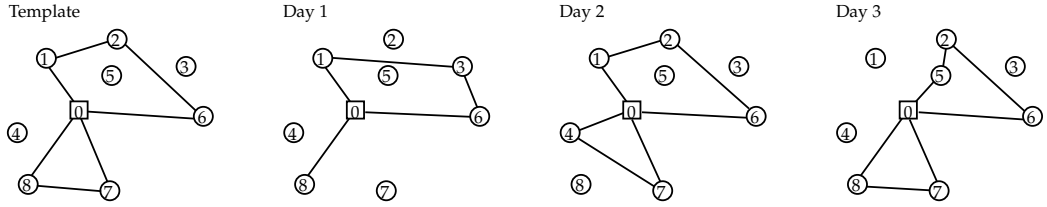

Figure 1: Example: template and corresponding solution.

By using the template concept, driver consistency is always guaranteed since the customer-to-driver assignment is not changed during the template's resolution. Additionally, the customer sequence in the template routes is transferred to all daily routes. The resulting precedence principle supports the compliance with the time consistency requirement.

In order to create template routes that can easily be transformed into a feasible solution, daily request information must be incorporated into the template construction. This is done in two ways. First, for each frequent customer we compute an artificial demand q_{ai} and an artificial service time s_{ai}. These values are used throughout the template construction. In contrast to Groër et al. [7], they are defined as the mean of the original demand and service time values over all days in D and not only the days a customer requests service. Thus, our variant implicitly considers the service frequency of each customer; q_{ai} and s_{ai} are computed as follows:

$$q_{ai} = \frac{\sum_{d \in D} q_{id}}{|D|} \ \forall i \in N_f \tag{15}$$

$$s_{ai} = \frac{\sum_{d \in D} s_{id}}{|D|} \ \forall i \in N_f \tag{16}$$

Second, artificial maximum tour lengths, T_a, and capacity values, Q_a, are used in the template construction. They control the number of routes in the template and thus, the number of routes in the daily schedules.

If T_a and Q_a are set to small values, each customer will be visited on a separate route in the template and in the corresponding solution. Such a solution would always be feasible with

a maximum arrival time difference of 0. The total travel time, however, would be very high. The effect of using tight artificial constraints in the template generation is shown in Figure 2.

Large T_a and Q_a values, on the other hand, lead to long template routes. Given the triangle inequality, these routes result in shorter travel times. Yet, when the template is resolved, it is more likely that the resulting solution violates the real T and Q constraints and the time consistency requirement. A solution obtained by using large T_a and Q_a values is illustrated in Figure 3.

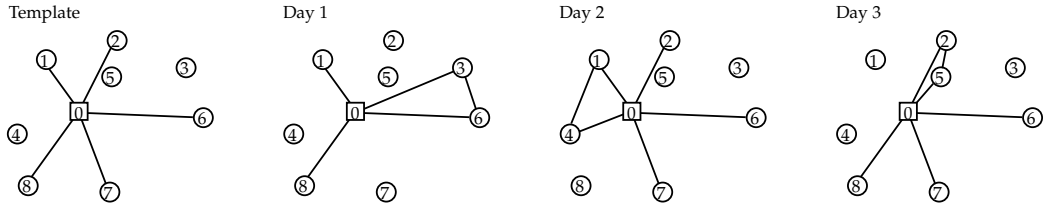

Figure 2: Artificial constraints, T_a and Q_a, are set to small values. The solution is feasible but the total travel time is very high.

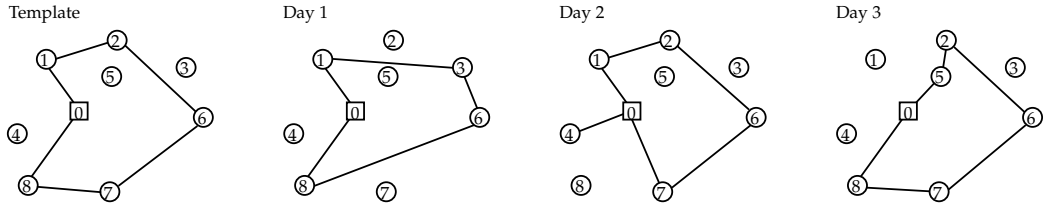

Figure 3: Artificial constraints, T_a and Q_a, are set to large values. The solution has a short total travel time but it might violate the maximum travel time T, the maximum capacity Q, or the time consistency requirement L.

For diversification purposes we use different T_a and Q_a values every time a new template is generated. The way we set the values deviates from the approach used in [7]. In [7] an estimate for both artificial constraints is made on the basis of the mean number of visits per day. Depending on whether the obtained solution is feasible or not, the constraints are either relaxed or tightened. In contrast to the estimation, we define upper and lower bounds for both, T_a and Q_a, and search the interval between the bounds. The upper bounds, $\overline{Q_a}$ and $\overline{T_a}$, are calculated as follows.

First, for each day d, two lower bounds on the number of vehicles are defined - one in terms of tour length and one in terms of capacity. They are denoted as $\underline{NV_{Td}}$ and $\underline{NV_{Q_d}}$, respectively. The daily lower bounds based on capacity considerations are defined as:

$$\underline{NV_{Q_d}} = \left\lceil \frac{\sum_{i \in N_f} q_{id}}{Q} \right\rceil \quad \forall d \in D. \tag{17}$$

In order to compute the daily lower bounds based on the maximum tour length, $\underline{NV_{Td}}$, we first estimate the total daily travel times. This is done by applying Kruskal's algorithm [10] to define the minimal spanning tree (MST) among all frequent customers requesting service on day d and the depot. We use the property that the total length of the MST on day d, denoted as $f(MST_d)$ plus the cost of the shortest edge from the depot to any customer is a lower bound for the cost of the optimal TSP solution and therefore also a lower bound for the cost of the optimal VRP solution. By, adding the aggregated service times on day d we get a lower bound

for the total duration of all tours. Dividing this value by the maximum tour length yields another lower bound for the vehicle number $\underline{NV_{Td}}$:

$$\underline{NV_{Td}} = \left\lceil \frac{f(MST_d) + \min_{i \in N \setminus \{0\}} t_{0i} + \sum_{i \in N_f} s_{id}}{T} \right\rceil \quad \forall d \in D \tag{18}$$

Let $f(MST_{template})$ denote the total travel time of the MST among all frequent customers $i \in N_f$ and the depot. Using artificial demands q_{a_i} and service times s_{a_i} (equations (15) and (16)), and $\underline{NV_{Qd}}$ and $\underline{NV_{Td}}$, we are now able to compute $\overline{Q_a}$ and $\overline{T_a}$:

$$\overline{Q_a} = \frac{\sum_{i \in N_f} q_{ai}}{\max_{d \in D} \underline{NV_{Qd}} - (1 - \varepsilon)} \tag{19}$$

$$\overline{T_a} = \frac{f(MST_{template}) + \min_{i \in N \setminus \{0\}} t_{0i} + \sum_{i \in N_f} s_{ai}}{\max_{d \in D} \underline{NV_{Td}} - (1 - \varepsilon)} \tag{20}$$

By bounding the artificial constraints to $\overline{Q_a}$ and $\overline{T_a}$ during the template generation, it is guaranteed that the number of template routes never falls below the bounds defined in (17) and (18). To avoid upper bounds that are lower than the optimal values, we reduce the maximum of the daily lower bounds on the vehicles ($\underline{NV_{Td}}, \underline{NV_{Qd}}$) by a value that is asymptotic to 1. This value is expressed by $1 - \varepsilon$, where ε is a small number. In our experiments we used $\varepsilon = 0.0001$.

Non-frequent customers are not considered in the calculation of $\underline{NV_{Qd}}$ and $\underline{NV_{Td}}$. Including all customers in the calculation would lead to lower $\overline{Q_a}$ and $\overline{T_a}$ values and consequently, more template routes would be required (even though not all customers are represented in the template). This approach would prevent solutions in which frequent customers are served by the minimum number of routes and all non-frequent customers are served on separate routes.

Lower bounds, $\underline{Q_a}$ and $\underline{T_a}$, are set to the artificial constraint values that yield the first feasible solution.

3.2 Constructing an initial solution

The construction heuristic builds the initial template routes. However, their resolution does not necessarily lead to a feasible solution. A template is accepted as the initial one if it results in a solution that satisfies the capacity and the tour length constraints. The time consistency may be violated. It is much harder to adhere to the time consistency requirement as it is only considered implicitly by the template concept.

To begin with, the artificial tour length T_a and capacity Q_a limits are set to their upper bounds (equations (19) and (20)). We then iteratively generate and resolve different templates while tightening the artificial tour length and capacity constraints until an initial template is found. More precisely, every violation of a constraint in the obtained solution leads to a reduction of the respective artificial limit, T_a and Q_a, by 1%.

We use a greedy approach [13, 17] to insert customers into the template during the construction phase. The heuristic inserts customers one after another at their cheapest position until all customers have been assigned. Every feasible insertion position is checked for every unscheduled customer. The customer who causes the least increase in the total travel time is then inserted at his best position. Formally speaking, let $\Delta f_{i,k}$ represent the change in the total travel time if customer i is inserted at his cheapest position into route k. For customers who cannot be inserted feasibly, $\Delta f_{i,k}$ is set to infinity. The customer i^* is inserted into template route k^* for which

$$(i^*, k^*) := \arg \min_{i \in N_f, k \in K} \Delta f_{i,k}. \tag{21}$$

The same approach is used to insert non-frequent customers during the resolution of the template. Here, unassigned non-frequent customer i^* is inserted at his best position into daily route k^* for which

$$(i^*, k^*) := \arg \min_{i \in N \setminus \{\{0\} \cup N_f\}, k \in K} \Delta f_{i,k}. \tag{22}$$

An empty route is added every time it is not possible to insert further customers into feasible positions.

In this context, feasibility refers to the artificial capacity and tour length constraints during the template construction and to the real constraints during the resolution. The check for time consistency is only performed once the schedule is completed.

3.3 Template based adaptive large neighborhood search

We use the TALNS to improve the initial template and consequently the initial solution. The TALNS integrates several sub-heuristics that are used to repeatedly destroy and repair the current template [13, 16, 17].

The pseudocode of the TALNS is presented in Algorithm 1. In each iteration, one destroy and repair sub-heuristic pair is selected based on its performance in past iterations (line 3). The chosen pair is applied to generate a new template (line 5). The destroy sub-heuristic removes assigned customers from the template routes. The removed customers are reinserted into the template by using the according repair sub-heuristic. The reinsertion is performed with regard to the selected artificial constraints T_a and Q_a (line 4). The new template is resolved (line 6), i.e, excessive customers are removed and non-frequent customers are inserted on each day as described in Section 3.2. The decision to move to the new template (respectively solution) or not is made by using the simulated annealing acceptance criterion (line 7).

In case the initial solution violates the time consistency constraint, the first priority is to find the first feasible solution. To ease the search, we further tighten the artificial template constraints and force the template routes to become shorter. We reduce both, T_a and Q_a, by 1% every 50^{th} iteration until the obtained solution complies with all constraints. The artificial constraints that led to the first feasible solution are set as lower bounds, $\underline{T_a}$ and $\underline{Q_a}$, for T_a and Q_a.

In the remaining TALNS iterations the artificial limits T_a and Q_a are selected randomly between their corresponding lower and upper bounds (line 4):

$$Q_a = \underline{Q_a} + y(\overline{Q_a} - \underline{Q_a}) \tag{23}$$

$$T_a = \underline{T_a} + y(\overline{T_a} - \underline{T_a}) \tag{24}$$

The random number y is chosen in the interval [0,1]. The last step is to update the performance of the applied destroy and repair sub-heuristic pair (line 14). The TALNS terminates as soon as a given number of iterations is reached.

Algorithm 1 TALNS

Require: initial template τ and corresponding solution s ▷ Section 3.2

$s_{best} = s$ ▷ $f(s) = \infty$ if s is infeasible
repeat
  select a pair of destroy and repair operators d and r ▷ Section 3.3.2
  select $T_a \in [\underline{T_a}, \overline{T_a}]$ and $Q_a \in [\underline{Q_a}, \overline{Q_a}]$ ▷ Equations (23) and (24)
  $\tau' =$ generateNewTemplate(τ, d, r, T_a, Q_a)
  $s' =$ resolveTemplate(τ')
  if accept(s',s) **then** ▷ Section 3.3.1
    $\tau = \tau'$
    $s = s'$
  end if
  if $f(s') < f(s_{best})$ **then**
    $s_{best} = s'$
  end if
  update joint performance of operators d and r ▷ Section 3.3.2
until maximum number of iterations is reached
return s_{best}

3.3.1 Acceptance criterion

A simulated annealing framework [8] is used to decide whether a new template τ' that produces solution s' should become the current incumbent template τ or not. We accept a new template τ' when the corresponding solution's objective value $f(s')$ is lower than that of the current incumbent solution, $f(s)$. Inferior solutions are accepted with probability $e^{-(f(s')-f(s))/\hat{t}}$. Parameter $\hat{t}$ denotes the current temperature and it is initialized only when the first feasible solution (with respect to all constraints) is found:

$$\hat{t} = -\frac{w_{\hat{t}}}{\ln 0.5} f(s) \tag{25}$$

Based on [13, 17] we set $\hat{t}$ such that a $w_{\hat{t}}$ % worse solution is accepted with a probability of 50%, where $w_{\hat{t}}$ is a parameter to be determined. The temperature is then decreased by using the geometric annealing schedule, $\hat{t} = \hat{t}c$; c denotes the cooling rate.

Infeasible solutions are rejected until $\hat{t}$ has been initialized ($f(s')$ is set to infinity). Afterwards, violations of the time consistency requirement are tolerated but penalized. If we obtain a solution that only violates the time consistency requirement we use an artificial objective function, $f_a(s')$, to calculate the acceptance probability.

$$f_a(s') = f(s') + (l_{max} - L)e^{(\frac{i_{TALNS}}{\delta} - p_{penalty})} \tag{26}$$

The impact of the penalization is controlled by parameters $p_{penalty}$, δ, and the current number of TALNS iteration i_{TALNS}. The higher i_{TALNS} (i.e., the further the search process), the lower the probability of accepting infeasible solutions. Setting δ to 6000 was found to work well during the implementation of the algorithm. l_{max} denotes the maximum arrival time difference of solution s'.

The best found solution s_{best} is replaced if s' is feasible and $f(s') < f(s_{best})$.

3.3.2 Selection of sub-heuristics

In each iteration of the TALNS, we generate a new template by applying one destroy sub-heuristic d, and one repair sub-heuristic r. The selection of these operators is performed by using the adaptive selection mechanism proposed by Pisinger and Ropke [13]. In our case,

however, the selection and remuneration of the destroy and repair operators are done pairwise rather than separately. Slightly better results could be achieved with this selection mechanism in [9]. We record the joint performance of the sub-heuristic pairs and assign weights ρ_{dr} according to their past performance. The better the results obtained by a pair, the higher its weight and therefore its probability ϕ_{dr} for being chosen in future selections. If η_d and η_r are the number of available destroy and repair sub-heuristics, respectively, we define the selection probabilities as follows:

$$\phi_{dr} = \frac{\rho_{dr}}{\sum_{d=1}^{\eta_d} \sum_{r=1}^{\eta_r} \rho_{dr}} \tag{27}$$

Roulette wheel selection is used to choose one pair in every iteration.

To link a pair's weight to its performance, the pairs can earn scores, ψ_{dr}, every time they are applied. The scores are collected during time segments of 100 iterations according to the following pattern:

$$\psi_{dr} = \begin{cases} \psi_{dr} + \sigma_1, & \text{if the destroy-repair pair yielded a solution that} \\ & \text{improved the global best solution } s_{best}; \\ \psi_{dr} + \sigma_2, & \text{if the destroy-repair pair yielded a solution that} \\ & \text{was not visited before and improved the incumbent solution } s; \\ \psi_{dr} + \sigma_3, & \text{if the destroy-repair pair yielded a solution that was not visited before and} \\ & \text{was accepted as the new incumbent solution } s, \text{ although it was worse;} \\ \psi_{dr}, & \text{otherwise.} \end{cases}$$

σ_1, σ_2 and σ_3 are user defined parameters. To find out whether a solution was visited before or not, we compare the solutions' objective values.

Initially, weights ρ_{dr} are set to one and the scores ψ_{dr} are set to zero. After each time segment, the weights are updated as follows and all ψ_{dr} values are reset to zero.

$$\rho_{dr} = p_{react} \frac{\psi_{dr}}{\max(1, \Theta_{dr})} + (1 - p_{react})\rho_{dr} \tag{28}$$

The influence of the new solutions on the weights is controlled by the reaction factor p_{react}; Θ_{dr} counts how often pair dr was selected in the current segment.

3.3.3 Destroy sub-heuristics

Before the chosen destroy sub-heuristic can be applied we have to determine the number of customers to remove, u. We choose u randomly in the interval
$[\min\{0.1|N_f|, 30\}, \min\{0.4|N_f|, 60\}]$ as suggested in [13].

In the following sections, the applied destroy sub-heuristics are presented and their characteristics to work on different types of neighborhoods are pointed out. They are all based on [13, 16, 17].

Random removal The random removal operator randomly chooses customers and removes them from the template routes. This is done repeatedly until u customers have been removed. The aim of the operator is to diversify the search.

Worst removal The worst removal operator iteratively identifies and removes customers who contribute the most to the template's total travel time. The idea is to favor the reinsertion of customers at cheaper insertion positions than the current ones. Let h denote the predecessor of customer i on its current route and let j be the successor of i. Then,

$$d(i, \tau) = t_{hi} + t_{ij} - t_{hj} \tag{29}$$

represents the saving obtained by temporarily removing customer i from the template, τ. To avoid outlying customers to be removed over and over again the removal is randomized. All $d(i,\tau)$ values are sorted in list L in decreasing order. In every iteration customer $i := L[y^{p_{worst}}|L|]$ is removed. y is a random number in the interval $[0,1)$ and p_{worst} the parameter that controls the impact of randomization. The $d(i,\tau)$ values are updated and one customer is removed in every iteration until u customers have been removed.

Related removal The related removal operator [19] is based on the observation that it is easier to interchange customers within a schedule when they are somehow related. The relatedness, $\mathcal{R}(i,j)$, between two customers i and j, is a composed expression consisting of geographical vicinity, t_{ij}, difference in demand, q_{ai}, and similarity in visit frequency, γ_{ij}. The smaller $\mathcal{R}(i,j)$, the higher is the similarity between two customers. The importance of each measure is controlled by parameters λ, μ, ν, respectively.

$$\mathcal{R}(i,j) = \lambda t_{ij} + \mu|q_{ai} - q_{aj}| - \nu\gamma_{ij} \tag{30}$$

Relatedness, γ_{ij}, in terms of visit frequency is characterized by the number of days on which either both customers i and j are serviced or none of them.

$$\gamma_{ij} = |D| - \sum_{d\in D} |w_{id} - w_{jd}| \tag{31}$$

Including γ_{ij} in $\mathcal{R}(i,j)$ does not favor the interchangeability of the customers in the template. Yet, it supports the grouping of similar customers which again supports the compliance with time consistency when the template is resolved. The procedure is initialized by removing a randomly chosen customer from the template and inserting it into the set of removed customers S. In each iteration one customer is chosen randomly from S to calculate the $\mathcal{R}(i,j)$ values. Just like in the worst removal heuristic, the removal is randomized to obtain a certain degree of diversification. Therefore, all $\mathcal{R}(i,j)$ values are sorted in list L in increasing order, and customer $i := L[y^{p_{related}}|L|]$ is removed from the template and added to S. Again, y is a random number in the interval $[0,1)$ and $p_{related}$ is the parameter that controls the impact of randomization. The removal is continued until u customers have been removed.

Cluster removal The cluster removal operator [16] shares the idea of removing related customers. Hence, it can be interpreted as a variant of the related removal operator. Relatedness, however, is defined only in terms of geographical distance. The cluster removal operator removes complete clusters of customers even if u would be exceeded. To identify clusters, the minimal spanning tree among all customers in the same template route is defined [10]. By deleting the longest arc of the tree we obtain two clusters. One of them is selected randomly and all customers belonging to the selected cluster are removed.

The first route is selected randomly. The following routes are selected as follows. A customer from the currently removed cluster is chosen randomly. The next route to destroy is the one that contains the customer who has the smallest distance to the previously selected customer and has at least three customers. The procedure is continued either until the number of removed customers is at least u or no more routes with at least three customers exist.

3.3.4 Repair sub-heuristics

After customers have been removed, the greedy heuristic and four variants of the regret heuristic, as applied in [13, 16, 17], are used to reinsert all removed customers back into the template routes. The operators work at the template level. So, only the artificial tour length and capacity constraints, T_a and Q_a, are considered during the repair phase.

Greedy heuristics The greedy heuristic for constructing the initial template is also used as a repair sub-heuristic. It inserts customers, one after another, until all frequent customers are inserted. For each unassigned customer $\in N_f$ each feasible insertion position is checked and the customer who can be inserted the cheapest is assigned to his cheapest position. For further details see Section 3.2.

Regret heuristics Similar to the greedy approach, the regret heuristic [15] inserts frequent customers one after another by checking every feasible insertion position. It includes, however, a look ahead component denoted as regret. This value represents the possible loss that may arise if a customer's insertion is postponed to later iterations. In the basic variant of the heuristic, the customer with the largest difference between inserting him into his best route at the best position and inserting him into his second best route at the best position is inserted in every iteration. This concept is extendable to consider more than two routes [13, 17]. So, difficulties in future insertions can be identified earlier. Let Δf_i^q denote the change in the total travel time for inserting customer i at his cheapest position in his q-cheapest route. If it is not possible to insert a customer into a route, Δf_i^q is set to infinity. In every iteration, the customer i^* to be inserted is given by:

$$i^* := \arg\max_{i \in N_f} \left\{ \sum_{h=2}^{\min(q,m)} (\Delta f_i^h - \Delta f_i^1) \right\} \tag{32}$$

Parameter q defines the number of routes considered in the current regret heuristic variant and m is the number of currently available routes. As in the greedy heuristic, an empty route is added whenever it is not possible to assign further customers.

Following [14], we use four regret heuristics, each with a different setting for q with $q \in \{2, 3, 4, m\}$.

3.4 Further improvements

Due to the complexity of the ConVRP caused by daily interdependencies, it is difficult to predict reasonable insertion positions based on partial solutions. Therefore, and to increase the degree of diversification beyond the level of the simulated annealing criterion, we also randomize the route construction procedures as suggested in [17]. The randomization is done by drawing a random number, denoted as noise term, p_{noise}, in the interval $[-\eta \max_{i,j \in N \setminus \{0\}} t_{ij}, \eta \max_{i,j \in N \setminus \{0\}} t_{ij}]$ and adding it to the customers' insertion cost. Parameter η controls the amount of randomization. Every time a customer's insertion cost C is calculated, we add p_{noise} to obtain C'. Whether a repair sub-heuristic inserts customers according to C or C' into the template, depends on the past performance of the heuristics with and without noise. The decision is made by using the adaptive selection mechanism as described in Section 3.3.2. For the insertion of non-frequent customers during the template's resolution, the decision whether to apply noise or not is made with equal probability.

The TALNS focuses only on the construction of minimal cost template routes. Thereby, it relies on the template concept to obtain good and feasible solutions. However, a low cost template does not necessarily result in a low cost schedule. This is especially true when only a small portion of the customers is considered in the template. On the other hand, a solution may be influenced negatively if the majority of the customers is inserted into the template but many customers have to be deleted from the routes during the resolution. The mentioned issue increases with longer planning periods.

To deal with these difficulties, we apply a truncated 2-opt operator [12] to the daily routes every time we obtain a solution that at least complies with the tour length and the capacity restrictions. Truncated means that only sequences with a maximum of three customers may be reversed. Given the time consistency requirement that prevents the reversal of long sequences, a lot of computational time can be saved by the restriction.

The effect of this operator is twofold: first, the routes obtained after employing the 2-opt operator may not comply with the precedence principle anymore; that is, customers $\in N_f$ may no longer be visited in the same order on all days. Experiments by Groër et al. [7] for the ConVRP show that in 2 out of 10 instances deviations from the precedence principle are needed to obtain the optimal solution. Second, non-frequent customers who are not restricted by consistency may be moved to different positions in the near neighborhood. Moving these customers to different positions might fill the holes caused by the removal of excessive customers.

The 2-opt operator accepts only feasible modifications. Therefore, besides improving the objective function value, it may also repair previously infeasible solutions.

In contrast to the TALNS, the consistent record-to-record solution approach presented in [7] operates only at the template level and does not include any post optimization. So, the solutions adhere to the template's precedence principle. The template-based tabu search approach [22] integrates a post-optimization phase that is run every time a new template is resolved. Here, different neighborhood operators are used to move non-frequent customers to different positions. Frequent customers are not moved. Therefore, the precedence principle of the underlying template is also not altered.

4 The ConVRP with shiftable starting times

In this section, we propose an alternative to the original ConVRP model and two solution approaches for it. The original ConVRP is quite rigid since all vehicles have to leave the depot at time 0 and waiting times between customer visits are not allowed (see inequalities (3) and (9) in Section 2). The consequence of such a rigidness is illustrated in the following example. Let us consider two customers, 1 and 2, with service time 0 who have to be serviced over a planning period of three days. The maximum arrival time difference L is 1. Tour length and capacity is unbounded. On the first day both customers require service, on the second day only customer 1 and on the third day only customer 2. All possible connections require a travel time of 1 time unit. The optimal solution that requires one vehicle is presented in Figure 4.

When all vehicles have to leave the depot at time 0, the obtained solution results in a current maximum arrival time difference $l_{max} = 1$, and the total travel time $TT = 7$. If L was set to a value smaller than 1, two vehicles, one for each customer, would be needed to solve the example. The resulting cost would be $TT = 8$.
Now, consider that a delayed departure from the depot is allowed. In this case, as can be seen in Figure 5, the vehicle could depart at time 1 on day 3. This would result in a solution with $l_{max} = 0$.

In our relaxed ConVRP variant, the constraint for the starting times to be 0 is omitted (inequality (3)). That is, a later departure from the depot is allowed.

Given the altered model, solutions that are infeasible because of the time consistency requirement can be repaired by adjusting the vehicles' departure times and, therefore, the customers' arrival times. To integrate this concept into the solution method, we call a repair mechanism every time the TALNS produces a template solution that cannot be resolved feasibly because of the time consistency requirement. The repair mechanism then tries to make the daily plan feasible by shifting the routes' starting times. The template routes are not affected by the repair mechanism.

In the next two sections we propose an exact and a heuristic repair approach.

4.1 An exact repair approach

To optimize the starting times a_{0kd} of all tours on all days, we solve the following LP model with IBM's ILOG Cplex 12.1, Concert Technology 2.9.

$$Minimize \quad l_{max} \tag{33}$$

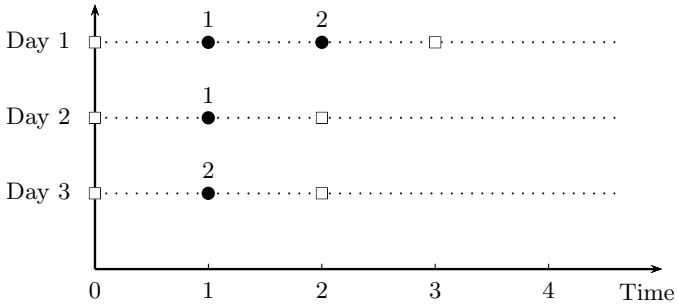

Figure 4: Solution: departure from depot = 0, $l_{max} = 1$

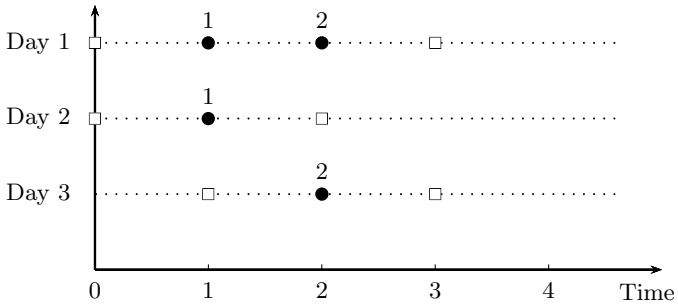

Figure 5: Solution: departure from depot is shiftable, $l_{max} = 0$

subject to

$$(a_{i\alpha} - a_{i\beta})w_{i\alpha}w_{i\beta} \leq l_{max} \qquad \forall\ i \in N_f, \alpha, \beta \in D, \alpha \neq \beta \tag{34}$$

$$a_{ikd} + s_{id} + t_{ij} = a_{jkd} \qquad \forall\ (i,j) \in A_{kd}, j \neq 0, d \in D, k \in K \tag{35}$$

$$a_{ikd} + w_{id}(s_{id} + t_{i0}) \leq Tw_{id} \qquad \forall\ i \in N\backslash\{0\}, k \in K, d \in D \tag{36}$$

$$a_{ikd} \geq 0 \qquad \forall\ i \in N, k \in K, d \in D \tag{37}$$

The objective (33) is to minimize the current maximum arrival time difference, l_{max}, to make the solution feasible. The arrival time difference for each frequent customer is defined by constraints (34). Equations (35) (derived from (8) and (9)) link the arrival times of consecutive customers where set A_{kd} gives the chosen arcs of route k on day d. Constraints (36) ensure that the vehicles return to their depot until T. To avoid deliveries beyond usual business hours we do not bound the tour duration but the time interval in which a tour can be conducted. Non-negativity is stated in constraints (37).

4.2 A heuristic approach

To repair a routing plan heuristically, we iteratively identify the customer i^* with the largest difference in his arrival times (computed as in constraints (34)). Let us denote the day when customer $i^{*\prime}s$ service starts earliest as α and the day when $i^{*\prime}s$ visit starts the latest as β. If $i^{*\prime}s$ route k on day β is already delayed, we first try to reduce that delay by decreasing the departure time from the depot as follows:

$$a_{0k\beta} = a_{0k\beta} - (l_{max} - L) \tag{38}$$

If this is not possible because of the non-negativity constraint (37), we try to delay route $k^{\prime}s$ starting time on day α:

$$a_{0k\alpha} = a_{0k\alpha} + (l_{max} - L) \tag{39}$$

The shifting of the starting times is performed until the routing plan is either repaired ($l_{max} \leq L$), l_{max} is not decreasing from one iteration to the next or when the latest return to the depot, T, would be exceeded.

5 Computational Results

The described algorithm is implemented in C++ and run on an Intel Xeon X5550 computer with 2.67GHz. Results are obtained by running the TALNS with 30000 iterations 10 times per instance. All results reported in the following sections are average values of these 10 runs.

5.1 Data sets

The data set used to test the described algorithm was proposed by Groër et al. [7]. It is based on the Christofides benchmark instances for the vehicle routing problem [2] and consists of 12 instances with a planning horizon of five days each. The number of customers range from 50 to 199 and the visit frequency is 70%. The visit frequency is the probability with which requests were assigned to customers and days during the instance generation. We refer to this data set as data set A.

To allow for more extensive tests, we extended the original benchmark data set and generated two modified sets. The modifications concern the visit frequency, which was set to 50% and 90%, respectively. The demands and the service times were set to their original values. The lower the visit frequency in the instance generation, the higher the probability that a customer has no request at all. In case a customer had no request, we assigned one visit on a random day.

In order to investigate the influence of the maximum arrival time difference constraint on the results, we tested four different L values for each instance. To define reasonable L values we ran the algorithm on all instances without bounding the arrival time differences. The vector of maximum arrival time differences obtained during these runs is denoted L_1. The vectors with the reduced maximum arrival time differences, $L_{0.8}$, $L_{0.6}$ and $L_{0.4}$, are calculated by multiplying L_1 by 0.8, 0.6 and 0.4, respectively. For easier handling we rounded all values to integers. We collect the modified instances in data set B.

As the real-world data set used by Groër et al. [7] is not available, we also adapted Gehring and Homberger's [6] benchmark instances for the vehicle routing problem with time windows (VRPTW) to test the algorithm's behavior on larger problem instances (data set C). These are divided into six different classes. Each class is characterized by a different spatial distribution of the customer locations (clustered (C1, C2), random (R1, R2) and random clustered (RC1, RC2)) and by different scheduling horizons (short scheduling horizon (R1, C1, RC1), long scheduling horizon (C2, R2, RC2)). We used one instance of each class with 1000 customers and extended them to a planning horizon of 25 days (i.e., five consecutive weeks with five days each). To turn the single period instances into periodic ones we chose a service frequency of 50%. The demands and service times were chosen randomly according to a Poisson distribution with the mean equal to the demands and service times in the original VRPTW instances. The time windows were omitted.

All generated instances are available at `http://prolog.univie.ac.at/research/ConVRP/`.

5.2 Parameter tuning

The TALNS is controlled by several parameters. The initial setting was chosen according to the values suggested by Ropke and Pisinger [17]. We then used the benchmark data set of Groër et al. [7] and their L values to fine tune the parameters. A sequential approach was used to tune one parameter after another. The parameter value that resulted in the best average objective value over five runs was chosen.

$w_{\hat{t}}$	c	p_{react}	σ_1	σ_2	σ_3	$p_{penalty}$	η	p_{worst}	$p_{related}$	λ	μ	ν
0.01	0.9999	0.18	24	3	4	2	0.025	2	4	6	7	12

Table 2: Parameter setting

Within the acceptance criterion, we tuned parameter $w_{\hat{t}}$ that controls the starting temperature, cooling rate c, and the penalty factor for infeasible solutions, $p_{penalty}$. The tuned parameters for the TALNS are the reaction factor p_{react} and the scores for performance σ_1, σ_2 and σ_3. In the repair phase, the noise parameter η was adjusted. In the destroy phase, the randomization parameters p_{worst} and $p_{related}$ and the weights for the relatedness elements λ, μ and ν were set. The parameters were tested in the same order as listed in Table 2. The sequential tuning of all parameters was repeated three times. Table 2 provides the final values.

The parameter tuning led to an improvement of 0.1% on average. Therefore, we conclude that the algorithm behaves robustly with respect to the chosen parameters.

5.3 Results for benchmark instances (data set A)

In a first step we examine the impact of the number of TALNS iterations on the solution quality. Table 3 shows the total travel times plus the aggregated service times [1], TT, of the best solutions found during the development of the algorithm. In the first column we indicate the instance names and the corresponding number of customers that have to be served. The remaining columns give the average gap to the best results found and the running time, CPU, in seconds for the solutions found after 10, 30, 50, and 70 thousand iterations, respectively.

Clearly, the solution quality improves with longer running times. To be competitive in terms of solution quality and running time we chose 30000 iterations for our experiments. This number of iterations is sufficient to obtain stable results over several runs. The resulting average variation coefficient (standard deviation / mean) is 0.0097 for a sample of 10 runs.

	Best found	10000 iterations		30000 iterations		50000 iterations		70000 iterations	
Instances (# Customers)	TT	Gap(%)	$CPU(s)$	Gap(%)	$CPU(s)$	Gap(%)	$CPU(s)$	Gap(%)	$CPU(s)$
Christofides_1_5_0.7 (50)	2124.21	6.09	1.80	3.33	5.45	1.80	9.25	1.45	12.79
Christofides_2_5_0.7 (75)	3530.01	3.00	4.92	2.13	14.69	1.75	24.35	1.56	33.74
Christofides_3_5_0.7 (100)	3285.55	2.79	8.74	1.60	25.58	1.44	43.46	1.39	61.22
Christofides_4_5_0.7 (150)	4484.89	4.27	29.13	2.54	84.31	2.10	139.14	2.05	189.63
Christofides_5_5_0.7 (199)	5556.13	4.31	44.84	2.33	122.24	2.05	199.47	1.93	269.09
Christofides_6_5_0.7 (50)	4051.48	0.01	2.19	0.00	6.63	0.00	11.11	0.00	15.41
Christofides_7_5_0.7 (75)	6653.48	3.51	6.20	2.29	18.33	2.11	30.52	1.93	43.21
Christofides_8_5_0.7 (100)	7096.88	1.96	10.65	1.35	32.24	1.29	53.41	1.22	75.53
Christofides_9_5_0.7 (150)	10331.80	3.06	34.32	1.14	97.39	0.96	153.42	0.76	206.32
Christofides_10_5_0.7 (199)	12973.60	3.76	54.81	2.04	146.32	1.67	232.13	1.55	316.02
Christofides_11_5_0.7 (120)	4459.06	2.33	13.15	1.07	35.96	0.69	56.96	0.68	76.73
Christofides_12_5_0.7 (100)	3487.50	2.24	8.92	1.22	25.60	1.08	42.72	0.88	58.45
Average	5669.55	3.11	18.31	1.75	51.23	1.41	82.99	1.28	113.18

Table 3: Impact of TALNS iterations on the solution quality, TT, and the computation time, CPU

An analysis of the destroy and repair sub-heuristics is presented in Tables 4 and 5, respectively. The tables list the application frequency (number of calls / TALNS iterations), AF, for each sub-heuristic for each instance. The average computation time, CPU, for a single call is given in milliseconds. The average application frequency and the average computation time over all instances are shown in the last row of the tables.

With regard to the destroy operators, the random removal sub-heuristic is chosen in the majority of the cases with 41% on average. This result indicates the need for diversification in

[1] We add the aggregated service times to the objective value to be consistent with the literature.

the ConVRP. The regret sub-heuristics with $q = 2$ and $q = 3$ are together used in 57% of the iterations to repair the destroyed templates. The application frequency of the least often applied destroy and repair operator is 10% and 11% respectively. These values show that all operators contribute to the actual performance of the algorithm. The repair operators are randomized in 49% of the cases (Section 3.4).

There are significant differences in the sub-heuristics' computation times that can lead to unpredictable total running times. The TALNS adapts the application frequency of the sub-heuristics on the basis of the test instances to be solved. Yet, it is not possible to estimate the share of each sub-heuristic in advance. As a result, instances with similar parameters (e.g., number of customers) may require different computation times. Stability in terms of running time can be achieved by restricting the selection of complicated sub-heuristic pairs, e.g., by normalizing the score ψ_{dr} in equation (28) ([13, 14]).

Stable running times are not a major concern in our applications. Therefore, we allow the adaptive mechanism to choose freely without any bias.

	Random removal		Worst removal		Related removal		Cluster removal	
Instances	*AF*	CPU(ms)	*AF*	CPU(ms)	*AF*	CPU(ms)	*AF*	CPU(ms)
Christofides_1_5_0.7	0.37	0.002	0.23	0.066	0.30	0.052	0.10	0.035
Christofides_2_5_0.7	0.48	0.003	0.12	0.153	0.30	0.128	0.11	0.037
Christofides_3_5_0.7	0.37	0.004	0.25	0.296	0.31	0.230	0.07	0.085
Christofides_4_5_0.7	0.49	0.006	0.18	0.751	0.28	0.584	0.04	0.098
Christofides_5_5_0.7	0.52	0.008	0.21	1.151	0.23	0.872	0.04	0.126
Christofides_6_5_0.7	0.30	0.003	0.25	0.064	0.27	0.045	0.17	0.029
Christofides_7_5_0.7	0.47	0.003	0.15	0.163	0.17	0.130	0.22	0.030
Christofides_8_5_0.7	0.38	0.004	0.24	0.307	0.31	0.237	0.08	0.074
Christofides_9_5_0.7	0.41	0.005	0.12	0.714	0.33	0.545	0.14	0.078
Christofides_10_5_0.7	0.41	0.007	0.09	1.119	0.36	0.836	0.15	0.110
Christofides_11_5_0.7	0.36	0.005	0.26	0.450	0.34	0.341	0.03	0.139
Christofides_12_5_0.7	0.42	0.004	0.17	0.281	0.34	0.233	0.07	0.058
Average	0.41	0.004	0.19	0.460	0.30	0.353	0.10	0.075

Table 4: Analysis of destroy sub-heuristics (*AF* denotes the application frequency of the respective sub-heuristic)

	Greedy heuristic		Regret heuristic q=2		Regret heuristic q=3		Regret heuristic q=4		Regret heuristic q=m	
Instances	*AF*	CPU(ms)	*AF*	CPU(ms)	*AF*	CPU(ms)	*AF*	CPU(ms)	*AF*	CPU(ms)
Christofides_1_5_0.7	0.21	0.056	0.22	0.089	0.19	0.084	0.19	0.091	0.20	0.095
Christofides_2_5_0.7	0.10	0.176	0.36	0.366	0.28	0.357	0.20	0.357	0.06	0.364
Christofides_3_5_0.7	0.13	0.370	0.27	0.578	0.23	0.569	0.23	0.574	0.14	0.609
Christofides_4_5_0.7	0.12	1.503	0.30	2.358	0.23	2.402	0.24	2.439	0.11	2.514
Christofides_5_5_0.7	0.04	2.495	0.41	3.437	0.31	3.538	0.20	3.626	0.04	3.889
Christofides_6_5_0.7	0.19	0.071	0.23	0.111	0.20	0.116	0.19	0.114	0.19	0.118
Christofides_7_5_0.7	0.15	0.297	0.31	0.526	0.24	0.526	0.22	0.544	0.09	0.582
Christofides_8_5_0.7	0.16	0.536	0.26	0.827	0.22	0.836	0.20	0.834	0.16	0.827
Christofides_9_5_0.7	0.07	1.743	0.39	2.662	0.29	2.938	0.20	2.909	0.05	2.946
Christofides_10_5_0.7	0.10	2.655	0.48	4.263	0.20	4.526	0.17	4.554	0.05	4.886
Christofides_11_5_0.7	0.05	0.550	0.32	0.808	0.30	0.824	0.23	0.855	0.10	0.854
Christofides_12_5_0.7	0.12	0.381	0.24	0.627	0.23	0.612	0.26	0.623	0.15	0.650
Average	0.12	0.903	0.32	1.388	0.25	1.444	0.21	1.460	0.11	1.528

Table 5: Analysis of repair sub-heuristics (*AF*, denotes the application frequency of the respective sub-heuristic)

In a next step we compare the results of different TALNS configurations: the TALNS as described in Section 3, the TALNS without using the truncated 2-opt operator from Section 3.4, and the TALNS with the exact repair mechanism (TALNS+ER) and the heuristic repair mechanism (TALNS+HR).

We also examine a variant of the TALNS that allows a partial violation of the driver consistency. More precisely, we perform a post processing step on the final solution found by the

TALNS (TALNS+PP). In this step a small subset of the customers may be assigned to a second driver to avoid extremely expensive insertion positions. We sort all customers in decreasing order of the saving that could be obtained by temporarily removing them on the day he causes the highest insertion cost (equation (29)). The customers are then reassigned, one after another, to their cheapest position in a different driver's route. The insertion is performed by the greedy heuristic described in Section 3.2. The procedure stops when 5% of the customers are served by two drivers or if the solution cannot be improved by further reassignments.

In Table 6 we report the total travel times plus the aggregated service times, TT, and the computation times, CPU, for the mentioned TALNS configurations on the 12 benchmark instances from data set A. The table shows the benefit of the truncated 2-opt operator that comes with a small increase in the average computation time. As expected, the variants that integrate a method to repair infeasible solutions perform better than the pure TALNS. Nevertheless, there is little difference between the average results of the TALNS and the TALNS+ER. This is because the wide maximum arrival time difference of the instances allows good results even without adjusting the departure times. Noticeable is the comparison between TALNS+ER and TALNS+HR since for some instances the heuristic repair achieves better results than the exact repair approach. There are several reasons for this observation. First, due to the wide maximum arrival time differences the repair approaches play only a minor role. Second, the randomization of the TALNS causes the repair methods to face different solutions with different opportunities to adjust the departure times. Furthermore, the repair methods themselves generate different current maximum arrival time differences, l_{max}. Since l_{max} is considered in the acceptance criterion (equation (26)), it also effects the score ψ_{dr} that can be earned by the destroy and repair sub-heuristics.

The application of a post processing phase that allows a partial violation of the driver consistency (TALNS+PP) improves the TALNS solutions by 0.34% on average. The degree of this improvement indicates the ability of the TALNS to avoid extremely inconvenient insertion positions despite the strict driver consistency requirement.

	TALNS wo. 2-opt		TALNS		TALNS+ER		TALNS+HR		TALNS+PP	
Instances	TT	$CPU(s)$	TT	$CPU(s)$	TT	$CPU(s)$	TT	$CPU(s)$	TT	$CPU(s)$
Christofides_1_5_0.7	2136.20	4.93	2194.93	5.45	2130.99	40.50	2128.76	5.50	2186.05	5.45
Christofides_2_5_0.7	3605.04	14.65	3605.03	14.69	3589.88	30.04	3584.70	15.12	3580.13	14.69
Christofides_3_5_0.7	3361.24	24.26	3338.03	25.58	3324.94	81.86	3330.32	26.23	3332.53	25.58
Christofides_4_5_0.7	4617.69	80.97	4598.78	84.31	4600.26	132.90	4575.78	86.86	4584.61	84.31
Christofides_5_5_0.7	5695.15	125.54	5685.55	122.24	5706.27	161.86	5739.10	132.15	5662.88	122.24
Christofides_6_5_0.7	4064.34	5.79	4051.50	6.63	4051.48	20.57	4051.48	6.65	4050.86	6.63
Christofides_7_5_0.7	6828.52	18.61	6805.99	18.33	6779.03	26.53	6779.03	19.74	6756.08	18.33
Christofides_8_5_0.7	7234.34	30.28	7192.45	32.24	7199.34	49.38	7207.18	32.63	7173.27	32.24
Christofides_9_5_0.7	10524.83	94.83	10450.04	97.39	10488.87	110.40	10488.87	97.97	10411.60	97.39
Christofides_10_5_0.7	13273.72	148.74	13238.57	146.32	13193.79	207.79	13185.10	144.05	13200.49	146.32
Christofides_11_5_0.7	4531.70	34.65	4506.59	35.96	4481.99	130.18	4481.91	37.66	4499.16	35.96
Christofides_12_5_0.7	3545.98	23.65	3530.16	25.60	3543.34	63.50	3532.73	25.87	3519.06	25.60
Average	5784.90	50.57	5766.47	51.23	5757.52	87.96	5757.08	52.53	5746.39	51.23

Table 6: Comparison of TALNS variants

To test the performance of the developed algorithm, we compare it to the template based record-to-record travel algorithm for the ConVRP (ConRTR) [7] and to the tabu search algorithm (TTS) [22] that is also template based. The results of the ConRTR and the TTS are taken from the respective papers.

The performance of the three algorithms for the 12 benchmark instances is listed in Table 7. The first column gives the names of the test instances. In the following columns, TT and the obtained maximum arrival time difference, l_{max}, are listed for all algorithms. The TTS is stochastic and the reported results are the best of five runs (TT_{min}). The ConRTR algorithm is deterministic. For the TALNS, we report the average results over 10 runs (TT_{avg}) and the best of five randomly chosen runs (TT_{min}). The computation time of the best of five runs is

given for TTS (CPU_{min}) and the average computation time is given for the TALNS (CPU_{avg}). The computation times of the ConRTR are not available. The last two columns show the gap of the results produced by our TALNS to those of the ConRTR (with respect to TT_{avg}) and the TTS (with respect to TT_{min}). In the last row, the average results over all instances are given.

Since the results reported for the ConRTR approach were obtained without bounding the maximum arrival time differences, the L values for the TTS and the TALNS are set to the l_{max} values produced by the ConRTR method.

The authors of the ConRTR pointed out that their aim was to develop a simple algorithm that relies on the template concept. The structure of the TALNS is more complex. Yet, it generates solutions that are on average 5.84% better than those of the ConRTR. Furthermore, all results obtained by the ConRTR are improved.

When comparing the TALNS to the TTS approach the difference between the results is smaller. The TALNS performs on average 1.89% better than the TTS. Additionally, on a similar processor the TTS requires an average computation time of 408 seconds while the TALNS requires only 51.

The master and daily scheduler heuristic for the courier delivery problem [21] was also applied to solve the ConVRP benchmark instances. As the authors do not bound the maximum arrival time difference only instances in which TALNS obtains smaller or equal l_{max} values are compared. This is the case in 9 out of 12 instances. Here, TALNS improves the total travel time by 6.5% on average and it decreases l_{max} by 28.7%.

These comparisons indicate that the TALNS is a competitive solution method for the ConVRP.

	ConRTR [7]		TTS [22]			TALNS					
Instances	TT	l_{max}	TT_{min}	l_{max}	$CPU_{min}(s)$[1]	TT_{avg}	TT_{min}	l_{max}	$CPU_{avg}(s)$[2]	Gap (%) to ConRTR[3]	Gap (%) to TTS[4]
Christofides_1_5_0.7	2282.14	24.38	2210.56	21.99	80.00	2194.93	2124.21	23.72	5.45	-3.82	-3.91
Christofides_2_5_0.7	3872.86	34.26	3622.71	27.75	93.00	3605.03	3600.41	31.86	14.69	-6.92	-0.62
Christofides_3_5_0.7	3628.22	22.87	3451.10	21.92	369.00	3338.03	3326.12	22.21	25.58	-8.00	-3.62
Christofides_4_5_0.7	4952.91	27.53	4572.00	25.15	388.00	4598.78	4556.33	24.19	84.31	-7.15	-0.34
Christofides_5_5_0.7	6416.77	26.93	5732.62	19.99	550.00	5685.55	5664.06	22.69	122.24	-11.40	-1.20
Christofides_6_5_0.7	4084.24	63.47	4096.87	55.38	70.00	4051.50	4051.48	63.26	6.63	-0.80	-1.11
Christofides_7_5_0.7	7126.07	83.96	6752.36	63.28	161.00	6805.99	6770.49	76.62	18.33	-4.49	0.27
Christofides_8_5_0.7	7456.19	73.04	7279.39	62.01	539.00	7192.45	7129.79	65.97	32.24	-3.54	-2.06
Christofides_9_5_0.7	11033.54	106.43	10585.10	84.76	947.00	10450.04	10381.9	88.85	97.39	-5.29	-1.92
Christofides_10_5_0.7	13916.80	60.17	13120.40	57.17	1052.00	13238.57	13102.7	57.95	146.32	-4.87	-0.13
Christofides_11_5_0.7	4753.89	16.10	4721.09	15.68	480.00	4506.59	4485.37	15.33	35.96	-5.20	-4.99
Christofides_12_5_0.7	3861.35	17.58	3607.88	16.91	172.00	3530.16	3497.93	16.50	25.60	-8.58	-3.05
Average	6115.42	46.39	5812.67	39.33	408.42	5766.47	5724.23	42.43	51.23	-5.84	-1.89

[1] Running times in seconds on a 2.8 GHz Intel Xeon CPU
[2] Running times in seconds on a 2.67 GHz Intel Xeon CPU
[3] With respect to TT_{avg}
[4] With respect to TT_{min}

Table 7: Comparison to existing approaches on data set A

5.4 Results for new instances (data set B)

In this section we show the effect of decreasing maximum arrival time differences on the results obtained by the pure TALNS and the TALNS with the integrated repair mechanisms. Therefore, we use the newly generated instances with varying visit frequencies and maximum arrival time differences (data set B).

Tables 8, 9 and 10 contain the average results of the TALNS variants over all modified benchmark instances with service frequencies 0.5, 0.7 and 0.9, respectively.[2] The first column gives the L vectors (Section 5.1) that are applied on the corresponding instances. In the following columns, the average results achieved by the TALNS with the exact repair mechanism

[2]Detailed results for data set B are reported in Appendix A.2.

	TALNS+ER	$CPU(s)$	TALNS+HR	$CPU(s)$	Gap (%) to TALNS+ER	TALNS	$CPU(s)$	Gap (%) to TALNS+ER
L_1	4192.47	34.01	4192.47	34.01	0.00	4192.47	34.01	0.00
$L_{0.8}$	4204.81	115.21	4207.88	35.73	0.07	4215.56	35.15	0.26
$L_{0.6}$	4222.67	132.13	4218.23	34.97	-0.11	5176.83	47.52	22.60
$L_{0.4}$	4260.50	134.23	4421.42	39.80	3.78	10085.78	92.92	136.73
Average	4220.11	103.90	4260.00	36.13	0.94	5917.66	52.40	39.89
Gap (%) $L_1 - L_{0.4}$	1.62		5.46			140.57		

Table 8: Comparison of TALNS variants on data set B with service frequency=0.5

(TALNS+ER), with the heuristic repair mechanism (TALNS+HR) and without any repair mechanism are presented with the corresponding computation times, CPU. The gaps to the results obtained by the TALNS+ER are given for the TALNS+HR and the pure TALNS. In the last but one row the average results over all L vectors are shown. The last row gives the gaps between the results obtained with L_1 and $L_{0.4}$ constraints for the three solution approaches.

A comparison between the different solution approaches reveals large differences in the results and highlights interesting saving potentials for companies that are facing similar problems.

The pure TALNS is able to provide good solutions for wide L constraints. However, when the constraint gets tighter it is more difficult to obtain low travel time solutions. In the worst case, a 60% decrease in L leads to a 186.16% increase in the total travel time (see Table 9). The substantial increase in the objective values as well as in the computation times is due to the high number of vehicles needed to cope with the tighter arrival time difference constraints.

The possibility to shift the vehicles' arrival times enables the generation of solutions that are almost independent of the maximum allowed arrival time difference. With small differences, this is true for the exact and the heuristic adjustment of the departure times. The total travel time only increased on average between 0.63% and 1.62% with the TALNS+ER method and between 1.55% and 5.46% with the TALNS+HR method, while the L values decreased by 60%. As explained above, the results obtained by the TALNS+HR method might be better than those of TALNS+ER due to the randomization of the TALNS and the effect of the maximum arrival time difference on the scores that can be earned by the repair and destroy sub-heuristics.

The effect of decreased interdependence between the total travel time and the maximum arrival time difference seems to increase with the service frequency. Test instances with high service frequency have a larger number of frequent customers. Therefore, the corresponding template contains more customers and is not altered much during the resolution. A high service frequency enables the generation of a template that gives a more accurate reflection of the daily schedules and produces better results.

To obtain good solutions for instances with lower service frequencies, interdependencies during the insertion of non-frequent customers must be considered. One can think of interdependencies when the insertion of a customer would lead to a violation of the time consistency, but the next insertion on another day could repair the previous infeasibility. The TALNS considers this difficulty only indirectly by randomizing the insertion through the noise term and applying a truncated 2-opt operator (Section 3.4). The tabu search algorithm by Tarantilis et al. [22] integrates a post optimization phase in which exclusively non-frequent customers are moved to different insertion positions. Unfortunately, it is not possible to compare these two approaches on the basis of the existing benchmark instances: the service frequency of 70% results in instances with only 3% non-frequent customers on average. Problem instances with a higher number of non-frequent customers would be more adequate for a meaningful comparison.

5.5 Results for large instances (data set C)

We showed that the TALNS performs well when solving short planning horizon problems. This approach, however, is not appropriate when considering a dynamic long-term environment in

	TALNS+ER	$CPU(s)$	TALNS+HR	$CPU(s)$	Gap (%) to TALNS+ER	TALNS	$CPU(s)$	Gap (%) to TALNS+ER
L_1	5752.85	52.01	5752.85	52.01	0.00	5752.85	52.01	0.00
$L_{0.8}$	5760.89	121.58	5756.37	53.28	-0.08	5770.88	50.70	0.17
$L_{0.6}$	5773.59	149.36	5763.57	52.70	-0.17	6266.31	61.51	8.53
$L_{0.4}$	5811.67	147.40	5894.96	52.35	1.43	16462.33	133.28	183.26
Average	5774.75	117.59	5791.94	52.59	0.30	8563.09	74.38	47.99
Gap (%) $L_1 - L_{0.4}$	1.02		2.47			186.16		

Table 9: Comparison of TALNS variants on data set B with service frequency=0.7

	TALNS+ER	$CPU(s)$	TALNS+HR	$CPU(s)$	Gap (%) to TALNS+ER	TALNS	$CPU(s)$	Gap (%) to TALNS+ER
L_1	6729.94	55.75	6729.94	55.75	0.00	6729.94	55.75	0.00
$L_{0.8}$	6730.55	120.92	6735.30	60.00	0.07	6757.17	54.70	0.40
$L_{0.6}$	6734.61	170.40	6743.22	56.42	0.13	7047.31	62.80	4.64
$L_{0.4}$	6772.21	180.96	6833.95	58.19	0.91	15113.41	149.77	123.17
Average	6741.83	132.01	6760.60	57.59	0.28	8911.96	80.76	32.05
Gap (%) $L_1 - L_{0.4}$	0.63		1.55			124.57		

Table 10: Comparison of TALNS variants on data set B with service frequency=0.9

which we have to create routing plans for consecutive planning periods. Solving each period separately would disregard both, long-term time and driver consistency. Figure 6 shows an example in which customers are serviced over several planning periods. The service requirements and the customers are changing over time. As a consequence, the solution algorithm is rerun for each planning period (1-7). The changing input data leads to different templates and therefore to inconsistent solutions.

We follow Groër et al. [7] to cope with this issue and design a template based on historical data and apply it to derive the solutions for future planning periods. This approach is illustrated in Figure 7. A template is generated by using customer data from the past periods 1 to 4. The template is then used to produce solutions in periods 5, 6, and 7.

By using the same template for several periods it is guaranteed that all customers who are represented in the template obtain a consistent service. However, the service consistency of new customers who are not represented in the template is not considered. The same is true for customers who have to be removed from the template to make it feasible. We are talking about an infeasible template if its resolution produces routes that do not satisfy the capacity or tour length constraints. All frequent customers that are not part of the template are inserted at their best insertion positions only with regard to the total travel time.

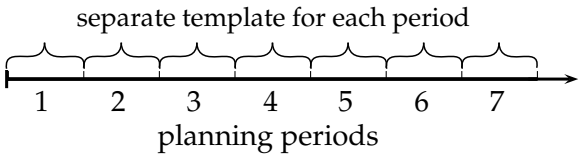

Figure 6: Solving each period separately results in inconsistent long-term solutions

In this section, we investigate the algorithm's behavior when the maximum arrival time difference, L, cannot be fixed to a reasonable value in advance. Typically, this is the case in long-term planning environments with changing customer requests. We perform experiments on the large instances (data set C) without bounding L and examine the consistency of the resulting solutions. Additionally, we compare the solutions obtained by using the historic template with those obtained by using a template based on current data.

Tables 11 and 12 show the average results of five runs for week five of the large instances.

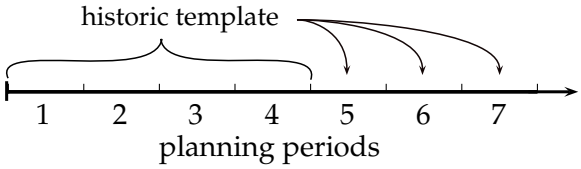

Figure 7: Consistent long-term solutions can be generated by using the same historic template

Like Groër et al. [7] we rely on the template's precedence principle to obtain time consistent routing plans even when the maximum arrival time difference is not bounded explicitly.

The results in Table 11 were obtained without using the 2-opt operator and those in Table 12 with using it. The first columns of both tables indicate the instance names and the corresponding maximum shift length, T. T can be interpreted as a trivial upper bound for the maximum arrival time difference. In columns 2 and 3 we give the total travel time plus service times, TT, and the number of vehicles, NV. Columns 4-7 show the average arrival time difference, l_{avg}, the maximum arrival time difference, l_{max}, the ratio of l_{max} and T, and the computation time in minutes, CPU, when depot departure times are fixed to 0 (TALNS). Columns 8-11 show the same items but with variable depot departure times (TALNS+ER). We have no explicit bound on the arrival time differences to adhere to, so we apply the exact repair approach described in Section 4.1 to the final solution only. Accordingly, the adjustment of departure times hardly effects the total computation time.

The results highlight the tradeoff between total travel time and the maximum arrival time differences. The application of the truncated 2-opt operator (reverse at the maximum three customers) can decrease the total travel time at the expense of increased arrival time differences: TT decreases by 0.75% on average while the average ratio of l_{max} and T increases from 19% to 23%. A preference to use one of the two approaches depends on the decision makers, so no evaluation can be made here. What can be evaluated is that the possibility to shift the vehicles' departure times can almost halve the maximum arrival time difference and should be considered whenever possible.

To investigate the TALNS' ability to produce templates for future planning periods we solve the ConVRP for week 1 - 4 (planning horizon $|D| = 20$ days) and apply the obtained template to solve week 5. Our aim is to provide long-term consistency for the most frequent customers. Therefore, all customers with an average of at least two visits per week are included into the template. To resolve the template for week 5, all customers that are represented in the template but have no service request on a specific day are removed. If the capacity or tour length constraints are exceeded despite of these removals, further customers are deleted from the solution. We repeatedly remove the customer with the lowest service frequency (in week 5) until all routes become feasible. Finally, we insert all customers that have been removed or are not represented in the template but require service by using the greedy heuristic (Section 3.2). This insertion happens without considering consistency.

Columns 2 and 3 of Table 13 show the number of customers considered in the template derived from week 1-4 and the computation time in minutes, CPU. The remaining columns show the number of frequent customers $|N_f|$ in week 5 and the number of frequent customers that are not represented in the template.

Tables 14 and 15 give the results of week 5 produced from the historic template. The results in Table 14 are obtained by resolving the template as described above. The results in Table 15 are obtained by additionally performing a post-optimization with the truncated 2-opt operator. The structure of the tables is similar to Tables 11 and 12. The only exceptions are that the computation times refer to the time needed for the template resolution in seconds and that the number of customers who have been removed to make routes feasible is also indicated.

			TALNS				TALNS+ER			
Instances (T)	TT	NV	l_{avg}	l_{max}	l_{max}/T	$CPU(min)$	l_{avg}	l_{max}	l_{max}/T	$CPU(min)$
R1_10_1_week_5 (1925)	325789.00	66.80	42.89	432.74	0.22	25.22	42.68	205.18	0.11	25.22
R2_10_1_week_5 (7697)	95627.32	10.40	68.54	331.22	0.04	36.26	55.76	168.97	0.02	36.26
C1_10_1_week_5 (1824)	551133.60	90.20	128.44	674.08	0.37	28.82	117.82	327.01	0.18	28.82
C2_10_1_week_5 (3914)	319889.60	18.20	232.81	1166.82	0.30	36.46	205.17	639.34	0.16	36.46
RC1_10_1_week_5 (1821)	322549.40	66.20	43.01	343.06	0.19	27.72	41.78	163.98	0.09	27.72
RC2_10_1_week_5 (7284)	94075.44	10.40	58.70	306.72	0.04	31.87	52.64	157.62	0.02	31.87
Average	284844.06	43.70	95.73	542.44	0.19	31.06	85.97	277.02	0.10	31.06

Table 11: Results for week 5 of the large instances without 2-opt

			TALNS				TALNS+ER			
Instances (T)	TT	NV	l_{avg}	l_{max}	l_{max}/T	$CPU(min)$	l_{avg}	l_{max}	l_{max}/T	$CPU(min)$
R1_10_1_week_5 (1925)	323128.60	68.20	49.93	473.47	0.25	30.08	51.18	281.09	0.15	30.08
R2_10_1_week_5 (7697)	95599.68	10.20	65.70	320.73	0.04	32.62	57.18	170.76	0.02	32.62
C1_10_1_week_5 (1824)	544885.20	88.60	138.49	810.66	0.44	34.67	149.39	521.22	0.29	34.68
C2_10_1_week_5 (3914)	319547.80	18.00	240.59	1290.64	0.33	36.09	217.57	673.77	0.17	36.09
RC1_10_1_week_5 (1821)	319051.80	65.00	49.04	509.51	0.28	28.08	50.16	294.19	0.16	28.08
RC2_10_1_week_5 (7284)	94053.22	10.20	53.46	259.77	0.04	31.67	47.00	133.65	0.02	31.67
Average	282711.05	43.37	99.54	610.80	0.23	32.20	95.41	345.78	0.13	32.20

Table 12: Results for week 5 of the large instances with 2-opt

When comparing the solutions from Tables 14 and 15 to those in Tables 11 and 12 one must pay attention to two antagonistic factors that affect the results. On the one hand, the solutions in Tables 14 and 15 are derived from a template based exclusively on historical data. This is obviously suboptimal in terms of total travel time even though it enables long-term consistency. On the other hand, there is a higher degree of freedom because almost 70 customers (on average) do not adhere to any consistency requirement. These customers are placed at their best positions regarding only total travel time. Therefore, one must keep in mind that a low total travel time is always associated with low consistency at least for some customers.

If a decision maker is willing to make this compromise, TALNS is applicable. It produces historic templates that can be applied to future planning periods with approximately 1.3% increase in the total travel times compared to solutions produced from a specially built template. With respect to time consistency both templates, historic and current, lead to comparable average arrival time differences. The repair mechanism to adjust the starting times can again almost halve the maximum arrival time differences. The minimization of l_{max}, however, can cause the average maximum arrival time difference to increase in some cases (see, e.g., instances R2 and RC2 in Table 15). Furthermore, driver consistency is not guaranteed for approximately 7% of the customers.

	template (week 1-4)		week 5	
Instances	# customers	CPU(min)	$\|N_f\|$	$\|N_f\|$ not in template
R1_10_1	864	73.85	821	52
R2_10_1	876	142.71	810	58
C1_10_1	850	160.96	820	72
C2_10_1	875	123.92	806	65
RC1_10_1	860	69.39	819	75
RC2_10_1	856	182.35	816	74
Average	863.50	125.53	815.33	66.00

Table 13: Properties of historic templates and week 5 of the large instances

Instances (T)	TT	NV	removed	TALNS l_{avg}	TALNS l_{max}	TALNS l_{max}/T	TALNS CPU(s)	TALNS+ER l_{avg}	TALNS+ER l_{max}	TALNS+ER l_{max}/T	TALNS+ER CPU(s)
R1_10_1_week_5 (1925)	344930.80	75.80	7.20	39.93	368.61	0.19	0.00	38.63	179.71	0.09	0.05
R2_10_1_week_5 (7697)	99239.68	11.00	1.00	75.74	1684.64	0.22	0.10	226.76	924.34	0.12	0.15
C1_10_1_week_5 (1824)	552262.20	105.00	4.20	102.83	707.77	0.39	0.00	94.85	297.05	0.16	0.05
C2_10_1_week_5 (3914)	294020.40	19.60	1.20	223.38	1335.50	0.34	0.01	196.34	639.31	0.16	0.05
RC1_10_1_week_5 (1821)	343862.00	75.40	5.80	34.16	369.22	0.20	0.00	31.77	153.73	0.08	0.06
RC2_10_1_week_5 (7284)	91613.80	10.40	1.80	62.03	700.25	0.10	0.06	90.39	389.83	0.05	0.11
Average	287654.81	49.53	3.53	89.68	861.00	0.24	0.03	113.12	430.66	0.11	0.08

Table 14: Results for week 5 of the large instances produced from historic template without 2-opt

Instances (T)	TT	NV	removed	TALNS l_{avg}	TALNS l_{max}	TALNS l_{max}/T	TALNS $CPU(s)$	TALNS+ER l_{avg}	TALNS+ER l_{max}	TALNS+ER l_{max}/T	TALNS+ER $CPU(s)$
R1_10_1_week_5 (1925)	342652.00	75.80	7.20	44.91	406.60	0.21	0.02	44.23	235.33	0.12	0.08
R2_10_1_week_5 (7697)	98945.14	11.00	1.00	76.31	1685.97	0.22	0.11	243.57	928.88	0.12	0.17
C1_10_1_week_5 (1824)	551463.80	105.00	4.20	114.15	798.11	0.44	0.01	116.49	433.91	0.24	0.06
C2_10_1_week_5 (3914)	293263.80	19.60	1.20	227.95	1337.25	0.34	0.01	199.78	656.41	0.17	0.05
RC1_10_1_week_5 (1821)	342081.60	75.40	5.80	38.09	445.65	0.24	0.01	37.88	261.42	0.14	0.06
RC2_10_1_week_5 (7284)	91297.00	10.40	1.80	61.72	694.00	0.10	0.07	89.69	388.64	0.05	0.12
Average	286617.22	49.53	3.53	93.86	894.59	0.26	0.04	121.94	484.10	0.14	0.09

Table 15: Results for week 5 of the large instances produced from historic template with 2-opt

6 Conclusion

Consistency improves service quality, service quality increases customer satisfaction, and customer satisfaction is one of the key factors of competitive advantage. The consistent vehicle routing problem (ConVRP), in response to this real world challenge, combines traditional vehicle routing constraints with the requirements for service consistency. In this article we first presented a solution approach called template based adaptive large neighborhood search (TALNS) for the described problem. It embeds the principle of deriving a multi-day routing plan from a set of template routes into the adaptive large neighborhood search framework. Experimental comparisons with other algorithms proved the competitiveness of the TALNS in terms of solution quality and computation time. The algorithm provides slightly better results than the best competitor while being about 8 times faster, thus representing a new state-of-the-art.

We construct and provide additional benchmark instances by varying customer service frequencies and maximum arrival time differences. This is important, since the maximum arrival time differences were determined by Groër et al. [7] by first ignoring any arrival time differences, and then defining what was obtained as maximum arrival time differences. Consequently these maximum arrival time differences are not really tight. Also, in the original benchmark instances by Groër et al. [7], the service frequencies were rather high so that only very few (on average 3%) non-frequent customers occurred. By varying these model parameters more challenging instances are obtained and interesting effects can be identified.

We used our TALNS to investigate the effect of different customer service frequencies and varying maximum arrival time differences on the total travel times. The numerical experiments showed that the TALNS is capable of generating very good solutions when the time consistency is loose. With tight maximum arrival time difference constraints, however, the cost increases sharply. We modified the original ConVRP model and allowed for a delay in the departure times from the depot. This realistic relaxation nearly caused a decoupling of total travel time and time consistency. In other words, while total cost increases sharply if the maximum arrival time difference constraint becomes very tight, this effect can be almost completely neutralized by appropriate adaption of the starting times of the routes. The important managerial implication for a service provider is that such a relaxation can lead to much higher service quality at

Instances	n	$\lvert N_f \rvert$ $x = 0.5$	$x = 0.7$	$x = 0.9$
Christofides_1_5_x	50	39	48	50
Christofides_2_5_x	75	59	74	75
Christofides_3_5_x	100	79	95	100
Christofides_4_5_x	150	116	149	150
Christofides_5_5_x	199	153	195	199
Christofides_6_5_x	50	40	48	50
Christofides_7_5_x	75	59	75	75
Christofides_8_5_x	100	79	98	100
Christofides_9_5_x	150	116	147	150
Christofides_10_5_x	199	153	193	199
Christofides_11_5_x	120	91	116	120
Christofides_12_5_x	100	79	97	100
Average	114	88.58	111.25	114

Table 16: Number of customers (overall and frequent). Wildcards x in the instance names can be replaced by the service frequencies 0.5, 0.7 and 0.9

practically no extra cost.

Future research in this field can be split into model-specific and solution method-specific topics. Concerning the model, the monetary differentiation between high and low quality services or extensions that balance the staff's hours of labor seem to be reasonable. In terms of solution methods, the template concept provides good solutions as long as the service frequency is high and so is the number of frequent customers. As the number of non-frequent customers increases, it becomes more important to find appropriate insertion approaches that are able to consider interdependencies over the entire planning horizon.

Acknowledgments The authors would like to thank Marc Reimann and the two anonymous referees for constructive comments and suggestions. The second author is supported by the Austrian Science Fund (FWF): T514-N13. This support is gratefully acknowledged.

A Appendix

A.1 Data set details

Table 16 presents the number of customers, n, that have to be served in the benchmark instances. Furthermore, the number of frequent customers, $|N_f|$, or customers who are considered in the template, is presented for the new and the original instances with the corresponding service frequencies, $x \in \{0.5, 0.7, 0.9\}$. The last row shows the average numbers of customers. The maximum arrival time difference, L, for each instance is listed in Table 17. The names of the instances are given in the first column. The following columns report the different L values for the instances with 50%, 70% and 90% service frequencies.

	x=0.5			x=0.7			x=0.9		
Instances	$L_{0.8}$	$L_{0.6}$	$L_{0.4}$	$L_{0.8}$	$L_{0.6}$	$L_{0.4}$	$L_{0.8}$	$L_{0.6}$	$L_{0.4}$
Christofides_1_5_x	54	41	27	34	26	17	32	24	16
Christofides_2_5_x	41	31	20	35	26	17	20	15	10
Christofides_3_5_x	46	34	23	30	22	15	23	17	11
Christofides_4_5_x	37	28	18	22	17	11	18	14	9
Christofides_5_5_x	31	23	15	28	21	14	10	7	5
Christofides_6_5_x	70	53	35	64	48	32	52	39	26
Christofides_7_5_x	61	46	30	67	50	33	56	42	28
Christofides_8_5_x	83	62	41	59	44	29	49	36	24
Christofides_9_5_x	78	59	39	71	53	35	43	32	21
Christofides_10_5_x	58	43	29	61	46	30	34	25	17
Christofides_11_5_x	38	29	19	15	11	7	16	12	8
Christofides_12_5_x	22	17	11	17	13	8	10	7	5

Table 17: Maximum arrival time differences. Wildcards x in the instance names can be replaced by the service frequencies 0.5, 0.7 and 0.9

A.2 Detailed results for data set B

Detailed results for all instances in data set B with varying maximum arrival time differences are given in Tables 18 - 26. The first three tables include the problem instances with a service frequency of 50%, the next three the instances with 70% . The final three tables include the instances with 90% service frequency. Tables 18, 21 and 24 show the results obtained by the TALNS for the original ConVRP. Here, shifts in the departure times are not allowed. The results given in Tables 19, 22 and 25 show the results for the TALNS+ER method where the vehicles' departure times are adjusted by an exact approach and Tables 20, 23 and 26 illustrate the results obtained by the TALNS+HR method that integrates a heuristic shifting mechanism. The mentioned tables are organized as follows. The first column gives the instance names and the following column pairs give the total travel time plus the aggregated service times, TT, and the obtained maximum arrival time difference, l_{max}, for the respective maximum arrival time difference constraints, L. The last rows show the average results. The results for the instances where the maximum arrival time difference is not bounded (L_1) are only given for the TALNS without repair mechanism, since the repair mechanism is only called when there is a violation of L.

Instances	L_1		$L_{0.8}$		$L_{0.6}$		$L_{0.4}$	
	TT	l_{max}	TT	l_{max}	TT	l_{max}	TT	l_{max}
Christofides_1_5_0.5	1651.19	70.70	1668.58	51.65	1754.54	37.92	1850.18	25.44
Christofides_2_5_0.5	2588.58	59.33	2602.12	33.41	2600.87	30.03	3585.00	18.39
Christofides_3_5_0.5	2689.36	52.46	2687.97	41.81	2712.04	32.93	4348.61	22.18
Christofides_4_5_0.5	3397.97	56.61	3399.87	34.34	3731.7	26.88	14136.58	16.52
Christofides_5_5_0.5	4045.53	37.53	4082.70	28.44	4150.04	22.40	20650.27	11.68
Christofides_6_5_0.5	2868.67	91.14	2883.88	64.41	2946.17	51.26	3820.50	31.37
Christofides_7_5_0.5	4701.86	89.57	4705.68	57.84	4819.25	44.22	8360.45	25.24
Christofides_8_5_0.5	5348.96	95.35	5353.45	76.18	5371.67	60.27	9908.53	39.10
Christofides_9_5_0.5	7461.90	100.23	7488.12	74.09	7739.97	57.00	15777.09	36.49
Christofides_10_5_0.5	9366.86	87.50	9403.01	57.02	19942.71	41.90	24670.81	28.03
Christofides_11_5_0.5	3293.99	75.69	3395.02	26.18	3423.62	24.38	3633.71	18.53
Christofides_12_5_0.5	2894.83	25.78	2916.31	18.76	2929.35	16.04	10287.67	8.47
Average	4192.47	70.16	4215.56	47.01	5176.83	37.10	10085.78	23.45

Table 18: TALNS, departure times=0, service frequency=0.5

Instances	$L_{0.8}$		$L_{0.6}$		$L_{0.4}$	
	TT	l_{max}	TT	l_{max}	TT	l_{max}
Christofides_1_5_0.5	1661.55	35.53	1653.01	32.65	1685.19	25.80
Christofides_2_5_0.5	2605.23	28.62	2623.90	22.11	2619.03	16.01
Christofides_3_5_0.5	2692.92	43.48	2703.48	21.00	2705.88	20.93
Christofides_4_5_0.5	3399.49	25.45	3403.92	22.11	3543.11	16.83
Christofides_5_5_0.5	4072.17	28.38	4093.25	15.45	4090.38	13.17
Christofides_6_5_0.5	2873.02	43.15	2870.30	43.13	2878.42	34.71
Christofides_7_5_0.5	4702.95	58.02	4745.82	36.22	4764.38	24.84
Christofides_8_5_0.5	5349.66	75.80	5357.24	38.85	5355.78	36.71
Christofides_9_5_0.5	7478.44	60.67	7489.27	47.52	7493.43	36.53
Christofides_10_5_0.5	9387.27	41.42	9388.77	39.14	9628.98	28.19
Christofides_11_5_0.5	3330.48	29.58	3433.58	21.93	3456.14	16.01
Christofides_12_5_0.5	2904.55	18.26	2909.46	12.39	2905.22	9.32
Average	4204.81	40.70	4222.67	29.38	4260.50	23.25

Table 19: TALNS+ER, departure times are shiftable, service frequency=0.5

Instances	$L_{0.8}$		$L_{0.6}$		$L_{0.4}$	
	TT	l_{max}	TT	l_{max}	TT	l_{max}
Christofides_1_5_0.5	1654.75	54.00	1654.95	41.00	1679.40	27.00
Christofides_2_5_0.5	2602.80	35.12	2606.07	30.67	2598.77	20.00
Christofides_3_5_0.5	2693.32	43.26	2708.02	33.83	2713.47	23.00
Christofides_4_5_0.5	3399.12	36.18	3401.60	28.00	3698.60	18.00
Christofides_5_5_0.5	4086.47	28.03	4093.06	22.92	4107.91	15.00
Christofides_6_5_0.5	2871.14	70.00	2870.68	53.00	2923.31	35.00
Christofides_7_5_0.5	4703.34	59.22	4731.01	45.99	4758.82	30.00
Christofides_8_5_0.5	5353.49	78.62	5363.67	62.00	5370.33	41.00
Christofides_9_5_0.5	7486.77	75.69	7459.19	58.59	7528.82	39.00
Christofides_10_5_0.5	9393.91	57.42	9385.69	43.00	11363.27	29.00
Christofides_11_5_0.5	3339.07	32.81	3418.82	24.55	3397.00	18.87
Christofides_12_5_0.5	2910.37	19.95	2925.99	16.42	2917.28	11.00
Average	4207.88	49.19	4218.23	38.33	4421.42	25.57

Table 20: TALNS+HR, departure times are shiftable, service frequency=0.5

Instances	L_1		$L_{0.8}$		$L_{0.6}$		$L_{0.4}$	
	TT	l_{max}	TT	l_{max}	TT	l_{max}	TT	l_{max}
Christofides_1_5_0.7	2119.84	40.41	2123.50	31.44	2143.64	24.66	3075.78	15.68
Christofides_2_5_0.7	3575.76	44.25	3568.94	33.85	3661.66	25.49	7968.27	16.01
Christofides_3_5_0.7	3306.11	35.91	3320.43	28.17	3339.36	21.41	10198.49	13.80
Christofides_4_5_0.7	4561.01	31.56	4663.42	21.27	6129.96	16.01	26084.59	6.96
Christofides_5_5_0.7	5720.26	28.87	5703.62	23.60	5726.60	19.46	6604.87	13.47
Christofides_6_5_0.7	4051.12	77.34	4051.57	63.24	4063.18	47.23	4376.52	31.11
Christofides_7_5_0.7	6785.91	77.35	6832.47	58.54	6914.61	48.16	10354.65	31.85
Christofides_8_5_0.7	7208.72	67.66	7210.89	54.80	7774.82	43.10	16546.28	28.07
Christofides_9_5_0.7	10507.94	108.20	10511.70	66.15	10691.63	52.00	20072.92	33.62
Christofides_10_5_0.7	13194.36	79.32	13234.58	58.86	15795.68	45.70	33655.92	28.50
Christofides_11_5_0.7	4477.14	20.56	4500.79	14.67	5386.16	10.65	40213.56	4.96
Christofides_12_5_0.7	3526.07	20.44	3528.65	15.84	3568.47	12.62	18396.13	7.06
Average	5752.85	52.66	5770.88	39.20	6266.31	30.54	16462.33	19.26

Table 21: TALNS, departure times=0, service frequency=0.7

Instances	$L_{0.8}$		$L_{0.6}$		$L_{0.4}$	
	TT	l_{max}	TT	l_{max}	TT	l_{max}
Christofides_1_5_0.7	2123.81	29.64	2127.49	17.88	2131.91	15.00
Christofides_2_5_0.7	3579.36	25.77	3585.74	20.32	3615.43	15.43
Christofides_3_5_0.7	3313.12	27.58	3331.35	17.43	3343.10	14.28
Christofides_4_5_0.7	4632.09	15.84	4589.22	15.19	4691.71	10.50
Christofides_5_5_0.7	5707.10	23.12	5718.77	17.56	5716.18	10.67
Christofides_6_5_0.7	4051.48	63.30	4063.18	47.23	4064.34	31.73
Christofides_7_5_0.7	6794.35	47.10	6815.14	38.95	6833.13	28.76
Christofides_8_5_0.7	7234.27	48.92	7236.82	34.06	7225.29	27.39
Christofides_9_5_0.7	10465.20	55.72	10578.57	40.86	10579.78	29.98
Christofides_10_5_0.7	13210.41	42.46	13215.13	32.15	13217.64	28.71
Christofides_11_5_0.7	4486.11	10.11	4485.92	9.62	4772.98	6.40
Christofides_12_5_0.7	3533.38	16.57	3535.76	8.21	3548.52	6.31
Average	5760.89	33.84	5773.59	24.96	5811.67	18.76

Table 22: TALNS+ER, departure times are shiftable, service frequency=0.7

Instances	$L_{0.8}$		$L_{0.6}$		$L_{0.4}$	
	TT	l_{max}	TT	l_{max}	TT	l_{max}
Christofides_1_5_0.7	2122.48	31.41	2133.91	25.74	2134.52	17.00
Christofides_2_5_0.7	3584.01	33.32	3576.75	25.91	3604.38	17.00
Christofides_3_5_0.7	3318.44	27.88	3331.68	21.97	3352.33	15.00
Christofides_4_5_0.7	4576.47	21.75	4567.86	17.00	4843.61	11.00
Christofides_5_5_0.7	5724.67	22.81	5698.33	19.02	5727.61	14.00
Christofides_6_5_0.7	4051.48	63.33	4063.18	47.23	4064.34	32.00
Christofides_7_5_0.7	6800.61	66.82	6821.98	49.77	6841.60	33.00
Christofides_8_5_0.7	7227.17	54.30	7232.61	44.00	7246.52	29.00
Christofides_9_5_0.7	10489.90	67.84	10495.53	52.95	10540.45	35.00
Christofides_10_5_0.7	13172.06	60.07	13203.58	46.00	13428.93	30.00
Christofides_11_5_0.7	4484.17	14.93	4491.44	11.00	5373.77	7.00
Christofides_12_5_0.7	3524.95	16.79	3546.04	12.99	3581.51	8.00
Average	5756.37	40.10	5763.57	31.13	5894.96	20.67

Table 23: TALNS+HR, departure times are shiftable, service frequency=0.7

Instances	L_1		$L_{0.8}$		$L_{0.6}$		$L_{0.4}$	
	TT	l_{max}	TT	l_{max}	TT	l_{max}	TT	l_{max}
Christofides_1_5_0.9	2489.82	41.98	2491.17	29.67	2498.73	22.26	3223.89	15.15
Christofides_2_5_0.9	4042.34	25.60	4036.93	19.04	4094.53	12.23	4459.08	8.47
Christofides_3_5_0.9	3998.96	29.22	4014.12	22.47	4045.57	16.27	4325.71	10.36
Christofides_4_5_0.9	5045.35	20.15	5033.98	16.02	5067.58	13.03	25656.10	7.00
Christofides_5_5_0.9	6466.02	17.79	6704.79	7.93	6977.31	6.73	27586.40	3.17
Christofides_6_5_0.9	4761.20	80.46	4764.94	42.97	4814.62	38.39	6049.26	24.68
Christofides_7_5_0.9	7748.61	82.94	7744.94	43.60	7730.79	36.88	7944.97	27.55
Christofides_8_5_0.9	8744.73	64.61	8742.17	46.58	8842.28	34.59	12515.23	22.58
Christofides_9_5_0.9	12477.58	56.59	12480.13	41.83	13477.05	30.91	30716.88	19.81
Christofides_10_5_0.9	15971.12	41.21	16018.11	32.32	17936.27	24.11	47915.45	15.70
Christofides_11_5_0.9	5000.61	22.01	5041.08	14.22	5051.55	11.39	6791.61	7.69
Christofides_12_5_0.9	4013.00	13.43	4013.63	9.13	4031.45	4.94	4176.3	4.62
Average	6729.94	41.33	6757.17	27.15	7047.31	20.98	15113.41	13.90

Table 24: TALNS, departure times=0, service frequency=0.9

Instances	$L_{0.8}$		$L_{0.6}$		$L_{0.4}$	
	TT	l_{max}	TT	l_{max}	TT	l_{max}
Christofides_1_5_0.9	2491.14	29.40	2497.63	18.92	2499.17	11.71
Christofides_2_5_0.9	4037.79	18.14	4049.83	9.70	4045.92	9.07
Christofides_3_5_0.9	4018.52	19.97	4016.00	13.50	4018.88	9.49
Christofides_4_5_0.9	5042.75	16.43	5036.07	8.41	5027.71	7.97
Christofides_5_5_0.9	6473.41	6.71	6448.74	6.13	6623.67	3.88
Christofides_6_5_0.9	4764.65	43.36	4767.65	20.99	4767.65	20.99
Christofides_7_5_0.9	7749.79	48.19	7743.38	36.08	7741.18	15.93
Christofides_8_5_0.9	8737.30	46.40	8749.40	21.78	8758.24	21.95
Christofides_9_5_0.9	12472.95	34.16	12487.40	21.90	12648.34	19.09
Christofides_10_5_0.9	15908.79	28.64	15964.44	19.29	16076.38	16.14
Christofides_11_5_0.9	5055.98	14.13	5039.34	7.46	5043.93	7.46
Christofides_12_5_0.9	4013.50	8.94	4015.40	3.72	4015.40	3.72
Average	6730.55	26.21	6734.61	15.66	6772.21	12.28

Table 25: TALNS+ER, departure times are shiftable, service frequency=0.9

Instances	$L_{0.8}$		$L_{0.6}$		$L_{0.4}$	
	TT	l_{max}	TT	l_{max}	TT	l_{max}
Christofides_1_5_0.9	2492.30	29.01	2496.89	23.56	2504.55	16.00
Christofides_2_5_0.9	4030.44	18.40	4043.58	15.00	4055.95	10.00
Christofides_3_5_0.9	4005.73	22.38	4007.55	16.81	4031.40	11.00
Christofides_4_5_0.9	5045.04	16.44	5030.51	13.90	5070.25	9.00
Christofides_5_5_0.9	6481.25	9.57	6445.75	7.00	6668.00	5.00
Christofides_6_5_0.9	4764.13	44.82	4767.65	39.00	4767.65	26.00
Christofides_7_5_0.9	7738.28	41.97	7751.14	36.38	7759.23	28.00
Christofides_8_5_0.9	8739.99	45.61	8751.64	36.00	8756.29	24.00
Christofides_9_5_0.9	12476.39	41.51	12510.38	32.00	12940.78	21.00
Christofides_10_5_0.9	15992.53	33.42	16020.62	25.00	16393.93	17.00
Christofides_11_5_0.9	5040.46	14.69	5077.50	11.84	5044.01	8.00
Christofides_12_5_0.9	4017.07	8.88	4015.46	7.00	4015.40	5.00
Average	6735.30	27.23	6743.22	21.96	6833.95	15.00

Table 26: TALNS+HR, departure times are shiftable, service frequency=0.9

References

[1] Campbell, A. and Thomas, B. (2008). Probabilistic traveling salesman problem with deadlines. *Transportation Science*, 42(1):1–21.

[2] Christofides, N. and Eilon, S. (1969). An algorithm for the vehicle-dispatching problem. *OR*, 20(3):309–318.

[3] Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2012). Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, 24(1):270 – 287.

[4] Cordeau, J., Laporte, G., Pasin, F., and Ropke, S. (2010). Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, 13(4):393–409.

[5] Feillet, D., Garaix, T., Lehuédé, F., Péton, O., and Quadri, D. (2010). A new Consistent Vehicle Routing Problem for the transportation of handicapped persons. Technical Report emse-00529945 - version 1, Ecole Nationale Supérieure des Mines de Saint-Etienne, France.

[6] Gehring, H. and Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In Miettinen, K., Mäkelä, M., and Toivanen, J., editors, *Proceedings of EUROGEN99: Short Course on Evolutionary Algorithms in Engineering and Computer Science*, pages 57–64. University of Jyväskylä.

[7] Groër, C., Golden, B., and Wasil, E. (2009). The consistent vehicle routing problem. *Manufacturing & Service Operations Management*, 11(4):630–643.

[8] Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

[9] Kovacs, A. A., Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling*, 15(5):579–600.

[10] Kruskal, J. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50.

[11] Li, F., Golden, B., and Wasil, E. (2005). Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers & Operations Research*, 32(5):1165–1179.

[12] Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269.

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

[14] Pisinger, D. and Ropke, S. (2010). Large neighborhood search. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of Metaheuristics*, volume 146, pages 399–419. Springer, New York.

[15] 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.

[16] Ropke, S. and Pisinger, D. (2006a). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3):750–775.

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

[18] 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.

[19] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In Maher, M. and Puget, J.-F., editors, *Principles and Practice of Constraint Programming CP98*, volume 1520, pages 417–431. Springer, Berlin Heidelberg.

[20] Smilowitz, K., Nowak, M., and Jiang, T. (2012). Workforce management in periodic delivery operations. *Transportation Science*. doi: 10.1287/trsc.1120.0407.

[21] Sungur, I., Ren, Y., Ordez, F., Dessouky, M., and Zhong, H. (2010). A Model and Algorithm for the Courier Delivery Problem with Uncertainty. *Transportation Science*, 44(2):193–205.

[22] Tarantilis, C., Stavropoulou, F., and Repoussis, P. (2012). A template-based tabu search algorithm for the consistent vehicle routing problem. *Expert Systems with Applications*, 39(4):4233 – 4239.

[23] Wong, R. (2008). Vehicle routing for small package delivery and pickup services. In Golden, B., Raghavan, S., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 475–485. Springer, New York.