Restricted dynamic programming: a flexible framework for solving realistic VRPs

J. Gromicho[a], J.J. van Hoorn[a], A.L. Kok[b], J.M.J. Schutten[b,*]

[a] *Vrije Universiteit, Amsterdam & ORTEC, Gouda, The Netherlands*

[b] *University of Twente, Enschede, The Netherlands*

Abstract

Most successful solution methods for solving large vehicle routing and scheduling problems are based on local search. A drawback of these approaches is that they are designed and optimized for specific types of vehicle routing problems (VRPs). As a consequence, it is hard to adapt these solution methods to handle new restrictions without losing solution quality. This is a particular drawback, since VRPs appearing in practice typically accommodate restrictions that are not accommodated in classical VRP models, such as time-dependent travel times and driving hours regulations. We present a new framework for solving VRPs that can handle a wide range of different types of VRPs. In addition, this framework accommodates various restrictions that are not considered in classical vehicle routing models, but that regularly appear in practice. Within this framework, restricted dynamic programming is applied to the VRP through the giant-tour representation. This algorithm is a construction heuristic which for many types of restrictions and objective functions leads to an optimal algorithm when applied in an unrestricted way. We demonstrate the flexibility of the framework for various restrictions appearing in practice. The computational experiments demonstrate that the framework outperforms state of the art local search methods when more realistic constraints are considered than in classical VRPs. Therefore, this new

*Corresponding author. Tel.: +31 53 489 4676; fax: +31 53 489 2159
E-mail adresses: JGromicho@ortec.nl, JHoorn@feweb.vu.nl, a.l.kok@utwente.nl, j.m.j.schutten@utwente.nl

framework for solving VRPs is a very promising approach for practical applications.

Keywords: Restricted DP; Giant-Tour Representation; VRP; Realistic Restrictions

1 Introduction

Local search methods have proved to be very successful in solving large vehicle routing and scheduling problems. Funke et al. [1] present an extensive overview of local search methods for vehicle routing and scheduling problems. These methods have been designed for a wide range of different types of VRPs and they develop high quality solutions for these problems.

A drawback of local search methods, however, is that it is difficult to generalize a local search method to various types of VRPs. The main difficulty is to adapt local search methods when new restrictions are introduced to the problem. Moreover, VRPs appearing in practice typically involve various restrictions that are not considered in classical vehicle routing models. If such a restriction is added, then the neighborhood structure and the search strategy need to be carefully redesigned in order to obtain high quality solutions. Therefore, adding a single restriction requires the development of a new algorithm as the extensive lists in Parragh et al. [2, 3] show.

In practice, extensions of various types of VRPs need to be solved, such as the capacitated VRP (CVRP), the VRP with time windows (VRPTW), and the pickup and delivery problem (PDP). Toth and Vigo [4] give an extensive overview of different types of VRPs and proposed solution methods. In addition, companies such as logistic service providers and distribution firms have their own set of restrictions of which a certain part may be general to all companies, but most companies also have some unique restrictions. As a consequence, each company requires a unique solution method to solve their routing problems. Therefore, solution strategies for the VRP that are flexible with respect to the addition of new restrictions and still produce high quality solutions are extremely valuable in practice. In general, solution methods based on local search lack this property.

We propose a flexible framework for solving large vehicle routing and scheduling problems. This framework covers a wide range of different types of VRPs. Moreover, it accommodates various restrictions that appear in practice, but that have been generally ignored in VRP literature, such as time-dependent travel times and driving hours regulations. The solution approach within this framework is a construction heuristic, as opposed to the majority of the solution approaches in VRP literature in which the focus is on route improvement heuristics. The solution approach is a generalization of the restricted dynamic programming heuristic for the TSP of Malandraki and Dial [5] (which can be seen as a form of beam search, see Bisiani [6]). We apply restricted dynamic programming to the VRP through the giant-tour representation, which was introduced by Funke et al. [1]. The giant-tour representation allows us to handle single tour and multiple tour problems in a similar way.

We show the quality of our framework by first solving the classical Solomon [7] benchmark instances for the VRPTW. Then, we compare the performance of our framework on the Solomon instances with the performance of our framework on a recent modification of the Solomon instances for the VRPTW with the European Community (EC) social legislation on driving and working hours by Goel [8]. We selected these problem sets, since the Solomon instances are standard reference in VRP literature, while Goel's instances form a more realistic set of instances that includes generally applicable regulations on driving and working hours. The results illustrate the power of this framework when more realistic VRPs are considered: the framework substantially improves the solutions found by a state of the art solution method for the VRPTW with the EC social legislation, while the framework cannot compete with state of the art solution methods for the classical Solomon instances.

The contributions of this paper are the following. First, we propose a new exact algorithm for the VRP. In the framework of this algorithm, various realistic restrictions and VRP variants can be accommodated. Second, we demonstrate this flexibility by showing how the algorithm can be applied to known VRP variants, and by showing how various additional realistic restrictions can be accommodated. Third, in addition to the way Malandraki and Dial [5] propose to restrict the state space to reduce computation times, we propose an additional way to restrict the state space which reduces computation times even further while maintaining or even improving solution quality. As a result of restricting the state space, the algorithm runs in practical (polynomial) computation times and produces high quality solutions for realistic types of VRPs.

Our paper is organized as follows. Section 2 first describes dynamic programming for the TSP, then describes the giant-tour representation of VRP solutions, and finally describes our framework for solving VRPs. Section 3 describes a way to reduce the state space of this dynamic programming formulation to obtain solutions within practical computation times. Section 4 demonstrates the flexibility of our framework by showing how known types of VRPs can be solved within this framework, and how various additional realistic restrictions can be accommodated. Section 5 presents the results of the computational experiments. Section 6 summarizes the main findings in this paper.

2 DP applied to the VRP

Our framework for solving VRPs is based on the the restricted dynamic programming (DP) heuristic for the TSP proposed by Malandraki and Dial [5], which is applied to the VRP through the giant-tour representation (GTR) introduced by Funke et al. [1]. Therefore, we first describe the DP for the TSP and the GTR of vehicle routing solutions.

2.1 Dynamic programming for the TSP

The restricted dynamic programming heuristic for the TSP is based on the exact dynamic programming algorithm for the TSP of Held and Karp [9] and Bellman [10]. The exact dynamic programming algorithm for the TSP can be described as follows.

The TSP considers the problem of visiting a set $V = \{0, 1, ..., n-1\}$ of n cities exactly once, starting and ending at city 0, and minimizing the total travel distance. The travel distance between each pair of cities $i, j \in V$ is given by c_{ij}.

A state $(S, j), j \in S, S \subseteq V \backslash 0$ in the DP algorithm represents a path starting at city 0, visiting all cities in S exactly once, and ending in city j. The cost $C(S, j)$ of a state is given by the length of the smallest of such paths. In the first stage, the costs of the states are determined by $C(\{j\}, j) = c_{0j}, \forall j \in V \backslash 0$. Next, in each successive stage the costs of the states are calculated with the recurrence relation $C(S, j) = \min_{i \in S \backslash j} \{C(S \backslash j, i) + c_{ij}\}$. Finally, the length of the optimal TSP tour is given by $\min_{j \in V \backslash 0} \{C(V \backslash 0, j) + c_{j0}\}$.

Since there are $\sum_{|S|=1}^{n-1} \binom{n-1}{|S|} \approx 2^n$ subsets S and each subset S contains $|S| \leq n-1$ possible end nodes, the total number of states is $\mathcal{O}(n2^n)$. Next each state is calculated by comparing at most $n-2$ additions, resulting in an algorithm with a running time complexity of $\mathcal{O}\left(n^2 2^n\right)$. The optimal TSP tour can be backtracked by saving for each state (S, j) the city $i \in S \backslash j$ that minimizes $C\left(S \backslash j, i\right) + c_{ij}$.

Since this approach constructs only one route, it cannot be applied directly to the VRP. We propose to apply it to the VRP through the GTR of vehicle routing solutions.

2.2 Giant-tour representation

Funke et al. [1] introduce the GTR of vehicle routing solutions, because it allows to handle single and multiple route problems in a similar way. Besides, it is a 'natural' representation of vehicle routing solutions. We use the GTR for the development of our general framework for solving VRPs. The GTR can be described as follows.

The basis of any routing problem is a directed graph $G = (V, A)$, in which the node set V consists of request nodes $R \subset V$, origin nodes $O \subset V$, and destination nodes $D \subset V$, and the arc set A represents feasible travels between these nodes. For a VRP, the request nodes R correspond to all customer requests. Furthermore, for each vehicle there is one origin and one destination node, which all may represent the same location. Therefore, if m is the number of vehicles available, we get $|O| = |D| = m$. If we order the vehicle routes $r^v, v = 1, ..., m$ in a routing solution, then the GTR of this solution is a cycle in the graph G in which each route end node d^v is connected to the route start node of the next vehicle route o^{v+1}. Finally, the cycle is closed by connecting d^m with o^1. In Figure 1, we present an example of a vehicle routing solution with three vehicles, two depots (A and B) and nine customers. Vehicle 1 starts at depot A and ends at depot B, vehicle 2 starts and ends at depot B, and vehicle 3 starts and ends at depot A. Figure 2 presents the same solution with its corresponding GTR.

2.3 Dynamic programming for the VRP

By using the GTR we transform the VRP into a sequencing problem and we can use the DP formulation for the TSP to solve it. However, we need to ensure that the DP solution is the GTR of a feasible VRP solution. A general way to do this is by checking the feasibility of a partial solution while expanding a state. We only call an expansion feasible if it represents the

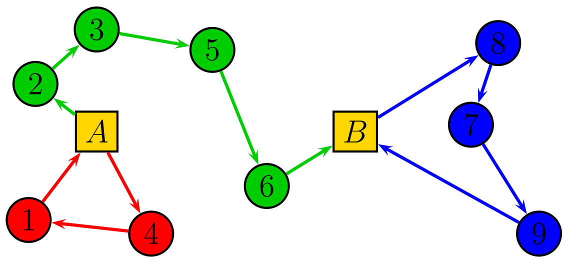

Figure 1: Example of a solution to a VRP with three vehicles

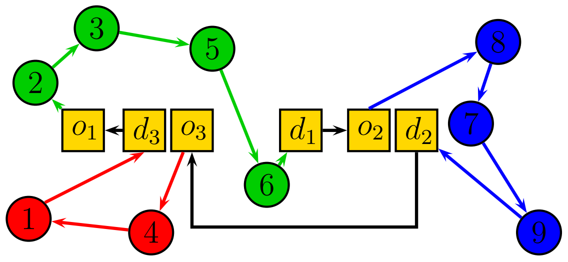

Figure 2: The giant-tour representation of a solution to the VRP of Fig. 1

giant tour of a feasible partial VRP solution. So for a state (S, d^v) where the partial solution ends with node d^v, the only feasible expansion is o^{v+1}. Furthermore, o^{v+1} can only be an expansion of a state with end node d^v. Therefore, unlike the TSP, not every expansion is feasible for the VRP and we must perform a feasibility check when expanding a state. This seems a downside, but it actually gives us the power to use this framework for almost every type of VRP.

To derive the running time complexity of the DP algorithm for the VRP, observe that we have to add $2m$ nodes to the $|R|$ nodes for the customers. This would lead to a total of $2m + |R|$ nodes. However, each end node of a vehicle is attached to the start node of the following vehicle leaving only m extra nodes in consideration $(i_1, i_2, \ldots, i_m)$, i.e., $n = m + |R|$. Furthermore, these nodes have a fixed order in the GTR which reduces the state space considerably.

The ordering of the vehicles imposes a serial precedence relation of m nodes $i_1 \to i_2 \to \ldots \to i_m$. For every state (S, j) for which holds that $i_a \notin S$, $i_b \in S$ and $i_a \to i_b$, there exists no feasible partial solution, because the precedence relation is not satisfied. Therefore, for every (S, j) that does have a feasible partial solution, there exists a $k, 0 \leq k \leq m$, such that $i_a \in S$ if $a \leq k$ and $i_a \notin S$ if $a > k$. We first derive the fraction of the total number of subsets that have this property, since this fraction equals the fraction of states that contain feasible partial solutions.

Suppose $|S| = l$ and let $i_1, ..., i_k \subseteq S$. Then it must hold that $k \leq l$ and $l - k \leq n - m$, with $l - k$ request nodes in S and $n - m$ request nodes in total. The total number of such subsets S equals $\binom{n-m}{l-k}$. If we sum this over all k and l, we get $\sum_{k=0}^{m} \sum_{l=k}^{n-m+k} \binom{n-m}{l-k} = (m+1)2^{n-m}$ subsets. If we divide this by the total number of subsets $S \subseteq V$, we get $\frac{(m+1)2^{n-m}}{2^n} = \frac{m+1}{2^m}$, which is the fraction of states that can contain feasible partial solutions given the precedence relations $i_1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_m$. This implies that each independent precedence relation $i \rightarrow j$ reduces the state space by $\frac{1}{4}$ (in this case $m = 2$ such that the fraction of states that contain feasible partial solutions is $\frac{3}{4}$).

Finally, observe that any given state can possibly end in $|R| + m = n$ nodes, but can only be expanded to $|R| + 1$ nodes (because of the serial precedence relation). Therefore, the running time complexity of the DP algorithm for the VRP is

$$\mathcal{O}\left((|R| + m)(|R| + 1)\frac{m+1}{2^m}2^{|R|+m}\right) = \mathcal{O}\left(n^2 m 2^{n-m}\right).$$

3 Restricting the state space

Although dynamic programming has the best running time complexity of all known exact algorithms for the TSP (see Woeginger [11]), it is not fast enough to solve problems of realistic sizes in practical computation times. Therefore, Malandraki and Dial [5] propose a restricted version of this algorithm, in which the number of states in each stage is bounded by a parameter H. This bounding procedure works as follows.

In each stage, we only use the H states with the smallest costs to expand in the next stage. Since each state reflects a partial tour and in each stage all partial tours visit the same number of customers, a state with low costs is more likely to yield the first part of a good TSP solution than a state with high costs. Therefore, continuing the algorithm with only the H states with smallest costs will, although it does not guarantee to find the optimal TSP tour, probably still result in a good TSP solution. Next, each of these H

states is expanded in all possible ways with one extra city. If certain states are reached multiple times (e.g., expanding (S, j) with customer k results in the same state as expanding (S, i) with customer k, i.e., $(S \cup k, k)$), then, according to the original recurrence relation, only the one with lowest costs is maintained.

Malandraki and Dial [5] show that restricted dynamic programming is a flexible approach for solving TSPs by applying it to the TSP with time-dependent travel times. They also show that increasing the value of H results in better solutions, but also in substantial higher computation times. Note that setting $H = 1$ results in the nearest neighbor heuristic and setting $H = \infty$ results in the exact dynamic programming algorithm for the TSP.

We restrict the state space even further, using a kind of beam search (Bisiani [6]), by expanding each state (S, j) only to the E nearest unvisited nodes. We think this is reasonable because edges in the optimal solution will most likely be between two nodes that are near neighbors of each other, as observed by Rego and Glover [12].

The same principle can be applied to the DP algorithm for the VRP. However, from these E expansions we only use the *feasible* ones. We respect the precedence relations on the route start and end nodes when expanding to the E nearest unvisited nodes. We observe that our beam through the state space will require a polynomial effort, i.e., for each fixed H and E we expand to $\mathcal{O}(nHE)$ states. The expansion of a state can be done in $\mathcal{O}(1)$. However, since we need to select the H best states we get a running time complexity of $\mathcal{O}(n^2 HE \log(H))$.

4 The flexibility of our solution approach

To demonstrate the flexibility of the DP algorithm, we show two techniques to accommodate various problem extensions. Then, we show how various realistic constraints, both ones that are considered in classical vehicle routing models and ones that have never or hardly been considered in VRP literature, can be accommodated in the DP framework. Note that any conceivable combination of these constraints is equally suited. Therefore, our generic approach is unique in comparison with the large variety of approaches found in literature, see Parragh et al. [2, 3], all of which are diverse and specific for some variants of the VRP.

4.1 Extra state dimensions

For the CVRP, as well as the VRPTW, we add state dimensions on capacity or time. When expanding a state, we perform a feasibility check to ensure that there is enough slack in capacity or time. However, we have to be careful not to lose the optimality guarantee of the (unrestricted) DP algorithm for the VRP. We demonstrate this by the following example. Suppose two states (S, j) and (S, i) can be feasibly expanded with the same node k such that the first expansion results in a partial solution with lower costs than the second expansion, but with less slack in capacity or time. According to the original recurrence relation, the first expansion will be selected. However, it may prove impossible to complete the first expansion to a feasible complete solution, while the second expansion can be completed to a feasible solution. This is resolved by making two copies of this state such that one represents the partial solution with lower costs, while the other represents the partial solution with more slack. This is formalized in dominance rules as in Dumas et al. [13], which can result in several copies of a single state. However, in practice costs and slack are correlated, such that for all states with the same visited node set S and end node $j \in S$ it is unlikely that no state is dominated by another one.

4.2 Input characteristics and precedence relations

In certain cases it may be infeasible to visit a certain customer with a specific vehicle. For example, frozen goods need to be transported in a refrigerated truck. This feasibility and others, even depending on the actual visit sequence, can be checked when expanding a state. To apply our model to VRPs with sequencing restrictions (e.g., the PDP), we add precedence constraints to the nodes involved, thereby reducing the state space. Furthermore, we need to add a feasibility check when a vehicle returns to a depot to ensure that the vehicle only returns if it has visited all the deliveries corresponding to the pickups that it has visited. These precedence relations also have a big impact on the running time, because every independent relation of a pickup and delivery reduces the state space by $\frac{1}{4}$ (see Section 2.3).

4.3 Realistic constraints

Several other realistic constraints fit within our algorithmic framework. State dimensions and input characteristics allow for various extensions of the VRP. Table 1 presents an overview of classical VRP extensions and how they fit within our algorithmic framework. Table 2 presents an overview of practical VRP extensions that have never or hardly been considered in literature, but can be accommodated in the DP framework using the techniques described above. This list of extensions is motivated by practice [14], but is certainly not exhaustive.

VRP extension	Inclusion within DP framework
Capacities (CVRP)	controlled with feasibility checks and a state dimension on the remaining capacity of the current vehicle.
Time windows (VRPTW)	controlled with feasibility checks and a state dimension on the departure time from the last visited node.
Pickup and deliveries (PDP)	controlled with precedence relations and feasibility checks.
Open VRP [a]	controlled with the input: set distances to route end nodes to 0.
Heterogeneous fleet	different vehicle capacities and availability times are controlled with the input and with feasibility checks.
Multiple depots	different depot locations for different vehicles are controlled with the input.
Multiple routes for a single vehicle	modeled by separate vehicles for each route and a flexible time window depending on the return time of the preceding vehicle.

[a] In the open VRP, vehicles do not have to return to the depot.

Table 1: Overview of classical VRP extensions that fit within the DP framework.

VRP extension	Inclusion within DP framework
Time-dependent travel times	state dimension on departure time is used to determine travel time.
Driving hours regulations	state dimensions control remaining driving and working times until a break is required (see Kok et al. [15]).
Rolling time horizon	state dimensions are set to the initial conditions of the planning period.
Multiple compartments	state dimensions control capacity of each compartment.
Vehicle characteristics	vehicle-customer feasibilities are checked.
Customer combinations	feasible customer combinations in the same vehicle are checked.
Drivers exchanging trucks at country borders	modeled by separate vehicle routes for each country and for both parts of each vehicle route from and to the country border. A flexible time window and flexible state dimensions on, e.g., driving hours regulations are used to ensure the right initialization of each route starting at the country border.

Table 2: Overview of VRP extensions from practice that fit within the DP framework

5 Computational experiments

We test our framework on known benchmark instances. Since most benchmark instances in literature are proposed for classical VRPs in which only few realistic constraints are considered, they are not well suited for this study. The recently developed benchmark instances for the VRPTW with the EC social legislation on driving and working hours proposed by Goel [8] form an exception on these classical benchmark instances. Therefore, we selected these instances to test our solution framework. To illustrate the difference in performance of our framework on classical VRP variants instead of more realistic VRPs, we first solve the well-known Solomon [7] instances with our solution framework. Moreover, the Solomon instances are standard reference in VRP literature, whereas Goel's instances are modifications of the Solomon instances. We implemented the DP framework in Delphi 7 and ran our experiments on a PC with a Core 2 Quad, 2.83 GHz CPU and 4 GB of RAM.

5.1 VRPTW

The Solomon instances consist of 6 problem sets: in the c1 and c2 instances customer locations are clustered, in the r1 and r2 instances they are random, and in the rc1 and rc2 instances they are semi-clustered; the 2-instances have a relatively longer time horizon and larger vehicle capacities than the 1-instances allowing for larger vehicle routes in terms of number of customers. The primary objective is to minimize the number of vehicles used, the secondary objective is to minimize the total travel distance.

In order to satisfy the time window and capacity constraints, we add state dimensions t and q, indicating the departure time from the last visited customer and the remaining vehicle capacity, respectively. If states A and B visit the same customer set S and end in the same node $j \in S$, then state A dominates state B if its costs are not larger than the costs of state B, its departure time is not larger than the departure time in state B, and its remaining capacity is not smaller than the remaining capacity in state B. This ensures that we do not exclude the optimal solution. If in a certain stage the number of states exceeds its maximum H, then only the most promising H states are maintained. To determine the most promising states, we use the following hierarchical criteria: number of vehicles used, total distance traveled, departure time from the last visited node, remaining vehicle capacity of the active vehicle. The first two criteria correspond to the objective function; the last two criteria select the most flexible solutions with respect to adding more customers (smaller departure times imply more time for visiting additional customers; larger remaining capacities imply more capacity available for visiting additional customers). We set $H = 100{,}000$ and we first set E to its maximum value n.

Table 3 presents the results for the six Solomon problem sets. The first three columns present the average number of vehicles used, the average travel distance, and the cpu time in seconds, respectively. The last two columns present the objective values for the best known solutions identified by heuristics.

Problem	DP Algorithm			Best Known Solutions	
Set	# Veh.	Dist.	cpu (s)	# Veh.	Dist.
c1	10.00	852.71	662	10.00	828.38
c2	3.00	608.72	958	3.00	589.86
r1	14.08	1333.52	642	11.92	1205.39
r2	4.18	1111.24	832	2.73	951.91
rc1	14.13	1510.09	602	11.50	1384.16
rc2	4.25	1336.96	755	3.25	1119.35

Table 3: results for the VRPTW

On average, the DP algorithm results in 1.27 more vehicle routes than state of the art local search methods for the VRPTW. The total travel distance is 11.0% larger, on average. The average computation time per problem instance is 738 seconds. Increasing H results in substantial reductions of the gaps with the best known solutions. We run our experiments also for $H = 10{,}000$ and $H = 1{,}000{,}000$, and it turns out that increasing H reduces the gaps in number of vehicles used and total travel distance considerably: increasing H from 10,000 to 100,000 reduces these gaps by 14.5% and 18.3%, respectively, and increasing H from 100,000 to 1,000,000 reduces these gaps by 12.7% and 20.5%, respectively. In these experiments, computation times increase (approximately) by a factor 11 when H increases by a factor 10.

Table 4 presents the impact of different E values on the performance of the algorithm. We set the values of E to n, $0.5n$, $0.25n$, and $0.125n$ (we round in case of fractional values) and we present average gaps and average computation times over all problem instances. The results indicate that it is possible to decrease E such that computation times also decrease, while the solution quality is maintained. Setting E to $0.5n$ even improves the solutions, both in number of vehicles and in total travel distance. When we further reduce E to $0.25n$, then the number of vehicles used increases slightly, but the total travel distance reduces again, while the computation times are only about 50% of the original computation times with $E = n$. The distance is even further decreased by setting $E = 0.125n$, but this gives an additional increase in the number of vehicles used. Smaller values for E allow only state expansions to near neighbors, which has a positive impact on the total travel distance, but may turn out worse for the number of vehicles used (due to, e.g., time windows).

E	# vehicles[a]	travel distance[b]	cpu (s)
n	1.27	11.02%	738
$0.5n$	1.25	10.99%	592
$0.25n$	1.30	10.71%	373
$0.125n$	1.38	10.55%	225

[a]absolute average gap
[b]relative average gap

Table 4: Impact of different E values

The DP algorithm cannot compete with state of the art methods for the VRPTW. For example, Pisinger and Ropke [16] attain the minimum number of vehicles for all problem instances, and the total travel distance is within 1% of the best known solutions. However, the DP framework is designed for solving realistic VRPs instead of classical VRPs. In the next section, we demonstrate that the DP framework outperforms state of the art solution methods when more realistic VRPs are considered.

5.2 VRPTW with EC social legislation

We illustrate the better performance of the DP framework when more realistic restrictions are accommodated by applying the algorithm to a set of benchmark instances for a more realistic VRP. The Solomon benchmarks have recently been adapted for the VRPTW with the EC social legislation on driving and working hours. Due to the generality of this legislation (it is valid for all member countries of the European Union, and the EC social legislation is more restrictive than the US Hours-Of-Service Regulations [17]), we selected this set of problem instances as benchmarks for more realistic VRPs.

The EC social legislation poses restrictions on the amount of driving and working times before a mandatory rest is taken. The legislation considers different time horizons for scheduling breaks and rest periods: single driving periods, daily driving and working times, and weekly driving and working times. After a restricted amount of accumulated driving and working time, a break or rest of certain minimum duration must be scheduled. For a detailed description of the rules in the legislation, we refer to Kok et al. [15]. Kok et al. [15] also propose a break scheduling heuristic that schedules breaks and rest periods such that the vehicle routes satisfy the requirements posed by the EC social legislation on driving and working hours. They embed this break scheduling method in the DP framework. Goel [8] proposes the following modification of the Solomon instances for the VRPTW with the EC social legislation.

Goel proposes to consider the depot opening time as a period of 144 hours, corresponding to a weekly working period, and to scale the customer time windows accordingly. Next, Goel suggests a driving speed of 5 distance units per hour, and to set all service times to 1 hour. Due to the required breaks and rest periods, it may turn out to be impossible to reach certain customers before their due dates, or the vehicle may not be able to return in time to the depot after serving a customer at his ready time. Therefore, Goel suggests to adjust such time windows in such a way that the ready time equals the earliest time the vehicle can reach the customer, and the due date is such that starting service at this due date and directly returning to the depot results in a return time at the depot's due date, respectively. Table 5 presents the results found by the DP algorithm, including the break scheduling heuristic of Kok et al. [15], for $H = 10{,}000$ and $E = n$ compared with the best known solutions for this problem, which were obtained by a state of the art large neighborhood search algorithm proposed by Goel [8].

In contrary to the classical Solomon instances, the DP algorithm significantly improves the VRPTW solutions when the EC social legislation is accommodated in the problem. The number of vehicles used reduces by 1.8, on average, and the travel distance by 5.4%. The results were obtained with an average computation time of 65 seconds. This illustrates the opportunities of applying the DP framework in practice: when realistic restrictions are added to the problem, the performance of the DP framework substantially improves with respect to state of the art local search methods, even outper-

Problem	DP Algorithm			Best Known Solutions	
Set	# Veh.	Dist.	cpu (s)	# Veh.	Dist.
c1	10.33	965.44	55	11.11	1054.45
c2	5.00	770.42	72	8.38	954.64
r1	9.67	1152.39	63	10.92	1144.23
r2	7.55	1100.83	66	10.27	1107.14
rc1	10.25	1300.60	71	11.13	1347.75
rc2	8.13	1266.64	69	10.00	1347.26

Table 5: results for the VRPTW with EC social legislation

forming the state of the art heuristic for the VRPTW with the EC social legislation on driving and working hours.

6 Conclusions

This paper introduces a highly flexible framework based on dynamic programming for Vehicle Routing Problems that is able to model and solve different types of problems previously tackled by tailored algorithms. Moreover, this framework accommodates various realistic restrictions that have been generally ignored in classical vehicle routing models, such as time-dependent travel times and driving hours regulations. Formerly, if a type of Vehicle Routing Problem would emerge from a known type by just changing or adding some constraints, one would often be forced to go back to the drawing board and devise a whole new tailored approach to handle it. Therefore, a general framework that is able to accommodate different types of problems offers the added benefit to handle several variants of problems which are hybrid variants of existing ones.

In order to demonstrate that the benefit of our framework extends further than just theoretical possibilities, the paper also shows how to control the size of the state space by restricting the number of expansions of a state leading to practical and efficient polynomial construction heuristics. The quality of these heuristics for solving realistic Vehicle Routing Problems is demonstrated by significantly improving the best known solutions for the VRPTW with the EC social legislation on driving and working hours, which were obtained by state of the art local search methods. Such achievements are generally only the realm of powerful neighborhood search methods, and hence a remarkable outcome of a generic construction algorithm.

Acknowledgment

This work was financially supported by Stichting Transumo through the project ketensynchronisatie.

References

[1] B. Funke, T. Grünert, and S. Irnich. Local search for vehicle routing and scheduling problems: Review and conceptual integration. *Journal of heuristics*, 11(4):267–306, 2005.

[2] S. N. Parragh, K.F. Doerner, and R.F. Hartl. A survey on pickup and delivery problems. part i: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

[3] S. N. Parragh, K.F. Doerner, and R.F. Hartl. A survey on pickup and delivery problems. part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008.

[4] P. Toth and D. Vigo. *The vehicle routing problem.* SIAM Monographs on Discrete Mathematics and Applications. Philadelphia, 2002.

[5] C. Malandraki and R.B. Dial. A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *European Journal of Operational Research*, 90(1):45–55, 1996.

[6] R. Bisiani. Beam search. In S. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 56–58. Wiley & Sons, 1987.

[7] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

[8] A. Goel. Vehicle scheduling and routing with drivers' working hours. *Transportation Science*, 43(1):17–26, 2009.

[9] M. Held and R.M. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196, 1962.

[10] R. Bellman. Dynamic programming treatment of the travelling salesman problem. *Journal of the Association for Computing Machinery*, 9(1):61, 1962.

[11] G.J. Woeginger. Exact algorithms for np-hard problems: A survey. In *Combinatorial Optimization–Eureka! You shrink!, Lecture Notes in Computer Science, Vol. 2570*, pages 185–207. Springer, Berlin, 2003.

[12] C. Rego and F. Glover. Local search and metaheuristics. In G. Gutin and A. Punnen, editors, *Traveling Salesman Problem and Its Variations*, pages 309–367. Kluwer Academic Publishers Dordrecht, 2002.

[13] Y. Dumas, J. Desrochiers, E. Gelinas, and Marius M. Solomon. An optimal algorithm for the traveling salesman problem with time windows. *Operations research*, 43(2):367–371, 1995. 0030-364X.

[14] ORTEC Algorithm Team. Internals of the VRP algorithms. *Internal confidential document*, 2009.

[15] A. L. Kok, C.M. Meyer, H. Kopfer, and J.M.J. Schutten. Dynamic programming algorithm for the vehicle routing problem with time windows and EC social legislation. *Beta working paper series*, 270, 2008.

[16] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8):2403–2435, 2007. 0305-0548.

[17] Hours-Of-Service Regulations. *Federal Motor Carrier Safety Administration*, 2005. `http://www.fmcsa.dot.gov/rules-regulations/topics/hos/hos-2005.htm`.