Fast Ejection Chain Algorithms for Vehicle Routing with Time Windows

H. M. J. Sontrop[1], S. P. van der Horn[2], G. Teeuwen[3], and M. Uetz[1]

[1] Maastricht University, Quantitative Economics, P.O. Box 616, 6200 MD Maastricht, The Netherlands. E-mail: {Herman.Sontrop@gmail.com, M.Uetz@ke.unimaas.nl}
[2] Heezerweg 198, 5614 HJ, Eindhoven, The Netherlands. E-mail: PietervanderHorn@gmail.com
[3] Centre for Quantitative Methods CQM BV, P.O.Box 414, 5600 AK Eindhoven, The Netherlands, E-mail: G.Teeuwen@cqm.nl

Abstract. This paper introduces new ejection chain strategies to effectively target vehicle routing problems with time window constraints (VRPTW). Ejection chain procedures are based on the idea of compound moves that allow a variable number of solution components to be modified within any single iteration of a local search algorithm. The yardstick behind such procedures is the underlying reference structure, which is the structure that is used to coordinate the moves that are available for the local search algorithm. The main contribution of the paper is a new reference structure that is particularly suited in order to handle the asymmetric aspects in a VRPTW. The new reference structure is a generalization of the doubly rooted reference structure introduced by Glover, resulting in a new, powerful neighborhood for the VRPTW. We use tabu search for the generation of the ejection chains. On a higher algorithmic level, we study the effect of different meta heuristics to steer the tabu chain ejection process. Computational results confirm that our approach leads to very fast algorithms that can compete with the current state of the art algorithms for the VRPTW.

1 Introduction

Recently, the famous Lin-Kernighan (LK) procedures [14] were dethroned as champion procedures for solving large scale traveling salesman problems by so-called *Stem & Cycle* (SC) ejection chains; see [6]. This is remarkable, since Lin-Kernighan type procedures had dominated this field for the last decades.

In the words of Glover [9], 'ejection chain procedures are based on the notion of generating compound sequences of moves, leading from one solution to another, by linked steps in which changes in selected elements cause other elements to be "ejected from" their current state, position or value assignment'. A key difference over the LK-based approaches is the fact that ejection chain procedures explicitly identify a so called *reference structure*. A reference structure is a structure similar to, but slightly different from a solution. Via a set of *transition rules*, the reference structure guides the generation of acceptable moves from one reference structure to another. Actual solutions to the problem, usually called

trial solutions in this context, are obtained from any reference structure, again via a predefined set of transition rules.

In this paper we address the vehicle routing problem with time window constraints (VRPTW). Given is a number of customers in the plane, with known demands, a given service or delivery time, and a fleet of identical vehicles with known limited capacities. We are asked to find a set of routes starting and ending at a central depot, such that each client's demand is delivered, and each client is visited exactly once. Clearly, any route must not violate the capacity constraints of the vehicle. In addition, each client must be serviced within its so called *time window*. The time window specifies an earliest and a latest time at which the delivery must begin. If a vehicle arrives at a customer before the opening of the time window, the vehicle must wait. Arriving after the end of the time window is not allowed. Two different solutions with the same number of vehicles are usually ranked by the total distance travelled by the vehicles (sometimes, also the waiting time is taken into account). The objective considered in this paper is the total distance travelled by the vehicles, utilizing as many vehicles as required[4].

The VRPTW has been extensively studied. The best *exact* procedures can still only handle small instances, small being in the order of 50-100 customers [20]. Meta-heuristic procedures mostly minimize the number of vehicles, and among solutions with the same number of vehicles, prefer those with small total distance travelled. Since the overall literature is extensive, we refer to [20] for a thorough introduction into vehicle routing in general, and many references. For concepts of tabu search, simulated annealing and other meta-heuristics, we refer to Aarts and Lenstra [1]. As a matter of fact, tabu search based procedures are the majority among the most effective algorithms for VRPTW, see, e.g., [18, 21, 12], but also other meta heuristic frameworks proved to be effective, such as, e.g., a multiple ant colony system of [5].

Our main contribution is a new reference structure, generalizing previous structures, that particularly targets the asymmetric nature of the VRPTW. This reference structure lies at the heart of a traditional ejection chain procedure that is based on tabu search. The ejection chain itself is driven by a higher level meta heuristic. So far, ejection chain studies have mainly focused on the lower level of control, the ejection chain process itself. In addition to providing a new reference structure, a novelty in our contribution is therefore also a study of different (higher level) meta heuristics that are used to steer the ejection chain process. Here we test two different approaches, iterated local search, and simulated annealing. Our computational results on established standard test sets confirm that the resulting hybrid meta heuristic is fast and extremely effective.

2 The Cyclic Doubly Rooted reference structure

It can be considered both a strength and a weakness of LK-based procedures that certain types of moves require a *path reversal*. In the absence of (tight) time

[4] This is not a limitation of our approach, since it could be adopted to handle other objectives, too. We opted for this objective in order to be able to compare our results to the known optimal solutions, which have been obtained using the same objective.

windows, path reversals are an important building block of powerful moves. In the presence of time windows, however, path reversals often violate time windows, especially if the time windows are tight. Therefore, our idea to create a powerful ejection chain procedure for the VRPTW is to have a reference structure that still provides a strong connectivity between solutions, but does not rely on path reversals in order to generate moves.

The current champion TSP procedures use the SC reference structure. Figure 1 shows a instance of this structure. It is a spanning subgraph that consists of one cycle and a path. The path is called the *stem*, the end of the stem is called the *tip*. The vertex that is shared by the cycle and the stem is called the *root*. Note that the root and the tip have odd degrees. Figure 2 shows a *Doubly Rooted*

Fig. 1. SC

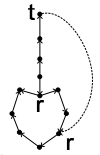


Fig. 2. DR

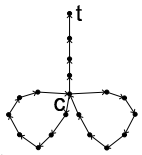


Fig. 3. Flower

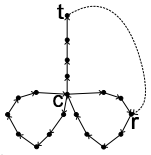


Fig. 4. CDR

(DR) reference structure. It may be conceived as arising from a SC by adding an edge (t, j) which connects the tip to an arbitrary node j on the cycle.[5] Again, this results in a structure that has two vertices that have an odd degree. In [8] Glover refers to both these vertices as *roots*. Glover shows that the DR structure has several advantages over the SC structure. He proves a connectivity result showing that the DR allows for more direct trajectories between solutions. Most important is that the connectivity result also holds for asymmetric problems. This is our main reason to base the new reference structure on the DR concept instead of the SC concept. In addition, the DR structure provides access to moves unavailable to a SC structure. Rego [17] generalizes the SC into the *Flower* reference structure[6], depicted in Figure 3. As can be seen it is a SC structure joined with multiple cycles. The intersection of the cycles is called the *core*. A SC structure can also be seen as a (trivial) Flower. The Flower concept proved to be very fruitful for VRP problems without time windows. The reference structure we propose is closely related to both the DR structure and the Flower concept. It may be conceived from a Flower by adding an edge (t, j) which connects the tip to an arbitrary node j on a cycle, as shown in Figure 4.

Figure 5 shows an instance of this new reference structure. We call it a *Constrained Doubly Rooted* reference structure (CDR). We use the word 'constrained' since the core, the depot vertex, always represents one of the roots. In Section 7 we briefly discuss an even more general version. Note that a CDR, too, has two vertices that have odd degree. We will refer to the vertex that represents the depot as the *core*, denoted by *c*. The other vertex with odd degree is called the

[5] Another form of a DR arises when we connect the tip to a vertex located on the stem. This form is not considered in this paper.

[6] Rego [17] refers to the vertex t as the *root*. We use the term *tip* as in [8], however.

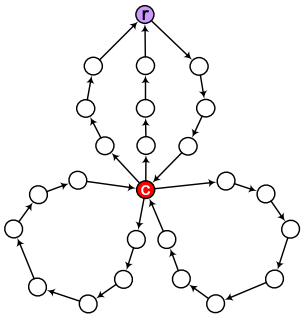
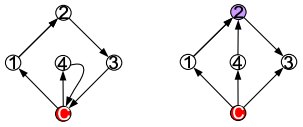
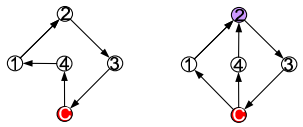
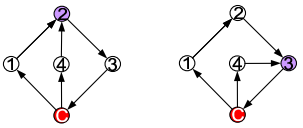
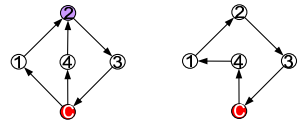


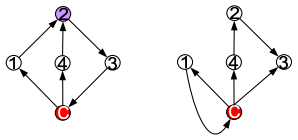
Fig. 5. CDR reference structure

root, denoted by r. A vertex v such that the edge (v, r) exists is called a *sub-root*. Note that when we eject such an edge (v, r) we obtain a flower structure, therefore v could also be called an *implicit tip*.

3 Transition rules

Next we state five simple rules that enable us to exploit the CDR structure to create a powerful ejection chain procedure. None of these rules involves a path reversal. The moves, however, can easily be adapted to include path reversals, if desired. In the figures, the core c is always the lowest vertex, and the root, if present, is displayed shaded.

S1-rule (Solution to CDR) Eject (s, c) Add (s, j) where $j \neq c$	
S2-rule (Solution to CDR) Add (c, p) where $p \neq c$ Eject (q, p) where q is the predecessor of p Add (q, j) where $j \neq c$ and $j \neq p$	
E1-rule (CDR to CDR) Eject (s, r) where $s \neq c$ Add (s, j) where $j \neq r$ and $j \neq c$	
T1-rule (CDR to Solution) Eject (s, r) where $s \neq c$ Add (s, j) where $s \neq j$ Eject (c, j)	

T2-rule. (CDR to Solution) Eject (s, r) Add (s, c)	

In order to not deteriorate the quality of either a solution or a CDR by one of the transition rules, and to speed up the procedure, whenever an arc (v, w) is added, we require one of the following two conditions. Either, vertex w must be a direct neighbor of vertex v in the *Delaunay triangulation* of the point set of customers of the underlying instance[7], or (v, w) is one of the 12 shortest edges leaving v such that this edge does not imply a time window violation on its own. No conditions are placed on edges that involve the depot vertex. Let us call a transition *admissible* if one these conditions are satisfied[8].

4 Ejection Chain Construction

Figure 6 shows an instance of an ejection chain with depth 5. Starting at a start solution S, we move to the best CDR structure that can be obtained from S, using any of the admissible S-transition. Then we use 4 consecutive admissible

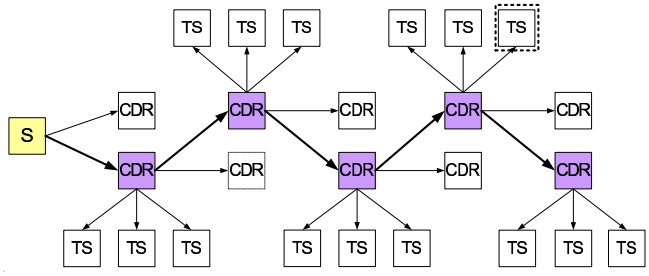


Fig. 6. Ejection chain example

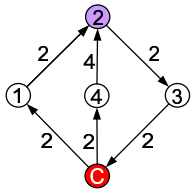
E-transitions, moving to the respective best available CDR structure, and at each level, we determine the best admissible trial solution (TS), using any of the T-transition rules. Eventually, the compound move would consist of moving from S to the overall best found trial solution, the boxed trial solution in Figure 6.

In order to be able to move to the 'best' admissible CDR structure for any S- or E-transition, we must decide on the quality of a CDR reference structure (recall that it does not represent a feasible solution). To this end, we associate a simple cost function with the CDR reference structure. Figure 7 provides an

[7] See [7] for definitions regarding Delaunay triangulations.

[8] Glover uses slightly stricter conditions, so called *legitimacy conditions*, to determine the edges that are susceptible to being added or ejected, see [8]. The same legitimacy conditions have been adopted also by Rego [17].

example how to determine the cost of a sample CDR structure. The idea is to just count the total length of arcs included in the structure, incremented by a penalty term for the total time window violations of the implicit routes. To understand



TW	Route A	Route B
C: [0,10]	A_C: 0 V_C: 0	A_C: 0 V_C: 0
1: [2,4]	A_1: 2 V_1: 0	A_4: 2 V_4: 0
2: [4,5]	A_2: 4 V_2: 0	A_2: 6 V_2: 1
3: [5,7]	A_3: 6 V_3: 0	A_3: 8 V_3: 1
4: [2,5]	A_C: 8 V_C: 0	A_C: 10 V_C: 0

Fig. 7. Instance for determining the cost of a CDR structure

what is meant, consider the instance of Figure 7. On the left we have a CDR structure. All arcs are of length 2, except arc (4,2) that has length 4. Service times are 0. Vertex 2 represents the root of the CDR, vertex C represents the core, while vertices 1 and 4 are the subroots. This CDR has two implicit routes, route A: {C,1,2,3,C} and route B: {C,4,2,3,C}. (At least one of these routes will be destroyed at the next move to either another CDR or to a solution.) The table on the right of Figure 7 shows the time windows (TW), the arrival times A_i and time window violations V_i for all vertices i. Note that Route A has a violation of 0 while Route B has a violation of $1 + 1 = 2$. The total cost is the sum of the lengths of all arcs and the total time window violation of all (implicit) routes multiplied by a certain penalty P. Therefore, we would assign a total cost of $(2 + 2 + 2 + 2 + 2 + 3) + P * (0 + 2) = 13 + 2P$.

Finally, it is important to notice that each ejected edge is declared tabu for the remainder of the ejection chain. In addition, all edges ejected in one ejection chain are declared tabu also for the next θ ejection chains considered by the algorithm, where θ is a randomized parameter that is explained in more detail in Section 6.

5 Higher Level Meta Heuristics

Ejection chain procedures are able to manipulate multiple solution components within a single compound move. Based on the theory developed by Glover [8] for the DR reference structure, the CDR-structure proposed in this paper is conjectured to provide a strong form of connectivity between any two solutions. Therefore, a search can (and indeed does) easily get stuck in a so-called *basin of attraction*, see [15]. Hence we need a mechanism that will provide an escape trajectory from such a basin. We implemented two mechanisms, iterated local search as well as simulated annealing.

5.1 Iterated Local Search

A simple, yet surprisingly effective technique is to apply an *Iterated Local Search* (ILS) strategy, see [15, 2] We propose an ILS where the embedded heuristic is

based on tabu controlled ejection chains and uses random vertex ejections as *kick moves*. From a given start solution S1 a basic ILS scheme can be stated as follows.

```
Apply a kick move on S1 yielding S2
Perform a local search on S2 resulting in S3
Choose wether or not to accept S3
If S3 is accepted restart from S3, otherwise restart from S1
```

We use the random ejection of a single vertex v as a kick move. The ejected vertex v forms a new route on its own, while we link v's predecessor to its successor. Note that the kick move never introduces infeasibility. One of the main strengths of our procedure is its ability to merge routes in a way similar, but much more efficient, than the well known Clark & Wright savings procedure [3]. Therefore creating single vertex routes presents no problems and, in a way, only makes it easier for our procedure to find new moves, since ejecting vertices always creates more 'freedom' in existing routes. Note that, through the objective, the procedure is strongly biased to decrease the number of routes, so creating a new route by ejecting a vertex through a kick is a move not likely to be made by the procedure otherwise, even though the kick move is available as a forced combination of an S2 and T2 move. We thus follow the view expressed in [16], namely that a kick move should correspond to a modification of the structure that is not easily accessible to the moves already available or is unlikely to be chosen by the procedure itself.[9] The following pseudo code summarizes the ILS procedure we used.

```
ILS_EjectionChainProcedure:
{
     while( n < MaxNrOfIterations )
     {
         while( Depth < MaxDepth )
         {
             MoveToNewReferenceStructure
             GetOptimalTrialSolution
             Depth++
         }
         DetermineOptimalTrialSolutionOverCurrentEjectionChain
         UpdateBestKnownSolution
         if( n mod KickFrequency = 0 )
         {
             ReturnToCurrentBestKnownSolution
             ApplyKickMove
         }
         UpdateTabuList
         n++
     }
}
```

The inner while-loop processes new ejection chains. Each chain is initialized via an S-rule. At each level we check all admissible trial solutions via T-rules, storing the best. We keep adding levels via best admissible E-moves until we reach the maximum depth. The procedure then returns the best found trial solution to the ILS procedure, the outer while-loop. Periodically the procedure applies a

[9] Since we use Tabu search, instead of strict local search, the kick move is temporarily "irreversible", thus the procedure is forced to find an alternative way to repair the sustained "damage" obtained from the kick move. The "strength" of the kick is rather low. Ejecting more than one vertex per kick proved unsuccessful.

kick move on the best known solution. Note that the ILS procedure we use only accepts improvements. Other acceptance criteria could be used; see [4, 15].

The behavior of our ILS procedure is such that, in a sense, we always stay close the the current best solution value. This is in line with the so called *Pyramid Principle*, as stated by Glover [9]. The ILS enables us to search with a low tabu tenure for most of the time. Our computational results in Section 6 confirm that the ILS procedure proved to be able to generate high quality solutions in a small number of iterations, where one iteration is understood to be one pass of the outer while-loop.

5.2 Simulated Annealing

Instead of using ILS, we can also use Simulated Annealing (SA) as meta heuristic to control the ejections chain procedure. An interesting variant arises when we return each generated trial solution to an SA process. The SA process then either accepts the current trial solution, terminating the current chain or it rejects the trial solution, enabling the chain to grow further. If no trial solution is accepted the procedure will select the best encountered trial solution in the current chain and start a new chain from there.

```
SA_EjectionChainProcedure:
{
     while( n < MaxNrOfIterations )
     {
         while( Depth < MaxDepth )
         {
             MoveToNewReferenceStructure
             GetOptimalTrialSolution
             Depth++
             if( AnnealingAccept )
             {
                TerminateCurrentChain
             }
         }
         DetermineOptimalTrialSolutionOverCurrentEjectionChain
         UpdateBestKnownSolution
         UpdateTabuList
         n++
     }
}
```

This procedure has the ability to automatically converge from an SA process to a strict tabu search. In the beginning of the search, when the temperature is high, ejection chains will be terminated prematurely because the SA will easily accept. However, when the temperature becomes very low, we always reach the maximum depth of the chain. Therefore the best trial solution will be chosen, just as a strict tabu search process would do.

This procedure adds two important qualities to a strict SA process. First, the search is much more guided in the early stages. This is because the generation of neighbors is done via a tabu controlled process, the ejection chain, instead of a random generation. Second, the time at the end of the search is spent more efficiently: in the end a strict SA process will generate a great number of trial solutions that will all be rejected. This kind of behavior is not desired, particularly when the checking of feasibility is expensive, as is often the case in time window constrained settings.

6 Computational results

This section contains the test results for the two proposed meta heuristics. The algorithms were programmed in C++. Testing was done on a 2.5Ghz, AMD machine with 512MB ram. We performed the computational test on the well known Solomon test cases [19]. Testing was done on the 50 as well as the 100 vertex instances. Since we minimize the total distance, we can compare to the global optimal values, as stated in [20]. The computational results stated here are therefore restricted to those instances where **global** optimal values are known. In addition we tested the procedure on larger instances generated by Homberger [10], again restricted to instances where global optima are known [11]. The Homberger test sets extend the Solomon sets and share the same design.

Both procedures use the same design of tabu ejection chains. After each ejection chain, all ejected edges are declared tabu for an additional θ iterations, where θ is randomly drawn from the interval $[10, 30]$. As a diversification measure, we increased this interval to [40,50] (50 vertices) or [70,100] (100 vertices) for 100 iterations, if the procedure did not find an improvement for 1000 subsequent iterations. The maximum depth for an ejection chain was always 50. The time window violation penalty P was set to 100. For both procedures we used the same, trivial start solution where each client forms an individual route. For the 50 vertex sets we performed 50,000 iterations, and for the 100 vertex and Homberger sets 100,000 iterations.

The results of Table 1 confirm that we get extremely close to the optimal solutions, on all tested instances. We display the globally optimal solutions (vehicles used, travel distance), the vehicles used by our solutions, as well as the deviation from the optimum (in %). The columns labelled ARO display a statistic for the tightness of the time windows, the average relative overlap (ARO)[10].

We observe that the ILS procedure clearly outperforms the SA procedure. This is probably due to the effectiveness of the kick moves; in fact, the ILS scheme performed a kick very often, every 5 iterations. The comparably worse performance of the SA procedure might also be caused by a too simplistic cooling schedule. It proved difficult to find a reasonable cooling scheme for the larger test sets. Therefore we only list the results for the ILS procedure here.

The ARO statistic provides a clue on the performance of our procedure. It measures the average overlap between any two time windows. We observe that, on average, when the procedure performs worse, the ARO is high. This indeed makes sense, since a high ARO implies that the instance more closely resembles a VRP without time windows, while our procedure is explicitly designed for settings where the time windows are tight. In fact, in settings with a lot of overlap between time windows, path reversals might be useful. But as a tribute to (tight) time windows, path reversals are explicitly excluded in our algorithm design.

Finally, we note that the computation times are very reasonable, although we did not tune our implementation for speed. Within 10 seconds running time

[10] $\text{ARO} = 100 \cdot \sum_i \sum_{j>i} \left(\frac{Overlap_{ij}}{|TW_i|} + \frac{Overlap_{ij}}{|TW_j|} \right) / (n(n-1))$, where $|TW_i|$ is the length of time window i and n is the number of clients (the depot is not considered a client).

on each 50 vertex instance, the ILS provides solutions on average 1.98% away from the optimum. 1 minute for each 100 vertex instance yields 2.69% deviation from the optimum on average. Computation times, however, can be decreased considerably. For example, using earliest departure times and latest arrival times as defined in [13], one can check the feasibility of trial solutions in constant time.

50 VERTICES	GLOBAL		ILS		SA	
SET	ARO	Veh	Dist	Veh	Dev	Veh	Dev
C101	5.64	5	362.4	5	0.23	5	0.23
C102	30.43	5	361.4	5	0.21	5	0.21
C103	52.54	5	361.4	5	0.21	5	5.21
C104	80.66	5	358	5	0.25	5	4.06
C105	11.07	5	362.4	5	0.23	5	0.65
C106	8.09	5	362.4	5	0.23	5	0.65
C107	17.17	5	362.4	5	0.23	5	0.65
C108	23.82	5	362.4	5	0.23	5	2.40
C109	36.75	5	362.4	5	0.23	5	2.04
C201	3.90	3	360.2	3	0.44	3	0.44
C202	29.07	3	360.2	3	0.44	3	0.44
C203	52.06	3	359.8	3	0.45	3	0.45
C204	80.78	2	350.1	2	0.46	2	0.46
C205	9.17	3	359.8	3	0.45	3	0.45
C206	14.25	3	359.8	3	0.45	3	1.96
C207	24.13	3	359.6	3	0.45	3	0.70
C208	19.02	2	350.5	2	0.46	2	0.46
R101	6.08	12	1044	12	0.26	12	0.26
R102	30.33	11	909	11	0.27	11	1.06
R103	52.54	9	772.9	9	0.46	9	1.23
R104	79.39	6	625.4	6	2.67	6	2.96
R105	19.03	9	899.3	10	2.23	9	1.67
R106	39.29	5	793	8	0.28	8	0.28
R107	58.00	7	711.1	7	4.31	7	0.68
R108	81.64	6	617.7	6	0.41	6	3.91
R109	38.66	8	786.8	8	1.07	8	0.96
R110	58.07	7	697	7	0.34	7	1.17
R111	59.02	7	707.2	7	0.80	7	4.28
R112	79.00	6	630.2	6	1.88	6	2.36
R201	13.00	6	791.9	6	0.31	6	0.51
R202	35.62	5	698.5	6	1.18	5	3.42
R205	28.19	5	690.9	4	0.96	4	1.29
RC101	21.37	8	944	9	1.25	8	0.17
RC102	40.47	7	822.5	8	2.16	8	6.41
RC103	58.65	6	710.9	6	0.54	6	1.79
RC104	82.06	5	545.8	5	0.27	5	5.82
RC105	40.75	8	855.3	8	0.19	8	6.68
RC106	44.58	6	723.2	6	4.66	6	8.65
RC107	65.40	6	642.7	6	0.47	6	5.95
RC108	81.55	6	598.1	6	0.18	6	4.58
RC201	14.23	5	684.8	5	0.22	5	1.53
RC205	28.37	5	631	5	0.16	5	3.36
	39.38				0.79		2.20

100 VERTICES	GLOBAL		ILS		
SET	ARO	Veh	Dist	Veh	Dev
C101	6.37	10	827.3	10	0.20
C102	29.72	10	827.3	10	0.20
C103	52.78	10	826.3	10	0.21
C104	76.01	10	822.9	10	0.81
C105	12.19	10	827.3	10	0.20
C106	15.37	10	827.3	10	0.20
C107	18.17	10	827.3	10	0.20
C108	25.07	10	827.3	10	0.20
C109	37.63	10	827.3	10	0.20
C201	4.33	3	589.1	3	0.42
C202	28.32	3	589.1	3	0.42
C203	52.19	3	588.7	3	0.42
C205	9.35	3	586.4	3	0.42
C206	14.65	3	586	3	0.43
C207	18.41	3	585.8	3	0.42
C208	19.74	3	585.8	3	0.43
R101	6.23	20	1637.7	20	0.32
R102	29.31	18	1466.6	18	0.48
R103	51.22	14	1208.7	15	1.47
R105	18.62	15	1355.3	16	1.29
R106	38.34	13	1234.6	13	1.86
R107	57.06	11	1064.6	12	4.06
R109	38.13	13	1146.9	13	2.53
R110	57.21	12	1068	12	3.80
R111	56.13	12	1048.7	12	4.55
R201	13.012	8	1143.2	8	0.82
RC101	19.44	15	1619.8	17	2.28
RC102	38.21	14	1457.4	15	2.16
RC103	56.15	11	1258	12	2.74
RC105	35.84	15	1513.7	17	3.89
RC201	14.23	9	1261.8	10	1.83
	30.63				1.27

HOMBERGER	GLOBAL		ILS	
SET	ARO	Veh	Dist	Veh	Dev
c1_2_1	5.76	20	2698.6	20	0.22
c1_2_2	29.02	20	2694.3	20	1.08
c1_2_5	11.89	20	2694.9	20	1.23
c1_2_6	16.04	20	2694.9	20	0.43
c1_2_7	17.95	20	2694.9	20	0.23
c1_2_8	24.00	20	2684	20	0.80
c1_4_1	6.18	40	7138.8	40	0.22
r1_2_1	1.82	23	4667.2	26	3.20
	14.08				0.93

Table 1: Computational Results.

7 Conclusions and recommendations

One of the key reasons for the good performance of our algorithms for VRPTW is, we believe, due to the ability to generate powerful compound moves that do not require a path reversal. It must be noted, however, that LK-based methods, that do perform path reversals, are extremely efficient in settings without time windows. We observed that, on average, when there is high overlap in the time

windows, the procedure performs less effective. It is likely that the absence of path reversals is a reason. However, our concept can be adapted quite easily to include moves that use path reversals, too.

It can be considered a strength that the procedures do not, in any form, use pre-processing or post-processing. Also, the procedures can not be considered two-phased methods, since they always use the same, extremely simple, start solution. A strong feature of the iterated local search is that the kick move can be made very problem-specific, and can be used to decrease any possibly sustained infeasibility during the search.

Generalizing the reference structure by relaxing the constraint that one of the roots must be the core is likely to further improve the procedure. The resulting *Generalized Doubly Rooted* reference structure (GDR) is shown in figure 8. The stated rules can be used as guidelines to create new S, E and T-type rules to exploit the GDR structure. E rules can be constructed that are able to increase or decrease the number of routes. In contrast, in the stated procedure, the total number of routes can not be changed by more than one route per ejection chain. Further research is necessary to examine the potential of the GDR structure over the CDR structure. Finally, the procedure can, in all probability, easily be

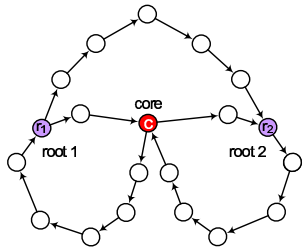


Fig. 8. Generalized Doubly Rooted reference structure (GDR)

extended to the multi-depot case by using the reference structure in Figure 8, but allowing the roots to lie on cycles not necessarily joined by the same depot.

Acknowledgements. This research was performed on behalf of the Center for Quantitative Methods, CQM BV, Eindhoven, The Netherlands. We are very grateful for all the support during the project. In addition, the authors would like to thank Fred Glover and Emile Aarts for all their valuable comments, suggestions and encouragement. We found their collaboration very inspiring.

References

1. E. H. L. Aarts and J. K. Lenstra. *Local search in combinatorial optimization.* Wiley, Chichester, UK, 1996.
2. C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, 2003.
3. G. Clark and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

4. M. den Besten, T. Stützle, and M. Dorigo. Design of iterated local search algorithms. In E. J. W. Boers, J. Gottlieb, P. L. Lanzi, R. E. Smith, S. Cagnoni, E. Hart, G. R. Raidl, and H. Tijink, editors, *Applications of Evolutionary Computing*, volume 2037 of *Lecture Notes in Computer Science*, pages 441–451, 2001.
5. L. M. Gambardella, E. Taillard, and G. Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 63–76. McGraw-Hill, 1999.
6. D. Gamboa, C. Rego, and F. Glover. Implementation analysis of efficient heuristic algorithms for the traveling salesman problem. *Computers and Operations Research*, 2004. To appear.
7. P. L. George and H. Borouchaki. *Delaunay Triangulation and Meshing, Applications to Finite Elements*. Hermes, 1998.
8. F. Glover. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65(1-3):223–253, 1996.
9. F. Glover and M. Laguna. *Tabu Search*. Kluwer, Dordrecht, NL, 1998.
10. J. Homberger. Extended solomon's VRPTW instances. www.fernuni-hagen.de/WINF/touren/inhalte/probinst.htm, 2000.
11. B. Kallehauge, J. Larsen, and O. B. G. Madsen. Lagrangian duality applied to the vehicle routing problem with time windows. *Computers and Operations Research*, 2005. To appear.
12. P. Kilby, P. Prosser, and P. Shaw. Guided local search for the vehicle routing problem. In S. Voss, S. Martello, I. H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 473–486. Kluwer, Boston, MA, 1997.
13. G. A. P. Kindervater and M. W. P. Savelsbergh. Vehicle routing: Handling edge exchanges. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 337–360. Wiley, Chichester, UK, 1997.
14. S. Lin and B. W. Kernighan. An effective heuristic for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.
15. H. R. Lourenco, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *The Handbook of Metaheuristics*, pages 321–353. Kluwer, Norwell, MA, 2002.
16. O. C. Martin and S. W. Otto. *Combining Simulated Annealing with Local Search Heuristics*, volume 63 of *Annals of Operations Research*, pages 57–75. 1996.
17. C. Rego. A subpath ejection method for the vehicle routing problem. *Management Science*, 44(10):1447–1459, 1998.
18. Y. Rochat and É. D. Taillard. Probalistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1:147–167, 1995.
19. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35:254–265, 1987.
20. P. Toth and D. Vigo. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2002.
21. J. Xu and J. Kelly. A network-flow based tabu search heuristic for the vehicle routing problem. *Transportation Science*, 30:379–393, 1996.