Randomized Adaptive Spatial Decoupling For Large-Scale Vehicle Routing with Time Windows

Russell Bent and **Pascal Van Hentenryck**
Brown University Box 1910

Abstract

In recent years, the size of combinatorial applications and the need to produce high-quality solutions quickly have increased steadily, providing significant challenges for optimization algorithms. This paper addresses this issue for large-scale vehicle routing problems with time windows, a class of very difficult optimization problems involving complex spatial and temporal dependencies. It proposes a randomized adaptive spatial decoupling (RASD) scheme for vehicle routing with time windows in order to produce high-quality solutions quickly. Experimental results on hard instances with 1,000 customers and 90 vehicles show that the RASD scheme, together with large neighborhood search, significantly improves the quality of the solutions under time constraints. Interestingly, the RASD scheme, when allowed to run longer, also improves the best available solutions in almost all the tested instances.

Introduction

The scale of optimization problems and the need for finding high-quality solutions quickly has grown steadily in recent years as optimization systems are increasingly deployed in operational, integrated settings. This trend generates significant issues for optimization research, changing its focus from finding optimal solutions to delivering high-quality solutions under time constraints. This paper examines the underlying algorithmic issues in the context of multiple vehicle routing with time windows (VRPTWs), which arise in many transportation applications including courier services, the scheduling of repairs in telecommunication companies, and supply-chain logistics. VRPTWs are particularly interesting in this respect, since instances with even 100 customers have not been solved optimally despite intense research. Hence finding high-quality solutions under time constraints for problems with 1,000 customers is a significant challenge.

Spatial and temporal decouplings (Hunsberger 2002) are natural avenues for speeding up optimization algorithms. Unfortunately they do not apply easily to large-scale VRPTWs which involve complex spatial and temporal dependencies. To remedy this limitation, this paper proposes a randomized adaptive spatial decoupling (RASD) scheme which iteratively selects random subproblems that can be optimized independently and reinserted into an existing solution. The decouplings obtained by the RASD scheme are adaptive since they depend on the current solution, not simply the instance data. The RASD scheme is also independent from the underlying optimization algorithm.

The RASD scheme was evaluated on large VRPTW instances with 1,000 customers and about 90 vehicles. The experimental results indicate that RASD($\mathcal{A}$), the RASD scheme with algorithm $\mathcal{A}$ to optimize the subproblems, produces significant improvements in solution quality over algorithm $\mathcal{A}$, when both algorithms must produce solutions within time constraints. Moreover, and perhaps surprisingly, the RASD scheme found new best solutions on almost all tested instances when allowed to run for an hour.

This paper reviews VRPTWs, their decouplings, and the difficulty in finding good decompositions. It then presents the RASD scheme, alternative algorithms, experimental results, and related work.

VRPTWs

A VRPTW instance is specified by the set $\mathcal{C}$ of customers, the set $\mathcal{V}$ of vehicles, and a depot d. Elements of $Sites = \mathcal{C} \cup \{d\}$ are called sites.

Every customer c has a demand $q_c \geq 0$ and a service time $s_c \geq 0$. The travel cost between sites i and j is t_{ij}. Each customer c has a time window $[e_c, l_c]$ constraining when it can be visited, where e_c and l_c represent the earliest and latest arrival times. Vehicles must arrive at customer c before the end of the time window l_c. They may arrive early but they have to wait until time e_c to be serviced. The depot also has a time window specifying when the vehicles may start and must return. Each vehicle has a capacity Q.

Solutions are specified in terms of vehicle routes and routing plans. A vehicle route starts from the depot, visits a number of customers at most once, and returns to the depot. It is thus a sequence $\langle d, c_1, \ldots, c_n, d\rangle$ or $\langle c_1, \ldots, c_n\rangle$ for short, where all c_i are different. The customers of a route $r = \langle c_1, \ldots, c_n\rangle$, denoted by *cust(r)*,

is the set $\{c_1, \ldots, c_n\}$. The size of a route, denoted by $|r|$, is $|cust(r)|$. The demand of a route, denoted by $q(r)$, is the sum of the demands of its customers. A route satisfies its capacity constraint if $q(r) \leq Q$. The travel cost *t(r)* of a route $r = \langle c_1, \ldots, c_n \rangle$ is the cost of visiting all its customers.

A routing plan is a set of routes $\{r_v \mid v \in \mathcal{V}\}$ in which every customer is visited exactly once. Observe that a routing plan assigns a unique earliest arrival time a_c for each customer c. It also assigns a unique return time $a(r)$ to the depot for each route r.

A solution to the VRPTW is a routing plan σ satisfying the capacity and time window constraints, i.e.,

$$\forall r \in \sigma : \; q(r) \leq Q \;\&\; a(r) \leq l_0 \;\wedge\; \forall c \in \mathcal{C} : a_c \leq l_c.$$

The size $|\sigma|$ of a routing plan σ is the number of non-empty routes in σ. The VRPTW problem consists of finding a solution σ which minimizes a lexicographic function consisting of the number of vehicles and the total travel cost, i.e., $f(\sigma) = \langle |\sigma|, \sum_{r \in \sigma} t(r) \rangle$. Modern algorithms for the VRPTW are organized in two stages, first minimizing the number of vehicles and then minimizing travel distance.

VRPTW Decouplings

This paper aims at finding decouplings to speed up the solving of large-scale VRPTWs. The goal of the decouplings is to decompose a VRPTW into subproblems that can be solved independently. More precisely, a decoupling of a VRPTW $\mathcal{P} = (\mathcal{C}, \mathcal{V})$ is a set of VRPTWs $\{\mathcal{P}_1 = (\mathcal{C}_1, \mathcal{V}_1), \ldots, \mathcal{P}_n = (\mathcal{C}_n, \mathcal{V}_n)\}$ such that

- $\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ is a partition of $\mathcal{C}$;
- $\{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ is a partition of $\mathcal{V}$;
- $\mathcal{P}_i$ $(1 \leq i \leq n)$ is feasible.

Each subproblem $\mathcal{P}_i$ in a VRPTW decoupling can then be solved independently to obtain a routing plan. The resulting plans σ_i can be smoothly integrated into a routing plan $\sigma = \{r \mid r \in \sigma_i \;\&\; 1 \leq i \leq n\}$ for $\mathcal{P}$, since they do not share customers or vehicles. Similarly, given a routing plan σ for $\mathcal{P}$, the routing plan for problem $\mathcal{P}_i$ is obtained by selecting in σ the routes of the vehicles in $\mathcal{V}$, i.e., $\text{PROJECT}(\mathcal{P}_i, \sigma) = \{r_v \in \sigma \mid v \in \mathcal{V}_i\}$.

Decoupling VRPTWs is Difficult

Because of the spatial and temporal nature of VRPTWs, finding decouplings seems a natural way to speed up the solution to large-scale instances. Indeed, spatial decouplings could exploit the geographical clustering of customers. Similarly, temporal decouplings may exploit the time windows of the customers. Unfortunately, these two aspects of VRPTWs often conflict, making it difficult to obtain spacial and temporal decouplings. Consider Figure 1 which depicts a high-quality solution to an instance with 1,000 customers and 90 vehicles. Even knowing this high-quality solution, there is no obvious spatial decoupling: some vehicles operate in relatively narrow regions, while others

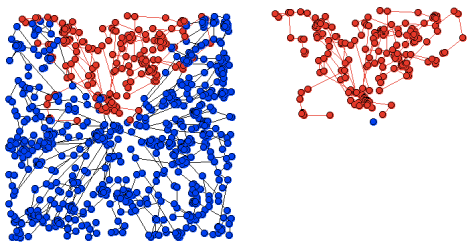

Figure 1: The Difficulty in VRPTW Decouplings

cover a wide area. The right part of Figure 1 shows only the red customers and their vehicles; it illustrates that, even in high-quality solutions, vehicles may travel over a large region. Hence static spatial decouplings are hard to find and may not produce high-quality solutions. Moreover, each vehicle in high-quality solutions often serves customers with diverse (tight or loose) time windows, making temporal decouplings difficult to find.

One might argue that VRPTW decouplings are too strong and other decompositions, allowing vehicles to be shared across spacial regions, should be considered. The difficulty then becomes how to recompose routes from the subproblems. Because of time windows and because of the fixed size of the fleet, such recompositions are typically difficult and little success was achieved using such an approach. The RASD scheme alleviates these difficulties by taking a more adaptive approach, deriving decouplings dynamically using both spacial information and the routing plan at hand.

The RASD Scheme

The RASD scheme is based on two main principles:

1. Starting from plan σ_0, it produces a sequence of plans $\sigma_1, \ldots, \sigma_j$ such that $f(\sigma_0) \geq f(\sigma_1) \geq \ldots \geq f(\sigma_j)$.
2. At step i, the scheme uses σ_{i-1} to obtain a decoupling $(\mathcal{P}_o, \mathcal{P}_s)$ of $\mathcal{P}$ with projected plan σ_o and σ_s. It reoptimizes $\mathcal{P}_o$ to obtain σ_o^* and the new plan $\sigma_i = \sigma_o^* \cup \sigma_s$.

The decoupling is spatial: it views the customer region as a circle and randomly select a wedge W to define $\mathcal{P}_o$. Unfortunately, as mentioned earlier, a wedge does not yield a decoupling in general, since some vehicles serving customers inside W may also serve customers outside the wedge. To remedy this problem, the RASD scheme proceeds in two steps. First, it collects the set of vehicles $\mathcal{V}_o$ serving customers in W in the current plan σ. Second, it defines $\mathcal{C}_o$ as all customers served in σ by the vehicles in $\mathcal{V}_o$, resulting in the subproblem $\mathcal{P}_o = (\mathcal{C}_o, \mathcal{V}_o)$. The "stable" subproblem $\mathcal{P}_s$ is defined as $\mathcal{P}_s = (\mathcal{C} \setminus \mathcal{C}_o, \mathcal{V} \setminus \mathcal{V}_o)$.

It remains to specify how to choose the wedge W. The idea is not to use wedge size only for the decoupling: Indeed some regions may have higher customer densities than others, leading to subproblems of very different nature for the same wedge size. Instead the the RASD scheme aims at producing problems with roughly the

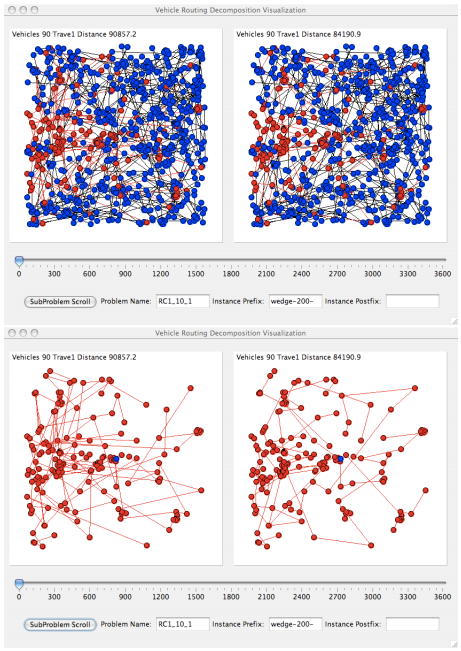

Figure 2: The First Decoupling of RASD.

Figure 3: The Decoupling of RASD After 5 Minutes.

same number C of customers. It first chooses the lower angle α of the wedge randomly. It then selects the upper angle β as the smallest angle greater than α producing a wedge with at least C customers.

Figures 2 and 3 depict the behavior of the RASD scheme visually. Figure 2 shows the initial plan σ_0 (top left) and the plan σ_1 (top right) after the first decoupling and optimization. The customers in the subproblem $\mathcal{P}_o$ are in red, the remaining ones in blue. The bottom part of Figure 2 shows the projected solution σ_o for subproblem $\mathcal{P}_o$ (bottom left) and its reoptimization σ_o^* (bottom right). As can be seen, the first subproblem is quite spread out, illustrating the difficulty in finding good decompositions. Figure 3 (top) shows the decoupling obtained after 5 minutes of execution. Here the decoupling is much nicer, the RASD solution after 5 minutes, being already of high-quality. Figure 3 (bottom) depicts the projected plan for subproblem $\mathcal{P}_o$ and its reoptimization for that decoupling.

The RASD scheme is depicted in Figure 4. The core of the algorithm is in lines 4–6 which decouple the VRPTW (line 4), reoptimizes subproblem $\mathcal{P}_o$ using algorithm $\mathcal{A}$ and the projected routing plan σ_o for $\mathcal{P}_o$ (line 5), and reinsert the new optimized subplan σ_o^* to obtain the new solution (line 6). These main steps are repeated until the time limit is reached. The decoupling is given in lines 10–15. The RASD scheme selects a random wedge W (line 11), collects all vehicles serving a customer in the wedge (line 12), and all the customers served by these vehicles (line 13). The customers and vehicles so obtained define the first subproblem and the other subproblem consisting of the remaining customers and vehicles (line 14). The wedge selection is given in lines 16–22. The lower angle α is first chosen randomly (line 17). The second angle β is chosen such that the wedge (α, β) contains at least N customers (a parameter of the implementation) and is the smallest such wedge. In other words, any other wedge (α, γ) with $\gamma < \beta$ does not contain N customers.

Alternative Algorithms

This section describes some alternative schemes to justify some of the design decisions underlying RASD. The first alternative, the SPASD scheme depicted in Figure 5, starts by partitioning the customer region with k wedges $W_1, \ldots, W_k$. It then considers each wedge in sequence, performing the decoupling and optimization for each of them once. The SPASD scheme guarantees that all regions are included in at least one optimization and is still adaptive in the sense that the decouplings depend on the current routing plan. Its main inconvenient is to consider each region only once, which is a significant drawback as discussed later.

The second alternative, the MPASD scheme depicted in Figure 6, executes the core of the SPASD algorithm multiple times. It can be viewed as a systematic version of the RASD scheme with the guarantee that all regions will be covered by multiple optimizations. It serves to evaluate the impact of randomization which contributes to the simplicity of RASD.

Experimental Results

This section presents the experimental results primarily on the class RC1 of benchmarks for 1,000 customers. These benchmarks, which are available at `www.top.sintef.no/vrp/benchmarks.html`, contain a

```
1.  function RASD(A, σ0) {
2.    σ ← σ0;
3.    repeat
4.      (Po, Ps) ← DECOUPLE(P,σ);
5.      σ*o ← A(Po, PROJECT(Po, σ));
6.      σ ← σ*o ∪ PROJECT(Ps, σ);
7.    until time limit
8.    return σ;
9.  }

10. function DECOUPLE(P, σ) {
11.   W ← SELECTWEDGE(P,σ);
12.   Vo ← {v ∈ V | ∃ c ∈ rv : c lies in W};
13.   Co ← ⋃v∈Vi cust(rv);
14.   return {(Co, Vo), (C \ Co, V \ Vo)};
15. }

16. function SELECTWEDGE(P, σ) {
17.   select α ∈ [0, 359];
18.   select β > α such that the wedge W = (α, β)
19.     (a) contains at least N customers;
20.     (b) is the smallest wedge satisfying (a);
21.   return W;
22. }
```

Figure 4: The RASD Scheme for VRPTW Decouplings

mix of loose and tight time windows and are representative of other problem classes. Results on classes R and C show similar behaviors for finding high-quality solutions quickly and finding new best solutions. Recall that the difficulty in these problems, once two-stage algorithms are considered, is mostly in optimizing travel distances. Hence the experimental results mostly focus on this second stage, and uses a solution with the minimal number of vehicles from the first phase. The experimental results use the large neighborhood search (LNS) (Shaw 1998) for algorithm $\mathcal{A}$. LNS is one of the most effective algorithmd for optimizating travel distances; it also has the benefits of easily accommodating side constraints, which is important in practical implementations. Some results for the first phase are also reported using ejection chains and simulated annealing. The experiments report the solution quality under various time constraints (e.g., 2.5, 5, 10, 15, ... minutes). Each reported result is the average of 50 runs on an AMD Athlon Dual Core Processor 3800.

Benefits of RASD Table 1 describes the solution quality under various time constraints for LNS and RASD(LNS). Each column describes a RC1 instance with 1000 customers and 90 vehicles. The clusters of rows consider various time constraints: 1, 2.5, 5, and 10 minutes. The row BK specifies the travel distance of the best known solution (prior to this research). The rows %Gap describes the improvement in solution quality of RASD(LNS) in terms of the best known solution

```
1.  function SPASD(A, σ0) {
2.    σ ← σ0;
3.    select random wedges W1, ..., Wk;
4.    for i = 1 to k
5.      (Po, Ps) ← DECOUPLEFORWEDGE(P,σ,Wi);
6.      σ*o ← A(Po, PROJECT(Po, σ));
7.      σ ← σ*o ∪ PROJECT(Ps, σ);
8.    return σ;
9.  }
10. function DECOUPLEFORWEDGE(P, σ,W) {
11.   Vo ← {v ∈ V | ∃ c ∈ rv : c lies in W};
12.   Co ← ⋃v∈Vi cust(rv);
13.   return {(Co, Vo), (C \ Co, V \ Vo)};
14. }
```

Figure 5: The SPASD Scheme for VRPTWs

```
1.  function MPASD(A, σ0) {
2.    σ ← σ0;
3.    repeat
3.      select random wedges W1, ..., Wk;
4.      for i = 1 to k
5.        (Po, Ps) ← DECOUPLEFORWEDGE(P,σ,Wi);
6.        σ*o ← A(Po, PROJECT(Po, σ));
7.        σ ← σ*o ∪ PROJECT(Ps, σ);
7.    until time limit
8.    return σ;
9.  }
```

Figure 6: The MPASD Scheme for VRPTWs

and is given by $100(\text{LNS} - BK)/(\text{RASD(LNS)} - Bk)$. RASD(LNS) is run with $N = 200$, i.e., the wedge must contain at least 200 customers. The results show that RASD(LNS) produces significant improvements in solution quality under time constraints. In average, it produces improvements of 35%, 29%, 17%, and 6% when the time constraints require solutions to be found within 1, 2.5, 5, and 10 minutes respectively. Figure 7 depicts the typical behaviour of LNS and RASD(LNS) on one of the benchmarks. The figure shows the dramatic improvements in solution quality under time constraints. It also shows that RASD(LNS) still dominates LNS when both algorithms run for an hour.

New Best Solutions RASD(LNS) was also instrumental in improving the best known solution for these benchmarks when allowed to run for an hour. Table 2 describes the previous best known solutions, the best solutions found during our experiments, and the value N used to obtain these solutions. These improvements are significant, can reach about 3%, and are typically obtained for small values of N (few customers in the wedges). It certainly interesting to observe that RASD(LNS) is also effective in finding very high-quality solutions when given more time. On the instances of class R, RASD(LNS) found four new best

BK	47143.9	44906.6	43782.6	41917.1	47632.3	46391.6	46157.7	45585.1	45405.5	45041.6
LNS (1)	70994.1	73614.3	77136.2	74131.2	69651.2	70753.1	70169.6	70196.6	69670.2	70730.8
RASD (1)	55974.2	54404.6	54210.1	53357.3	55723.4	57149.8	58179.0	55573.0	55952.7	55781.7
%Gap (1)	31.8	42.77	52.36	49.55	29.24	29.32	25.97	32.07	30.21	33.18
LNS (2.5)	62113.4	62727.3	64174.0	60959.1	61049.5	61782.1	61041.0	61064.6	60510.6	61002.4
RASD (2.5)	50006.9	47211.7	46218.4	45436.7	49525.3	49758.4	49590.0	48986.9	48959.0	48658.0
%Gap (2.5)	25.6	34.55	41.01	37.03	24.19	25.97	24.80	26.49	25.44	27.40
LNS (5)	55917.0	62727.3	54918.4	51958.0	55337.1	55674.0	54824.1	54700.8	54133.2	54127.5
RASD (5)	48254.1	47211.7	44739.4	43891.1	48036.4	48047.4	47723.1	47277.4	47303.0	46902.9
%Gap (5)	15.12	20.29	23.24	19.24	15.32	16.43	15.38	16.28	15.04	16.03
LNS (10)	51032.9	55170.6	47557.0	45599.5	50252.0	50589.7	49794.9	49331.2	48994.2	48655.9
RASD (10)	48019.7	46057.6	44116.4	43891.1	47418.3	47420.9	46906.6	46493.7	46484.1	46088.5
%Gap (10)	5.89	7.83	7.85	5.59	5.94	6.83	6.25	6.22	5.52	5.70

Table 1: Solution Quality Under Time Constraints.

RC1_10	1	2	3	4	5	6	7	8	9	10
BK	47143.9	44906.6	43782.6	41917.1	47632.3	46391.6	46157.7	45585.1	45405.5	45041.6
RASD	46747.9	44543.7	42979.4	42053.8	46169.3	45961.4	45495.4	44955.3	44955.3	44587.9
N	150	100	50	50	200	150	100	50	150	50

Table 2: Best Solutions Found Within an Hour.

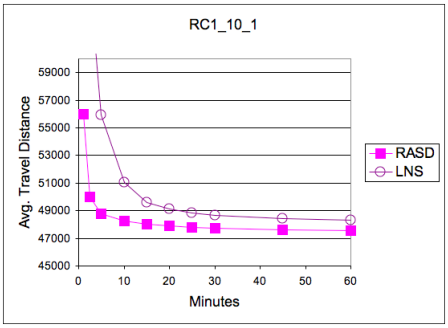

Figure 7: Benefits of RASD on RC1_10_1.

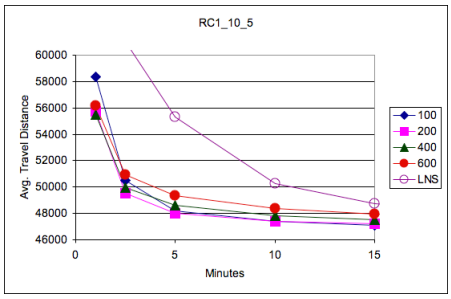

Figure 8: Impact of the Wedge Size N on RC1_10_5.

solutions, including a 10% improvement on R1_10_2.

The Impact of the Wedge Size Figure 8 depicts the impact of the wedge size N on solution quality for RC1_10_5 (other problems show similar results) for up to 15 minutes. In general, smaller wedge sizes give better results although the difference are not substantial. Larger wedge sizes (e.g., $N = 400$) are sometimes more efficient when time is limited to 1 minute since they are more likely to cover all regions. Wedge size $N = 200$ seems a good compromise overall for finding high-quality solutions quickly. Smaller wedge sizes resulted in better solutions in the long run (to a point).

Alternative Algorithms Figure 9 presents the solution quality of the alternative algorithm SPASD(LNS) and MPASD(LNS) on RC1_10_4. Algorithm SPASD(LNS) is clearly dominated as soon as 2.5 minutes or more are available. It even becomes worse than LNS after 10 minutes, which never happens for RASD(LNS) even after an hour. When only one minute of CPU time is available, SPASD(LNS) performs well, because it is guaranteed to see all customers, which is not necessarily the case for RASD(LNS), indicating that using SPASD(LNS) as a starting point is probably a good idea. MPASD(LNS) is typically close to, but dominated by, RASD(LNS), indicating there is little advantage to being systematic here.

Vehicle Reduction As mentioned earlier, vehicle reduction is generally much faster than the minimization of travel distance and high-quality solutions are generally reached much more quickly. However, VRPTW decouplings also improves this first phase of the algorithm under tight time constraints. Figure 10 examplifies this behavior whenever simulated annealing or an ejection chain algorithm is used in the first phase. What is interesting here is that the RASD scheme behaves similarly regardless of the underlying optimization algorithm.

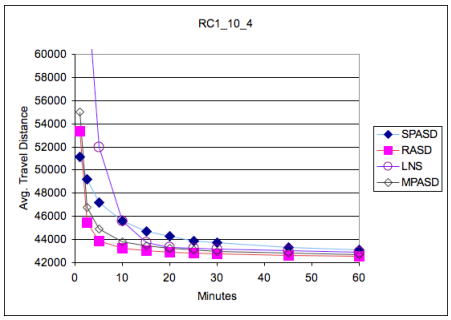

Figure 9: Performance of Alternative Algorithms.

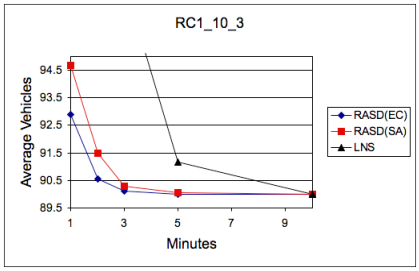

Figure 10: Decouplings for Vehicle Reduction.

Related Work

There are literally hundreds of papers discussing vehicle routing problems and their variations. See (Cordeau *et al.* 2001; Braysy & Gendreau 2005a; 2005b) for recent surveys. Almost all papers focus on problems of relatively small size which, as mentioned earlier, are already extremely difficult. Unfortunately, many of the proposed techniques do not scale well and some recent papers specifically address large-scale problems. (Bouthillier, Crainic, & Kropf 2005; Bouthillier & Crainic 2005) use parallel computation for scalability. Their main contribution is an architecture allowing different search strategies to run in parallel and to communicate their progress. Their experimental results are not competitive, probably due the simplicity of search strategies used. Their approach can also be thought of as a decomposition of the search strategy, whereas the RASD scheme relies on problem decomposition. To date, the most successful approach for solving large-scale VRPTWs is an advanced evolutionary technique (Mester & Braysy 2005) building upon the success of earlier algorithms (e.g., (Braysy, Dullaert, & Gendreau 2004; Homberger & Gehring 1999)). The main innovation in (Mester & Braysy 2005) are the incorporation of sophisticated diversification schemes (e.g., using guided local search) into an evolutionary framework.

It is useful to relate the RASD to the approach in (Hunsberger 2002) which impose specific temporal constraints to obtain decouplings. RASD uses spatial decouplings that constrain specific subsets of customers to be served by designated vehicles. Moreover, the use of decoupling is fundamentally different. *The idea is to iteratively obtain new decouplings to optimize an existing plan by re-optimizing subproblems.* This use of decouplings also contrast with traditional decomposition techniques in constraint satisfaction (Dechter 2003).

It is important to contrast LNS (Shaw 1998) and the RASD scheme. In LNS, the basic step consists of removing related customers from a plan σ and to reinsert them in σ using an optimization algorithm. The RASD scheme can also be thought of as removing related customers with a fundamental difference: *the removed customers defines a VRPTW subproblem of (significantly) smaller size which can solved independently.* This is critical for finding high-quality solution quickly. Obviously, the two approaches are synergetic since our results are obtained using RASD(LNS).

Conclusion

This paper proposes a randomized adaptive spatial decoupling (RASD) scheme for producing high-quality solutions to large-scale VRPTWs quickly. Based on the current plan, the RASD scheme repeatedly and adaptively obtains random spatial VRPTW decouplings which are re-optimized and re-inserted in the plan. Experimental results on hard instances with 1,000 customers show that the RASD scheme, together with LNS, significantly improves the quality of the solutions under time constraints, while also producing new best solutions when allowed to run for about an hour.

References

Bouthillier, A. L., and Crainic, T. 2005. A Cooperative Parallel Meta-Heuristic for the VRPTW. *C&OR* 32.

Bouthillier, A. L.; Crainic, T.; and Kropf, P. 2005. A Guided Cooperative Search for the VRPTW. *IEEE Intelligent Systems* 20 (4):36–42.

Braysy, O., and Gendreau, M. 2005a. VRPTW Part I: Route Construction and Local Search Algorithms. *Transportation Science* 39:104–118.

Braysy, O., and Gendreau, M. 2005b. VRPTW, Part II: Metaheuristics. *Transportation Science* 39:119–139.

Braysy, O.; Dullaert, W.; and Gendreau, M. 2004. Evolutionary Algorithms for the VRPTW. *Journal of Heuristics* 20:587–611.

Cordeau, J.-F.; Desaulniers, G.; Desrosiers, J.; Solomon, M.; and Soumis, F. 2001. The VRPTW. *The Vehicle Routing Problem: SIAM Monographs on Discrete Mathematics and Applications* 157–194.

Dechter, R. 2003. *Constraint Processing.* Morgan Kaufmann.

Homberger, J., and Gehring, H. 1999. Two Evolutionary Metaheuristics for the VRPTW. *INFOR* 37:297–318.

Hunsberger, L. 2002. Algorithms for a Temporal Decoupling Problem in Multi-Agent Planning. In *AAAI'02.*

Mester, D., and Braysy, O. 2005. Active Guided Evolution Strategies for Large Scale VRPTWs. *C&OR* 32:1593–1614.

Shaw, P. 1998. Using Constraint Programming and Local Search Methods to Solve VRPs. In *CP'98*, 417–431.