Waiting and Relocation Strategies in Online Stochastic Vehicle Routing

Russell Bent and Pascal Van Hentenryck
Brown University
{rbent,pvh}@cs.brown.edu

Abstract

This paper considers online stochastic multiple vehicle routing with time windows in which requests arrive dynamically and the goal is to maximize the number of serviced customers. Contrary to earlier algorithms which only move vehicles to known customers, this paper investigates waiting and relocation strategies in which vehicles may wait at their current location or relocate to arbitrary sites. Experimental results show that waiting and relocation strategies may dramatically improve customer service, especially for problems that are highly dynamic and contain many late requests. The decisions to wait and to relocate do not exploit any problem-specific features but rather are obtained by including choices in the online algorithm that are necessarily sub-optimal in an offline setting.

1 Introduction

Vehicle routing with time windows is a hard combinatorial optimization problem with many important applications in distribution and transportation scheduling. It has received considerable attention in the last decades and sophisticated algorithms are now available to find near-optimal solutions in reasonable time. In recent years, attention has shifted to online and/or stochastic versions of the problem. The stochastic and online versions are motivated by the inherent uncertainties arising in many industrial problems and technological developments, such as onboard computers and communication systems, which give transportation systems the opportunity to update plans even after the vehicle has been deployed.

In online stochastic problems, customers arrive dynamically as the algorithm proceeds and each customer request has a time window during which it can be served. The algorithm must decide whether to accept or reject the request upon arrival. If it is accepted, the online algorithm must serve the request. The online algorithm typically has two black-boxes available to make decisions: an optimization algorithm for the deterministic version of the problem and a conditional sampling procedure to generate future requests (e.g., [Chang *et al.*, 2000; Benoist *et al.*, 2001]).

Online stochastic vehicle routing was first studied in [Bent and Van Hentenryck, 2003; Bent and Van Hentenryck, 2004c] using a sampling-based approach. Their idea is to generate scenarios consisting of existing and sampled customers, to solve the scenarios using large neighborhood search [Shaw, 1998], and to make online decisions based on the scenario solutions. Compared to other online stochastic problems such as packet scheduling [Chang *et al.*, 2000][1] and reservation systems [Benoist *et al.*, 2001], online stochastic vehicle routing introduces an additional difficulty: it takes time to serve a request. Indeed serving a request involves moving to the customer, and then processing the request. In addition, a routing plan for a scenario may schedule a "sampled" customer (in contrast to an actual request) on a vehicle, something that never occurs in these other applications. *How to best proceed when this situation arises is the key issue studied in this paper.*

[Bent and Van Hentenryck, 2004c] took a conservative approach to this issue: Their algorithm, which is reviewed subsequently, first filters all the "sampled" customers from the plans, leaving only actual requests. Vehicles, when they become idle, are then sent to actual customers, chosen from the filtered solutions of the scenarios. Apparently, their design decision was motivated by the fact that sampled customers may never actually place a request. However, the fact that a sampled customer is served next in a scenario solution provides insight into the nature of the uncertainty and solutions.

The key contribution of this paper is to *propose two novel strategies, waiting and relocation, to address this issue and to better exploit stochastic information in online vehicle routing*. The waiting strategy recognizes that it is sometimes beneficial for a vehicle to wait at the current location instead of moving to a known customer. The relocation strategy goes one step further and may move vehicles to customer locations where no requests were placed (yet). These new strategies make several fundamental contributions:

- The vehicles can wait or relocate anywhere and at any time during the algorithm execution. This contrasts with earlier approaches (e.g., [Larsen *et al.*, 2004; van Hemert, 2004]) where waiting and relocation points are defined a priori using knowledge of the distribution, clustering of the customers, and heuristics.
- The decisions of when and where to wait are systematically derived from stochastic information. Indeed, wait-

[1] This paper also contains a comparison between online stochastic optimization and POMDP approaches.

ing and relocation simply make available to the online algorithms decisions that are necessarily sub-optimal in the offline setting. This contrasts with heuristics to distribute waiting time in routing plans (e.g., [Mitrovic-Minic *et al.*, 2004; Mitrovic-Minic and Laporte, 2004]) which do not use stochastic information.

- A decision to wait or relocate solely relies on the scenario solutions, not on specific properties of the distribution which is used as a black-box. Hence the strategies should naturally transfer to a variety of applications.

Experimental results show that the waiting and relocation strategies may produce dramatic improvements in customer service compared to earlier algorithms, bridging much of the gap between the online solution and an offline, a posteriori solution where all the uncertainty has been revealed. The improvements are particularly impressive for highly dynamic instances with many late customers, which are particular challenging for earlier algorithms.

The rest of the paper is organized as follows. Sections 2–3 present the offline and online problems. Sections 4–6 present the online stochastic algorithm in stepwise refinements, concluding with the waiting and relocation strategies. Section 7 presents the problem instances and the experimental results.

2 The Offline Problem

The Input Data A vehicle routing problem is specified by a number of customers that must be visited by a pool of vehicles. Each customer makes a request that must be served within a time window and takes some capacity from the vehicle. Each vehicle starts at the depot, serves some customers, and must return to the depot by the deadline.

Each problem contains a set R of n customers and a depot o. The set S of sites is thus $R \cup \{o\}$. The travel time between sites i and j is denoted by $d(i, j)$. Each request is associated with a customer and, since each customer makes at most one request, we use customer and request interchangeably. Every request c has a demand $q(c) \geq 0$ and a service time $p(c) \geq 0$. Each instance has a pool of m identical vehicles with capacity Q. Each vehicle starts from the depot.

Each customer c has a time window specified by an interval $[e(c), l(c)]$ satisfying $e(c) \leq l(c)$. The time window represents the earliest and latest possible arrival times of a vehicle serving customer c. In other words, the service for customer c may start as early as $e(c)$ and as late as $l(c)$. A customer c may not be served before $e(c)$ but a vehicle arriving early to serve c may wait at the site until time $e(c)$ before beginning service. The depot has a time window $H = [e_0, l_0]$, which represents the earliest departure and latest possible return for the vehicles. Typically, e_0 denotes the beginning of the day and l_0 is the deadline by which all vehicles must return.

Routing Plans Optimization algorithms for vehicle routing typically return a routing plan that specifies the order in which each vehicle visits its customers. A vehicle route, or route for short, starts at the depot, serves some customers, and returns to the depot. A customer appears at most once on a route. Hence a route is a sequence $\langle o, c_1, \ldots, c_n, o \rangle$, where $c_i \in R$ and all c_i are distinct. The capacity of a route ρ is the sum of its customer capacities, i.e., $q(\rho) = \sum_{i=1}^{n} q(c_i)$. A routing plan is a tuple of routes $(\rho_1, \ldots, \rho_m)$ one for each vehicle, in which each customer appears at most once. We also use $cust(\rho)$ and $cust(\gamma)$ to denote the customers of a route ρ and a plan γ. A routing plan assigns a unique successor and predecessor for each served customer and depot. For a plan γ, the successor of site c is denoted by c^+ and the predecessor is denoted by c^-.

Departure Times Routing plans do not prescribe departure times for the vehicles. These departure times are typically not uniquely defined: a vehicle may depart at different times from specific customers and still visit all its assigned customers before the deadline. In addition to the routing plan, a solution will also consist of an assignment $\sigma : R \to H$ of starting times to all customers.

The Vehicle Routing Problem We are now in position to describe the vehicle routing problem. A solution to a vehicle routing problem with time windows (VRPTW) is a routing plan $\gamma = (\rho_1, \ldots, \rho_m)$ and a starting time assignment σ satisfying the capacity and time window constraints, i.e.,

$$C(\gamma) \equiv \begin{cases} q(\rho_j) \leq Q & (1 \leq j \leq m) \\ \sigma(c) - p(c) \leq l(c) & (c \in cust(\gamma)) \\ \sigma(c) \geq \max(e(c), & \\ \quad \sigma(c^-) + d(c^-, c)) + p(c) & (c \in cust(\gamma)) \\ \sigma(c) + d(c, c^+) \leq l(o) & (c \in cust(\gamma)) \end{cases}$$

The objective is to find a solution maximizing the number of served customers $|cust(\gamma)|$. This objective function differs from the optimization criterion used the Solomon benchmarks, where the goal is to minimize the number of vehicles and, in case of ties, to minimize the total travel time. This highlights the difference between strategic planning and operational decision making. As maximizing the number of served customers is very difficult on the problems considered here, the cost associated with relocating or waiting (as factored into the travel time) is negligible.

Notations If S is a non-empty sequence, $\text{FIRST}(S)$ and $\text{LAST}(S)$ denote the first and the last element of a non-empty sequence. The concatenation of two sequences S_1 and S_2 is denoted by $S_1 : S_2$. If S is a sequence and S^- is a prefix of S, then $S - S^-$ denotes the suffix S^+ such that $S = S^- : S^+$. If S is a sequence and R is a set, $\text{FILTER}(S, R)$ denotes the sequence obtained by removing the elements of R from S.

3 The Online Problem

In the online problem, requests arrive dynamically as the algorithm proceeds. The online algorithm maintains a partial routing plan $\gamma^- = \langle \rho_1^-, \ldots, \rho_m^- \rangle$ consisting of a partial route ρ_i for each vehicle i. It also maintains a partial assignment σ^- of starting times for all customers in $cust(\gamma^-) \setminus \{\text{LAST}(\rho_1), \ldots, \text{LAST}(\rho_m)\}$. These times specify when the vehicle serving a given customer has departed to the next customer on the same vehicle. The last customers on the vehicles have no departure times, since they have not been served yet.

The online algorithms have at their disposal an optimization algorithm $\mathcal{O}$. Given a set of customer requests R and

ONLINE ALGORITHM $\mathcal{A}(\langle R_1, \ldots, R_h \rangle)$
1 $\rho_0 \leftarrow \langle\rangle$;
2 $\sigma_0 \leftarrow \sigma_\perp$;
3 $\Gamma \leftarrow$ GENERATESOLUTIONS(ρ_0, σ_0, R_1);
4 **for** $t \in H$
5 **do** $\boldsymbol{A_t} \leftarrow$ ACCEPTREQUESTS$(\rho_{t-1}, \sigma_{t-1}, R_t, \boldsymbol{A_{t-1}}, \Gamma)$;
6 $\Gamma \leftarrow$ UPDATEPLANS$(\rho_{t-1}, \sigma_{t-1}, \boldsymbol{A_t}, \Gamma)$;
7 **if** IDLE$(\rho_{t-1}, \sigma_{t-1})$
8 **then** $s_t \leftarrow$ CHOOSEREQUEST$(\rho_{t-1}, \boldsymbol{A_t}, \Gamma)$;
9 $\rho_t \leftarrow \rho_{t-1} : s_t$;
10 $\sigma_t \leftarrow \sigma_{t-1}[\text{LAST}(\rho_{t-1}) \leftarrow t]$;
11 $\Gamma \leftarrow \{\ \rho \in \Gamma \mid \text{FIRST}(\rho - \rho_{t-1}) = s_t\ \}$;
12 **else** $\rho_t \leftarrow \rho_{t-1}$;
13 $\sigma_t \leftarrow \sigma_{t-1}$;
14 $\Gamma \leftarrow \Gamma \cup$ GENERATESOLUTIONS$(\rho_t, \sigma_t, \boldsymbol{A_t})$;
15 **return** (ρ_h, σ_h);

GENERATESOLUTIONS$(\rho_t, \sigma_t, \boldsymbol{A_t})$
1 $\Gamma \leftarrow \emptyset$;
2 **repeat**
3 $R \leftarrow$ SAMPLE(t);
4 $\Gamma \leftarrow \Gamma \cup \{\mathcal{O}(\rho_t, \sigma_t, \boldsymbol{A_t}, R)\}$;
5 **until** time $t + 1$
6 **return** Γ;

Figure 1: Online Stochastic Routing

a pair (γ^-, σ^-), $\mathcal{O}(\gamma^-, \sigma^-, R)$ returns a routing plan $\gamma^+ = \langle \rho_1^- : \rho_1^+, \ldots, \rho_m^- : \rho_m^+ \rangle$ maximizing $|cust(\gamma^+)|$ and satisfying $C^-(\gamma^+)$, where C^- denotes the problem-specific constraints C where the time windows of each customer c in $cust(\gamma^-) \setminus \{\text{LAST}(\rho_1), \ldots, \text{LAST}(\rho_m)\}$ have been tightened to $[\sigma^-(c), \sigma^-(c)]$. The online algorithms also use a procedure SAMPLE(t) to conditionally sample the request distribution from time t to the time horizon.

The rest of this paper describes the online stochastic algorithms in stepwise refinements using the algorithms of [Bent and Van Hentenryck, 2004a] as a basis to clearly identify our contributions. The algorithms are presented for a single vehicle, their generalization to multiple vehicles being obtained naturally using pointwise decisions [Bent and Van Hentenryck, 2004a].

4 Online Vehicle Routing

We now present the generic online routing algorithm. Since there is only one vehicle, a routing plan is simply the vehicle route and we use both terms interchangeably. The generic online algorithm is depicted in Figure 1. It maintains a set of plans Γ representing scenario solutions that are used to make decisions over the course of the computation. At every time t, the algorithm also maintains a partial routing plan ρ_t, its associated departure times σ_t, and R_t the requests that become available. Finally, the algorithm assumes that the set of requests R_1 is available before the start of the computation. The implementation also includes service guarantees: Once a request is accepted, it must be served.

Lines 1–2 initialize the partial routing plan and the departure times, while line 3 generates the initial set of plans used in the decisions. The body of the algorithm (lines 5–15) first

CHOOSEREQUEST-$\mathcal{C}(\rho_t, \boldsymbol{R_t}, \Gamma)$
1 $F \leftarrow \bigcup_{i=1}^{t} R_i$;
2 **for** $r \in F$
3 **do** $f(r) \leftarrow 0$;
4 **for** $\rho \in \Gamma$
5 **do** $r \leftarrow$ FIRST(FILTER$(\rho - \rho_t, F)$);
6 $f(r) \leftarrow f(r) + 1$;
7 **return** $argmax(r \in F)\ f(r)$;

Figure 2: Consensus for Online Stochastic Routing.

determines whether to accept any new requests that have arrived. Next those plans that cannot accommodate the new accepted requests $\boldsymbol{A_t}$ (line 6) are removed from Γ. It is important to stress that a least one plan $p \in \Gamma$ should be able to accommodate the requests in $\boldsymbol{A_t}$ (through insertion or replacement of an equivalent sampled customer) since otherwise the algorithm cannot provide the necessary service guarantees. In this paper, customers are accepted greedily whenever a routing plan can accommodate them. Using stochastic information for accepting/rejecting did not bring significant improvements. The algorithm then determines whether the vehicle is idle, that is whether service is completed for the last customer in ρ_{t-1} given the departure times in σ_{t-1}. If the vehicle is busy traveling or servicing the last customer in ρ_{t-1}, the routing plan and departure times remain the same (lines 12–13) and the algorithm simply continues generating plans (line 14). Otherwise, the vehicle is idle and the algorithm chooses a request s_t to serve using the plans in Γ (line 8), augments the routing plan (line 9) and the departure times (line 10), and updates Γ to remove the plans incompatible with the decisions (line 11). It is useful to review some of the details of the algorithm.

- a vehicle is idle at time t for a plan $\langle s_1, \ldots, s_k \rangle$ and departure times σ if

  $$k = 0 \vee \max(\sigma(s_{k-1}) + d(s_{k-1}, s_k), e(s_k)) + p(s_k) \leq t.$$

  In other words, a vehicle is idle when its route has no customers or when it has finished serving its last customer s_k by time t.
- The algorithm assigns the departure time of the last customer in ρ_{t-1} to time t in line 10. The vehicle thus departs for customer s_t at time t.
- A routing plan ρ that next visits a customer other than s_t must be removed from Γ since its decisions are incompatible with ρ_t (line 11).

Figure 1 also depicts how to generate plans. Line 3 of function GENERATESOLUTIONS generates a scenario by sampling the distribution from time t to the horizon (the time in which the vehicles must return to the depot). Line 4 calls the optimization algorithm with the routing plan and departure times at time t. It remains to specify how to make decisions. Figure 2 shows how to implement function CHOOSEREQUEST to obtain the consensus algorithm $\mathcal{C}$ from [Bent and Van Hentenryck, 2004d] where the details can be found. Algorithm $\mathcal{C}$ considers all known requests F (line 1) and initializes their evaluations (lines 2–3). It then considers each routing plan $\rho \in \Gamma$ (line 4), retrieves the request served

CHOOSEREQUEST-CW($\rho_t, \boldsymbol{A_t}, \Gamma$)
1 $F \leftarrow \bigcup_{i=1}^{t} A_i$;
2 **for** $r \in F \cup \{\perp\}$
3 **do** $f(r) \leftarrow 0$;
4 **for** $\rho \in \Gamma$
5 **do** $r \leftarrow$ FIRST$(\rho - \rho_t)$;
6 **if** $r \in F$
7 **then** $f(r) \leftarrow f(r) + 1$;
8 **else** $f(\perp) \leftarrow f(\perp) + 1$;
9 **return** $argmax(r \in F \cup \{\perp\})\ f(r)$;

Figure 3: The Consensus Algorithm with a Waiting Strategy.

next in ρ, and increments its credit (line 6). The request in F with the best evaluation is selected in line 7.

It is important to emphasize a critical point in this implementation. A solution $\rho \in \Gamma$ is a routing plan $\rho = \rho_{t-1} : \rho^+$ starting with partial route ρ_{t-1} followed by a sequence of requests coming from F and the sampling. As a consequence, there is no guarantee that the request s served next on the vehicle, i.e., $s =$ FIRST(ρ^+), is an actual request ($s \in F$), not a sampled customer ($s \notin F$). This is precisely why the implementation in Figure 2 uses FILTER(ρ^+, F) to prune plan ρ^+ and keep only the requests in F. This guarantees that FIRST(FILTER(ρ^+, F)) returns a real customer and that the vehicle departs for a customer who requested service.

5 A Waiting Strategy

The algorithm by [Bent and Van Hentenryck, 2004c] filters sampled customers before selecting the request. This conservative approach ensures that the vehicle always moves to a known customer, not a sampled request. This section investigates a waiting strategy based on the recognition that it may be beneficial for the vehicle to wait at its current location instead of serving customers too eagerly. For instance, the fact that the solution ρ to a scenario at time t starts with a sampled customer, that is

$$\rho = \rho_{t-1} : \rho^+ \ \wedge \ \text{FIRST}(\rho^+) \notin F,$$

indicates that it may be beneficial to wait since the sampled request may materialize, in which case it must be served before the first accepted customer. The difficulty is to decide when to wait in a systematic fashion given that the algorithm has solved multiple scenarios, all of which may have different customers to serve next in their routing plans. Figure 3 depicts a natural implementation. Its key idea is to add a wait action $\perp$ to the accepted requests. When considering a plan $\rho \in \Gamma$, the algorithm retrieves the request r to serve next in the scenario (line 5). In the case of an accepted request ($r \in F$), the evaluation of r is incremented. Otherwise, if r is a sampled customer ($r \notin F$), the evaluation of the wait action is incremented. The implementation then selects the element of $F \cup \{\perp\}$ with the best evaluation, which may be either an accepted request or the wait action. The online generic routing algorithm must also be generalized slightly to wait. When the request is the wait action, the algorithm modifies neither the routing plan nor the departure times.

Waiting heuristics have attracted considerable attention recently (see, for instance, [Mitrovic-Minic *et al.*, 2004; Mitrovic-Minic and Laporte, 2004]). The beauty in the algorithm presented here is that the choice of when and where to wait is fully automatic and guided by the scenarios.

CHOOSEREQUEST-CR($\rho_t, \boldsymbol{A_t}, \Gamma$)
1 **for** $r \in Customers$
2 **do** $f(r) \leftarrow 0$;
3 **for** $\rho \in \Gamma$
4 **do** $r \leftarrow$ FIRST$(\rho - \rho_t)$;
5 $f(r) \leftarrow f(r) + 1$;
6 **return** $argmax(r \in Customers)\ f(r)$;

Figure 4: Consensus with a Relocation Strategy.

6 A Relocation Strategy

The waiting strategy recognizes that it may be beneficial to wait at the current location instead of serving an accepted request. It is especially appropriate for problems in which the bottleneck is to minimize travel times and it is reasonably easy to serve the customers. When the challenge is in maximizing the number of served requests, it is appealing to explore a relocation strategy and to consider moving to the location of sampled customers. Once again, the difficulty is to determine when and where to move. Figure 4 proposes a natural relocation strategy. Its fundamental idea is to avoid differentiating between accepted and sampled customers: the vehicle simply moves to the request with the best evaluation. Lines 1–2 initialize the evaluation of all customers, lines 4–5 increments the first request, and line 6 selects the request with the best evaluation. The selected request may be either an accepted or a sampled request.

A relocation strategy may be beneficial for improving the number of served requests because it anticipates future requests and positions the vehicle to serve them quickly. It is never advantageous when minimizing travel times, since it may move to locations where no requests will ever materialize. Observe also that when and where to relocate is also fully automatic and systematically derived from the scenario solutions. This contrasts with other approaches (e.g., [Larsen *et al.*, 2004; van Hemert, 2004]) where relocation points are created using heuristics based on a priori information for specific problems, instances, and distributions.

7 Experimental Results

We now describe the experiments that compare our algorithms with those of [Bent and Van Hentenryck, 2004c].

The Benchmarks The online vehicle-routing problems are generated from the Solomon benchmarks [Solomon, 1987], a collection of very challenging vehicle-routing problems with 100 customers, many of which have yet to be solved optimally. The stochastic versions were developed by [Bent and Van Hentenryck, 2004c] where the details are found. We review the salient features of these benchmarks. The problems are divided into classes 1 through 5. The degree of a dynamism (DOD) of a problem is the ratio of the number of stochastic customers over the number of total customers. The class 1 problems are characterized by early arriving requests. and class 2 problems by more late arriving requests. The third

class mixes class 1 and 2. For these three classes, the average DOD is 44%. Class 4 considers problems with more late arriving customers than class 2 and have an average DOD of 59%. Class 5 considers problems with a higher proportion of stochastic customers, i.e. an average DOD of 81%. In all problems, the expected number of customers is 100.

The Algorithms The results compare local optimization (LO) with the consensus and regret algorithms which may include the waiting and relocation strategies. Algorithm LO is a generalization of the parallel tabu-search algorithm in [Gendreau *et al.*, 1999]. It generates multiple routing plans on the accepted customers. These plans are then used to accept or reject new customers and to select the decisions at each time step. LO is thus close to algorithm $\mathcal{C}$, the main difference being that no stochastic information is exploited. The consensus algorithms $\mathcal{C}$, $\mathcal{CW}$, and $\mathcal{CR}$ have been fully described in this paper: $\mathcal{C}$ is the algorithm originally proposed by [Bent and Van Hentenryck, 2004c], while $\mathcal{CW}$ and $\mathcal{CR}$ respectively include the waiting and relocation strategies. The regret algorithms, $\mathcal{R}$, $\mathcal{RW}$, and $\mathcal{RR}$, improved upon the consensus algorithms by using a sub-optimality approximation to evaluate the value of scheduling each request next on the vehicles and are described in detail in [Bent and Van Hentenryck, 2004b]. They use a simple and fast sub-optimality approximation whose details are described in [Bent *et al.*, 2005]. Consider the decision of choosing which customer to serve next on vehicle v and let s_t be the next customer on the route of vehicle v at time t. To evaluate the regret of another customer r on the same vehicle v, the sub-optimality approximation determines whether there is a feasible swap of r and s_t on v, in which case the regret is zero. Otherwise, if such a swap violates the time-window constraints, the regret is 1.

Initial Plans and Online Process The online stochastic algorithms generate and solve 50 scenarios to select the decisions at time 0. The algorithms also generate and solve 50 additional samples to create plans given the set of decisions at time 0. As these scenarios are generated and solved ahead of time, the number of scenarios can be arbitrarily large depending on the application. Each such optimization is allocated one minute. During the online execution, each optimization runs for 10 seconds and uses the LNS procedure from [Shaw, 1998]. All algorithms are executed on an AMD Athlon 64 3000 processor with 512MB of RAM running Linux. Each of the instances is run 50 times to account for the nondeterministic nature of the algorithms. In the results, we often omit the words "in the average" for brevity.

The Results Figures 5 and 6 summarize all the results for the regret and consensus algorithms regarding the number of unserviced customers. (All customers can be served in the offline, a posteriori problems.) The figures depict the results for the specific instances and a linear regression for each class of algorithms. Some specific results are cropped by the graph extent to maintain its readability. The main result is the outstanding behavior of $\mathcal{RR}$, i.e., the regret algorithm with a relocation strategy. From the interpolations, it can be seen that algorithm $\mathcal{RR}$ dominates all the algorithms for DODs over 50% and that the improvements increase substantially as the

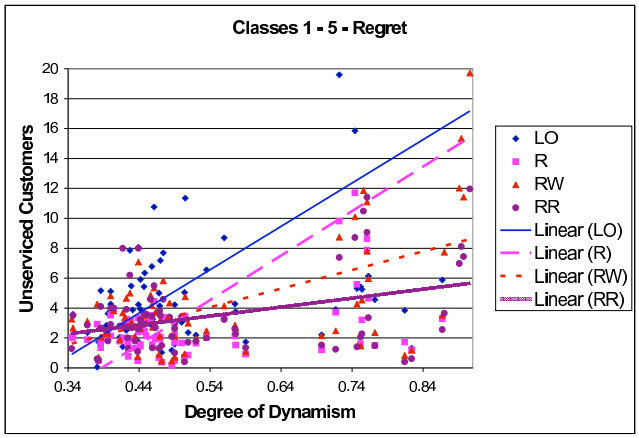

Figure 5: Unserviced Customers for Regret.

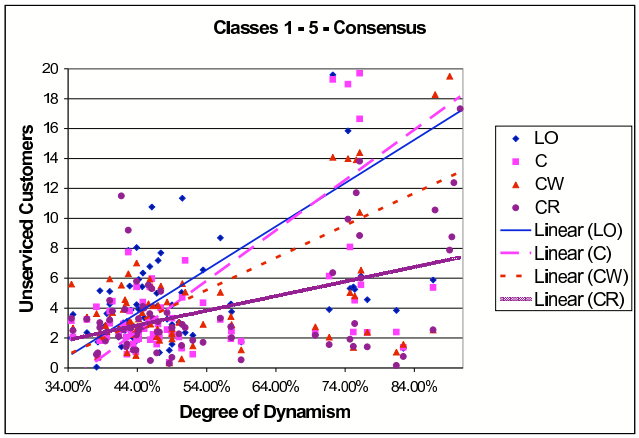

Figure 6: Unserviced Customers for Consensus.

DOD grows. The improvement over $\mathcal{R}$ is significant, indicating the importance of using relocation on this set of instances. Subsequent results will characterize more precisely when the relocation strategy is of paramount importance. Algorithm $\mathcal{RW}$ is also quite effective in general, but it is dominated by $\mathcal{RR}$ as the DOD increases. Similar results can be observed for the consensus algorithms $\mathcal{C}$, $\mathcal{CW}$, and $\mathcal{CR}$ but, in general, they serve fewer customers than their regret counterparts.

Results on Class 4 Figure 7 depicts the results on class 4. We highlight the best and second best result for each problem in boldface and italics respectively. Algorithms $\mathcal{RR}$ and $\mathcal{RW}$ are reasonably close. For small DODs, $\mathcal{R}$ performs the best,

	DOD	LO	$\mathcal{C}$	$\mathcal{CW}$	$\mathcal{CR}$	$\mathcal{R}$	$\mathcal{RW}$	$\mathcal{RR}$
101-1	46.3%	2.08	2.24	4.16	3.30	**1.94**	3.72	*2.68*
101-2	45.8%	6.78	5.42	5.94	3.62	*3.50*	4.18	**3.44**
101-3	50.0%	3.06	*2.06*	3.06	2.28	**1.66**	3.46	3.42
101-4	45.6%	2.90	**3.16**	4.30	5.54	*3.28*	4.86	4.58
101-5	47.4%	7.70	*4.02*	5.48	5.12	**3.38**	5.82	4.58
102-1	59.0%	1.74	1.78	1.22	**0.54**	*0.92*	1.10	1.34
102-2	57.5%	4.28	**1.94**	3.44	2.76	*2.12*	2.86	2.60
102-3	56.0%	8.70	**3.24**	5.06	*3.32*	3.38	4.14	**3.24**
102-4	52.0%	2.18	**0.92**	*1.48*	1.84	1.64	1.96	1.78
102-5	57.6%	3.76	2.46	2.90	*2.02*	**1.52**	2.68	2.28
104-1	76.1%	21.1	19.7	14.4	13.8	**8.64**	*11.1*	11.4
104-2	75.6%	25.6	28.6	13.9	11.7	20.1	*11.9*	**10.5**
104-3	76.1%	20.9	16.6	10.4	8.84	*7.88*	**7.80**	9.06
104-4	72.2%	19.6	19.3	14.1	**6.36**	9.80	8.74	*7.38*
104-5	74.4%	15.9	19.0	14.0	*9.94*	11.7	10.1	**8.72**

Figure 7: Unserviced Customers on Class 4.

	DOD	LO	$\mathcal{C}$	$\mathcal{CW}$	$\mathcal{CR}$	$\mathcal{R}$	$\mathcal{RW}$	$\mathcal{RR}$
101-1	75.4%	5.26	4.52	4.82	*2.96*	3.20	4.54	**2.28**
101-2	74.7%	5.32	8.08	5.04	*1.92*	5.58	4.26	**1.40**
101-3	69.7%	2.22	2.36	2.76	2.18	**1.20**	2.16	*1.52*
101-4	71.7%	3.90	6.14	2.06	*1.56*	3.72	2.48	**1.24**
101-5	76.3%	6.14	5.56	6.54	6.00	*4.66*	5.98	**3.36**
102-1	77.2%	4.56	2.36	2.40	**1.40**	1.50	2.36	*1.48*
102-2	86.7%	5.88	5.38	**2.54**	**2.54**	3.30	3.52	*2.56*
102-3	81.4%	3.84	2.40	1.06	**0.16**	1.72	0.84	*0.40*
102-4	75.2%	5.42	2.40	1.38	**1.44**	1.62	*1.46*	2.06
102-5	82.4%	1.32	1.38	1.58	*0.76*	1.22	1.18	**0.62**
104-1	89.4%	25.3	26.9	23.0	8.76	26.1	*15.3*	**8.12**
104-2	90.6%	25.0	28.9	31.3	17.3	35.0	*19.7*	**12.0**
104-3	89.1%	26.3	27.3	19.5	7.88	24.5	*12.0*	**6.98**
104-4	89.7%	30.2	31.2	22.0	12.4	33.9	*11.4*	**7.44**
104-5	87.0%	22.7	31.8	18.3	10.6	28.3	*7.74*	**3.66**

Figure 8: Unserviced Customers on Class 5.

however, what is noteworthy is the significant gain $\mathcal{RR}$ and $\mathcal{RW}$ show as the DOD increases. These results show the significance of the waiting and relocation strategies on class 4. Observe that LO misses more than 25 customers on instance `rc104-2`, while only about 10 customers are not served by $\mathcal{RR}$. Their consensus counterparts also behave well.

Results on Class 5 Class 5 contains the most difficult instances and the experimental results are depicted in figure 8. On these instances, some algorithms may miss up to 30 customers. This is the case of algorithms LO, $\mathcal{C}$, and $\mathcal{R}$ on `rc104-4`. Once again, algorithm $\mathcal{RR}$ is the most effective algorithm overall and only misses 7 customers on `rc104-4`. The improvements of $\mathcal{RR}$ over other algorithms are quite dramatic on class 5 and grow with the degree of dynamism. They clearly demonstrate the value of a relocation strategy on online vehicle routing with time windows.

It is important to point out that the relocation is completely guided by the stochastic information and does not include any problem-specific knowledge: a vehicle simply moves to the selected customer whether this is an accepted customer or a sampled customer. As a consequence, the relocation strategy is simple to implement, yet it is critical to obtain high-quality solutions on these instances. The regret algorithm $\mathcal{RW}$ with a waiting strategy is also effective and provides significant over the other algorithms, but it is dominated by $\mathcal{RR}$.

8 Conclusion

This paper considered the online vehicle routing problem where customer requests arrive dynamically. It demonstrated the effectiveness of two *novel strategies to further exploit stochastic information: waiting and relocation*. The strategies were shown as natural instantiations of the generic stochastic optimization framework by making available actions that are sub-optimal in an offline setting. The effectiveness of the strategies were demonstrated and validated in the experimental results. As future work, it will be interesting to see the performance of these techniques on classes of VRPs where waiting and relocation may have more impact on cost. Finally, as the decision to relocate or wait solely relies on scenario solutions, not on any prior knowledge of the problem or distribution, the results should naturally extend to a variety of applications in planning and scheduling.

Acknowledgments

This research is partly supported by NSF Award DMI-0600384 and ONR DEPSCOR Award N000140610607.

References

[Benoist *et al.*, 2001] T. Benoist, E. Bourreau, Y. Caseau, and B. Rottembourg. Towards Stochastic Constraint Programming: A Study of On-line Multi-Choice Knapsack with Deadlines. In *CP-01*, 61–76, 2001.

[Bent and Van Hentenryck, 2003] R. Bent and P. Van Hentenryck. Dynamic Vehicle Routing with Stochastic Requests. In *IJCAI'03*, 2003.

[Bent and Van Hentenryck, 2004a] R. Bent and P. Van Hentenryck. Online Stochastic and Robust Optimization. In *ASIAN-04*, 286–300, 2004.

[Bent and Van Hentenryck, 2004b] R. Bent and P. Van Hentenryck. Regrets Only! Online Stochastic Optimization Under Time Constraints. In *AAAI-04*, 501–506, 2004.

[Bent and Van Hentenryck, 2004c] R. Bent and P. Van Hentenryck. Scenario-Based Planning for Partially Dynamic Vehicle Routing with Stochastic Customers. *Operations Research*, 52 (6):977–987, 2004.

[Bent and Van Hentenryck, 2004d] R. Bent and P. Van Hentenryck. The Value of Consensus in Online Stochastic Scheduling. In *ICAPS-04*, 219–226, 2004.

[Bent *et al.*, 2005] R. Bent, I. Katriel, and P. Van Hentenrck. Sub-Optimality Approximation. In *CP-05*, 2005.

[Chang *et al.*, 2000] H. Chang, R. Givan, and E. Chong. Online Scheduling Via Sampling. In *AIPS-00*, 62–71, 2000.

[Gendreau *et al.*, 1999] M. Gendreau, F. Guertin, J. Potvin, and E. Taillard. Parallel Tabu Search for Real-Time Vehicle Routing and Dispatching. *Transportation Science*, 33 (4):381–390, 1999.

[Larsen *et al.*, 2004] A. Larsen, O. Madsen, and M. Solomon. The A-priori Dynamic Traveling Salesman Problem With Time Windows. *Transportation Science*, 38:459–472, 2004.

[Mitrovic-Minic and Laporte, 2004] M. Mitrovic-Minic and G. Laporte. Waiting Strategies for the Dynamic Pickup and Delivery Problem With Time Windows. *Transportation Research Record Part B*, 38:635–655, 2004.

[Mitrovic-Minic *et al.*, 2004] M. Mitrovic-Minic, R. Krishnamurti, and G. Laporte. Double-Horizon Based Heuristics for the Dynamic Pickup and Delivery Problems With Time Windows. *Transportation Research Record Part B*, 38:669–685, 2004.

[Shaw, 1998] P. Shaw. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In *CP-98*, 417–431, 1998.

[Solomon, 1987] M. Solomon. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35 (2):254–265, 1987.

[van Hemert, 2004] J. van Hemert. Dynamic Routing With Fruitful Regions: Models and Evolutionary Computation. *Parallel Problem Solving from Nature VIII*, 2004.