Regrets Only!
Online Stochastic Optimization under Time Constraints

Russell Bent and **Pascal Van Hentenryck**
Brown University, Providence, RI 02912
{rbent,pvh}@cs.brown.edu

Abstract

This paper considers online stochastic optimization problems where time constraints severely limit the number of offline optimizations which can be performed at decision time and/or in between decisions. It proposes a novel approach which combines the salient features of the earlier approaches: the evaluation of every decision on all samples (expectation) and the ability to avoid distributing the samples among decisions (consensus). The key idea underlying the novel algorithm is to approximate the regret of a decision d. The regret algorithm is evaluated on two fundamentally different applications: online packet scheduling in networks and online multiple vehicle routing with time windows. On both applications, it produces significant benefits over prior approaches.

Introduction

Online scheduling and routing problems arise naturally in many application areas and have received increasing attention in recent years. Contrary to offline optimization, the data is not available a priori in online optimization. Rather it is incrementally revealed during algorithm execution. In many online optimization problems, the data is a set of requests (e.g., packets in network scheduling or customers in vehicle routing) which are revealed over time and the algorithm must decide which request to process next.

Online stochastic optimization under time constraints assumes the distribution of future requests, or an approximation thereof, is available for sampling. This is typically the case in many applications, where either historical data and predictive models are available. Moreover, it assumes that the time to make a decision is severely constrained, so that only a few offline optimizations can be performed at decision time and in between decisions. Online problems of this kind arise in many applications, including vehicle routing, taxi dispaching, packet scheduling, and online deliveries.

One of the critical issues faced by online stochastic algorithms is how to use their time wisely, since only a few samples can be optimized within the time constraints. In other words, the algorithm must find an effective approach to optimize the samples and extract information from their solutions in order to make more informed decisions. When time is not a factor, a traditional approach (see, e.g., (Chang, Givan, & Chong 2000)) consists of using an expectation algorithm: at time t, generate a number of samples, solve each sample once per available request r by serving r at t, and select the best request overall. Unfortunately, the expectation approach does not perform well under time constraints, since it must distribute its available optimizations across all requests. This issue was recognized and addressed in (Bent & Van Hentenryck 2004) where a consensus approach was proposed. Its key idea is to solve as many samples as possible and to select the request which is chosen most often in the sample solutions at time t. The consensus approach was shown to outperform the expectation method on online packet scheduling under time constraints. However, as decision time increases, the quality of the consensus approach levels off and is eventually outperformed by the expectation method. It is also possible to hybridize the expectation and consensus approaches but the resulting method loses some of the benefits of consensus under strict time constraints.

This paper reconsiders online stochastic optimization under time constraints and proposes a *regret* approach which truly combines the salient features of the consensus and expectation approaches. Like consensus (and unlike expectation), the regret approach avoids distributing the optimizations across the requests. Like expectation (and unlike consensus), the regret algorithm collects information on all requests for each optimization. Its key insight is to recognize that, in many applications, it is reasonably easy to approximate the regret of a request r, i.e., the cost of replacing the request served at time t by r. As a consequence, once the regret is available, the best solution serving request r at time t for a given sample can be approximated without optimization. The regret approach was evaluated on online packet scheduling and online vehicle routing with time windows. On both applications, the regret algorithm exhibits significant benefits compared to earlier approaches. The key contributions of this paper can be summarized as follows:

1. It proposes a novel approach to online stochastic optimization under time constraints which outperforms prior approaches and combines their salient benefits.
2. It provides a fundamental insight about the relationship between the expectation and consensus approaches, a link missing in (Bent & Van Hentenryck 2004) and explaining why consensus was successful in online vehicle routing.

```
ONLINEOPTIMIZATION(H)
1  R ← ∅;
2  w ← 0;
3  for t ∈ H
4  do R ← AVAILABLEREQUESTS(R, t) ∪ NEWREQUESTS(t);
5     r ← CHOOSEREQUEST(R, t);
6     SERVEREQUEST(r, t);
7     w ← w + w(r);
8     R ← R \ {r};
```

Figure 1: The Generic Online Algorithm

3. It evaluates the regret approach on two radically different applications, i.e., online packet scheduling and vehicle routing, which are two extreme points in the spectrum of online stochastic optimization. On both applications, it provides significant benefits over earlier approaches.

Online Stochastic Optimization

This section presents the online stochastic optimization framework. Although the framework is reasonably simple, it applies directly to a variety of applications (e.g., online packet scheduling) and can be easily adapted to more complex problems. Its main benefit is to crystallize the main focus of this paper: *how to use a limited number of offline optimizations to make more informed decisions.*

The framework assumes a discrete model of time. Its offline problem considers a time horizon $H = [\underline{H}, \overline{H}]$ and a number of requests R. Each request r is associated with a weight $w(r)$ which represents the gain if the request is served. A solution to the offline problem serves a request r at each time $t \in H$ and can be viewed as a function $H \rightarrow R$. Solutions must satisfy the problem-specific constraints which are left unspecified in the framework. The goal is to find a feasible solution σ maximizing $\sum_{t \in H} w(\sigma(t))$.

In the online version, the requests are not available initially and become progressively available at each time step. The distribution, which is seen as a black-box, is also available for sampling. The online algorithms have at their disposal a procedure to solve, or approximate, the offline problem. However, at each time step, they may only use the offline procedure $\mathcal{O}$ times because of the time constraints. More precisely, the approaches presented herein are expressed in terms of the two black-boxes:

1. a function OPTIMALSOLUTION(R, t) which, given a set R of requests and a time t, returns an optimal solution for R over $[t, \overline{H}]$;
2. a function GETSAMPLE(H) returning a set of requests over horizon H by sampling the arrival distribution.

Note that, in practice, it may not be practical to sample the distribution for the entire time horizon. As a consequence, we simply assume that the samples extend Δ time steps in the future, where Δ is an implementation parameter.

Online Oblivious Algorithms

All the approaches described in this paper share the same overall structure which is depicted in Figure 1. The approaches only differ in the way they implement function CHOOSEREQUEST. The online optimization schema simply considers the set of available and new requests at each time step and chooses a request r which is then served and removed from the set of available requests. Function AVAILABLEREQUEST(R, t) returns the set of requests available for service at time t and function SERVEREQUEST(r, t) simply serves r at time t ($\sigma(t) \leftarrow r$). Subsequent sections specify the online algorithms, i.e., their implementations of function CHOOSEREQUEST. To illustrate the framework, here is how two oblivious algorithms, not using stochastic information, can be specified as instantiations of the generic algorithm. These algorithms serve as a basis for comparison.

Greedy (G): This heuristic serves the available request with highest weight. It can be specified formally as

```
CHOOSEREQUEST-G(R, t)
1  A ← READY(R, t);
2  return argmax(r ∈ A) w(r);
```

Local Optimal (LO): This algorithm chooses the next request to serve at time t by finding the optimal solution for the available requests at t. It can be specified as

```
CHOOSEREQUEST-LO(R, t)
1  σ ← OPTIMALSOLUTION(R, t);
2  return σ(t);
```

Online Stochastic Algorithms

We now review a number of existing stochastic algorithms.

Expectation (E): This is the primary method proposed by (Chang, Givan, & Chong 2000) for online packet scheduling. Informally speaking, the method generates future requests by sampling and evaluates each available requests against that sample. A simple implementation can be specified as follows:

```
CHOOSEREQUEST-E(R, t)
1  A ← READY(R, t);
2  for r ∈ A
3     do f(r) ← 0;
4  for i ← 1 ... O/|A|
5     do S ← R ∪ GETSAMPLE([t + 1, t + Δ]);
6        for r ∈ A
7           do σ ← OPTIMALSOLUTION(S \ {r}, t + 1);
8              f(r) ← f(r) + w(r) + w(σ);
9  return argmax(r ∈ A) f(r);
```

Line 1 computes the requests which can be served at time t. Lines 2-3 initialize the evaluation function $f(j)$ for each request r. The algorithm then generates a number of samples for future requests (line 4). For each such sample, it computes the set R of all available and sampled request at time t (line 5). The algorithm then considers each available request r successively (line 6), it implicitly schedules r at time t, and applies the optimal offline algorithm (line 7) using $S \setminus \{r\}$ and the time horizon. The evaluation of request r is updated in line 8 by incrementing it with its weight and the score of the corresponding optimal offline solution σ. All scenarios are evaluated for all available request and the algorithm then returns the request $r \in A$ with the highest evaluation.

Observe Line 4 of Algorithm E which distributes the available offline optimizations across all available requests.

When $\mathcal{O}$ is small (due to the time constraints), each available request is only evaluated with respect to a small number of samples and hence the expectation method does not yield much information. This is precisely why online vehicle routing algorithms (Bent & Van Hentenryck 2001) do not use this method, since the number of requests is very large (about 50 to 100), time between decisions is limited, and the optimization is computationally demanding.

Note also that practical implementations of Algorithm E must be careful to avoid distributing the samples across similar requests and/or to avoid evaluating requests which are formally dominated by others. These problem-specific considerations can be encapsulated in function READY.

Consensus (C): We now turn to a consensus algorithm which uses stochastic information in a fundamental different way. Algorithm C was introduced in (Bent & Van Hentenryck 2004) as an abstraction of the sampling method used in online vehicle routing (Bent & Van Hentenryck 2001). Instead of evaluating each possible request at time t with respect to each sample, algorithm C executes the offline algorithm on the available and sampled requests and to count the number of times a request is scheduled at time t in each resulting solution. Then the request with the highest count is selected. Algorithm C can be specified as follows:

```
CHOOSEREQUEST-C(R, t)
1  for r ∈ R
2     do f(r) ← 0;
3  for i ← 1 ... O
4     do S ← R ∪ GETSAMPLE([t + 1, t + Δ]);
5        σ ← OPTIMALSOLUTION(S, t);
6        f(σ(t)) ← f(σ(t)) + 1;
7  return argmax(r ∈ R) f(r);
```

Observe line 5 which calls the offline algorithm with all available and sampled requests and a time horizon starting at t and line 6 which increments the number of times request $\sigma(t)$ is scheduled first. Line 7 simply returns the request with the largest count. Algorithm C has several appealing features. First, it does not partition the available samples between the requests, which is a significant advantage when the number of samples is small and/or when the number of requests is large. Second, it avoids the conceptual complexity of identifying symmetric or dominated requests.

Consensus+Expectation (C+E$_k$): Algorithms E and C both have advantages and limitations. Algorithm C+E$_k$ attemps to combine their strengths while minimizing their drawbacks. Letting $\mathcal{O} = S_c + kS_e$, its key idea is to run algorithm C first to identify k promising requests (using S_c optimizations) and to run algorithm E to discriminate between them in a precise fashion (using kS_e iterations).

Limitations of Consensus Figure 2 evaluates the various approaches on online packet scheduling under different time constraints (specified by the number of available offline optimizations). The figure exhibits a number of interesting points. First, it clearly shows that stochastic information pays off, since consensus is always significantly better than algorithms G and LO. Second, it shows that consensus significantly outperforms algorithm E when few offline optimizations are available. Third, it shows that consensus levels off and is dominated by algorithm E when the number of available optimizations increases. Finally, the hybrid algorithm exploits some of the benefits of C and E, but it is dominated by C when there are few offline optimizations available. Note also that the bottom line is the offline solution, i.e., a lower bound.

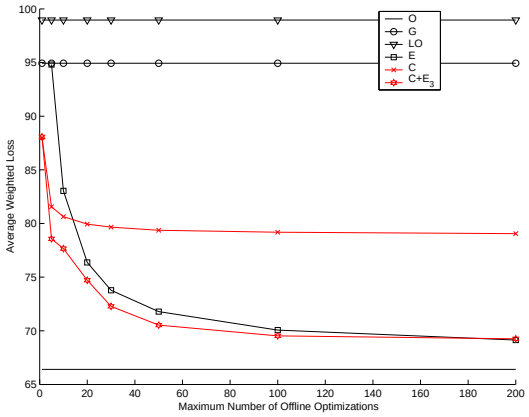

Figure 2: Existing Algorithms on Online Packet Scheduling

Consensus appears to have two main limitations. First, it is purely qualitative as it ignores the score of the optimal solutions. Second, it is extremely *elitist* in that only the optimal choice receives a credit for a given sample. The first limitation can be addressed by crediting the value of the optimal solution instead of incrementing a counter, i.e., replacing line 6 by $f(\sigma(t)) \leftarrow f(\sigma(t)) + w(\sigma)$; This "quantitative" consensus QC does not bring many benefits on the online packet scheduling and vehicle routing problems, where this modification seem to favor the best requests even more.

The real limitation of consensus is its *elitism*. Only the best request is given some credit for a given sample, while other requests are simply ignored. It ignores the fact that several requests may be essentially similar with respect to a given sample. Moreover, it does not recognize that a request may never be the best for any sample, but may still be extremely robust overall. *The fundamental issue is thus to determine if it is possible to gather that kind of information from the sample solutions without solving additional optimization problems.*

The Regret Online Stochastic Algorithm

The key insight to resolve this issue lies in the recognition that, in many applications, *it is possible to quickly estimate the regret of a request r at time t*, i.e., the difference in objective value between the optimal solution σ and the best solution serving request r at time t.

Definition 1 (Regret) *Let R be the set of requests at time t, $r \in$ READY(R, t), and $\sigma =$ OPTIMALSOLUTION(R, t). The regret of r wrt R and t, denoted by* REGRET(R, t, r)*, is defined as*

$$w(\sigma) - (w(r) + w(\text{OPTIMALSOLUTION}(R, t + 1))).$$

This paper assumes the availability of an application-specific regret function REGRETUB which provides an up-

per bound to the regret, i.e.,

$$\text{REGRETUB}(\sigma, R, t, r) \geq \text{REGRET}(R, t, r).$$

for $\sigma = \text{OPTIMALSOLUTION}(R, t)$. Of course, the computational complexity of REGRETUB is assumed to be negligible compared to the cost of the offline optimization.

Such regret approximations typically exist in practical applications. In an online facility location problem, the regret of opening a facility f can be estimated by evaluating the cost of closing the selected facility $\sigma(t)$ and opening f. In vehicle routing, the regret of serving a customer c next can evaluated by swapping c with the first customer on the vehicle serving c. In packet scheduling, the regret of serving a packet p can be estimated by swapping and/or serving a constant number of packets. In all cases, the cost of approximating the regrets is small compared to the cost of the offline optimization. The experimental sections discuss the regrets for specific applications in detail. Note that there is an interesting connection to local search, since computing the regret may be viewed as evaluating the cost of a local move for the application at hand. We are now ready to present the regret algorithm R:

```
CHOOSEREQUEST-R(R, t)
1  for r ∈ R
2     do f(r) ← 0;
3  for i ← 1 ... O
4     do S ← R ∪ GETSAMPLE([t + 1, t + Δ]);
5        σ ← OPTIMALSOLUTION(S, t);
6        f(σ(t)) ← f(σ(t)) + w(σ);
7        for r ∈ READY(R, t) \ {σ(t)}
8           do f(r) ← f(r) + (w(σ) − REGRETUB(σ, R, t, r));
9  return argmax(r ∈ R) f(r);
```

Its basic organization follows algorithm QC, the "quantitative" version of consensus. However, instead of assigning some credit only to the request selected at time t for a given sample s, algorithm R (lines 7-8) uses the regret estimation to compute, each available request r, an approximation of the best solution of s serving r at time t, i.e.,

$$w(\sigma) - \text{REGRETUB}(\sigma, R, t, r).$$

As a consequence, every available request is given an evaluation value for every sample at time t for the cost of (essentially) a single offline optimization. Observe that algorithm R performs $\mathcal{O}$ offline optimizations at time t.

Relationships between E, C, R

The regret algorithm combines the salient features of the expectation and consensus algorithms. Like C (and unlike E), it considers $\mathcal{O}$ samples and avoids distributing the offline optimizations among the requests. Like E (and unlike C), each available request is evaluated for every sample. As a consequence, the algorithm combines the fundamental features of both approach and addresses their limitations.

The regret algorithm also establishes a clear relationship between the algorithms. Indeed, algorithm E is simply algorithm R with an exact regret, i.e.,

$$\text{REGRETUB}(\sigma, R, t, r) = \text{REGRET}(R, t, r).$$

Of course, an exact computation of the regret typically requires an offline optimization. Algorithm QC, the "quantitative" version of consensus, is simply algorithm R with the worst-case regret, i.e., $\text{REGRETUB}(\sigma, R, t, r) = w(\sigma)$. As a consequence, algorithms E and QC are two extreme instantiations in the spectrum of regret algorithms. Alternatively, algorithm E may be viewed as an ideal to achieve but time constraints make it impractical. Algorithm R then provides a general way to obtain fast approximations using regrets and QC is its worst-case instantiation.

Readers may wonder, as we did, why the "qualitative" version of consensus behaves so well in online vehicle routing and packet scheduling. The basic reason is that the difference between an optimal choice $\sigma(t)$ at time t and another available choice r is typically very small (we will come back to this issue later in the paper). It is over time that significant differences between the algorithms accumulate.

Packet Scheduling

This section reports experimental results on the online packet scheduling problem was initially proposed by (Chang, Givan, & Chong 2000) to study the benefits of stochastic information on the quality of the schedules.

Problem Description The offline version can be specified as follows. We are given a set *Jobs* of jobs partitioned into a set of classes C. Each job j is chararacterized by its weight $w(j)$, its arrival date $a(j)$, and its class $c(j)$. Jobs in the same class have the same weight (but different arrival times). We are also given a schedule horizon $H = [\underline{H}, \overline{H}]$ during which jobs must be scheduled. Each job j requires a single time unit to process and must be scheduled in its time window $[a(j), a(j) + d]$, where d is the same constant for all jobs (i.e., d represents the time a job remains available to schedule). In addition, no two jobs can be scheduled at the same time and jobs that cannot be served in their time windows are dropped. The goal is to find a schedule of maximal weight, i.e., a schedule which maximizes the sum of the weights of all scheduled jobs. This is equivalent to minimizing weighted packet loss. More formally, assume, for simplicity and without loss of generality, that there is a job scheduled at each time step of the schedule horizon. Under this assumption, a schedule is a function $\sigma : H \rightarrow Jobs$ which assigns a job to each time in the schedule horizon. A schedule σ is feasible if

$$\forall\, t_1, t_2 \in H : t_1 \neq t_2 \rightarrow \sigma(t_1) \neq \sigma(t_2)$$
$$\forall\, t \in H : a(\sigma(t)) \leq t \leq a(\sigma(t)) + d.$$

The weight of a schedule σ, denoted by $w(\sigma)$, is given by $w(\sigma) = \sum_{t \in H} w(\sigma(t))$. The goal is to find a feasible schedule σ maximizing $w(\sigma)$. This offline problem can be solved in quadratic time $O(|J||H|)$.

Experimental Setting The experimental evaluation is based on the problems of (Chang, Givan, & Chong 2000; Bent & Van Hentenryck 2004), where all the details can be found. In these problems, the arrival distributions are specified by independent HMMs, one for each job class. The results are given for the reference 7-class problems and for an online schedule consisting of 200,000 time steps. Because

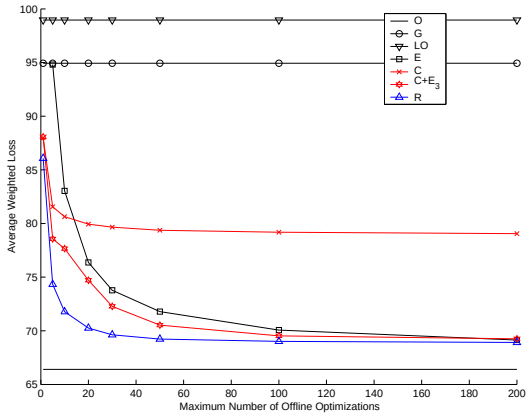

Figure 3: The Regret Algorithm on Packet Scheduling

it is unpractical to sample the future for so many steps, the algorithms use a sampling horizon of 50, which seems to be an excellent compromise between time and quality.

The Regret Function The online packet scheduling fits directly in the online stochastic framework of this paper. To evaluate the algorithms, it suffices to specify how to compute an upper bound to the regret. Our experimental evaluation uses a fast approximation based on a simple case analysis. Consider a job $r \in \text{READY}(R, t)$. If job r is not scheduled (i.e., $r \notin \sigma$), the key idea is to try rescheduling $\sigma(t)$ instead of the job of smaller weight in σ. The regret becomes

$$\min(s \in [t, a(\sigma(t)) + d])\ w(\sigma(s)) - w(r)$$

since the replaced job is removed from σ and r is added to the schedule. In the worst case, the replaced job is $\sigma(t)$ and the regret is $w(\sigma(t)) - w(r)$. If job r is scheduled at time t_r, the key idea is to select the best possible unscheduled job which may be scheduled at t_r. The regret now becomes

$$w(\sigma(t)) - \max(r \in U_r)\ w(u)$$

where $U_r = \{j \in \text{READY}(R, t_r) \mid j \notin \sigma\}$, since job $\sigma(t)$ is lost in the schedule and job j is in the schedule already. Note that, in the best case, job $\sigma(t)$ is inserted at time t_r, in which case the regret is 0. This upper bound is $O(\max(d, |C|))$, which is sublinear in $|J|$ and $|H|$ and essentially negligible for this application.

The Results Figures 3 compares the various algorithms on the 7-class problems. The benefits of the regret algorithm are clearly apparent. Algorithm R indeed dominates all the other algorithms, including consensus (C) when there are very few offline optimizations (strong time constraints) and expectation (E) even when there are a reasonably large number of them, (weak time constraints).

Vehicle Routing

We now evaluate the regret algorithm on a significantly more complicated problem: online multiple vehicle routing with time windows. This problem was studied initially in (Bent & Van Hentenryck 2001) to show the value of stochastic information in vehicle routing.

Problem Formulation The vehicle routing problems are specified formally in (Bent & Van Hentenryck 2001) where all the details can be found. Each problem contains a depot, a number of customer regions and a number of customer service requests from the regions. Each request has a demand, a service time, and a time window specified by an interval $[e, l]$, which represents the earliest and latest possible arrival times respectively. There are a number of identical vehicles available for use, each with capacity Q. A vehicle route starts at the depot, serves some customers at most once, and returns to the depot. The demand of a route is the summation of the demand of its customers. A routing plan is a set of routes servicing each customer exactly once. A solution to the offline VRPTW is a routing plan that satisfies the capacity constraints on the vehicle and the time window constraints of the requests. The objective is to find a solution maximizing the number of served customers. In the dynamic VRPTW, customer requests are not known in advance and become available during the course of the day. In general, a number of requests are available initially.

The online VRPTW is much more complicated than the online packet scheduling and does not fit directly in the stochastic framework presented earlier. First, decisions are triggered by two types of events: new customer requests and the arrival of a vehicle at a customer site. Second, a decision consists of choosing which customers to serve next on a given vehicle. Third, the online algorithm must decide whether to accept or reject customer requests immediately and must service the accepted requests. Finally, the VRPTW is a hard NP-complete problem whose instances are extremely difficult to solve optimally. Only 2 to 10 offline optimizations can be solved in between two events and the number of events is large (e.g., 50 different requests). Hence, the expectation method is not practical at all.

Experimental Setting The experimental results are based on the class-4 problems from (Bent & Van Hentenryck 2001), where all details can be found. They are derived from the Solomon benchmarks which are very challenging and involve 100 customers. The 15 instances exhibit various degrees of dynamism (i.e., the ratio between known and dynamic customers), different distributions of early and late requests, as well as time windows of very different sizes. Hence they cover a wide spectrum of possibilities and structures. The number of vehicles available for the dynamic algorithms was determined by solving the offline problems and adding two vehicles.

The Stochastic Algorithm The experimental evaluation is based on the MSA algorithm proposed in (Bent & Van Hentenryck 2001). This algorithm continously generates samples and "solves" the offline problems using large neighborhood search (LNS) (Shaw 1998). Plans are kept by the MSA algorithm as long as they are consistent with the decisions. This makes it possible to have a reasonable number of samples on which to base the decisions, since only very few samples can be solved in between events. This is due partly to the complexity of solving these complex vehicle routing problems and partly to the short time between events. In general, only a small number of samples (about 5) are avail-

Prob	DOD	MSA^c	MSA^r
20-20-60-rc101-1	46.3%	3.00	3.48
20-20-60-rc101-2	45.8%	4.32	4.84
20-20-60-rc101-3	50.0%	3.24	3.46
20-20-60-rc101-4	45.6%	5.08	5.38
20-20-60-rc101-5	47.4%	6.08	5.72
20-20-60-rc102-1	59.0%	1.10	1.94
20-20-60-rc102-2	57.5%	3.66	3.70
20-20-60-rc102-3	56.0%	4.12	3.60
20-20-60-rc102-4	52.0%	2.58	3.12
20-20-60-rc102-5	57.6%	2.88	2.90
20-20-60-rc104-1	76.1%	13.38	11.26
20-20-60-rc104-2	75.6%	13.86	12.16
20-20-60-rc104-3	76.1%	10.64	8.98
20-20-60-rc104-4	72.2%	14.32	9.42
20-20-60-rc104-5	74.4%	13.38	10.20

Figure 4: Regret on Online Vehicle Routing

able to make a decision. *The main goal of our experimental evaluation is to evaluate two versions of the MSA algorithm: MSA^c which uses consensus to choose the next customer on a given vehicle and MSA^r which uses regret for the same decision.* The rest of the algorithm is left unchanged.

The Regret Function The regret function is simple and fast. Consider the decision of choosing which customer to serve next on vehicle v and let s be the first customer on the route σ of vehicle v. To evaluate the regret of another customer r on a vehicle v, the key idea is to determine if there is a feasible swap of r and s on v, in which case the regret is zero. Otherwise, if such a swap violates the time window constraints, the regret is 1. *The main benefit of this regret function is to recognize that some choices of customers are essentially equivalent.*

Results Table 4 depicts the results on the 15 instances of the Solomon benchmarks. Each instance is solved 50 times because of the nondeterministic nature of the sampling and LNS algorithms. The second column gives the degree of dynamism and the third and fourth columns report the number of missed customers by MSA^c and MSA^r. The regret algorithm does not bring any benefit on the first two classes of problems with lower degrees of dynamism. *However, it produces some dramatic improvements on the third class of instances, where the degree of dynamism is high (about 70%) inducing stricter time constraints.* On this last class, the regret algorithm reduces the number of missed customers by about 35% on some problems and always produces reductions above 12%. This is a very interesting result, since consensus is particularly effective on these problems. Indeed, at a time t, the difference in the number of missed customers between a good and a bad choice is typically 1 and hence consensus is already a very good approximation of E. However, by recognizing "equivalent" choices, the regret algorithm further improves the approximation and produces significant benefits for the most time-constrained instances.

Related Work

Online algorithms (e.g., (Fiat & Woeginger 1998)) have been addressed for numerous years but research has traditionally focused on techniques oblivious to the future and on competitive ratios (Karlin *et al.* 1988). It is only recently that researchers have begun to study how information about future uncertainty may improve the performance of algorithms. This includes scheduling problems (Chang, Givan, & Chong 2000), vehicle routing problems (Bent & Van Hentenryck 2001), (Cambell & Savelsbergh 2002) and elevator dispatching (Nikovski & Branch 2003) to name a few. Research on these problems has varied widely, but the unifying theme is that probabilistic information about the future significantly increases quality. The expectation method was the primary method used in (Chang, Givan, & Chong 2000), They also pointed out why POMDPs are too general for this class of problems. The consensus approach was motivated by online stochastic vehicle routing (Bent & Van Hentenryck 2001) and applied to online packet scheduling in (Bent & Van Hentenryck 2004).

Conclusion

This paper proposed a novel algorithm for online stochastic optimization under time constraints. Like the consensus approach, the new regret algorithm solves as many samples as possible and avoids distributing them among requests. Like the expectation approach, it extracts information on all available requests for every sample by using an upper bound to the regret. The novel algorithm was evaluated on two very different applications, online packet scheduling and online vehicle routing, and was shown to outperform prior algorithms significantly in most cases. Moreover, the regret algorithm provides a fundamental insight in the relationships between the expectation and consensus approaches. This link, which was missing in earlier work, explains why consensus was so successful in online vehicle routing.

Acknowledgment

This research is partially supported by NSF ITR Award DMI-0121495.

References

Bent, R., and Van Hentenryck, P. 2001. Scenario Based Planning for Partially Dynamic Vehicle Routing Problems with Stochastic Customers. *Operations Research.* (to appear).

Bent, R., and Van Hentenryck, P. 2004. The Value of Consensus in Online Stochastic Scheduling. In *ICAPS 2004.*

Cambell, A., and Savelsbergh, M. 2002. Decision Support for Consumer Direct Grocery Initiatives. *Report TLI-02-09, Georgia Institute of Technology*.

Chang, H.; Givan, R.; and Chong, E. 2000. On-line Scheduling Via Sampling. In *AIPS'2000*, 62–71.

Fiat, A., and Woeginger, G. 1998. *Online Algorithms: The State of the Art*.

Karlin, A.; Manasse, M.; Rudolph, L.; and Sleator, D. 1988. Competitive Snoopy Caching. *Algorithmica* 3:79–119.

Nikovski, D., and Branch, M. 2003. Marginalizing Out Future Passengers in Group Elevator Control. In *UAI'03.*

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