Approximately Optimal Risk-Averse Routing Policies via Adaptive Discretization

Darrell Hoy
Electrical Engineering and Computer Science
Northwestern University
darrell.hoy@u.northwestern.edu

Evdokia Nikolova
Electrical and Computer Engineering
University of Texas at Austin
nikolova@austin.utexas.edu

Abstract

Mitigating risk in decision-making has been a longstanding problem. Due to the mathematical challenge of its nonlinear nature, especially in adaptive decision-making problems, finding optimal policies is typically intractable. With a focus on efficient algorithms, we ask how well we can approximate the optimal policies for the difficult case of general utility models of risk.

Little is known about efficient algorithms beyond the very special cases of linear (risk-neutral) and exponential utilities since general utilities are not separable and preclude the use of traditional dynamic programming techniques. In this paper, we consider general utility functions and investigate efficient computation of approximately optimal routing policies, where the goal is to maximize the expected utility of arriving at a destination around a given deadline. We present an adaptive discretization variant of successive approximation which gives an ϵ-optimal policy in polynomial time. The main insight is to perform discretization at the utility level space, which results in a nonuniform discretization of the domain, and applies for any monotone utility function.

Introduction

A central question in decision making under uncertainty is how to mitigate risk. Under the expected utility framework this means to find the decision (or a sequence of decisions) that maximizes the expected utility of the decision, for some nonlinear utility function that represents the agent's risk-averse preferences.

In adaptive or sequential decision-making, due to the large space of possible decisions, it is especially challenging to compute an optimal policy—that is, an optimal sequence of actions, efficiently. In the prevalent framework of Markov Decision Processes (MDPs), efficient algorithms are known for the special cases of linear utility (that corresponds to a risk-neutral agent) and exponential utility, which was used to define risk-sensitive MDPs (Howard and Matheson 1972). The latter allows for efficient computation since the exponential utility function of a sum of rewards separates into a product of terms, specifying a utility for each action.

Little is known about efficient computation of the optimal policy for utility functions beyond linear and exponential, and we provide examples suggesting that computing the optimal policy in such situations is likely intractable. With the goal of retaining efficient computation, we focus on finding an approximately optimal policy and, in particular, on the question how well can we approximate routing decisions in polynomial time?

Our main result is that the optimal routing policy can be approximated arbitrarily well, in polynomial time, under arbitrary monotone utility functions. At the core of our algorithm, we present a general-purpose technique based on adaptive discretization that works for any monotone utility function. The main insight is to perform discretization at the utility level space, which results in a nonuniform discretization of the domain (cost or travel time). We use this technique to generate ϵ-approximate routing policies in time polynomial in the input size and $\frac{1}{\epsilon}$.

In contrast to much of the literature, our approximation is to the optimal *continuous*-time policy, not the optimal policy to the time-discretized problem. This is essential in nonlinear utility models since standard constant-step discretization of time incurs approximation errors that may become significant when the utility function is a step function (e.g., the probability of arriving on time) or a steep function with a large gradient.

While we discuss agents who have non-linear utilities over arrival times, our model applies equally well when agents have a non-linear utility over the expenditure of a different resource, for instance money or fuel.

Related Work

Risk-sensitive Markov Decision Processes (MDPs) (Howard and Matheson 1972) have seen a recent boom in risk models expanding from the exponential utility function model to other nonlinear and general utilities (Bäuerle and Rieder 2014; Ruszczyński 2010; Wu and Lin 1999; Ermon et al. 2011; 2012; Liu and Koenig 2005; 2006; 2008; Dean et al. 1993; Xu and Mannor 2011; Osogami 2011; Le Tallec 2007). Recognizing the challenge of nonlinear cumulative reward functions, most recent work focuses on characterizing properties of such MDPs or providing heuristics. Little remains known about efficient computation of approximately optimal policies. Ruszczyński (2010) observes that we cannot expect a Markov optimal policy if our attitude to risk depends on the whole past. In a mean-variance risk model, Mannor and Tsitsiklis (2011) show hardness results in settings where standard MDPs for expected reward maximizers are polynomial-time solvable.

Most closely related to our approach, Li and Littman (2005) perform a discretization adaptively by holding off on the discretization of each step until necessary, however they do not prove any polynomial convergence, and leave open the exact discretization involved in each utility step.

Rigorous and efficient approximation of optimal policies has remained open when restricted to routing policies, as well. Fan and Nie (2006) consider the 0-1 utility function (referred to as the stochastic on time arrival—SOTA problem), and show that using successive approximation, convergence to the optimal solution will happen in acyclic graphs. Nie and Fan (2006) consider a discretization via uniform time-steps approach and give an algorithm which converges in pseudo-polynomial time to the optimal policy under the given discretization scheme. Samaranayake, Blandin, and Bayen (2011) noted that in settings where edge lengths are lower-bounded by a constant, the convergence can happen much faster; they also discuss using a Fast Fourier Transform (FFT) and 'zero-delay convolutions' in a subsequent paper (Samaranayake, Blandin, and Bayen 2012) to speed up the convolution at each step. Nie et al. (2012) recently considered an adaptive discretization approach for the 0-1 utility function.

The stochastic Canadian Traveler Problem (CTP) seeks routing policies for minimizing the expected travel time when the edge travel times come from given distributions and are observed upon reaching one endpoint of the edge (Papadimitriou and Yannakakis 1991), as opposed to at the end of traversing an edge in our setting. The CTP is #P-hard and optimal policies have been computed efficiently for directed acyclic graphs and undirected graphs consisting of disjoint paths or trees (Nikolova and Karger 2008), and complexity results have been given for other CTP variants (Fried et al. 2013).

Problem statement and preliminaries

Given a directed graph $G = (V, E)$, an agent wishes to route from source node s to destination node d. The non-negative cost of each edge e is drawn independently from a continuous cumulative distribution function (c.d.f.) F_e, and the agent does not see the edge cost until traversing the edge. In the case of a graph with cycles, traversing an edge multiple times yields different, independently drawn costs for each time the edge is traversed. The agent's utility upon arriving at the destination d at time t is given by the utility function $u(t)$.

The agent is allowed to change her route at any node. As such, instead of looking for the optimal path, we are interested in the optimal routing strategy. We call a *routing policy* $P(v, t)$ a function that gives the desired neighbor to travel to from node v upon arrival at time t.

This is a generalization of the stochastic on time arrival problem (Fan and Nie 2006; Samaranayake, Blandin, and Bayen 2012), in which travelers aim to find a policy that maximizes the chance that they arrive at their destination within a certain travel budget. Our generalization allows the use of arbitrary monotone utility functions when arriving at the destination, rather than the hard deadline of the earlier work that corresponds to a 0-1 utility function.

Notation

When speaking about a deadline, it is more intuitive to refer to edge travel times or delays. When we do, time and delay will be equivalent to the notion of edge cost here, as will be clear from the context.

A routing policy $P(v, t)$ maps a vertex-time pair (v, t) to a neighboring vertex w (namely, a vertex w such that there exists an edge $e = (v, w)$). A routing policy generation problem is a tuple $(G, (s, d), u)$ where $s, d \in V$, and $u(t)$ denotes the expected utility of arriving at d at time t. We assume u to be a monotone non-increasing function, namely arriving early is always weakly better than arriving later.

Let $U_P^*(v, t)$ be the expected utility of following policy P at vertex v and time t; hence $U_P^*(d, t) = u(t)$. We will often use $U_P(v, t)$ to denote an estimate of the expected utility from following P at vertex v, time t.

Denote by P^* the optimal such routing policy. The Bellman equation tells us that this policy satisfies

$$U_{P^*}^*(d, t) = u(t), \tag{1}$$

$$U_{P^*}^*(v, t) = \max_{e=(v,w)} \int_t^{\overline{T}} f_e(z - t) U_{P^*}^*(w, z) dz, \tag{2}$$

where z represents the total travel time from the source to node u and $\overline{T}$ is an upper bound on the travel time (for example, defined as the value of $\overline{T}$ for which the utility function becomes zero: $U(d, \overline{T}) = 0$).

Examples

We consider a few examples of the behavior of optimal policies to develop intuition about their properties and challenges. We first show that the optimal policy can be arbitrarily better than the optimal path, even when the path can be recalculated along the way.

We then show that the optimal policy can take exponential space in the size of the graph, which implies that a polynomial-time exact algorithm for calculating the optimal policy is unlikely. Both examples assume a 0-1 step utility function at a deadline, namely, the expected utility is to maximize the probability of arriving before the deadline.

Example 1. *The optimal policy can be arbitrarily better than the optimal path.*

Consider routing across the network shown in Figure 1, from s to d with a deadline of 6—that is, $u(t) = 1$ if $t \leq 6$ and 0 otherwise. Each top edge has cost 2^i half the time, and is free the rest of the time. If an edge is ever free, then the policy can route the agent directly to d to arrive exactly on time—hence ensuring arrival $7/8$ of the time. Choosing a fixed path ends up with arriving on time only $1/2$ of the time. Taking the number of vertices to be larger and taking the probability of a free edge to be smaller, these arrival probabilities can be made to approach 1 and 0 respectively.

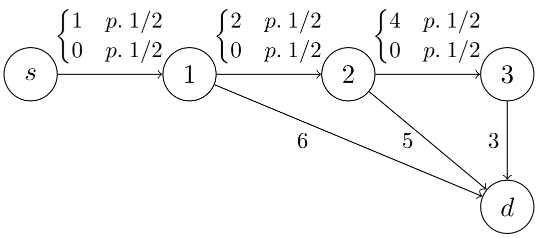
$\begin{cases}1 & p.\ 1/2\\ 0 & p.\ 1/2\end{cases}$
$\begin{cases}2 & p.\ 1/2\\ 0 & p.\ 1/2\end{cases}$
$\begin{cases}4 & p.\ 1/2\\ 0 & p.\ 1/2\end{cases}$
s
1
2
3
6
5
3
d

Figure 1: With a deadline of $D = 6$, the optimal policy arrives on time $7/8$ of the time while any fixed path manages $4/8$. As the number of vertices increase, and the probability of a free edge decreases, this gap can approach 1.

One way of dealing with uncertainty in practice is to allow the agent to recalculate their path at every node. However, by only calculating the best path a agent is not accounting for the possibility of changing her plan. Thus, with only light changes to the example we can prevent the agent from being able to take advantage of her adaptability.

Example 2. *The optimal policy can be arbitrarily better than the strategy of following and recalculating the optimal path at every node in the graph.*

Consider adding an s-d edge to the graph in Figure 1 that takes time 6 with probability $1/2 + \epsilon$ and time 7 otherwise. Then the s-d edge is the best path to take and there are no intermediate nodes to take advantage of the freedom to recalculate the path.

Example 3. *The optimal policy can take space exponential in the size of the graph.*

Consider the graph shown in Figure 2, with a deadline of D. At vertex 2, the optimal policy is simple—take the safe high edge if $t + 16 \leq D$, otherwise take the riskier bottom edge. From vertex 1, the optimal policy requires 4 entries, each alternating between the high and low edges, and the optimal policy from s requires 8 entries:

$$P^*(s,t) = \begin{cases} H & \text{for } t \leq D-28, \\ L & \text{for } D-28 < t \leq D-26, \\ H & \text{for } D-26 < t \leq D-24, \\ \vdots & \\ L & \text{for } D-16 < t \leq D-14. \end{cases}$$

The same process continues when increasing the number of vertices; thus the number of entries in the optimal policy can grow exponentially with the size of the graph.

Discretization

In this section, we discuss approaches to discretization for approximately solving the dynamic programming problem at each state. For each node v, we consider the subproblem of finding its next-hop neighbor at a given time t and an estimate of the utility from traversing the edge to that neighbor, for given estimates of utilities $U_P(w,t)$ for each neighboring node $w \in N_v$.

Solving the given routing problem relies on being able to quickly calculate a convolution of the edge length distribution with the expected utility of arriving at the end of the edge at the given time.

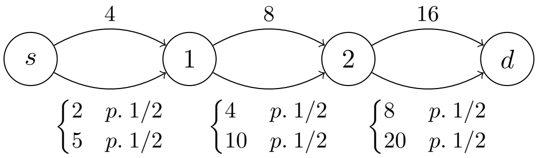
4
8
16
s
1
2
d
$\begin{cases}2 & p.\ 1/2\\ 5 & p.\ 1/2\end{cases}$
$\begin{cases}4 & p.\ 1/2\\ 10 & p.\ 1/2\end{cases}$
$\begin{cases}8 & p.\ 1/2\\ 20 & p.\ 1/2\end{cases}$

Figure 2: The optimal policy for routing from s to d with a deadline of D can take exponential space. From any vertex, there will be one interval in the policy table for each possible path to the destination vertex d.

An adaptive discretization scheme is considered in (Nie et al. 2012), in which the process is split into two operations: first, splitting the region into many uniform intervals, and second, consolidating nearby intervals that do not have enough individual probability mass.

We present a more general and efficient adaptive discretization scheme that enables an improvement in the calculation of precise error bounds.

Discretization with step-functions

Consider the case of an edge $e = (v, w)$, with a constant time computable c.d.f. $F_{e(t)}$, and an expected utility of arriving at w of $U_{w(t)}$ over the interval $[0, \overline{T}]$. Then, the utility of taking edge e from v at time t is $\int_t^{\overline{T}} f_e(z-t)U_w(z)dz$. The best such utility then, given the neighborhood N_v of v, is $\max_{w \in N_v} \int_t^{\overline{T}} f_e(z-t)U_w(z)dz$.

In the general case, this integral is challenging to compute and even more so to solve the problem optimally. However, in the special case when $U_w(z)$ is a step function with k steps, the integral can be easily computed in $O(k)$ time as follows: Write the step-function $U_d(z)$ as

$$U_d(z) = \begin{cases} a_1, & s_0 \leq z < s_1 \\ a_2, & s_1 \leq z < s_2 \\ \cdots & \\ a_k, & s_{k-1} \leq z < s_k, \end{cases}$$

for some constants $a_1, ..., a_k, s_0, ..., s_k$. Then, our convolution becomes:

$$\begin{aligned} \int_t^{\overline{T}} f_e(z-t)U_d(z)dz &= \sum\nolimits_{i=1}^{k} \int_{s_{i-1}}^{s_i} f_e(z-t)U_w(z)dz \\ &= \sum\nolimits_{i=1}^{k} \int_{s_{i-1}}^{s_i} f_e(z-t)a_i dz \\ &= \sum\nolimits_{i=1}^{k} a_i(F_e(s_i - t) - F_e(s_{i-1} - t)). \end{aligned}$$

The last quantity is easy to compute quickly in our problem. With n neighbors, each having a utility function that is a step function with $O(k)$ steps, the running time for computing the best edge to take at time t is $O(kn)$.

Alternative Discretization Techniques

We now give more details about our adaptive discretization and contrast it with other natural discretization approaches into constant time and constant utility intervals.

Constant time intervals The most natural and common approach is to discretize in constant units of time. For example, for each edge, for the desired level of accuracy ϵ, calculate this value for every desired discretization point $i\epsilon$ for i from 0 to $\overline{T}/\epsilon$. This is the discretization approach used by Nie and Fan (2006) and Samaranayake, Blandin, and Bayen (2011). However, we desire an approximation not in terms of time, but in terms of expected utility. In scenarios where small changes in time may result in large changes in expected utility, this approach may well result in a very inefficient use of space/time.

Constant utility intervals Ideally, we should do a discretization in expected utility space: in this way, we could choose the right places for the step function so that the quantity $a_i - a_{i-1}$ is a constant. Given a desired step size of δ, this would need only a maximum of $1/\delta$ steps. However, fully inverting the convolution will rarely be feasible and we need to consider approximation schemes instead.

Adaptive discretization We discretize time in a way that results in constant utility intervals by binary searching for effective discretization points. Specifically, we begin with a time interval $[0, \overline{T}]$, and desire to generate a step function represented by the following pairs $[(s_0, a_0), (s_1, a_1), \dots (s_{\overline{T}}, a_{\overline{T}})]$ such that $\forall i,\ a_i - a_{i-1} < \delta$. To do this, we add discretization points in the middle of intervals that are further away from each other than δ. We split the interval in two by adding a discretization point right in the middle and repeat until all intervals are closer than δ to one another.

1. Calculate $a_0 = U(0)$, $a_{\overline{T}} = U(\overline{T})$ and create a list of the time-utility pairs $[(s_0, a_0), (s_{\overline{T}}, a_{\overline{T}})]$.
2. While there is an adjacent set of pairs in the list (s_{i+1}, a_{i+1}) and (s_i, a_i) such that $a_{i+1} - a_i > \delta$:
   (a) Add $((s_{i+1} + s_i)/2, U((s_{i+1} + s_i)/2))$ to the list.

To minimize the number of points used, we merge adjacent intervals if together they do not account for a utility difference of more than δ. The optimal utility discretization needs a discretization point every time the utility changes by δ and we need at most two points for each increase by δ. Thus, we achieve a 2-approximation to the minimum number of points needed.

Algorithm

We consider the case that G is a directed acyclic graph (DAG). For real-world applications of interest, such as driving between a given source and destination in a road network, the relevant subgraph is close to a DAG. Our approach extends to more general graphs with the caveat that more assumptions must be made to ensure convergence within polynomial time. One such assumption is that all edges share a lower bound on edge delay.

Policy Generation

We take a standard policy iteration approach to solving the generic problem, making use of the above-mentioned adaptive discretization technique.

For each vertex, we keep track of the optimal policy as a set of intervals. Associated with each interval is the neighbor to travel to and the expected utility of traveling to the neighbor at the end of the interval.

Algorithm 1 BESTNEIGHBOR (v, t) — Calculate best neighbor from vertex v at time t

```
for each adjacent edge $e = (v, w)$ do
    $u_e \leftarrow \sum_{(a,b)} U_P(w, b) \cdot (F_e(b - t) - F_e(a - t))$
end for
$e^* \leftarrow \arg\max_e (u_e)$
return $(e^*, u_{e^*})$
```

Algorithm 2 ADAPTIVEDISC $(v, (a, b))$ — Adaptively discretize vertex v for times in interval (a, b).

```
if $U_P(v, a) - U_P(v, b) \leq \delta$ then
    return $[(a, U_P(v, a)), (b, U_P(v, b))]$
else
    $x \leftarrow (a + b)/2$
    $(P(v, x), U_P(v, x)) \leftarrow$ BESTNEIGHBOR$(v, x)$
    return ADAPTIVEDISC$(v, (a, x))$ + ADAPTIVEDISC$(v, (x, b))$
end if
```

Algorithm 3 UPDATEDAGPOLICY (G)

```
Sort vertices in reverse topological order from $d$
for each vertex $v$ do
    $U_P(v, 0) \leftarrow$ BESTNEIGHBOR$(v, 0)$
    $U_P(v, \overline{T}) \leftarrow$ BESTNEIGHBOR$(v, \overline{T})$
    ADAPTIVEDISC$(v, (0, \overline{T}))$
    Remove excess discretization points
end for
return $(P, U_P)$
```

Error Bounds

In this section, we prove tight bounds on the approximation error of our algorithm. Let $L(v)$ be the number of edges in the longest path from v to d.

Lemma 1. *Consider a policy $P(v, t)$ with estimated utility $U_P(v, t)$ from Algorithm 3. Then, for all vertices v, $U_P(v, t)$ is an underestimate of the actual expected utility from following P and is within $\delta L(v)$ of the actual expected utility. Specifically,*

$$U_P^*(v, t) - \delta L(v) \leq U_P(v, t) \leq U_P^*(v, t). \tag{3}$$

Proof. We use induction on the number of edges in the longest path from a vertex to the destination d. For d, the policy is initially the optimal policy, and so $U_P^*(d,t) = U_P(d,t)$.

Now, assume true for all vertices w such that $L(w) < \ell$. Consider a vertex v such that $L(v) = \ell$. As our graph is a DAG, for every $w \in N_v$, $L(w) < \ell$. Thus, by assumption, for any such w,$U_P^*(w,t) - \delta L(w) \leq U_P(w,t) \leq U_P^*(w,t)$.

We now consider the amount of error that is added in by the discretization explicitly at node v. This loss comes directly from the discretization and the possibility that the agent following the policy arrives at a vertex just after a discretization step and waits until the next time discretization step to move. The actual utility from leaving on arrival could differ from the waiting utility by as much as δ. At every realization of delay upon the edge, the agent could gain as much as δ extra utility.

Let $\overline{U_P}$ denote the optimal local choice at v, given the already computed policies of neighbors:

$$\overline{U_P}(v,t) = \max_{w\in N_v} \int_t^{\overline{T}} f_e(z-t)U_P(w,z)dz.$$

From our induction hypothesis, using $\overline{U_P}$ exactly here satisfies:

$$U_P^*(w,t) - \delta(L(w)-1) \leq \overline{U_P}(w,t) \leq U_P^*(w,t),$$

by taking the expectation over the neighbors. We lose one more δ due to discretization, hence

$$\overline{U_P}(v,t) - \delta \leq U_P(v,t) \leq \overline{U_P}(v,t).$$

Combining together gives $U_P^*(w,t) - \delta L(w) \leq U_P(w,t) \leq U_P^*(w,t)$. □

Now, we consider how much utility P can lose relative to the optimal policy. This reduces to the following question: If $P(v,t) = e$, how much better could traveling to a different outgoing edge e' be?

Lemma 2. *Consider a policy $P(v,t)$ and estimated utility $U_P(v,t)$ from Algorithm 3. Then, for every vertex v and time t, U_P is within $\delta L(v)$ of the utility from following the optimal policy $U_{P^*}^*$. In particular,*

$$U_{P^*}^*(v,t) - \delta L(v) \leq U_P(v,t). \qquad (4)$$

Proof. We will prove the lemma via induction on the number of edges in the longest path from a vertex to d.

Let $U_{P,e}(v,t)$ denote the utility from explicitly taking edge $e = (v,w)$ at t and following P otherwise. Hence, $U_{P^*}^*(v,t) = \max_{e=(v,w)} U_{P^*,e}^*(v,t)$.

Now, assume that Equation (4) holds for all vertices w such that $L(w) < \ell$. Consider a vertex v such that $L(v) = \ell$.

By the Bellman Equation, we have:

$$U_{P^*}^*(v,t) = \max_{e=(v,w)} \int_t^{\overline{T}} f_e(z-t)U_{P^*}^*(w,z)dz.$$

By our induction hypothesis, $U_P(w,z) + \delta L(w) \geq U_{P^*}^*(w,z)$ for all neighboring vertices u, hence

$$\begin{aligned}U_{P^*}^*(v,t) &\leq \max_{e=(v,w)} \int_t^{\overline{T}} f_e(z-t)(U_P(w,z) + \delta L(w))dz \\ &\leq \max_{e=(v,w)} \int_t^{\overline{T}} f_e(z-t)U_P(w,z)dz + \delta(L(w)) \\ &\leq \max_{e=(v,w)} \int_t^{\overline{T}} f_e(z-t)U_P(w,z)dz + \delta(L(v)-1).\end{aligned}$$

The second inequality follows by separating the $\delta(L(w))$ term from the general integral and integrating over the density. The last follows as our graph is a DAG.

Via discretization, we lose another δ, but we know:

$$U_P(v,t) \geq \max_{e=(v,w)} \int_t^{\overline{T}} f_e(z-t)U_P(w,z)dz - \delta.$$

Combining gives:

$$\begin{aligned}U_P(v,t) + \delta &\geq \max_{e=(v,w)} \int_t^{\overline{T}} f_e(z-t)U_P(w,z)dz \\ &\geq U_{P^*}^*(v,t) - \delta(L(v)-1) \qquad (5)\end{aligned}$$

Thus, $U_{P^*}^*(v,t) - \delta L(v) \leq U_P(v,t)$. □

Running time

Lemma 3. *For graphs with uniformly continuous edge delay distributions, Algorithm 3 runs in time $O(|E|/\delta^2 + |G|)$.*

Proof. BESTNEIGHBOR runs in $O(1/\delta)$ time for each edge and time assuming F_e runs in constant time and is called $O(1/\delta)$ times, under the assumption F_e is uniformly continuous. Thus, BESTNEIGHBOR accounts for $O(|E|/\delta^2)$ time. Analyzing the graph adds an extra $|G|$, yielding a running time of $O(|E|/\delta^2 + |G|)$. □

Theorem 4. *There is an additive polynomial-time approximation scheme (PTAS) for the optimal policy generation problem over DAGs.*

Proof. By Lemma 2, after running Algorithm 3, the policy has $U_P(s,0) \geq U_{P^*}^*(s,0) - \delta L(s)$. Hence, for a given desired error bound ϵ, running UPDATEDAGPOLICY with $\delta = \epsilon/L(s)$ gives additive error ϵ in time $O(|E|L(s)^2/\epsilon^2 + |G|)$, where $L(s) \leq |V|$ is the number of edges on the longest path from s to the destination. □

Experiments

We tested Algorithm 3 on $m \times n$ grid graphs, routing from the $(0,0)$ vertex to $(m-1, n-1)$ vertex. We compare adaptive discretization to uniform time based discretization, where time discretization is done with $L(s)/\epsilon$ steps to achieve a policy of about the same size as the adaptive policy.

The distribution on each edge is an independent gamma-distribution with lower support of 10, mean drawn uniformly between 25 and 35 and shape parameter drawn uniformly between 1 and 100. A 0-1 utility is used with a deadline of $30 \cdot ((m-1) + (n-1))$, ensuring a reasonable chance of arriving on time. 5 different graphs are tested for each configuration. Errors given are the standard deviation over these 5 graphs. Expected utilities are calculated by following the policy for 50000 realizations of delays on the graph. The implementation is in Python and single-threaded, and execution times are on an Intel Xeon E5-2630 2.3GHz processor.

Convergence Figure 3 details how the expected utility from the adaptive and uniform policies change as the desired

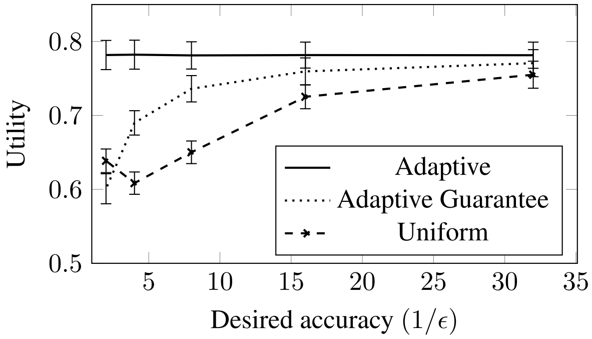

Figure 3: Utility of policies on a 10×10 grid graph with varying desired accuracy.

accuracy increases. The adaptive guarantee refers to the expected utility that is guaranteed by the algorithm. Note that the actual expected utility is much higher, reflecting the pessimism inherent in the analysis for the guarantee. Even the guarantee however converges much faster than the performance of uniform discretization.

Runtime Figure 4 details the effect of increasing desired accuracy on runtimes. The adaptive algorithm is slower for a given $1/\epsilon$, but it appears to be a constant factor. Controlling for achieved accuracy diminishes these. The results reflect that the runtime is quadratic in $1/\epsilon$, as proven in Lemma 3. Figure 5 shows the effect of increasing the size of a graph on policy generation time, supporting the polynomial bound from Lemma 3. The graphs tested are grid graphs with fixed width 10, and fixed error bound $\epsilon = 1/8$.

Extensions

In this section, we list possible improvements that would make our algorithm more practical.

Keeping track of errors. The calculations of error bounds assume that at every node all options are feasible, and that the potential optimal policy may involve traversing the longest path to the destination. This is more general than necessary and keeping track of more information at each point would improve the final error estimates.

Adaptively discretizing vertices. Vertices will contribute differently to the error relative to the optimal policy based on how likely it is they are reached. We can use this to discretize the policy more precisely at important vertices relative to unimportant vertices.

Pruning unusable edges. When keeping track of errors precisely, we can explicitly rule out utilizing some paths, if the optimal upper bound on using the edge is worse than the underestimate of policy utility from a given vertex v. In addition, there may be edges which are first-order stochastically dominated (FOSD) by other paths that can be removed. For instance, staying on a highway will dominate getting off of the highway and taking local roads.

Working from large to small δ. From our above mentioned techniques, it is very feasible to significantly reduce the necessary graph to consider after running the algorithm. One approach then is to start with a large δ, and continue to alternate pruning the graph and halving δ until the desired accuracy is reached. This algorithm still runs in polynomial time, but may be much more practical in a real-world implementation.

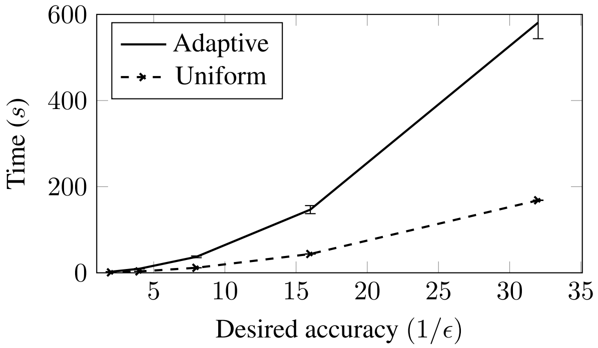

Figure 4: Policy generation time with varying desired accuracy on 10×10 graphs.

Conclusion

We have presented a new adaptive discretization scheme to approximate risk-averse routing policies efficiently and arbitrarily well. Our risk-averse model is very general, allowing for arbitrary monotone decreasing utility functions (reflecting that earlier arrival is weakly preferred to later arrival). The challenges with arbitrary nonlinear functions are that (1) the expected utility corresponding to a given route is difficult to evaluate and (2) such functions are in general nonadditive over edges so that traditional dynamic programming approaches fail. Our policy approximation scheme is based on adaptive nonuniform discretization of time guided by the utility function and our experiments confirm that the algorithm implementation runs in time polynomial in the size of the graph, as we also show theoretically. Our algorithmic technique may extend to more general risk-averse MDP formulations and other stochastic problems where a risk-averse solution is needed.

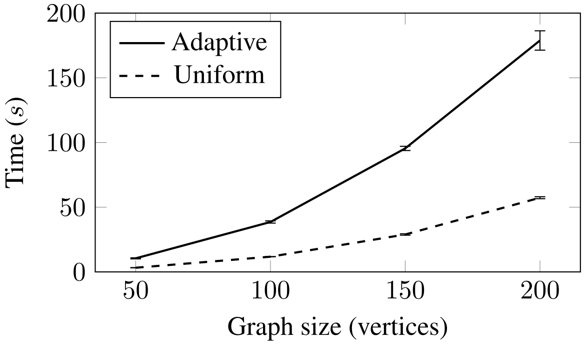

Figure 5: Policy generation time by number of vertices in graph for $\epsilon = 1/8$.

Acknowledgements. This work was supported in part by NSF CCF grants 1350823,1216103,1331863 and by a Google Faculty Research Award.

References

Bäuerle, N., and Rieder, U. 2014. More risk-sensitive markov decision processes. *Mathematics of Operations Research* 39(1):105–120.

Dean, T.; Kaelbling, L. P.; Kirman, J.; and Nicholson, A. 1993. Planning with deadlines in stochastic domains. In *In Proceedings of the Eleventh National Conference on Artificial Intelligence*, 574–579.

Ermon, S.; Conrad, J.; Gomes, C. P.; and Selman, B. 2011. Risk-sensitive policies for sustainable renewable resource allocation. In Walsh, T., ed., *IJCAI*, 1942–1948. IJCAI/AAAI.

Ermon, S.; Gomes, C. P.; Selman, B.; and Vladimirsky, A. 2012. Probabilistic planning with non-linear utility functions and worst-case guarantees. In van der Hoek, W.; Padgham, L.; Conitzer, V.; and Winikoff, M., eds., *AAMAS*, 965–972. IFAAMAS.

Fan, Y., and Nie, Y. 2006. Optimal Routing for Maximizing the Travel Time Reliability. *Networks and Spatial Economics* 6(3-4):333–344.

Fried, D.; Shimony, S. E.; Benbassat, A.; and Wenner, C. 2013. Complexity of Canadian traveler problem variants. *Theor. Comput. Sci.* 487:1–16.

Howard, R. A., and Matheson, J. E. 1972. Risk-sensitive markov decision processes. *Management Science* 18(7):356–369.

Le Tallec, Y. 2007. *Robust, Risk-Sensitive, and Data-driven Control of Markov Decision Processes*. Ph.D. Dissertation, MIT.

Li, L., and Littman, M. L. 2005. Lazy approximation for solving continuous finite-horizon mdps. In Veloso, M. M., and Kambhampati, S., eds., *AAAI*, 1175–1180. AAAI Press / The MIT Press.

Liu, Y., and Koenig, S. 2005. Risk-sensitive planning with one-switch utility functions: Value iteration. In Veloso, M. M., and Kambhampati, S., eds., *AAAI*, 993–999. AAAI Press / The MIT Press.

Liu, Y., and Koenig, S. 2006. Functional value iteration for decision-theoretic planning with general utility functions. In *AAAI*, 1186–1193. AAAI Press.

Liu, Y., and Koenig, S. 2008. An exact algorithm for solving mdps under risk-sensitive planning objectives with one-switch utility functions. In Padgham, L.; Parkes, D. C.; Mller, J. P.; and Parsons, S., eds., *AAMAS (1)*, 453–460. IFAAMAS.

Mannor, S., and Tsitsiklis, J. N. 2011. Mean-variance optimization in markov decision processes. In *ICML*.

Nie, Y., and Fan, Y. 2006. Arriving-on-time problem: discrete algorithm that ensures convergence. *Transportation Research Record* 1964(1):193–200.

Nie, Y. M.; Wu, X.; Dillenburg, J. F.; and Nelson, P. C. 2012. Reliable route guidance: A case study from Chicago. *Transportation Research Part A: Policy and Practice* 46(2):403–419.

Nikolova, E., and Karger, D. 2008. Route planning under uncertainty: The Canadian Traveller problem. In *Proceedings of the Twenty-Third Conference on Artificial Intelligence (AAAI)*.

Osogami, T. 2011. Iterated risk measures for risk-sensitive markov decision processes with discounted cost. In *UAI 2011*.

Papadimitriou, C., and Yannakakis, M. 1991. Shortest paths without a map. *Theoretical Computer Science* 84:127–150.

Ruszczyński, A. 2010. Risk-averse dynamic programming for markov decision processes. *Math. Program.* 125(2):235–261.

Samaranayake, S.; Blandin, S.; and Bayen, a. 2011. A tractable class of algorithms for reliable routing in stochastic networks. *Procedia - Social and Behavioral Sciences* 17:341–363.

Samaranayake, S.; Blandin, S.; and Bayen, A. 2012. Speedup techniques for the stochastic on-time arrival problem. In *ATMOS*, 1–12.

Wu, C., and Lin, Y. 1999. Minimizing risk models in markov decision processes with policies depending on target values. *Journal of Mathematical Analysis and Applications* 231(1):47 – 67.

Xu, H., and Mannor, S. 2011. Probabilistic goal markov decision processes. In *International Joint Conference on Artificial Intelligence*, volume 3, 2046–2052.