# On the Complexity of Shortest Path Problems on Discounted Cost Graphs *

Rajeev Alur, Sampath Kannan, Kevin Tian, and Yifei Yuan

University of Pennsylvania, Philadelphia, PA, US

**Abstract.** Discounted Cost Register Automata (DCRA) associate costs with strings in a regular manner using the operation of discounted sum. The min-cost optimization problem for DCRAs corresponds to computing shortest paths in graphs with more general forms of discounting than the well-studied notion of future discounting. We present solutions to two classes of such shortest path problems: in presence of both past and future discounting, we show the decision problem is NP-complete, but has a polynomial-time approximation scheme; in presence of two future discounting criteria that are composed in a prioritized manner, we show that the problem is solvable in NEXPTIME.

## 1 Introduction

The classical shortest path problem is to determine the minimum-cost path from a given source to a given target vertex in a finite directed graph whose edges are labeled with costs from a numerical domain, where the cost of a path is the sum of the costs of the edges it contains. In a *generalized* version of the shortest-path problem, each edge is labeled with a cost as well as a discount factor: at every step, the cost of each subsequent edge is scaled by the current discount factor (that is, the cost of a path consisting of the edges $e_1 e_2 \cdots e_n$ is given by the expression $\sum_i (c_i \prod_{j<i} d_j)$, where $c_i$ and $d_i$ denote respectively the cost and discount of the edge $e_i$) [8]. This form of *future discounting* is used in the study of Markov decision processes and more recently, in quantitative analysis of systems [6, 3]. The problem of computing the shortest path in presence of such future discounting can be solved in polynomial-time [9, 8].

While the existing work on generalized shortest paths considers only future discounting, there are some natural variations for associating costs with paths in presence of discounting. For example, we can define *past discounting*, where the cost of each preceding edge is scaled by the current discount factor (that is, the cost of a path is $\sum_i (c_i \prod_{j>i} d_j)$), and *global discounting*, where each cost is scaled by the discount factors of all the edges appearing in the path (that is, the cost of a path is $\sum_i (c_i \prod_j d_j)$). The goal of this paper is to initiate a systematic study of shortest path problems for such models with different notions of discounting.

* This research was partially supported by NSF awards CCF 1137084, CCF 1138996, and CCF 0915777.

Our framework for defining a general class of discounted shortest-path problems is based upon the recently proposed notion of *regular functions* that map strings over an input alphabet $\Sigma$ to a numerical domain with a specified set of operations [2]. For our purpose, the numerical domain $\mathbb{D}$ consists of pairs $(c, d)$, where $c$ ranges over incremental costs and $d$ ranges over discount factors, $(0, 1)$ is the identity, and the binary *discounted sum* operation is defined as $(c_1, d_1) \otimes (c_2, d_2) = (c_1 + d_1 * c_2, d_1 * d_2)$. A regular function is computed by a *discounted cost register automaton* (DCRA), a deterministic machine that maps strings over an input alphabet to cost values using a finite-state control and a finite set of registers containing values in $\mathbb{D}$. At each step, the machine reads an input symbol, updates its control state, and updates its registers using expressions involving the discounted sum operation (such as $x := (c, d) \otimes x$; $x := x \otimes (c, d)$; $x := x \otimes y$; $y := (0, 1)$, where $x$ and $y$ are registers). After processing the input, the machine outputs the cost stored in one of the registers. The appeal for the class of functions computed by DCRAs is based on its connection to the well understood theory of regular string-to-string transformations with multiple equivalent characterizations and closure properties [5, 4, 1]. For example, if a function $f$ is computable, then so is $f^R$ defined by $f^R(w) = f(w^R)$, where $w^R$ is the reverse of the string $w$; and if we were to allow a DCRA to make *speculative* decisions based on a regular property of the suffix of the input, instead of just the current input symbol, then such *regular-look-ahead* does not add to expressiveness. Different versions of discounted shortest-path problems turn out to be special cases of the *min-cost* problem for DCRAs, namely, given a function defined by a DCRA, find a string with minimal cost.

To solve the min-cost problem for DCRAs with one register, we focus on the shortest-path problem in a graph with *past-and-future discounting*: the cost of a path $e_1 e_2 \cdots e_n$ is given by $\sum_i (c_i \prod_{j<i} f_j \prod_{j>i} p_j)$, where $c_i$, $f_i$, and $p_i$ denote, respectively, the cost, future discount, and past discount, of the edge $e_i$. This generalizes problems such as future discounting, past discounting, and global discounting. We show that the decision version of the problem is NP-complete: while the strongly-connected-components in the input graph can be analyzed efficiently, the problem is NP-hard even for acyclic graphs. Then, we develop a polynomial-time approximation scheme to solve the problem.

Our next set of results focus on the min-cost problem for DCRAs with two registers. We first consider the *prioritized shortest-path problem*: each edge is labeled two cost-discount pairs $(c, d)$ and $(c', d')$, and the cost of a path $e_1 e_2 \cdots e_n$ corresponds to evaluating the expression $(c_1, d_1) \otimes (c_2, d_2) \cdots (c_n, d_n) \otimes (c'_1, d'_1) \otimes (c'_2, d'_2) \cdots (c'_n, d'_n)$. Thus, this corresponds to having two future-discounting functions, where the cost of a path is obtained by taking the discounted sum of a high-priority future-discounted cost with a low-priority future-discounted cost. While in the classical future discounting problems, for a given cycle, either it is beneficial to skip the cycle entirely, or it is beneficial to repeat it indefinitely, with prioritized discounting, the optimal number of times a cycle should be repeated depends on the cost/discount of the context. While the structure of an optimal path can be complex, we can establish an upper bound on the length

of such a path, leading to decidability. We show that the decision version of the prioritized shortest-path problem is solvable in NEXPTIME. A prioritized shortest path problem corresponds to a DCRA with two registers $x$ and $y$, where the registers are composed only at the end. If we allow repeated composition using the update $\{x := x \otimes y; y := (0,1)\}$ on any edge, we are still able to establish decidability. Decidability of the min-cost problem for the general class of DCRAs remains an open problem.

## 2 Discounted Cost Register Automata

We use $\mathbb{D}$ to denote the domain consisting of pairs $(c,d)$, where $c$ is a cost and $d$ is a discount factor. The typical choice for $\mathbb{D}$ is $\mathbb{D} = \mathbb{Q}^{\geq 0} \times \mathbb{Q}^{\geq 0}$, where $\mathbb{Q}^{\geq 0}$ denotes the set of nonnegative rational numbers. We define the *discounted sum* operator $\otimes$ for elements in $\mathbb{D}$ as follows. For any two elements $(c_1,d_1),(c_2,d_2)$ from $\mathbb{D}$, $(c_1,d_1) \otimes (c_2,d_2) = (c_1 + d_1 * c_2, d_1 * d_2)$. It is easy to check that $(\mathbb{D}, \otimes)$ forms a semigroup with the identity $(0,1)$. For an element $e = (c,d) \in \mathbb{D}$, we denote the first component of $e$ by $e.cost$ and the second component by $e.discount$, i.e. $e.cost = c$ and $e.discount = d$.

A discounted cost register automaton (DCRA) is a deterministic machine that maps strings over an input alphabet to costs using a finite-state control and a finite number of registers that hold values in $\mathbb{D}$. At every step, the machine reads an input symbol, updates its control state, and updates its registers using a parallel assignment. The right-hand-side in each assignment is an expression built from registers and constants in $\mathbb{D}$ using the discounted sum operation. The assignment is required to be *copyless*: no register appears more than once in the right-hand-sides of these assignments. The copyless restriction is common in the theory of transducers, and ensures that the output grows linearly in the size of the input. The output function associates with each accepting state an expression over the registers, evaluating which gives the resulting cost. The syntax and semantics of DCRAs is formalized in the following definitions.

A **discounted cost register automaton (DCRA)** $M$ is a tuple $(\Sigma, Q, q_0, F, X, \delta, \rho, \mu)$, $\Sigma$ is a finite input alphabet, $Q$ is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is a set of accepting states, $X$ is a finite set of registers, $\delta : Q \times \Sigma \to Q$ is the state transition function, $\rho : Q \times \Sigma \times X \to (X \cup \mathbb{D})^*$ is the register-update function with the copyless restriction: for each state $q \in Q$, each register $x \in X$, and every symbol $a \in \Sigma$, $x$ appears at most once in the multiset of strings $\{\rho(q,a,y) | y \in X\}$, $\mu : F \to (X \cup \mathbb{D})^*$ is the output function with the copyless restriction: for each state $q \in F$, each variable $x \in X$, $x$ appears at most once in $\mu(q)$.

A *configuration* of a DCRA $M = (\Sigma, Q, q_0, F, X, \delta, \rho, \mu)$ is a pair $(q,s)$, where $q \in Q$ is a state and $s : X \to \mathbb{D}$ is a valuation function that maps each register to an element in $\mathbb{D}$. The valuation function $s$ naturally extends to a mapping from $(X \cup \mathbb{D})^*$ to $\mathbb{D}$ by evaluating the discounted sum. The run of $M$ on an input string $w = a_1 \dots a_n \in \Sigma^*$ is a sequence of configurations $(q_0,s_0) \dots (q_n,s_n)$, where $q_0$ is the initial state, $s_0$ is the initial valuation that maps each register to the identity

$(0,1)$, and $q_{i+1} = \delta(q_i, a_{i+1})$, and for each $x \in X$, $s_{i+1}(x) = s_i(\rho(q_i, a_{i+1}, x))$, for $0 \leq i < n$. The DCRA $M$ defines a (partial) function $[\![M]\!]$ from $\Sigma^*$ to $\mathbb{Q}^{\geq 0}$: if $q_n \in F$ then $[\![M]\!](w) = s_n(\mu(q_n)).cost$, and $[\![M]\!](w)$ is undefined otherwise.

It is worth noting that a DCRA is basically the same as a *streaming string transducer* (SST) that maps strings over input alphabet to strings over output alphabet [1]: a DCRA interprets string concatenation as discounted sum and hence its output symbols and register values are elements of $\mathbb{D}$ rather than strings over the output alphabet as is the case for SSTs. SSTs have the same expressiveness as MSO-definable string-to-string transformations, two-way deterministic sequential transducers, and Macro string transducers [5, 4, 1]. This class of *regular* string-to-string transformations has appealing closure properties such as closure under reversal and closure under regular look-ahead, and these carry over to DCRA-definable functions. In other words, DCRAs capture a robust class of functions for associating costs with strings by composing elements in $\mathbb{D}$ using the discounted sum operation in a regular manner.

The following decision problems are natural for DCRAs:

1. Given a DCRA $M$, and a string $w$ over the input alphabet $\Sigma$, the *evaluation problem* is to compute the value of $[\![M]\!](w)$.
2. Given two DCRAs $M_1$ and $M_2$ over the same input alphabet $\Sigma$, the *equivalence checking problem* is to decide whether $[\![M_1]\!](w) = [\![M_2]\!](w)$ for every string $w$ over $\Sigma$.
3. Given a DCRA $M$ over input alphabet $\Sigma$, the *min-cost problem* is to decide the value of $inf\{[\![M]\!](w) | w \in \Sigma^*\}$.

It is easy to see that the evaluation problem for DCRAs is solvable in time linear in the size of input string. The equivalence checking problem for DCRAs can be solved in time polynomial in the number of states and exponential in the number of registers using the techniques discussed in [2]. In this paper we focus on the complexity of the min-cost problem for DCRAs with at most 2 registers.

## 3 Past-and-Future Discounting

First observe that to solve the min-cost problem for a DCRA, we can ignore the input symbols, and focus on computing shortest paths in the directed graph corresponding to the state-transition structure of a DCRA. We first look at discounted shortest-path problems corresponding to DCRAs with one register.

### 3.1 Generalized Shortest Path Problem

Given a DCRA $M$ with one register $x$, where each update is of the form $x := x \otimes (c, d)$, the min-cost problem for $M$ coincides with a problem called the *generalized shortest-path problem*, or shortest-paths in presence of only future-discounting, defined below.

Given a labeled directed graph $G = (V, E, L)$, where $L : E \to \mathbb{D}$ is the labeling function, and two vertices $s, t \in V$, the **generalized shortest path**

**problem** is to find the $s$-$t$ path $p$ that minimizes the cost $L(p).cost$, where the labeling function is extended to paths using the discounted sum operator, that is, for a path $p = e_1 \dots e_k$, $L(p) = L(e_1) \otimes L(e_2) \cdots L(e_k)$.

We denote $C(p) = L(p).cost$ and $D(p) = L(p)$ $.discount$. Polynomial-time algorithms for this optimization problem are given in [9].

**Theorem 1.** *Given a labeled directed graph $G = (V, E, L)$, the generalized shortest path problem can be solved in $O(mn^2 \log n)$, where $n = |V|$ and $m = |E|$.*

Here, we are more interested in the structure of the optimal path. A **lasso** is a path $p$ consisting of a simple path $p_0$ followed by a simple cycle $l$ such that $l$ is the only cycle in $p$. Given a lasso $p = p_0 l$ such that $D(l) < 1$, we define the limiting cost of $p$ to be the limit of the costs of paths in the sequence $(p_0 l, p_0 ll, \dots)$, which equals $C(p_0) + D(p_0)\frac{C(l)}{1-D(l)}$. It turns out that the optimal path must be either a simple path from $s$ to $t$, or a lasso. In other words, if it is beneficial to include a cycle to reduce the cost, then it is beneficial to repeat the cycle arbitrarily many times. This property holds even when the graph has finite number of vertices, but infinitely many edges.

**Lemma 1.** *Let $G = (V, E)$ be a graph, with $V$ finite but $E$ possibly infinite. For any vertices $s, t \in V$, and any labeling function $L : E \to \mathbb{D}$ there is an optimal path which is either a simple path from $s$ to $t$, or a lasso from $s$.*

*Proof.* Suppose the optimum $p$ is not a simple path from $s$ to $t$, then $p$ has the structure $p_0 l p_1$, where $l$ is the first cycle in $p$. If $C(l) + D(l)C(p_1) < C(p_1)$, we have that the limiting cost of the lasso $p_0 l$ is no more than $C(p_0 l^n p_1)$ for any natural number $n$. If $C(l) + D(l)C(p_1) \geq C(p_1)$, we have that $C(p_0 p_1) \leq C(p_0 l p_1)$. In either case, we can see by induction that the solution is a simple path or a lasso. □

### 3.2 Shortest Path in Past and Future Discounted Graphs

The generalized shortest path problem can be seen as a *future* discount problem – the discount at an edge applies to the costs of all future edges. Reversing the direction on the edges, we see that it is equivalent to a *past* discount problem, where the discount at an edge applies to all past edges. We consider the following variant: each edge $e$ is now given a cost $c(e)$, a past discount $p(e)$ to be applied to all preceding edges, and a future discount $f(e)$ to be applied to all succeeding edges. In this problem, we assume that discounts are in the range $[0, 1]$. The cost of a path $p = e_1 \dots e_k$ is $C(p) = \sum_{i=1}^{k} c(e_i) \prod_{j<i} f(e_j) \prod_{j>i} p(e_j)$. Given a directed graph $G = (V, E)$, vertices $s, t \in V$, and cost function $c : E \to \mathbb{Q}^{\geq 0}$ and discount functions $p, f : E \to [0, 1]$, the **past and future discount problem** seeks to find an $s$-$t$ path $p$ minimizing $C(p)$.

This variant corresponds to the DCRA with one register, where each update is of the form $x := (0, p') \otimes x \otimes (c', f')$. Note that on each edge, $c(e) = c'p', p(e) = p'$ and $f(e) = f'p'$.

First, we present a sequence of assumptions about the graph structure such that if each assumption does not hold, the optimal cost is easy to compute.

**Lemma 2.** *We may assume that any strongly connected component in $G$ satisfies at least one of the following: all of the past discounts are equal to 1, or all of the future discounts are equal to 1.*

*Proof.* Suppose not. Then let $S$ be a strongly connected component, and $e_1$ an edge with past discount $< 1$ and $e_2$ an edge with future discount $< 1$. Since this is a strongly connected component, there is a cycle containing $e_1$ and $e_2$. Thus, a path from $s$ to $S$, arbitrarily many iterations of a fixed cycle containing $e_1$ and $e_2$, and a path to $t$ has cost arbitrarily close to 0. □

**Lemma 3.** *We may assume that every non-trivial strongly connected component in $G$ has an edge with a past discount $< 1$ or an edge with a future discount $< 1$.*

*Proof.* Suppose not. Then we can construct a graph $G'$ on which this is true. Consider a strongly connected component $S$ in $G$ in which each edge has both discounts equal to 1. Then any optimal path $p$ reaching $S$ at $u$ and leaving $S$ at $v$ uses a shortest path (in the classical discount-less sense) from $u$ to $v$. Thus, we may replace $S$ as follows: for a vertex $v \in S$, we replace it with vertices $v_-$ and $v_+$. For an edge $(u, v)$ with $u \notin S$, replace the endpoint $v$ with $v_-$. Similarly, for $(v, u)$ with $u \notin S$, replace $v$ with $v_+$. Now, for all pairs of vertices $u, v \in S$, add an edge $(u_-, v_+)$ with cost equal to the cost of the (classical) shortest path from $u$ to $v$ in $S$ and both discounts 1. It is clear that replacing all non-trivial strongly connected components in this way yields a graph $G'$ with each component containing at least a discount, and that a solution in $G'$ determines a solution $G$ and vice versa. □

**Lemma 4.** *We may assume there is a topological ordering of the strongly connected components of $G$ such that all strongly connected components with a past discount less than 1 occur before components with a future discount less than 1.*

*Proof.* Suppose not. Then we have a path from $s$ to a component with a future discount less than 1, to a component with a past discount less than 1, to $t$. If we use a cycle repeatedly in the future discount component with discount $< 1$ and similarly use a cycle in the past discount component, we drive the cost to 0. □

Now, we are ready to prove our results.

**Theorem 2.** *The decision version of the past and future discount problem (that is, given $K$, is there path $p$ with $C(p) \leq K$) is* NP*-complete .*

*Proof.* First, the problem is NP. From Lemma 4, we know that in a topological ordering, the strongly connected components with a past discount occur before strongly connected components with a future discount. From Lemma 1, we know that the structure of an optimum subpath in the strongly connected components must be a simple path or a lasso. Thus, a sufficient proof would be the path itself.

The problem is NP-hard, by a reduction from subset product (which is shown to be NP-hard in [7]). □

Now we proceed to establish that the problem can be approximated efficiently.

**Theorem 3.** *The past and future discount problem has a polynomial-time approximation scheme.*

*Proof.* Given a graph $G = (V, E)$, cost and discount functions $c, f, p$, and an approximation factor $\varepsilon$, we will give an approximation scheme that finds a path with cost at most $(1 + \varepsilon)$ times that of the minimum cost path. We will assume that $G$ has the structure imposed by the preceding lemmas.

By Lemma 4, we note that in a topological ordering of the strongly connected components of $G$, all of the components with past discounts precede all of the components with future discounts. Let us denote the strongly connected components $S_1, \ldots, S_r$ (in topological sort order), and let us say that $k$ is such that for all $i \leq k$, $S_i$ is either trivial (consisting of a single vertex) or has a past discount, and for all $i > k$, $S_i$ is either trivial or has a future discount. We use dynamic programming from $s$ through $S_k$, and a similar, reversed process, from $t$ backwards through $S_{k+1}$, and then stitch the results together.

Let $\delta = \frac{\log(1+\varepsilon)}{2n}$ where $|V| = n$, and let $c$ be the highest cost value on any edge. At each vertex, we will store the path with the best future discount that has cost in the interval $[(1+\delta)^i, (1+\delta)^{i+1})$. Then the cost of a simple $s$-$t$ path is at most $n \cdot c$, and so we need to manage at most $O\left(\frac{n \log c}{\log(1+\varepsilon)}\right)$ such buckets. Let $T_v[i]$ denote the best (least) discount for paths in the bucket $[(1+\delta)^i, (1+\delta)^{i+1})$.

We start our dynamic program at $s$, where we have no paths. The program then processes entire strongly connected components at a time. For a component $S_i$, we first consider all of the edges from outside $S_i$ to a vertex of $S_i$. Let $e = (u, v)$ with $u \in S_{i'}$ for some $i' < i$ and $v \in S_i$. Then for each $l$ that $T_u[l]$ is not empty, we look in the bucket for the interval $j$ containing $p(e) \cdot (1+\delta)^{l+1} + c(e) \cdot T_u[l]$, and if $T_u[l] \cdot f(e)$ is less than $T_v[j]$, then we update $T_v[j]$ with this new value.

Having resolved any paths to vertices in $S_i$ that do not use any other vertices in $S_i$, we now address paths to vertices in $S_i$ through other vertices. Note that if a strongly connected component is trivial, this case does not arise. For each pair of vertices $u, v \in S_i$, and each $T_u[l]$ that is not empty, we build a graph which is just $S_i$, except we add a vertex $t'$ and an edge $(v, t')$ with cost $(1+\delta)^{l+1}$ and past discount $T_u[l]$. All other edges will have the same costs and discounts as in $G$. Now, since $S_i$ had only past discounts $< 1$, we can solve a past discount problem to find the best (possibly infinite) path from $u$ to $t'$. Note then that this represents a path from $s$ to $v$, going through $u$, where the subpath from $s$ to $u$ is the one stored in $T_u[l]$. Repeating this for each $l$ and each pair of vertices completely resolves the table $T_v$ for each $v \in S_i$.

We can do this for all $i$ up to $k$. We perform the analogous process for $S_{k+1}$ and subsequent components: we reverse the roles of past and future discounts, and reverse the orientations on the edges. So for $v \in S_i$ for $i \leq k$, $T_v[l]$ will have the path with the best future discount in the $l$-th bucket, while for $v \in S_i$ for $i > k$, $T_v[l]$ will have the path with the best past discount. Now, for all edges $(u, v)$ with $u \in S_i$ and $v \in S_j$ with $i \leq k < j$, and all $l_1, l_2$ for which $T_u[l_1]$ and $T_v[l_2]$ are non-empty, we consider the path adjoining the path in the $l_1$-th bucket

at $u$ to $(u, v)$ to the path in the $l_2$-th bucket at $v$ (note this is an $s$-$t$ path). Now the claim is that the minimum cost path from across all such $u, v, l_1, l_2$ has cost at most $(1 + \varepsilon)$ times the true optimum.

**Lemma 5.** *For each $u \in \bigcup_{i \le k} S_i$, and each $l$, the path from $s$ to $u$ realizing the discount $T_u[l]$ has cost at most $\frac{(1+\delta)^{l+1}}{1+\varepsilon}$.*

*Proof.* Observe that the only time a path's cost becomes overestimated is when it is placed in bucket $j$ (representing paths with approximate costs in the interval $[(1+\delta)^j, (1+\delta)^{j+1})$), where the cost can only be overestimated by a factor $(1+\delta)$. Since a path can be put in a bucket only twice per strongly connected component, we see that its cost is overestimated by at most $(1 + \delta)^{2n} = (1 + \frac{\log(1+\varepsilon)}{2n})^{2n} \le e^{\log(1+\varepsilon)} = 1 + \varepsilon$. □

A similar lemma applies for $u \in \bigcup_{i>k} S_i$ and $t$. Thus we have found the path with the least approximate cost, where for each path, the approximate cost is at most $(1 + \varepsilon)$ times its true cost, and in particular, this path has cost at most $(1 + \varepsilon)$ times the cost of the optimum path. □

## 4 Prioritized Discounting

In this section, we look at the prioritized discounting problem which corresponds to the min-cost problem for DCRAs with two registers where one register always leads the other in both update and output.

### 4.1 Shortest Path in Prioritized Discounted Graph

We first define the Prioritized Discounted Graph and the shortest path problem in a Prioritized Discounted Graph.

A **prioritized discounted graph** is a labeled directed graph $G = (V, E, L_x, L_y)$, where $L_x, L_y : E \to \mathbb{D}$ are labeling functions. Given a prioritized discounted graph $G = (V, E, L_x, L_y)$ and $s, t \in V$, the **prioritized shortest-path problem** seeks to find the $s$-$t$ path $p$ minimizing the cost defined as $C(p) = (L_x(p) \otimes L_y(p))\,.cost$.

The prioritized shortest-path corresponds to the min-cost problem of DCRAs with two registers $x$ and $y$, and each update is of the form $\{x := x \otimes (c_x, d_x); y := y \otimes (c_y, d_y)\}$ and the output function is $x \otimes y$.

We show that the decision version of the shortest path problem for prioritized discounted graph is decidable, assuming that $\mathbb{D} = \mathbb{Q}^{\ge 0} \times [0, 1]$.

**Theorem 4.** *Given a prioritized discount graph $G$ and a nonnegative rational $K$, deciding whether there is an $s$-$t$ path in $G$ such that $C(p) \le K$ is solvable in* NEXPTIME.

*Proof.* For a path $p$, let $(C_x(p), D_x(p)) = L_x(p)$ and $(C_y(p), D_y(p)) = L_y(p)$. We call cycle $l$ a "$x$-neutral cycle" if $L_x(l) = (0, 1)$. If there is a path $p$ such that $C(p) = C_x(p) + D_x(p)C_y(p) \le K$, then the $x$-cost of $p$ (i.e. $C_x(p)$) is at most

$K$. Let us call a path $p$ a "candidate path" if $C_x(p) \leq K$. If there is an edge $e$ in $p$ such that $D_x(e) = 0$, then $p$ has the simple structure as in the future discount problem. Therefore, we may assume $G$ doesn't contain such edges. We first consider the case where $G$ doesn't contain any $x$-neutral cycles. Let $p_0 l$ be the "best" lasso minimizing $\lim_{i\to\infty} C_x(p_0 l^i)$. If $K \geq \lim_{i\to\infty} C_x(p_0 l^i)$, $\lim_{i\to\infty} C(p_0 l^i) = \lim_{i\to\infty} C_x(p_0 l^i) \leq K$. Otherwise, the length of any candidate path cannot exceed $L$ for some natural number $L$. The following lemma shows an exponential upper bound for $L$.

**Lemma 6.** *Let $T$ be the least limiting $x$-cost among all the lassos. If $K < T$, the length of any candidate path is at most $L$, for some $L = 2^{O(nb)}$, where $n$ is the number of vertices in $G$ and $b$ is the maximum number of bits to describe the labeling functions and the bound $K$.*

*Proof.* Fix a vertex $v$. Let $m$ be the maximum number of occurrences of $v$ in a candidate path. Let's consider the candidate path $p$ with the least $x$-cost among all the paths in which $v$ appears $m$ times. Suppose $p = p_0 l_1 l_2 ... l_{m-1} p_m$. Here, $p_0$ is a path from $s$ to $v$ and $l_i$'s are cycles that start and end in $v$, $p_m$ is a path from $v$ to $t$. For brevity, let $c_i = C_x(l_i)$ and $d_i = D_x(l_i)$. By Lemma 1, we further assume that $p_0$, $p_m$ are simple paths and $l_i$'s are simple cycles.

First we claim that $c_i/(1-d_i) \leq c_{i+1}/(1-d_{i+1})$, for $i = 1...m-2$. Note that we define $c_i/(1-d_i) = \infty$, if $d_i = 1$. Suppose not, there is a cycle $l_i$ such that $c_i/(1-d_i) > c_{i+1}/(1-d_{i+1})$. This results in a path $p' = p_0 l_1 ... l_{i+1} l_i ... l_{m-1} p_m$ with the same occurrences of $v$ and $x$-cost less than $C_x(p)$.

Let $Q_i = C_x(l_i l_{i+1} ... l_{m-1} p_m)$. We claim that $Q_1 > ... > Q_m$. Suppose not, we have $Q_i \leq Q_{i+1}$ for some $i$. Then $T \leq \lim_{n\to\infty} C_x(p_0 l_1 ... l_{i-1} l_i^n) \leq K$.

Let $T_1$ and $T_2$ be the least and second least limiting $x$-cost among the simple cycles that start and end in $v$. Let $l_1, l_2, ..l_k$ be the cycles with the limiting $x$-cost $T_1$, so the limiting $x$-cost of $l_i (i > k)$ is at least $T_2$, and $Q_1 < T_1$.Consider any $i > k$. If $d_i = 1, Q_i - Q_{i+1} = c_i + d_i \cdot Q_{i+1} - Q_{i+1} = c_i > 0$. If $d_i < 1$, $Q_i - Q_{i+1} = c_i + d_i \cdot Q_{i+1} - Q_{i+1} = c_i - (1-d_i) Q_{i+1} = (1-d_i)(\frac{c_i}{1-d_i} - Q_{i+1}) \geq (1-d) \cdot (T_2 - T_1)$. Here $d$ is the largest $x$-discount among all the simple cycles other than 1. Therefore, $Q_{k+1} \geq Q_m + \delta(m-k-1)$, where $\delta = min\{c_i, (1-d)(T_2 - T_1) | i = 1...m-1, c_i > 0\}$. Also we know $Q_{k+1} < T_1$, so $m < \frac{T_1}{\delta} + k + 1 = k + 2^{O(nb)}$.

Finally, we show an exponential bound for $k$. Since $C_x(p_0 l_1 ... l_k) \leq K$,we know

$$C_x(p_0 l_1 ... l_k) = C_x(p_0) + D_x(p_0) T_1 (1 - \prod_{i=l}^{k} d_i) \leq K$$

Therefore, $k \leq \log(1 - \frac{K - C_x(p_0)}{T_1 D_x(p_0)}) / \log d \leq \frac{K}{(C_x(p_0) + T_1 D_x(p_0) - K) \log 1/d} = 2^{O(nb)}$. Therefore, $L \leq 2^{O(nb)}$. □

Now consider the case where $G$ contains $x$-neutral cycles. Let $p$ be a path such that $C(p) \leq K$. If $p$ does not contain any $x$-neutral cycles, either the best lasso has the limiting cost at most $K$ or by Lemma 6, the length of $p$ is at

most $L$. Otherwise $p = p_0 l p_1$ for some $x$-neutral simple cycle $l$. Without loss of generality, assume there is no $x$-neutral cycles in $p_0$ and $C_y(lp_1) < C_y(p_1)$. Then $\lim_{i\to\infty} C(p_0 l^i p_1') \leq C(p)$, where $p_1'$ is the path obtained by eliminating all the $x$-neutral cycles in $p_1$. By Lemma 6, the length of $p_0 l p_1'$ is at most $L + 2n$. □

We show a lower bound for the complexity of shortest path problem in a prioritized discounted graph.

**Theorem 5.** *Given a prioritized discounted graph $G$, and a source vertex $s$ and a target vertex $t$, deciding whether there is a path $p$ from $s$ to $t$ such that $C(p) \leq K$ is* NP*-hard.*

*Proof.* We reduce from subset product problem [7]. □

### 4.2 Shortest Path in Prioritized Past and Future Discounted Graph

Consider the most general min-cost problem for DCRAs with one register: each update is of the form $x := (c', d') \otimes x \otimes (c, d)$. In fact, these DCRAs can be modeled by DCRAs with two registers, where each update is $\{x := (c', d') \otimes x; y := y \otimes (c, d)\}$ and the output is $x \otimes y$. The min-cost problem for this set of DCRAs can be formalized as a variant shortest path problem in prioritized discounted graphs: Given a prioritized discounted graph $G = (V, E, L_x, L_y)$, and $s, t \in V$, we wish to find an $s$-$t$ path minimizing $\left(L_x(p^R) \otimes L_y(p)\right).cost$. Here, $p^R$ denote the reverse of $p$ (that is, if $p = (e_1 e_2 ... e_k)$,$p^R = (e_k e_{k-1} ... e_1)$). By applying similar idea as that in theorem 4 and analyzing the cost of the $L_x(p^R)$, it is easy to see that this variant shortest path problem is decidable for $\mathbb{D} = \mathbb{Q}^{\geq 0} \times [0, 1]$.

**Theorem 6.** *Given a prioritized discounted $G$ and a nonnegative rational $K$, deciding whether there is an $s$-$t$ path in $G$ such that $\left(L_x(p^R) \otimes L_y(p)\right).cost \leq K$ is solvable in* NEXPTIME.

### 4.3 Shortest Path in Prioritized Discounted Resetting Graph

A **prioritized discounted resetting graph** is a labeled directed graph $G = (V, E_1, E_2, L_x, L_y)$. $V$ is the set of vertices and there are two types of edges $E_1$ and $E_2$. The labeling functions are only defined on the edges in $E_1$, i.e. $L_x, L_y : E_1 \to \mathbb{D}$ are the labeling functions. $E_2$ is the set of resetting edges. We denote $p \in E_i^*$ if the path $p$ only consists of edges from $E_i$, for $i = 1, 2$. Given a prioritized discounted resetting graph $G = (V, E_1, E_2, L_x, L_y)$ and two vertices $s, t \in V$, the **shortest path problem** for $G$ is to find the $s$-$t$ path $p = (p_1 e_1 p_2 e_2 ... p_k e_k)$ which minimizes the cost defined as $C(p) = (L_x(p_1) \otimes L_y(p_1) \cdots L_x(p_k) \otimes L_y(p_k)).cost$. Here $p_i \in E_1^*, e_i \in E_2$ and we assume that every path from $s$ to $t$ ends with an edge in $E_2$ without loss of generality.

The shortest path problem for prioritized discounted resetting graphs corresponds to the min-cost problem of a subset of DCRAs, where there are two registers $x$, $y$ and each update is of the form $\{x := x \otimes (c_x, d_x); y := y \otimes (c_y, d_y)\}$(we model this update as the edges in $E_1$) or $\{x := x \otimes y, y := (0, 1)\}$ (we model this update as the edges in $E_2$), and the output function is $x \otimes y$.

Now we show that the decision version of the shortest problem in a prioritized discounted resetting graph is decidable, with the assumption that $\mathbb{D} = \mathbb{Q}^{\geq 0} \times [0, 1]$.

**Theorem 7.** *Given a prioritized discounted resetting graph $G = (V, E_1, E_2, L_x, L_y)$, and $s, t \in V$ and a nonnegative rational $K \in \mathbb{Q}^{\geq 0}$, deciding whether there is an $s$-$t$ path $p$ in $G$, such that $C(p) \leq K$ is decidable.*

*Proof.* We reduce the shortest path problem in $G$ to the generalized shortest path problem in a graph $G' = (V', E', L')$ with finitely many vertices but potentially infinitely many edges. First, for each edge $e \in E_2$, we construct a vertex $v_e$. We also create a source vertex $s'$ and target vertex $t'$. Thus, $V' = \{s', t', v_e | e \in E_2\}$. Now we show the construction of the edges in $G'$ and the labeling functions. Consider any edge $e = (u, v) \in E_2$, and any path $p \in E_1^*$ from $s$ to $u$, we construct an edge $e_p = (s', v_e)$ with label $L'(e_p) = L_x(p) \otimes L_y(p)$. Consider each (ordered) pair of edges $(e_1, e_2)$ such that $e_i = (u_i, v_i) \in E_2$ for $i = 1, 2$. For any path $p \in E_1^*$ from $v_1$ to $u_2$ we construct an edge $e_p = (v_{e_1}, v_{e_2})$ with label $L'(e_p) = L_x(e_p) \otimes L_y(p)$. Finally, for any edge $e = (v, t) \in E_2$, we construct an edge $e' = (v_e, t')$ with label $L'(e') = (0, 1)$. It is easy to see the equivalence between the shortest path in $G$ and $G'$.

**Lemma 7.** *For any $s$-$t$ path $p$ in $G$, there exists an $s'$-$t'$ path $p'$ in $G'$ such that $C(p) = C(p')$. For any $s'$-$t'$ path $p'$ in $G'$, there exists an $s$-$t$ path $p$ in $G$, such that $C(p) = C(p')$.*

Let $G_1 = (V, E_1, L_x, L_y)$ and $L_x(p) \otimes L_y(p) = (C_1(p), D_1(p))$ for any path $p \in E_1^*$. We now describe a nondeterministic algorithm to solve the shortest path in the prioritized discounted resetting graph $G$. By Lemma 1, there is a generalized shortest path in $G'$, which is a simple path or a lasso. First, the algorithm guesses the structure of the generalized shortest path $p'$ in $G'$. If $p'$ is a simple path, by Lemma 7, the shortest path in $G$ has the structure $p = (p_1 e_1 \dots p_k e_k)$, where $e_i = (u_i, v_i) \in E_2$ and $p_i \in E_1^*$ for $i = 1 \dots k$ and $k \leq |E_2|$. Second, since $C(p) \leq K$, the cost of the subpath $p_1 e_1$, which is from $s$ to $v_1$ through $u_1$, is at most $K$, i.e. $C_1(p_1) = C(p_1 e_1) = (L_x(p_1) \otimes L_y(p_1)).cost \leq K$. Therefore, the algorithm suffices to solve the prioritized shortest-path problem in $G_1$ with source $s$ and target $u_1$ and bound $K$. If there is an infinite sequence of paths $p'_j$ in $G_1$ with unbounded length and limiting cost at most $K$, then by theorem 4, $\lim_{j\to\infty} D_1(p'_j) = 0$. Therefore, the algorithm outputs "yes", since by taking any finite path $p'_2$ from $v_1$ to $t$, we have $\lim_{j\to\infty} C(p'_j e_1 p'_2) = \lim_{j\to\infty} C_1(p'_j) \leq K$; otherwise, there are finitely many candidate paths. Among all the candidate paths, the algorithm guesses one path $p_1$. Now, the algorithm will solve for the second subpath $p_2$ between $v_1$ and $u_2$. Note that, since $C(p_1 e_1 p_2 e_2) \leq K$, we know $C_1(p_2) = C(p_2 e_2) \leq K_1$, where $K_1 = \frac{K - C_1(p_1)}{D_1(p_1)}$. Therefore, the algorithm solves the prioritized shortest-path problem in $G_1$ with source $v_1$ and target $u_2$ and bound $K_1$. The algorithm solves other subpaths similarly. Finally, if there is a guessed path from $s$ to $t$ with cost at most $K$, output "yes", otherwise output "no". If the guessed path $p'$ in $G'$ is a lasso,

again by Lemma 7, the corresponding shortest path in $G$ has the structure $p = (p_1 e_1 \cdots p_j e_j \cdots p_k e_k p_{k+1} e_j \cdots p_k e_k p_{k+1} e_j \cdots)$, where $e_i = (u_i, v_i) \in E_2$ for $i = 1 \ldots k$, $k \leq |E_2|$, and the cycle $l = (e_j \cdots p_k e_k p_{k+1} e_j)$ repeats after the subpath $p_0 = (p_1 e_1 \cdots p_j)$. The algorithm behaves as the same when guessing the subpath $p_i$ between $v_{i-1}$ and $u_i$. If there is a guessed path $p$ with limiting cost $\lim_{i \to \infty} C(p_0 l^i) \leq K$, output "yes", otherwise output "no". □

## 5 Conclusions

The model of Discounted Cost Register Automata defines a robust class of functions for mapping strings to costs using the discounted sum operator in a regular manner. The min-cost problem for this class offers a unifying framework for generalizing the classical notion of discounting. While decidability of the min-cost problems for the general class of DCRAs remains an open problem, we have solved two interesting special cases. The shortest path problem in presence of past-and-future discounting is NP-complete with a polynomial-time approximation scheme. In prioritized discounting, two cost criteria with future discounting are combined using discounted sum. The structure of the optimal path becomes significantly more complex in this case, and we have established NEXPTIME upper bound for this problem, and also proved decidability for variants.

## References

1. Alur, R., Černý, P.: Streaming Transducers for Algorithmic Verification of Single-pass List-processing Programs. In: Proc. of Principles of Programming Languages. pp. 599–610 (2011)
2. Alur, R., D'Antoni, L., Deshmukh, J.V., Raghothaman, M., Yuan, Y.: Regular functions, cost register automata, and generalized min-cost problems (2012), available at http://www.cis.upenn.edu/ãlur/rca12.pdf
3. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. ACM Trans. Comput. Log. 11(4) (2010)
4. Courcelle, B.: Graph Operations, Graph Transformations and Monadic Second-Order Logic: A survey. Electronic Notes in Theoretical Computer Science 51, 122 – 126 (2002)
5. Engelfriet, J., Hoogeboom, H.J.: MSO definable String Transductions and Two-way Finite-State Transducers. ACM Transactions on Computational Logic 2(2), 216–254 (2001)
6. Filar, J., Vrieze, F.: Competitive Markov Decision Processes. Springer (1997)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1990)
8. Goldberg, A.V., Plotkin, S.A., Tardos, É.: Combinatorial Algorithms for the Generalized Circulation Problem. In: Foundations of Computer Science. pp. 432–443 (1988)
9. Oldham, J.D.: Combinatorial Approximation Algorithms for Generalized Flow Problems. In: Symposium On Discrete Algorithms. pp. 704–714 (1999)