The Robot Routing Problem for Collecting Aggregate Stochastic Rewards*

Rayna Dimitrova[1], Ivan Gavran[2], Rupak Majumdar[3], Vinayak S. Prabhu[4], and Sadegh Esmaeil Zadeh Soudjani[5]

1 **Max Planck Institute for Software Systems, Kaiserslautern, Germany**
`Rayna@mpi-sws.org`
2 **Max Planck Institute for Software Systems, Kaiserslautern, Germany**
`Gavran@mpi-sws.org`
3 **Max Planck Institute for Software Systems, Kaiserslautern, Germany**
`Rupak@mpi-sws.org`
4 **Max Planck Institute for Software Systems, Kaiserslautern, Germany**
`Vinayak@mpi-sws.org`
5 **Max Planck Institute for Software Systems, Kaiserslautern, Germany**
`Sadegh@mpi-sws.org`

Abstract

We propose a new model for formalizing reward collection problems on graphs with dynamically generated rewards which may appear and disappear based on a stochastic model. The *robot routing problem* is modeled as a graph whose nodes are stochastic processes generating potential rewards over discrete time. The rewards are generated according to the stochastic process, but at each step, an existing reward disappears with a given probability. The edges in the graph encode the (unit-distance) paths between the rewards' locations. On visiting a node, the robot collects the accumulated reward at the node at that time, but traveling between the nodes takes time. The optimization question asks to compute an optimal (or ϵ-optimal) path that maximizes the expected collected rewards.

We consider the finite and infinite-horizon robot routing problems. For finite-horizon, the goal is to maximize the total expected reward, while for infinite horizon we consider limit-average objectives. We study the computational and strategy complexity of these problems, establish NP-lower bounds and show that optimal strategies require memory in general. We also provide an algorithm for computing ϵ-optimal infinite paths for arbitrary $\epsilon > 0$.

1998 ACM Subject Classification I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases Path Planning, Graph Games, Quantitative Objectives, Discounting

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2017.13

1 Introduction

Reward collecting problems on metric spaces are at the core of many applications, and studied classically in combinatorial optimization under many well-known monikers: the traveling salesman problem, the knapsack problem, the vehicle routing problem, the orienteering problem, and so on. Typically, these problems model the metric space as a discrete graph whose nodes or edges constitute rewards, either deterministic or stochastic, and ask how to traverse the graph to maximize the collected rewards. In most versions of the problem,

* An extended version of this paper [9] is available at `https://arxiv.org/abs/1704.05303`.

rewards are either fixed or cumulative. In particular, once a reward appears, it stays there until collection. However, in many applications, existing rewards may disappear (e.g., a customer changing her mind) or have more "value" if they are collected fast.

We introduce the *Robot Routing problem*, which combines the spatial aspects of traveling salesman and other reward collecting problems on graphs with stochastic reward generation and with the possibility that uncollected rewards may disappear at each stage. The robot routing problem consists of a finite graph and a reward process for each node of the graph. The reward process models dynamic requests which appear and disappear. At each (discrete) time point, a new reward is generated for the node according to a stochastic process with expectation λ. However, at each point, a previously generated reward disappears with a fixed probability δ. When the node is visited, the entire reward is collected. The optimization problem for robot routing asks, given a graph and a reward process, what is the optimal (or ϵ-optimal) path a robot should traverse in this graph to maximize the expected reward?

As an illustrating example for our setting, consider a vendor planning her path through a city. At each street corner, and at each time step, a new customer arrives with expectation λ, and an existing customer leaves with probability δ. When the vendor arrives at the corner, she serves all the existing requests at once. We ignore other possible real-world features and behaviors e.g., customers leaving queues if the queue length is long. How should the vendor plan her path? Similar problems can be formulated for traffic pooling [25], for robot control [13], for patrolling [15], and many other scenarios.

Despite the usefulness of robot routing in many scenarios involving dynamic appearance and disappearance of rewards, algorithms for its solution have not, to the best of our knowledge, been studied before. In this paper, we study two optimization problems: the *value computation problem*, that asks for the maximal expected reward over a *finite* or *infinite* horizon, and the *path computation problem*, that asks for a path realizing the optimal (or ϵ-optimal) reward. The key observation to solving these problems is that the reward collection can be formulated as discounted sum problems over an extended graph, using the correspondence between stopping processes and discounted sum games.

For finite horizon robot routing we show that the value *decision* problem (deciding if the maximal expected reward is at least a certain amount) is NP-complete when the horizon bound is given in unary, and the value and optimal path can be computed in exponential time using dynamic programming.

For the infinite horizon problem, where the accumulated reward is defined as the long run average, we show that the value decision problem is NP-hard if the probability of a reward disappearing is more than a threshold dependent on the number of nodes. We show that computing the optimal long run average reward can be reduced to a 1-player mean-payoff game on an *infinite graph*. By solving the mean payoff game on a finite truncation of this graph, we can approximate the solution up to an arbitrary precision. This gives us an algorithm that, for any given ϵ, computes an ϵ-optimal path in time exponential in the size of the original graph and logarithmic in $1/\epsilon$. Unlike finite mean-payoff 2-player games, strategies which generate optimal paths for robot routing even in the 1-player setting can require memory. For the *non-discounted* infinite horizon problem (that is, when rewards do not disappear) we show that the optimal path and value problems are solvable in polynomial time.

Related work. The robot routing problem is similar in nature to a number of other problems studied in robot navigation, vehicle routing, patrolling, and queueing network control, but to the best of our knowledge has not been studied so far.

There exists a plethora of versions of the famous traveling salesman problem (TSP) which explore the trade-off between the cost of the constructed path and its reward. Notable examples include the orienteering problem [23], in which the number of locations visited in a limited amount of time is to be maximized, vehicle routing with time-windows [17] and deadlines-TSP [2], which impose restrictions or deadlines on when locations should be visited, as well as discounted-reward-TSP [4] in which soft deadlines are implemented by means of discounting. Unlike in our setting, in all these problems, rewards are static, and there is no generation and accumulation of rewards, which is a key feature of our model.

In the dynamic version of vehicle routing [7] and the dynamic traveling repairman problem [3], tasks are dynamically introduced and the objective is to minimize the expected task waiting time. In contrast, we focus on limit-average objectives, which are a classical way to combine rewards over infinite system runs. Patrolling [6] is another graph optimization problem, motivated by operational security. The typical goal in patrolling is to synthesize a strategy for a defender against a single attack at an undetermined time and location, and is thus incomparable to ours. A single-robot multiple-intruders patrolling setting that is close to ours is described in [15], but there again the objective is to merely detect whether there is a visitor/intruder at a given room. Thus, the patrolling environment in [15] is described by the probability of detecting a visitor for each location. On the contrary, our model can capture *counting patrolling problems*, where the robot is required not only to detect the presence of visitors but to register/count as many of them as possible. Another related problem is the information gathering problem [20]. The key difference between the information gathering setting and ours is that [20] assumes that making an observation earlier has bigger value than if a lot of observations have already been made. This restriction on the reward function is *not* present in our model, since the reward value collected when visiting node v at time t (making observation (v, t), in their terms) only depends on the last time when v was previously visited, and not on the rest of the path (the other observations made, in their terms).

Average-energy games [8, 5] are a class of games on finite graphs in which the limit-average objective is defined by a double summation. The setting discussed in [8, 5] considers static edge weights and no discounting. Moreover, the inner sum in an average-energy objective is over the whole prefix so far, while in our setting the inner sum spans from the last to the current visit of the current node, which is a crucial difference between these two settings.

Finally, there is a rich body of work on multi-robot routing [24, 1, 18, 10, 11] which is closely related to our setting. However, the approaches developed there are limited to static tasks with fixed or linearly decreasing rewards. The main focus in the multi-robot setting is the task allocation and coordination between robots, which is a dimension orthogonal to the aggregate reward collection problem which we study.

Markov decision processes (MDP) [19] seem superficially close to our model. In an MDP, the rewards are determined statically as a function of the state and action. In contrast, the dynamic generation and accumulation of rewards in our model, especially the individual discounting of each generated reward, leads to algorithmic differences: for example, while MDPs admit memoryless strategies for long run average objectives, strategies require memory in our setting and there is no obvious reduction to, e.g., an exponentially larger, MDP.

We employed the reward structure of this article in [13] with the goal of synthesizing controllers for reward collecting Markov processes in continuous space. The work [13] is mainly focused on addressing the continuous dynamics of the underlying Markov process where the authors use abstraction techniques [12] to provide approximately optimal controllers with formal guarantees on the performance while maintaining the probabilistic nature of the process. In contrast, we tackle the challenges of this problem with having a deterministic graph as the underlying dynamical model of the robot and study the computational complexity of the proposed algorithms thoroughly.

Contributions. We define a novel optimization problem for formalizing and solving reward collection in a metric space where stochastic rewards appear as well as disappear over time.

- We consider reward-collection problems in a novel model with *dynamic generation* and *accumulation* of rewards, where each reward *can disappear with a given probability.*
- We study the value decision problem, the value computation problem, and the path computation problem over a finite horizon. We show that the value decision problem is NP-complete when the horizon is given in unary. We describe a dynamic programming approach for computing the optimal value and an optimal path in exponential time.
- We study the value decision problem, the value computation problem, and the path computation problem over an infinite horizon. We show that for sufficiently large values of the disappearing factor δ the value decision problem is NP-hard. We provide an algorithm which for any given $\epsilon > 0$, computes an ϵ-optimal path in time exponential in the size of the original graph and logarithmic in $1/\epsilon$. We demonstrate that strategies (in the 1-player robot routing games) which generate infinite-horizon optimal paths can require memory.

2 Problem Formulation

Preliminaries and notation. A finite directed graph $G = (V, E)$ consists of a finite set of nodes V and a set of edges $E \subseteq V \times V$. A path $\pi = v_0, v_1, \ldots$ in G is a finite or infinite sequence of nodes in G, such that $(v_i, v_{i+1}) \in E$ for each $i \geq 0$. We denote with $|\pi| = N$ the length (number of edges) of a finite path $\pi = v_0, v_1, \ldots, v_N$ and write $\pi[i] = v_i$ and $\pi[0 \ldots N] = v_0, v_1, \ldots v_N$. For an infinite path π, we define $|\pi| = \infty$. We also denote the cardinality of a finite set U by $|U|$. We denote by $\mathbb{N} = \{0, 1, ..\}$ and $\mathbb{Z}_+ = \{1, 2, ..\}$ the sets of non-negative and positive integers respectively. We define $\mathbb{Z}[n, m] = \{n, n+1, \ldots, m\}$ for any $n, m \in \mathbb{N}$, $n \leq m$. We denote with $\mathbb{I}(\cdot)$ the indicator function which takes a Boolean-valued expression as its argument and returns 1 if this expression evaluates to true and 0 otherwise.

Problem setting. Fix a graph $G = (V, E)$. We consider a discrete-time setting where at each time step $t \in \mathbb{N}$, at each node $v \in V$ a reward process generates rewards according to some probability distribution. Once generated, each reward at a node decays according to a decaying function. A *reward-collecting* robot starts out at some node $v_0 \in V$ at time $t = 0$, and traverses one edge in E at each time step. Every time the robot arrives at a node $v \in V$, it collects the reward accumulated at v since the last visit to v. Our goal is to compute the maximum expected reward that the robot can possibly collect, and to construct an optimal path for the robot in the graph, i.e., a path whose expected total reward is maximal.

To formalize reward accumulation, we define a function Last_π which (for path π) maps an index $t \leq |\pi|$ and a node $v \in V$ to the length of the path starting at the previous occurrence of v in π till position t; and to $t + 1$ if v does not occur in π before time t:

$$\mathsf{Last}_\pi(t, v) := \min\left(t + 1,\ \{t - j \in \mathbb{N} \mid j < t, \pi[j] = v\}\right).$$

Reward functions. Let $\xi : \Omega \times V \to \mathbb{R}$ be a set of random variables defined on a sample space Ω and indexed by the set of nodes V. Then $\xi(\cdot, v)$, $v \in V$, is a measurable function from Ω to $\mathbb{R}$ that generates a random reward at node v at any time step. Let π be the path in G traversed by the robot. At time t, the position of the robot is the node $\pi[t]$, and the robot collects the uncollected decayed reward generated at node $\pi[t]$ (since its last visit to $\pi[t]$) up till and including time t. Then, the robot traverses the edge $(\pi[t], \pi[t + 1])$, and at time $t + 1$ it collects the rewards at node $\pi[t + 1]$.

The uncollected reward at time t at a node v given a path π traversed by the robot is defined by the random variable

$$\mathsf{acc}_\pi(t,v) \ := \sum_{j=0}^{\mathsf{Last}_\pi(t,v)-1} \gamma(v)^j \xi(w(t-j),v), \quad w(\cdot)\in\Omega.$$

The value $\gamma(v)$ in the above definition is a *discounting factor* that models the probability that a reward at node v survives for one more round, that is, the probability that a given reward instance at node v disappears at any step is $1-\gamma(v)$.

Note that the previous time a reward was collected at node v was at time $t-\mathsf{Last}_\pi(v,t)$, the time node v was last visited before t. Thus $\mathsf{acc}_\pi(t,v)$ corresponds to the rewards generated at node v at times $t, t-1,\dots, t-\mathsf{Last}_\pi(t,v)+1$, which have decayed by factors of $\gamma(v)^0, \gamma(v)^1, \dots, \gamma(v)^{\mathsf{Last}_\pi(t,v)-1}$, respectively. When traversing a path π, the robot collects the accumulated reward $\mathsf{acc}_\pi(t,\pi[t])$ at time t at node $\pi[t]$.

We define the *expected finite N-horizon sum reward* for a path π as:

$$r_{\mathsf{sum}}^{(N)}(\pi) \ := \ \mathbb{E}\left[\sum_{t=0}^{N} \mathsf{acc}_\pi(t,\pi[t])\right].$$

Let $\lambda : V \to \mathbb{R}_{\geq 0}$ be a function that maps each node $v \in V$ to the *expected value of the reward generated at node v* for each time step, $\lambda(v) = \mathbb{E}[\xi(\cdot,v)]$. We assume that the rewards generated at each node are independent of the agent's move. Thus, the function λ will be sufficient for our study, since we have

$$r_{\mathsf{sum}}^{(N)}(\pi) \ = \sum_{t=0}^{N} \mathsf{Eacc}_\pi(t,\pi[t]), \text{ where } \mathsf{Eacc}_\pi(t,v) := \sum_{j=0}^{\mathsf{Last}_\pi(t,v)-1} \gamma(v)^j\lambda(v). \tag{1}$$

For an infinite path π, the *limit-average* expected reward is defined as

$$r_{\mathsf{av}}(\pi) \ = \ \liminf_{N\to\infty} \frac{r_{\mathsf{sum}}^{(N)}(\pi)}{N+1}. \tag{2}$$

The finite and infinite-horizon *reward values* for a node v are defined as the best rewards over all paths originating in v: $R_{\mathsf{sum}}^{(N)}(v) = \sup_\pi \left\{ r_{\mathsf{sum}}^{(N)}(\pi) \mid \pi[0]=v, |\pi|=N\right\}$ and $R_{\mathsf{av}}(v) = \sup_\pi \{r_{\mathsf{av}}(\pi) \mid \pi[0]=v, |\pi|=\infty\}$, respectively. The choice of limit-average in (2) is due to the unbounded sum reward $r_{\mathsf{sum}}^{(N)}(\pi)$ when N goes to infinity. For a given path π, the sequence $r_{\mathsf{sum}}^{(N)}(\pi)/(N+1)$ in (2) may not converge. Thus we opt for the worst case limiting behavior of the sequence. Alternatively, one may select the best case limiting behavior $\limsup_{N\to\infty}$ in (2) with no substantial change in the results of this paper.

Node-invariant functions λ and γ and definition of cost functions. In the case when the functions λ and γ are constant, we write λ and γ for the respective constants. In this case, the expressions for $r_{\mathsf{sum}}^{(N)}(\pi)$ and $r_{\mathsf{av}}(\pi)$ can be simplified using the identity $1+\gamma+\gamma^2+\dots+\gamma^{q-1} = \frac{1-\gamma^q}{1-\gamma}$ for $\gamma<1$. Then we have

$$\begin{aligned} r_{\mathsf{sum}}^{(N)}(\pi) &= \sum_{t=0}^{N} \sum_{j=0}^{\mathsf{Last}_\pi(\pi[t],t)-1} \gamma^j\lambda = \lambda\cdot\sum_{t=0}^{N}\left(1+\gamma+\ldots+\gamma^{\mathsf{Last}_\pi(t,\pi[t])-1}\right) \\ &= \lambda\cdot\sum_{t=0}^{N}\frac{1-\gamma^{\mathsf{Last}_\pi(t,\pi[t])}}{1-\gamma} = \frac{(N+1)\lambda}{1-\gamma} - \frac{\lambda}{1-\gamma}\sum_{t=0}^{N}\gamma^{\mathsf{Last}_\pi(t,\pi[t])}. \end{aligned} \tag{3}$$

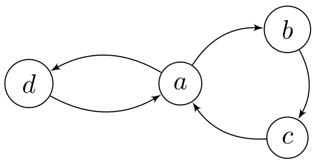


Figure 1 A graph $G_{\mathsf{e}} = (V_{\mathsf{e}}, E_{\mathsf{e}})$ with two simple cycles sharing a single node.

The expression $r_{\mathsf{av}}(\pi)$ can be simplified as:

$$r_{\mathsf{av}}(\pi) = \liminf_{N\to\infty} \frac{1}{N+1} r_{\mathsf{sum}}^{(N)}(\pi) = \frac{\lambda}{1-\gamma} - \frac{\lambda}{1-\gamma} \limsup_{N\to\infty} \frac{1}{N+1} \sum_{t=0}^{N} \gamma^{\mathsf{Last}_\pi(t,\pi[t])}. \tag{4}$$

For the special case $\gamma = 1$ (i.e., when the rewards are not discounted), the expression for the finite-horizon reward is $r_{\mathsf{sum}}^{(N)}(\pi) = \lambda \sum_{t=0}^{N} \mathsf{Last}_\pi(t, \pi[t])$.

We define *cost functions* that map a path π to a real valued finite- or infinite-horizon cost:

$$c_{\mathsf{sum}}^{(N)}(\pi) := \sum_{t=0}^{N} \gamma^{\mathsf{Last}_\pi(\pi[t],t)} \quad \text{and} \quad c_{\mathsf{av}}(\pi) := \limsup_{N\to\infty} \frac{c_{\mathsf{sum}}^{(N)}(\pi)}{N+1}. \tag{5}$$

From Equations (3) and (4), the computation of optimal paths for the reward functions $r_{\mathsf{sum}}^{(N)}$ and r_{av} corresponds to computing paths that minimize the cost functions $c_{\mathsf{sum}}^{(N)}(\pi)$ and $c_{\mathsf{av}}(\pi)$, respectively. Analogously to $R_{\mathsf{sum}}^{(N)}(v)$ and $R_{\mathsf{av}}(v)$, the infimums of the cost functions in (5) over paths are denoted by $C_{\mathsf{sum}}^{(N)}(v)$ and $C_{\mathsf{av}}(v)$ respectively.

▶ **Example 1.** Consider the graph $G_{\mathsf{e}} = (V_{\mathsf{e}}, E_{\mathsf{e}})$ in Figure 1 with $V_{\mathsf{e}} = \{a, b, c, d\}$, which we will use as a running example throughout the paper. The functions λ and γ are constant.

Consider the finite path $\pi_1 = adabcad$. For the occurrences of node a in π_1 we have $\mathsf{Last}_{\pi_1}(0, a) = 1$, $\mathsf{Last}_{\pi_1}(2, a) = 2$, $\mathsf{Last}_{\pi_1}(5, a) = 3$, and similarly for the other nodes in π_1. The reward for π_1 as a function of λ and γ is $r_{\mathsf{sum}}^{6}(\pi_1) = \frac{7\lambda}{1-\gamma} - \frac{\lambda}{1-\gamma}(\gamma+\gamma^2+\gamma^2+\gamma^4+\gamma^5+\gamma^3+\gamma^5)$ for $\gamma < 1$ and $r_{\mathsf{sum}}^{6}(\pi_1) = 22\lambda$ for $\gamma = 1$. For the infinite path $\pi_2 = (abc)^\omega$ we have $\mathsf{Last}_{\pi_2}(0, a) = 1$, $\mathsf{Last}_{\pi_2}(1, b) = 2$, $\mathsf{Last}_{\pi_2}(2, c) = 3$ and the value of Last is 3 in all other cases. Thus we have $r_{\mathsf{av}}(\pi_2) = \frac{\lambda}{1-\gamma} - \frac{\lambda}{1-\gamma}\gamma^3$ for $\gamma < 1$ and $r_{\mathsf{av}}(\pi_2) = 3\lambda$ for $\gamma = 1$. Similarly, for $\pi_3 = (abcad)^\omega$ we have $r_{\mathsf{av}}(\pi_3) = \frac{\lambda}{1-\gamma} - \frac{\lambda}{1-\gamma} \cdot \frac{(\gamma^2+\gamma^3+3\gamma^5)}{5}$ for $\gamma < 1$ and $r_{\mathsf{av}}(\pi_3) = 4\lambda$ for $\gamma = 1$.

Problem statements. We investigate optimization and decision problems for finite and infinite-horizon robot routing. The *value computation problems* ask for the computation of $R_{\mathsf{sum}}^{(N)}(v)$ and $R_{\mathsf{av}}(v)$. The corresponding decision problems asks to check if the respective one of these two quantities is greater than or equal to a given threshold $R \in \mathbb{R}$.

▶ **Definition 2** (Value Decision Problems). Given a finite directed graph $G = (V, E)$, an expected reward function $\lambda : V \to \mathbb{R}_{\geq 0}$, a discounting function $\gamma : V \to (0, 1]$, an initial node $v_0 \in V$ and a threshold value $R \in \mathbb{R}$,

- The *finite horizon value decision problem* is to decide, given N, if $R_{\mathsf{sum}}^{(N)}(v_0) \geq R$.
- The *infinite horizon value decision problem* is to decide if $R_{\mathsf{av}}(v_0) \geq R$.

For a finite directed graph $G = (V, E)$, expected reward and discounting functions $\lambda : V \to \mathbb{R}_{\geq 0}$ and $\gamma : V \to (0, 1]$ and $v_0 \in V$, a finite path π is said to be an *optimal path* for time-horizon N if (a) $\pi[0] = v_0$ and $|\pi| = N$, and (b) for every path π' in G with $\pi'[0] = v_0$

and $|\pi'| = N$ it holds that $r^{(N)}_{\mathsf{sum}}(\pi) \geq r^{(N)}_{\mathsf{sum}}(\pi')$. Similarly, an infinite path π is said to be *optimal for the infinite horizon* if $\pi[0] = v_0$ and for every infinite path π' with $\pi'[0] = v_0$ in G it holds that $r_{\mathsf{av}}(\pi) \geq r_{\mathsf{av}}(\pi')$. We can also define corresponding *threshold paths*: given a value R a path π is said to be threshold R-optimal if $r^{(N)}_{\mathsf{sum}}(\pi) \geq R$ or $r_{\mathsf{av}}(\pi) \geq R$, respectively. An ϵ-*optimal* path is one which is $R^{(N)}_{\mathsf{sum}}(v_0) - \epsilon$ or $R_{\mathsf{av}}(v_0) - \epsilon$ threshold optimal (for finite or infinite horizon respectively).

▶ **Example 3.** Consider again the graph G_{e} shown in Figure 1. Examining the expressions computed in Example 1, we have that $r_{\mathsf{av}}(\pi_2) > r_{\mathsf{av}}(\pi_3)$ for $\gamma = 0.1$ and $r_{\mathsf{av}}(\pi_3) > r_{\mathsf{av}}(\pi_2)$ for $\gamma = 0.9$. Thus, in general, the optimal value depends on γ. Due to the structure of the set of infinite paths in G_{e} we can analytically compute the optimal value $R_{\mathsf{av}}(v)$ for each $v \in V_{\mathsf{e}}$ as a function of γ and a corresponding optimal path (see the full version [9] for the proof):

- if $\gamma \in [0, a_1]$, then $R_{\mathsf{av}}(v) = \frac{\lambda}{1-\gamma} - \frac{\lambda}{1-\gamma}\gamma^3$ and the path $(abc)^\omega$ is optimal;
- if $\gamma \in [a_1, a_2]$, then $R_{\mathsf{av}}(v) = \frac{\lambda}{1-\gamma} - \frac{\lambda}{1-\gamma} \cdot \frac{(\gamma^2+4\gamma^3+2\gamma^5+\gamma^8)}{8}$ and $(abcabcad)^\omega$ is optimal;
- if $\gamma \in [a_2, 1]$, then $R_{\mathsf{av}}(v) = \frac{\lambda}{1-\gamma} - \frac{\lambda}{1-\gamma} \cdot \frac{(\gamma^2+\gamma^3+3\gamma^5)}{5}$ and the path $(abcad)^\omega$ is optimal.

The constants $a_1 \approx 0.2587$ and $a_2 \approx 0.2738$ are respectively the unique real roots of polynomials $\gamma^6+2\gamma^3-4\gamma+1$ and $5\gamma^6-14\gamma^3+12\gamma-3$ in the interval $(0, 1)$. Note that for $\gamma = 1$ we have $R_{\mathsf{av}}(v) = 4\lambda$ which is achieved by $(abcad)^\omega$. The path $(abc)(ad)(abc)^2(ad)^2 \ldots (abc)^n(ad)^n \ldots$, which is not ultimately periodic, also achieves the optimal reward.

Paths as strategies. We often refer to infinite paths as resulting from strategies (of the collecting agent). A *strategy* σ in G is a function that maps finite paths $\pi[0 \ldots m]$ to nodes such that if $\sigma(\pi)$ is defined then $(\pi[m], \sigma(\pi)) \in E$. Given an initial node v_0, the strategy σ generates a unique infinite path π, denoted as $\mathsf{outcome}(v_0, \sigma)$. Thus, every infinite path $\pi = v_0, v_1, \ldots$ defines a unique strategy σ_π where $\sigma_\pi(\pi[0 \ldots i]) = v_{i+1}$, and $\sigma_\pi(\epsilon) = v_0$, and σ_π is undefined otherwise. Clearly, $\mathsf{outcome}(v_0, \sigma_\pi) = \pi$. We say a strategy σ is optimal for a path problem if the path $\mathsf{outcome}(v_0, \sigma)$ is optimal. A strategy σ is *memoryless* if for every two paths $\pi'[0 \ldots m'], \pi''[0 \ldots m'']$ for which $\pi'[m'] = \pi''[m'']$, it holds that $\sigma(\pi') = \sigma(\pi'')$. We say that memoryless strategies suffice for the optimal path problem if there always exists a memoryless strategy σ such that $\mathsf{outcome}(v_0, \sigma)$ is an optimal path.

3 Finite Horizon Rewards: Computing $R^{(N)}_{\mathsf{sum}}(v)$

In this section we consider the finite-horizon problems associated with our model. The following theorem summarizes the main results.

▶ **Theorem 4.** *Given $G = (V,E)$, expected reward and discounting functions, node $v \in V$, and horizon $N \in \mathbb{N}$:*

1. *The finite-horizon value decision problem is NP-complete if N is in unary.*
2. *The value $R^{(N)}_{\mathsf{sum}}(v)$ for $v \in V$ is computable in exponential time even if N is in binary.*

Analogous results hold for the related reward problem where in addition to the initial node v, we are also given a destination node v_f, and the objective is to go from v to v_f in at most N steps while maximizing the reward.

The finite-horizon value problem is NP-hard by reduction from the Hamiltonian path problem (see the full version [9]), even in the case of node-invariant λ and γ. Membership in NP in case N is in unary follows from the fact that we can guess a path of length N and check that the reward for that path is at least the desired threshold value.

To prove the second part of the theorem, we construct a finite *augmented weighted graph*.

For simplicity, we give the proof for node-invariant λ and γ, working with the cost functions c_{sum} and c_{av}. The augmented graph construction in the general case is a trivial generalization by changing the weights of the nodes, and the dynamic programming algorithm used for computing the optimal cost values is easily modified to compute the corresponding reward values instead. For $\gamma < 1$ the objective is to minimize $c^{(N)}_{\mathsf{sum}}(\pi) = \sum_{t=0}^{N} \gamma^{\mathsf{Last}_\pi(t,\pi[t])}$ and for $\gamma = 1$ the objective is to maximize $r^{(N)}_{\mathsf{sum}}(\pi) = \lambda \sum_{t=0}^{N} \mathsf{Last}_\pi(t,\pi[t])$ over paths π.

Augmented weighted graph. Given a finite directed graph $G = (V, E)$ we define the *augmented weighted graph* $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ which "encodes" the values $\mathsf{Last}_\pi(t, v)$ for the paths in G explicitly in the augmented graph node. We can assume w.l.o.g. that $V = \{1, 2, \ldots, |V|\}$.

- The set of nodes $\widetilde{V}$ is $V \times \mathbb{Z}_+^{|V|}$ (the set $\widetilde{V}$ is infinite). A node $(v, b_1, b_2, \ldots, b_{|V|}) \in \widetilde{V}$ represents the fact that the current node is v, and that for each node $u \in V$ the last visit to u (before the current time) was b_u time units before the current time.
- The weight of a node $(v, b_1, b_2, \ldots, b_{|V|}) \in \widetilde{V}$ is γ^{b_v}.
- The set of edges $\widetilde{E}$ consists of edges $(v, b_1, b_2, \ldots, b_{|V|}) \to (v', b'_1, b'_2, \ldots, b'_{|V|})$ such that $(v, v') \in E$; and $b'_v = 1$, and $b'_u = b_u + 1$ for all $u \neq v$.

Let π be a path in G. In the graph $\widetilde{G}$ there exists a unique path $\widetilde{\pi}$ that corresponds to π:

$$\widetilde{\pi} = (\pi[0], 1, 1, \ldots, 1)\,, \ (\pi[1], b_1^1, b_2^1, \ldots, b_{|V|}^1)\,, \ (\pi[2], b_1^2, b_2^2, \ldots, b_{|V|}^2)\,, \ldots \tag{6}$$

starting from the node $(\pi[0], 1, 1, \ldots, 1)$ such that for all t and for all $v \in V$, we have $\mathsf{Last}_\pi(t, v) = b_v^t$. Dually, for each path $\widetilde{\pi}$ in $\widetilde{G}$ starting from $(v_0, 1, 1, \ldots, 1)$, there exists a unique path π in G from the node v_0 such that $\mathsf{Last}_\pi(t, v) = b_v^t$ for all t and v.

For a path $\widetilde{\pi}$ in the form of (6) let

$$\widetilde{c}^{(N)}_{\mathsf{sum}}(\widetilde{\pi}) := \sum_{t=0}^{N} \gamma^{b^t_{v_t}} \quad \text{and} \quad \widetilde{c}_{\mathsf{av}}(\widetilde{\pi}) := \limsup_{N\to\infty} \frac{1}{N+1} \sum_{t=0}^{N} \gamma^{b^t_{v_t}}. \text{ Observe that:}$$

- $\widetilde{c}^{(N)}_{\mathsf{sum}}(\widetilde{\pi})$ is the sum of weights associated with the first $N + 1$ nodes of $\widetilde{\pi}$.
- $\widetilde{c}_{\mathsf{av}}(\widetilde{\pi})$ is the limit-average of the weights associated with the nodes of $\widetilde{\pi}$.

Thus, $\widetilde{c}^{(N)}_{\mathsf{sum}}$ and $\widetilde{c}_{\mathsf{av}}$ define the classical total finite sum (shortest paths) and limit average objectives on weighted (infinite) graphs [26]. Additionally, $\widetilde{c}^{(N)}_{\mathsf{sum}}(\widetilde{\pi}) = c^{(N)}_{\mathsf{sum}}(\pi)$, and $\widetilde{c}_{\mathsf{av}}(\widetilde{\pi}) = c_{\mathsf{av}}(\pi)$ where π is the path in G corresponding to the path $\widetilde{\pi}$.

Now, define $\widetilde{C}^{(N)}_{\mathsf{sum}}((v_0, 1, 1, \ldots, 1))$ as the infimum of $\widetilde{c}^{(N)}_{\mathsf{sum}}(\widetilde{\pi})$ over all paths $\widetilde{\pi}$ with $\widetilde{\pi}[0] = (v_0, 1, 1, \ldots, 1)$, and similarly for $\widetilde{C}_{\mathsf{av}}((v_0, 1, 1, \ldots, 1))$. Then it is easy to see that $C^{(N)}_{\mathsf{sum}}(v_0) = \widetilde{C}^{(N)}_{\mathsf{sum}}((v_0, 1, 1, \ldots, 1))$ and $C_{\mathsf{av}}(v_0) = \widetilde{C}_{\mathsf{av}}((v_0, 1, 1, \ldots, 1))$. Thus, we can reduce the optimal path and value problems for G to standard objectives in $\widetilde{G}$. The major difficulty is that $\widetilde{G}$ is infinite. However, note that only the first $N + 1$ nodes of $\widetilde{\pi}$ are relevant for the computation of $\widetilde{c}^{(N)}_{\mathsf{sum}}(\widetilde{\pi})$. Thus, the value of $\widetilde{C}^{(N)}_{\mathsf{sum}}((v_0, 1, \ldots, 1))$ can be computed on a *finite* subgraph of $\widetilde{G}$, obtained by considering only the finite subset of nodes $V \times \mathbb{Z}[1, N + 1]^{|V|} \subseteq \widetilde{V}$.

For $\gamma < 1$, we obtain the value $\widetilde{C}^{(N)}_{\mathsf{sum}}(\widetilde{\pi})$ by a standard dynamic programming algorithm which computes the shortest path of length N on this finite subgraph starting from the node $(v_0, 1, 1, \ldots, 1)$ (and keeping track of the number of steps). For $\gamma = 1$, where the objective is to maximize $r^{(N)}_{\mathsf{sum}}(\pi) = \lambda \sum_{t=0}^{N} \mathsf{Last}_\pi(t, \pi[t])$ over paths π, we proceed analogously. Note that the subgraph used for the dynamic programming computations is of exponential size in terms of the size of G and the description of N. This gives the desired result in Theorem 4.

4 Infinite Horizon Rewards: Computing $R_{\mathsf{av}}(v)$

Since we consider finite graphs, every infinite path eventually stays in some strongly connected component (SCC). Furthermore, the value of the reward function $r_{\mathsf{av}}(\pi)$ does not change if we alter/remove a finite prefix of the path π. Thus, it suffices to restrict our attention to the SCCs of the graph: the problem of finding an optimal path from a node $v \in V$ reduces to finding the SCC that gives the highest reward among the SCCs reachable from node v. Therefore, we assume that the graph is strongly connected.

4.1 Hardness of Exact $R_{\mathsf{av}}(v)$ Value Computation

Since it is sufficient for the hardness results, we consider node-invariant λ and γ.

Insufficiency of memoryless strategies. Before we turn to the computational hardness of the value decision problem, we look at the *strategy complexity* of the optimal path problem and show that optimal strategies need memory.

▶ **Proposition 5.** *Memoryless strategies do not suffice for the infinite horizon problem.*

Proof. Consider Example 3. A memoryless strategy results in paths which cycle exclusively either in the left cycle, or the right cycle (as from node a, it prescribes a move to either only b or only d). As shown in Example 3, the optimal path for $\gamma \geq a_1$ needs to visit both cycles. Thus, memoryless strategies do not suffice for this example. ◀

For ω-regular objectives, strategies based on *latest visitation records* [14, 22], which depend only on the *order* of the last node visits (*i.e.*, for all node pairs $v_1 \neq v_2 \in V$, whether the last visit of v_1 was before that of v_2 or vice versa) are sufficient. However, we can show that such strategies do not suffice either. To see this, recall the graph in Figure 1 for which the optimal path for $\gamma = 0.26$ is $(abcabcad)^\omega$. Upon visiting node a this strategy chooses one of the nodes b or d depending on the *number* of visits to b since the last occurrence of d. On the other hand, every strategy based *only on the order of last visits* is not able to count the number of visits to b and thus, results in a path that ends up in one of $(abc)^\omega$, or $(ad)^\omega$, or $(abcad)^\omega$, which are not optimal for this γ. The proof is given in the full version [9]. It is open if finite memory strategies are sufficient for the infinite-horizon optimal path problem.

NP-Hardness of the value decision problem. To show NP-hardness of the infinite-horizon value decision problem, we first give bounds on $R_{\mathsf{av}}(v_0)$. The following Lemma proven in the full version [9], establishes these bounds.

▶ **Lemma 6.** *For any graph $G = (V, E)$ and any node $v_0 \in V$, $R_{\mathsf{av}}(v_0)$ is bounded as*

$$\lambda\frac{1-\gamma^p}{1-\gamma} \leq R_{\mathsf{av}}(v_0) \leq \lambda\frac{1-\gamma^{n_{\mathsf{v}}}}{1-\gamma}, \tag{7}$$

where $n_{\mathsf{v}} = |V|$ and p is the length of the longest simple cycle in G.

▶ **Corollary 7.** *If the graph $G = (V, E)$ contains a Hamiltonian cycle, any path $\pi = (v_1 v_2 \dots v_{|V|})^\omega$, with $v_1 v_2 \dots v_{|V|} v_1$ being a Hamiltonian cycle, is optimal and the optimal value of $R_{\mathsf{av}}(v_0)$ is exactly the upper bound in* (7).

The following lemma establishes a relationship between the value of optimal paths and the existence of a Hamiltonian cycle in the graph, and is useful for providing a lower bound on the computational complexity of the value decision problem.

▶ **Lemma 8.** *If the upper bound in* (7) *is achieved with a path π for some $\gamma < 1/|V|$, then the graph contains a Hamiltonian cycle ρ that occurs in π infinitely often.*

Proof. The proof is by contradiction. Suppose π does not visit any Hamiltonian cycle infinitely often. Then it visits each such cycle at most a finite number of times. Without loss of generality we can assume that the path doesn't visit any such cycles, since the total number of Hamiltonian cycles is finite. We have for $n_{\mathsf{v}} = |V|$

$$c_{\mathsf{sum}}^{(N)}(\pi) \geq \sum_{t=0}^{N} \gamma^{\mathsf{Last}_\pi(t,\pi[t])}\mathbb{I}(\mathsf{Last}_\pi(t,\pi[t]) < n_{\mathsf{v}}) \geq \gamma^{n_{\mathsf{v}}-1}\sum_{t=0}^{N}\mathbb{I}(\mathsf{Last}_\pi(t,\pi[t]) < n_{\mathsf{v}}).$$

Now let's look at $\pi_{n,n_{\mathsf{v}}} = \pi[n(n+1)\dots(n+n_{\mathsf{v}})]$ a finite sub-path of length n_{v}. There is at least one node repeated in $\pi_{n,n_{\mathsf{v}}}$ since the graph has n_{v} distinct nodes. Note that if $\pi[n] = \pi[n+n_{\mathsf{v}}]$, there must be another repetition due to the lack of Hamiltonian cycles in the path. In either case, there is an $i_n \in \mathbb{Z}[n, n+n_{\mathsf{v}}]$ such that $\mathsf{Last}_\pi(i_n, \pi[i_n]) < n_{\mathsf{v}}$.

$$c_{\mathsf{sum}}^{(N)}(\pi) \geq \gamma^{n_{\mathsf{v}}-1}\sum_{t=0}^{N}\mathbb{I}(\mathsf{Last}_\pi(t,\pi[t]) < n_{\mathsf{v}}) \geq \gamma^{n_{\mathsf{v}}-1}\left\lfloor\frac{N}{n_{\mathsf{v}}}\right\rfloor$$

$$\Rightarrow c_{\mathsf{av}}(\pi) \geq \gamma^{n_{\mathsf{v}}-1}\limsup_{N\to\infty}\frac{1}{N+1}\left\lfloor\frac{N}{n_{\mathsf{v}}}\right\rfloor = \gamma^{n_{\mathsf{v}}-1}\frac{1}{n_{\mathsf{v}}} \Rightarrow \gamma^{n_{\mathsf{v}}} \geq \left(\gamma^{n_{\mathsf{v}}-1}\right)/n_{\mathsf{v}} \;\Rightarrow\; \gamma \geq 1/n_{\mathsf{v}},$$

which is a contradiction with the assumption that $\gamma < 1/n_{\mathsf{v}}$. Then a necessary condition for $c_{\mathsf{av}}(\pi) = \gamma^{n_{\mathsf{v}}}$ with some $\gamma < 1/n_{\mathsf{v}}$ is the existence of a Hamiltonian cycle. ◀

▶ Remark. Following the same reasoning of the above proof it is possible to improve the upper bound in (7) as $R_{\mathsf{av}}(v_0) \leq \lambda\left(1 - \gamma^p/n_{\mathsf{v}}\right)/(1-\gamma)$ for small values of γ, where p is the length of the longest simple cycle of the graph.

▶ **Theorem 9.** *The infinite horizon value decision problem is NP-hard for $\gamma < 1/|V|$.*

Proof. We reduce the Hamiltonian cycle problem to the infinite horizon optimal path problem. Given a graph $G = (V, E)$, we fix some λ and $\gamma < 1/|V|$. We show that G is Hamiltonian iff $R_{\mathsf{av}}(v_0) \geq \lambda\left(1 - \gamma^{|V|}\right)/(1-\gamma)$. If G has a Hamiltonian cycle $v_1 v_2 \dots v_{|V|} v_1$, then the infinite path $\pi = (v_1 v_2 \dots v_{|V|})^\omega$ has reward $r_{\mathsf{av}}(\pi) = \lambda\left(1 - \gamma^{|V|}\right)/(1-\gamma)$, for any choice of γ. For the other direction, applying Lemma 8 with $\gamma < 1/|V|$ implies that G is Hamiltonian. ◀

Non-discounted rewards ($\gamma = 1$) and node-invariant function λ. Contrary to the finite-horizon non-discounted case, the infinite-horizon optimal path and value problems for $\gamma = 1$ can be solved in polynomial time. To see this, note that the reward expression $r_{\mathsf{sum}}^{(N)}(\pi)$ can be written as $r_{\mathsf{sum}}^{(N)}(\pi) = \lambda\sum_{v\in V} y(v,\pi,N)$, where $y(v,\pi,N)$ is defined as $y(v,\pi,N) = 1 + \max U$, for $U = \{j \in \mathbb{N} | j \leq N, \pi[j] = v\} \cup \{-1\}$. Then, we can bound the reward by

$$r_{\mathsf{av}}(\pi) \leq \lim_{N\to\infty}\frac{\lambda\sum_{i=1}^{|V(\pi)|}(N+1-i+1)}{N+1} = \lim_{N\to\infty}\frac{\lambda|V(\pi)|(2N-|V(\pi)|+3)}{2(N+1)} = \lambda|\mathsf{Inf}(\pi)|,$$

where $\mathsf{Inf}(\pi)$ is the set of nodes visited in the path π infinitely often. This indicates that the maximum reward is bounded by λ times the maximal size of a reachable SCC in the graph G. This upper bound is also achievable: we can construct an optimal path by finding a maximal SCC reachable from the initial node and a (not necessarily simple) cycle $v_1 \dots v_n v_1$ that visits all the nodes in this SCC. Then, a subset of optimal paths contains paths of the form $\pi_0 \cdot (v_1 \dots v_n)^\omega$, where π_0 is any finite path that reaches v_1. This procedure can be done with a computational time polynomial in the size of G.

▶ **Example 10.** Consider the graph in Figure 1. The optimal reward for the infinite-horizon non-discounted case is 4λ, achievable by the path $\pi = (abcad)^\omega$. Another path which is not ultimately periodic but achieves the same reward $\pi' = (abc)(ad)(abc)^2(ad)^2 \dots (abc)^n(ad)^n \dots$.

Note that for the case $\gamma = 1$ there always exists an ultimately periodic optimal path, such a path is generated by a finite-memory strategy.

4.2 Approximate Computation of $R_{\mathsf{av}}(v)$

In the previous section we discussed how to solve the infinite-horizon value and path computation problems for the non-discounted case. Now we show how the infinite-horizon path and value computation problems for $\gamma < 1$ can be effectively approximated. We first define functions that over and underapproximate $C_{\mathsf{av}}(v)$ (thus also $R_{\mathsf{av}}(v)$) and establish bounds on the error of these approximations. Given an integer $K \in \mathbb{N}$, approximately optimal paths and an associated interval approximating $C_{\mathsf{av}}(v)$ can be computed using a finite augmented graph $\widetilde{G}_K$ based on the augmented graph $\widetilde{G}$ of Section 3. Intuitively, $\widetilde{G}_K$ is obtained from $\widetilde{G}$ by pruning nodes that have a component greater than K in their augmentation. By increasing the value of K, the approximation error can be made arbitrarily small.

We describe the approximation algorithm for node-invariant λ and γ. The results generalize trivially to the case when λ and γ are not node-invariant by choosing K large enough to satisfy the condition that bounds the approximation error for each $\lambda(v)$ and $\gamma(v)$.

Approximate cost functions. Consider the following functions from $V^* \times \mathbb{N}$ to $\mathbb{R}_{\geq 0}$:

$$\bar{c}^{(N)}_{\mathsf{sum}}(\pi, K) := \sum_{t=0}^{N} \gamma^{\min\{K, \mathsf{Last}_\pi(t, \pi[t])\}} \quad \text{and}$$

$$\underline{c}^{(N)}_{\mathsf{sum}}(\pi, K) := \sum_{t=0}^{N} \gamma^{\mathsf{Last}_\pi(t, \pi[t])} \mathbb{I}(\mathsf{Last}_\pi(t, \pi[t]) \leq K).$$

Informally, for $\bar{c}^{(N)}_{\mathsf{sum}}(\pi, K)$, if the last visit to node $\pi[t]$ occurred more than K time units before time t, the cost is γ^K, rather than the original smaller amount $\gamma^{\mathsf{Last}_\pi(t,\pi[t])}$. For $\underline{c}^{(N)}_{\mathsf{sum}}(\pi, K)$, if the last visit to $\pi[t]$ occurred more than K time steps before time t, then the cost is 0. For both, if the last visit to the node $\pi[t]$ occurred less than or equal to K steps before, we pay the actual cost $\gamma^{\mathsf{Last}_\pi(t,\pi[t])}$. The above definition implies that $\underline{c}^{(N)}_{\mathsf{sum}}(\pi, K) \leq c^{(N)}_{\mathsf{sum}}(\pi) \leq \bar{c}^{(N)}_{\mathsf{sum}}(\pi, K)$ for every π. Then we have $\underline{C}(v_0, K) \leq C_{\mathsf{av}}(v_0) \leq \overline{C}(v_0, K)$, where we define

$$\underline{C}(v_0, K) := \inf_{\pi, \pi[0]=v_0} \limsup_{N \to \infty} \frac{\underline{c}^{(N)}_{\mathsf{sum}}(\pi, K)}{N+1} \quad \text{and}$$

$$\overline{C}(v_0, K) := \inf_{\pi, \pi[0]=v_0} \limsup_{N \to \infty} \frac{\bar{c}^{(N)}_{\mathsf{sum}}(\pi, K)}{N+1}.$$

The difference between the upper and lower bounds can be tuned by selecting K:

$$\bar{c}^{(N)}_{\mathsf{sum}}(\pi, K) - \underline{c}^{(N)}_{\mathsf{sum}}(\pi, K) = \sum_{t=0}^{N} \gamma^K \mathbb{I}(\mathsf{Last}_\pi(t, \pi[t]) \geq K+1)$$
$$\implies 0 \leq \bar{c}^{(N)}_{\mathsf{sum}}(\pi, K) - \underline{c}^{(N)}_{\mathsf{sum}}(\pi, K) \leq (N+1)\gamma^K$$
$$\implies \underline{c}^{(N)}_{\mathsf{sum}}(\pi, K) \leq \bar{c}^{(N)}_{\mathsf{sum}}(\pi, K) \leq \underline{c}^{(N)}_{\mathsf{sum}}(\pi, K) + (N+1)\gamma^K$$
$$\implies \underline{C}(v_0, K) \leq \overline{C}(v_0, K) \leq \underline{C}(v_0, K) + \gamma^K.$$

Therefore $C_{\mathsf{av}}(v_0)$ belongs to the interval $[\underline{C}(v_0,K),\overline{C}(v_0,K)] \subset [0,\gamma^K]$ and the length of the interval is at most γ^K. In order to guarantee the total error of $\epsilon > 0$ for the actual reward $R_{\mathsf{av}}(v_0)$[1], we can select $K \in \mathbb{N}$ according to $\frac{\lambda}{1-\gamma}\gamma^K \le \epsilon \Longrightarrow K \ge \ln\left[\frac{\epsilon(1-\gamma)}{\lambda}\right] / \ln\gamma$. The value $C_{\mathsf{av}}(v_0)$ can be computed up to the desired degree of accuracy by computing either $\overline{C}(v_0,K)$ or $\underline{C}(v_0,K)$. Next, we give the procedure for computing $\overline{C}(v_0,K)$ (the procedure for $\underline{C}(v_0,K)$ is similar). It utilizes a finite augmented weighted graph $\widetilde{G}_K$.

Truncated augmented weighted graph $\widetilde{G}_K$. Recall the infinite augmented weighted graph $\widetilde{G}$ from Section 3. We define a truncated version $\widetilde{G}_K = (\widetilde{V}_K, \widetilde{E}_K)$ of $\widetilde{G}$ where we only keep track of $\mathsf{Last}_\pi(t,v)$ values less than or equal to K. For $\widetilde{G}_K$ we define two weight functions $\overline{w}$ and $\underline{w}$, for $\bar{c}^{(N)}_{\mathsf{sum}}$ and $\underline{c}^{(N)}_{\mathsf{sum}}$ respectively.

- The set of nodes $\widetilde{V}_K$ is $V \times \mathbb{Z}[0,K]^{|V|}$.
- For a node $\widetilde{v} = (v, b_1, b_2, \dots, b_{|V|}) \in \widetilde{V}$,
  - the weight $\overline{w}(\widetilde{v})$ is γ^{b_v} if $b_v > 0$ and γ^K otherwise.
  - the weight $\underline{w}(\widetilde{v})$ is γ^{b_v} if $b_v > 0$ and 0 otherwise.
- The set of edges $\widetilde{E}_K$ consists of edges $(v, b_1, b_2, \dots, b_{|V|}) \to (v', b'_1, b'_2, \dots, b'_{|V|})$ such that $(v \to v') \in E$, $b'_v = 1$, and for $u \neq v$:
  - $b'_u = b_u + 1$ if $b_u > 0$ and $b_u + 1 \le K$,
  - $b'_u = 0$ if $b_u = 0$ or $b_u + 1 > K$.

  Thus, in $\widetilde{G}_K$ we specify two different weights for path π at time step t in the case when the previous visit to $\pi[t]$ in π was more than K time steps ago.

  Similarly to the infinite augmented graph we have
- $\bar{c}^{(N)}_{\mathsf{sum}}(\pi,K)$ is the sum of weights assigned by $\overline{w}$ to the first $(N+1)$ nodes of $\widetilde{\pi}$.
- $\underline{c}^{(N)}_{\mathsf{sum}}(\pi,K)$ is the sum of weights assigned by $\underline{w}$ to the first $(N+1)$ nodes of $\widetilde{\pi}$.

It is easy to see that $\overline{C}(v_0,K)$ is the least possible limit-average cost with respect to $\overline{w}$ in $\widetilde{G}_K$ starting from the node $\widetilde{v}_0 = (v_0, 1, 1, \dots, 1)$. The same holds for $\underline{C}(v_0,K)$ with $\underline{w}$. Below we show how to compute $\overline{C}(v_0,K)$. The case $\underline{C}(v_0,K)$ is analogous, and thus omitted.

Algorithm for computing $\overline{C}(v_0,K)$. We now describe a method to compute $\overline{C}(v_0,K)$ as the least possible limit-average cost in $\widetilde{G}_K$ with respect to $\overline{w}$. It is well-known that this can be reduced to the computation of the minimum cycle mean in the weighted graph [26], which in turn can be done using the algorithm from [16] that we now describe.

As before, first we assume that $\widetilde{G}_K$ is strongly connected. For every $\widetilde{v} \in \widetilde{V}_K$, and every $n \in \mathbb{Z}_+$, we define $W_n(\widetilde{v})$ as the minimum weight of a path of length n from $\widetilde{v}_0$ to $\widetilde{v}$; if no such path exists, then $W_n(\widetilde{v}) = \infty$. The values $W_n(\widetilde{v})$ can be computed by the recurrence

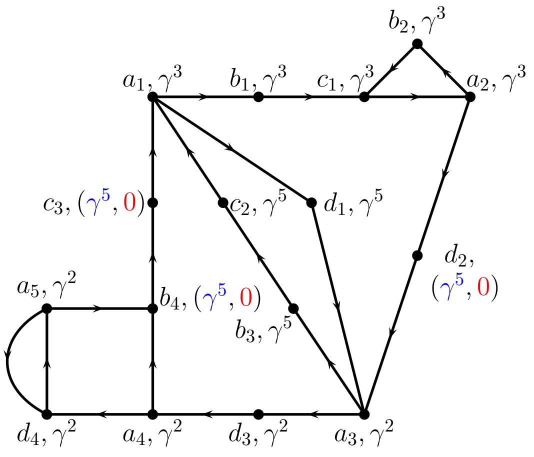
$$W_n(\widetilde{v}) = \min_{(\widetilde{u},\widetilde{v}) \in \widetilde{E}_K} \{W_{n-1}(\widetilde{u}) + \overline{w}(\widetilde{v})\}, \quad n = 1, 2, \dots, |\widetilde{V}_K|,$$

with the initial conditions $W_0(\widetilde{v}_0) = 0$ and $W_0(\widetilde{v}) = \infty$ for any $\widetilde{v} \neq \widetilde{v}_0$. Then, we can compute

$$\overline{C}(v_0,K) = \min_{\widetilde{v} \in \widetilde{V}_K} \max_{n \in \mathbb{Z}[0,|\widetilde{V}_K|-1]} \left[\frac{W_{|\widetilde{V}_K|}(\widetilde{v}) - W_n(\widetilde{v})}{|\widetilde{V}_K| - n}\right].$$

A cycle $\widetilde{\rho}$ with the computed minimum mean can be extracted by fixing the node $\widetilde{v}$ which achieves the minimum in the above value and the respective path length n and finding a

[1] Since $R_{\mathsf{av}}(v_0)$ is upper bounded by $\lambda/(1-\gamma)$, we assume that the required accuracy ϵ is less than this upper bound.



$a_1 = (a, 3, 2, 1, 4)$ $a_2 = (a, 3, 2, 1, 0)$
$a_3 = (a, 2, 4, 3, 1)$ $a_4 = (a, 2, 0, 5, 1)$
$a_5 = (a, 2, 0, 0, 1)$

$b_1 = (b, 1, 3, 2, 5)$ $b_2 = (b, 1, 3, 2, 0)$
$b_3 = (b, 1, 5, 4, 2)$ $b_4 = (b, 1, 0, 0, 2)$

$c_1 = (c, 2, 1, 3, 0)$ $c_2 = (c, 2, 1, 5, 3)$
$c_3 = (c, 2, 1, 0, 3)$

$d_1 = (d, 1, 3, 2, 5)$ $d_2 = (d, 1, 3, 2, 0)$
$d_3 = (d, 1, 5, 4, 2)$ $d_4 = (d, 1, 0, 0, 2)$

Figure 2 The SCC of the finite augmented graph $\widetilde{G}_5$ for the graph in Figure 1. The node labels are the values of the functions $\overline{w}$ and $\underline{w}$ (in black if $\overline{w} = \underline{w}$, otherwise respectively in blue and red).

minimum-weight path from $\widetilde{v}_0$ to $\widetilde{v}$ and a cycle of length $|\widetilde{V}_K| - n$ within this path. Thus, the path $\widetilde{\pi}$ in $\widetilde{G}_K$ obtained by repeating $\widetilde{\rho}$ infinitely often realizes this value. A path π from v_0 in G with $c_{\mathsf{av}}(\pi) = \overline{C}(v_0, K)$ is obtained from $\widetilde{\pi}$ by projection on V.

In the general case, when $\widetilde{G}_K$ is not strongly connected we have to consider each of its SCCs reachable from $\widetilde{v}_0$, and determine the one with the least minimum cycle mean.

For each SCC with m edges and n nodes the computation of the quantities W requires $O(n \cdot m)$ operations. The computation of the minimum cycle mean for this component requires $O(n^2)$ further operations. Since $n \leq m$ because of the strong connectivity, the overall computation time for the SCC is $O(n \cdot m)$. Finally, the SCCs of $\widetilde{G}_K$ can be computed in time $O(|\widetilde{V}_K| + |\widetilde{E}_K|)$ [21]. This gives us the following result.

▶ **Lemma 11.** *The value $\overline{C}(v_0, K)$ and a path π with limit average cost $c_{\mathsf{av}}(\pi) = \overline{C}(v_0, K)$ can be computed in time polynomial in the size of $\widetilde{G}_K$.*

The same result can be established for the under approximation $\underline{C}(v_0, K)$.

▶ Remark. The number of nodes of $\widetilde{G}_K$ is $|\widetilde{V}_K| = |V| \times (K+1)^{|V|}$. For the approximation procedure described above it suffices to augment the graph with the information about which nodes were visited in the last K steps and in what order. Thus, we can alternatively consider a graph with $|V| \times |V|^K$ nodes in the case when the computed K is smaller than $|V|$.

▶ **Example 12.** Figure 2 shows the SCC of the augmented graph $\widetilde{G}_K$ reachable from initial node $(a, 1, 1, 1, 1)$ for the graph given in Figure 1 and $K = 5$. The nodes in the SCC are denoted by shorthands in the picture, *e.g.*, $a_1 = (a, 3, 2, 1, 4)$. The labels on the nodes correspond to the values of the weight functions $\overline{w}$ and $\underline{w}$. As we can see, $\widetilde{G}_5$ already contains simple cycles $(a_2 b_2 c_2 a_2), (a_3 b_3 c_2 a_1 b_1 c_1 a_2 d_2 a_3), (a_3 b_3 c_2 a_1 d_1 a_3)$, which correspond to the optimal paths presented in Example 3. The outcome of the minimum cycle mean for $\widetilde{G}_5$ will be the same with the two sets of weights only for the first and third interval for γ determined in Example 3, but will be different for the second case in which the term γ^8 is replaced respectively by γ^5 and 0 for the upper and lower bounds.

Theorem 13, a consequence of Lemma 11, states the approximate computation result.

▶ **Theorem 13.** *Given a graph $G = (V, E)$, node $v_0 \in V$, rational values λ and $0 < \gamma < 1$, and error bound $\epsilon > 0$, we can compute in time polynomial in $|V|(K+1)^{|V|}$ for $K =$*

$\ln\left[\frac{\epsilon(1-\gamma)}{\lambda}\right] / \ln\gamma$ *(i.e., exponential in $|V|$), infinite paths π_{under} and π^{over} and values $r_{\mathsf{av}}(\pi_{\mathsf{under}})$ and $r_{\mathsf{av}}(\pi_{\mathsf{over}})$ such that $r_{\mathsf{av}}(\pi_{\mathsf{under}}) \leq R_{\mathsf{av}}(v_0) \leq r_{\mathsf{av}}(\pi_{\mathsf{over}})$ and $r_{\mathsf{av}}(\pi_{\mathsf{over}}) - r_{\mathsf{av}}(\pi_{\mathsf{under}}) \leq \epsilon$.*

4.3 Approximation via Bounded Memory

The algorithm presented earlier is based on an augmentation of the graph with a specific structure updated deterministically and whose size depends on the desired quality of approximation. Furthermore, in this graph there exists a memoryless strategy with approximately optimal reward. We show that this allows us to quantify how far from the optimal reward value is a strategy that is optimal among the ones with bounded memory of fixed size.

First, we give the definition of memory structures. A *memory structure* $\mathcal{M} = (M, m_0, \delta)$ for a graph $G = (V, E)$ consists of a set M, initial memory $m_0 \in M$ and a memory update function $\delta : M \times V \to M$. The memory update function can be extended to $\delta^* : V^* \to M$ by defining $\delta^*(\epsilon) = m_0$ and $\delta^*(\pi \cdot v) = \delta(\delta^*(\pi), v)$. A memory structure $\mathcal{M}$ together with a function $\tau : V \times M \to V$ such that $(v, \tau(v, m)) \in E$ for all $v \in V$ and $m \in M$, and an initial node $v_0 \in V$ define a strategy $\sigma : V^* \to V$ where $\sigma(\epsilon) = v_0$ and $\sigma(\pi \cdot v) = \tau(v, \delta^*(\pi))$. In this case we say that the strategy σ has memory $\mathcal{M}$. Given a bound $B \in \mathbb{N}$ on memory size we define the finite graph $G \times B = (V^{G\times B}, E^{G\times B})$, where $V^{G\times B} = V \times \mathbb{Z}[1, B]$; and $E^{G\times B} = \{((v, i), (v', j)) \in (V \times \mathbb{Z}[1, B]) \times (V \times \mathbb{Z}[1, B]) \mid (v, v') \in E\}$).

Memoryless strategies in this graph precisely correspond to strategies that have memory of size B. More precisely, for each strategy σ in $G = (V, E)$ that has memory $\mathcal{M} = (M, m_0, \delta)$ there exists a memoryless strategy $\sigma_{\mathcal{M}}$ in $G \times |M|$ such that the projection of $\mathsf{outcome}((v_0, m_0), \sigma_{\mathcal{M}})$ on V is $\mathsf{outcome}(v_0, \sigma)$. Conversely, for each memoryless strategy $\sigma_{\mathcal{M}}$ in $G \times B$ there exist a memory structure $\mathcal{M} = (M, m_0, \delta)$ with $|M| = B$ and a strategy σ with memory $\mathcal{M}$ in G such that the projection of $\mathsf{outcome}((v_0, m_0), \sigma_{\mathcal{M}})$ on V is $\mathsf{outcome}(v_0, \sigma)$.

▶ **Example 14.** Recall that in Example 3 we established that an optimal path for $\gamma = 0.26$ is the path $(abcabcad)^\omega$. In the graph $G \times 3$ there exists a simple cycle corresponding to this path, namely $(a, 1)(b, 1)(c, 2)(a, 2)(b, 2)(c, 3)(a, 3)(d, 1)(a, 1)$. Thus, the optimal path $(abcabcad)^\omega$ corresponds to a strategy with memory size of 3.

An optimal strategy among those with memory of a given size B can be computed by inspecting the memoryless strategies in $G \times B$ and selecting the one with maximal reward (these strategies are finitely but exponentially many).

A strategy returned by the approximation algorithm presented earlier uses a memory structure of size $(K+1)^{|V|}$. If $(K+1)^{|V|} \leq B$, then this strategy is isomorphic to some memoryless strategy σ in $G \times B$. Since the reward achieved by the optimal memoryless strategy in $G \times B$ is at least that of σ, its value is at most $\frac{\lambda}{1-\gamma}\gamma^K$ away from $R_{\mathsf{av}}(v_0)$. Now, $(K+1)^{|V|} \leq B$ iff $K \leq \left\lfloor \frac{\ln B}{\ln|V|} \right\rfloor - 1$. Thus, under a memory size B, we are guaranteed to find a strategy which has reward that at most $\frac{\lambda}{1-\gamma}\gamma^{\left\lfloor \frac{\ln B}{\ln|V|} \right\rfloor - 1}$ away from the optimal.

Next we sketch an algorithm for computing optimal B-memory bounded strategies. As mentioned previously, we can restrict our attention to memoryless strategies in $G \times B$. Memoryless strategies in this graph lead to lasso shaped infinite paths, with an initial non-cyclic path followed by a simple cycle which is repeated infinitely often. This means that we can restrict our attention to paths of length $|V| \cdot B$ which complete the lasso. Now, we apply this length bound to restrict our attention to the finite portion of the augmented graph $\widetilde{G}$ from Section 3 that corresponds to path lengths at most $|V| \cdot B$. This finite subgraph contains nodes $V \times \mathbb{N}[1, |V| \cdot B]^{|V|}$. The dynamic programming algorithm is polynomial time on this graph, hence we get a running time which is polynomial in $|V| \cdot (|V| \cdot B)^{|V|}$.

▶ **Theorem 15.** *Given a graph $G = (V, E)$, a node $v_0 \in V$, rational values λ and $0 < \gamma < 1$, and a memory bound $B > 1$, we can compute a B-memory optimal strategy σ with reward r_{av} at most $\frac{\lambda}{1-\gamma}\gamma^{\left\lfloor \frac{\ln B}{\ln |V|} \right\rfloor - 1}$ away from the optimal $R_{\mathsf{av}}(v_0)$ in time polynomial in $|V|^{(|V|+1)} \cdot B^{|V|}$.*

5 Conclusion and Open Problems

We have introduced the robot routing problem, which is a reward collection problem on graphs in which the reward structure combines spatial aspects with dynamic and stochastic reward generation. We have studied the computational complexity of the finite and infinite-horizon versions of the problem, as well as the strategy complexity of the infinite-horizon case. We have shown that optimal strategies need memory in general, and that strategies based on last visitation records do not suffice. However, the important question about whether finite-memory suffices for the infinite-horizon optimal path problem or infinite memory is needed remains open. In case finite-memory suffices, it will be desirable to provide an algorithm for precisely solving the infinite-horizon value problem. So far we have only given methods for approximating the optimal solution.

Another interesting line of future work is the generalization of the model to incorporate timing constraints. More specifically, in this work we have assumed that all the rewards accumulated at a node are instantaneously collected. This assumption is justified by the fact that in many request-serving scenarios the time taken to serve the requests at a given location is negligible compared to the time necessary to travel between the locations. A more precise model, however, would have to incorporate the serving time per node, which would depend on the amount of collected reward. This would imply that the elapsed time becomes a random variable, while in our current model it is deterministic.

Other extensions include the setting where the robot can react to the environment by observing the collected rewards, or observing the accumulated rewards at nodes of the graph, or where there is probabilistic uncertainty in the transitions of the robot in the graph. Finally, the robot routing problem presents new challenges for development of efficient approximation algorithms, which is a major research direction concerning path optimization problems.

References

1 Sofia Amador, Steven Okamoto, and Roie Zivan. Dynamic multi-agent task allocation with spatial and temporal constraints. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1384–1390. AAAI Press, 2014. URL: `http://dl.acm.org/citation.cfm?id=2615731.2616029`.

2 Nikhil Bansal, Avrim Blum, Shuchi Chawla, and Adam Meyerson. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing 2004*, pages 166–174. ACM, 2004. `doi:10.1145/1007352.1007385`.

3 Dimitris J. Bertsimas and Garrett J. van Ryzin. A stochastic and dynamic vehicle routing problem in the Euclidean plane. *Operations Research*, 39(4):601–615, 1991. `doi:10.1287/opre.39.4.601`.

4 Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. *SIAM J. Comput.*, 37(2):653–670, 2007. `doi:10.1137/050645464`.

5 Patricia Bouyer, Piotr Hofman, Nicolas Markey, Mickael Randour, and Martin Zimmermann. Bounding average-energy games. In *Foundations of Software Science and Computa-*

tion Structures: 20th International Conference, FOSSACS 2017, pages 179–195. Springer Berlin Heidelberg, 2017. doi:10.1007/978-3-662-54458-7_11.

6 Tomás Brázdil, Petr Hlinený, Antonín Kucera, Vojtech Rehák, and Matús Abaffy. Strategy synthesis in adversarial patrolling games. *CoRR*, abs/1507.03407, 2015.

7 Francesco Bullo, Emilio Frazzoli, Marco Pavone, Ketan Savla, and Stephen L. Smith. Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, 99(9):1482–1504, 2011. doi:10.1109/JPROC.2011.2158181.

8 Krishnendu Chatterjee and Vinayak S. Prabhu. Quantitative temporal simulation and refinement distances for timed systems. *IEEE Trans. Automat. Contr.*, 60(9):2291–2306, 2015. doi:10.1109/TAC.2015.2404612.

9 Rayna Dimitrova, Ivan Gavran, Rupak Majumdar, Vinayak S. Prabhu, and Sadegh Esmaeil Zadeh Soudjani. The robot routing problem for collecting aggregate stochastic rewards. *CoRR*, abs/1704.05303, 2017. URL: http://arxiv.org/abs/1704.05303.

10 Ali Ekici, Pinar Keskinocak, and Sven Koenig. Multi-robot routing with linear decreasing rewards over time. In *2009 IEEE International Conference on Robotics and Automation, ICRA 2009*, pages 958–963. IEEE, 2009. doi:10.1109/ROBOT.2009.5152803.

11 Ali Ekici and Anand Retharekar. Multiple agents maximum collection problem with time dependent rewards. *Computers & Industrial Engineering*, 64(4):1009–1018, 2013. doi:10.1016/j.cie.2013.01.010.

12 Sadegh Esmaeil Zadeh Soudjani and Alessandro Abate. Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes. *SIAM Journal on Applied Dynamical Systems*, 12(2):921–956, 2013. doi:10.1137/120871456.

13 Sadegh Esmaeil Zadeh Soudjani and Rupak Majumdar. Controller synthesis for reward collecting Markov processes in continuous space. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, HSCC '17, pages 45–54, New York, NY, USA, 2017. ACM. doi:10.1145/3049797.3049827.

14 Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *Proc. Symposium on Theory of Computing*, pages 60–65. ACM Press, 1982. doi:10.1145/800070.802177.

15 Satoshi Hoshino and Shingo Ugajin. Adaptive patrolling by mobile robot for changing visitor trends. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 104–110, 2016. doi:10.1109/IROS.2016.7759041.

16 Richard M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3):309–311, September 1978. doi:10.1016/0012-365X(78)90011-0.

17 Anthonius W. J. Kolen, Alexander H. G. Rinnooy Kan, and Henricus W. J. M. Trienekens. Vehicle routing with time windows. *Oper. Res.*, 35(2):266–273, April 1987. doi:10.1287/opre.35.2.266.

18 Justin Melvin, Pinar Keskinocak, Sven Koenig, Craig A. Tovey, and Banu Yuksel Ozkaya. Multi-robot routing with rewards and disjoint time windows. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2332–2337. IEEE, 2007. doi:10.1109/IROS.2007.4399625.

19 Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994. doi:10.1002/9780470316887.

20 Ruben Stranders, Enrique Munoz de Cote, Alex Rogers, and Nicholas R. Jennings. Near-optimal continuous patrolling with teams of mobile information gathering agents. *Artificial Intelligence*, 195:63 – 105, 2013. doi:10.1016/j.artint.2012.10.006.

21 Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972. doi:10.1137/0201010.

22 Wolfgang Thomas. On the synthesis of strategies in infinite games. In *STACS 95: Theoretical Aspects of Computer Science*, volume 900 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, 1995. `doi:10.1007/3-540-59042-0_57`.

23 Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011. `doi:10.1016/j.ejor.2010.03.045`.

24 Changyun Wei, Koen V. Hindriks, and Catholijn M. Jonker. Dynamic task allocation for multi-robot search and retrieval tasks. *Appl. Intell.*, 45(2):383–401, 2016. `doi:10.1007/s10489-016-0771-5`.

25 Tichakorn Wongpiromsarn, Alphan Ulusoy, Calin Belta, Emilio Frazzoli, and Daniela Rus. Incremental synthesis of control policies for heterogeneous multi-agent systems with linear temporal logic specifications. In *2013 IEEE International Conference on Robotics and Automation*, pages 5011–5018, 2013. `doi:10.1109/ICRA.2013.6631293`.

26 Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158(1&2):343–359, 1996. `doi:10.1016/0304-3975(95)00188-3`.