Motivating Time-Inconsistent Agents: A Computational Approach

Susanne Albers and Dennis Kraft(✉)

Department of Computer Science, Technical University of Munich, Munich, Germany
{albers,kraftd}@in.tum.de

Abstract. We study the complexity of motivating time-inconsistent agents to complete long term projects in a graph-based planning model as proposed by Kleinberg and Oren [5]. Given a task graph G with n nodes, our objective is to guide an agent towards a target node t under certain budget constraints. The crux is that the agent may change its strategy over time due to its present-bias. We consider two strategies to guide the agent. First, a single reward is placed at t and arbitrary edges can be removed from G. Secondly, rewards can be placed at arbitrary nodes of G but no edges must be deleted. In both cases we show that it is NP-complete to decide if a given budget is sufficient to guide the agent. For the first setting, we give complementing upper and lower bounds on the approximability of the minimum required budget. In particular, we devise a $(1 + \sqrt{n})$-approximation algorithm and prove NP-hardness for ratios greater than $\sqrt{n}/3$. Finally, we argue that the second setting does not permit any efficient approximation unless $\mathrm{P} = \mathrm{NP}$.

Keywords: Approximation algorithms · Behavioral economics · Computational complexity · Planning and scheduling · Time-inconsistency

1 Introduction

In this paper we study the phenomenon of *time-inconsistent behavior* from a computational perspective. Time-inconsistency is a fundamental problem in behavioral economics and has many examples in every day life including academia. For instance, consider a referee who agrees to evaluate a scientific proposal. Despite good intentions, the referee gets distracted and never submits a report. Or consider a student who enrolls in a course. After completing the first homework assignments, the student drops out without earning any credit. In general, these situations have a reoccurring pattern: An agent makes a plan to complete a set of tasks in the future, but changes the plan at a later point in time. This behavior is sometimes the result of unforeseen circumstances. However, in many cases the plan is changed or abandoned even if the circumstances stay the same. This paradox behavior of *procrastination* and *abandonment* is well-known in the

S. Albers and D. Kraft—Work supported by the European Research Council, Grant Agreement No. 691672.

field of behavioral economics and might severely affect the performance of agents in an economic or social domain, see e.g. [1,9,11].

A sensible explanation for time-inconsistent behavior is that agents assign disproportionately greater value to current cost than to future expenses. For example, consider a simple *car wash problem* in which Alice commissions Bob to wash her car. Each day Bob can either do the chore or postpone it to the next day. However, the longer he waits, the dirtier the car gets. On day i cleaning the car incurs a cost of $i/50$ while the cost of waiting another day is 0. After completing the task, Bob will receive a reward of 1 from Alice. Because Bob is present-biased, he perceives any current cost according to its true value, but discounts future costs and rewards by a factor of $\beta \in [0, 1]$. On day i he compares the cost of washing the car right away, which is $i/50$, to his perceived cost of washing it on the next day, which is $\beta(i+1)/50$. Suppose that $\beta = 1/3$. Because $i/50 > \beta(i+1)/50$, he procrastinates with good intentions of doing the job on the following day. On day $i = 50$, Bob's perceived cost for washing the car on the next day or any of the following days is at least $\beta(50 + 1)/50$. This exceeds his perceived reward of β and therefore he abandons the project.

Previous Work: There exists an extensive body of work on time-inconsistent behavior in the economic literature, cf. again [1,9,11]. In particular, the car wash problem as stated above is a special case of *quasi-hyperbolic discounting* [7]. We build on work by Kleinberg and Oren, who proposed a graph-based model that captures time-inconsistent behavior in general planning problems [5]. Since its introduction, their work has sparked an active line of research at the intersection of economics, mathematics and computer science, see e.g. [4,6].

We will give a formal definition of our model in Sect. 2. Essentially, it consists of a directed acyclic task graph G with n nodes. Each node represents a certain state of the project, whereas the edges are tasks necessary to transition between states. The workload of individual tasks is modeled by edge costs. To complete the project, an agent with bias factor $\beta \in [0, 1]$ must move from a designated source s to a target t. As a motivation, rewards are placed on the nodes of G. When located at some node of G, the agent considers all possible paths to t. However, because of its time inconsistency, the agent only evaluates the cost of incident edges accurately. All other costs and rewards are discounted by a factor of β. Let P be a path that minimizes the agent's *perceived net cost*. If this cost is at most 0, the agent traverses the first edge of P and then reassesses its plan. Otherwise the agent abandons the project. A graph in which the agent always reaches t is called *motivating*.

In this paper, we will take the perspective of a project designer, whose main objective is budget-efficiency. In other words, we try to minimize the reward we must spend to get the project completed. In general, various strategic arrangements can be made to increase budget-efficiency. Because the aim of such arrangements is to commit the agent to finish the project, they are also called *commitment devices* [2]. Consider, for instance, the car wash example. As the project designer Alice can introduce a deadline to keep Bob from procrastinating. As we will show in Sect. 2, this is beneficial to both of them. In general, the introduction of deadlines belongs to a broader range of popular commitment devices that reduce the agent's set of choices, see e.g. [9,10]. Note that the graph-based model lends itself to this approach as we can model any reduction of the agent's choices by simply removing the corresponding edges from G [5].

A second popular commitment device is to hand out rewards at intermediate states of the project [10]. In the graph-based model, we can do this by placing rewards at non-terminal nodes of G. We call such an assignment a *reward configuration*. This approach is especially interesting if the project designer's budget is only affected by rewards that are actually collected by the agent. As we will show in Sect. 2, this allows the construction of *exploitative* projects in which the agent is motivated by rewards it never claims. Considering the power and versatility of the two commitment devices mentioned above, Kleinbeg and Oren pose the complexity of computing motivating subgraphs and reward configurations as two important open problems [5].

In an unpublished manuscript, Tang et al. address both of these problems [12]. First, they show that it is NP-hard to decide if G contains a motivating subgraph for a fixed reward placed at t. Secondly, they give NP-hardness results for three variations of the reward configuration problem: One in which the rewards must be positive, one in which rewards may also be negative and one in which every reward that is laid out must be collected. In each setting, the project designer is charged the absolute sum of the rewards placed on G.

Our Contribution: We will thoroughly analyze the complexity and approximability of computing motivating subgraphs as well as reward configurations. In Sect. 3, we will settle the complexity of finding a motivating subgraph for a fixed reward at t. First, we will show that the problem is polynomially solvable if $\beta = 0$ or $\beta = 1$. We will then prove that it is NP-complete to decide the existence of a motivating subgraph for general $\beta \in (0, 1)$. Tang et al. showed NP-hardness via a reduction from 3-SAT [12]. In contrast, we use reduction from k DISJOINT CONNECTING PATHS. We believe this reduction to be simpler. More importantly, we will be able to generalize the reduction to obtain a hardness of approximation result at a later point.

Considering the hardness of the motivating subgraph problem, Sect. 4 will focus on an optimization version of the problem. More formally, we want to compute the minimum reward that must be placed at t such that G contains a motivating subgraph. We will propose a simple $(1 + \sqrt{n})$-approximation algorithm that outputs the reward and a corresponding motivating subgraph. As the main technical contribution of this paper, we will show that this approximation is asymptotically tight. In particular, we will prove that the problem cannot be approximated efficiently within a ratio less than $\sqrt{n}/3$ unless P = NP. Thus, we resolve the approximability of the motivating subgraph problem.

Finally, Sect. 5 will explore the problem of finding reward configurations within a fixed total budget of at most b. We will examine a version of the problem that, in our view, is the most sensible one. First, only positive rewards may be laid out. This assumption is reasonable as it is not entirely clear how negative rewards should be implemented in practice and how they are accounted for in the designer's budget. Secondly, the designer must only pay for rewards that are actually collected by the agent. This setting is fundamentally different from the settings analyzed by Tang et al. as it allows exploitative solutions. We show that the problem can be solved in polynomial-time if $\beta = 0$ or $\beta = 1$. Using a reduction from SET PACKING, we prove that deciding the existence of a motivating reward configuration is NP-complete for general $\beta \in (0, 1)$, even if $b = 0$. This immediately implies that the optimization problem of finding the minimum b for which a motivating reward configuration exists cannot be approximated efficiently within any ratio greater or equal to 1 unless P = NP.

2 The Formal Model

In the following, we will present Kleinberg and Oren's graph-based model [5]. Let $G = (V, E)$ be a finite directed acyclic graph. Associated with each edge (v, w) is a non-negative cost $c_G(v, w)$. Furthermore, the project designer may lay out positive rewards $r_G(v)$ at arbitrary nodes v. We call r a *reward configuration*. An agent with bias factor $\beta \in [0, 1]$ has to incrementally construct a path from a source s to a target t. Located at some node v different from t, the agent evaluates its *lowest perceived net cost*. For this purpose it considers all paths P from v to t. However, only the initial edge of P is accounted for by its actual value. All other costs and rewards along P are discounted by a factor of β. More precisely, let $d_{G,r}(w)$ denote the cost of a cheapest path from some node w to t with respect to the actual costs and rewards. Note that although $d_{G,r}(w)$ might be negative depending on r, no negative cycles can occur as G is acyclic. If no path exists, we assume that $d_{G,r}(w) = \infty$. The lowest perceived net cost is defined as $\zeta_{G,r}(v) = \min\{c_G(v, w) + \beta d_G(w) \mid (v, w) \in E\}$ if v has at least one outgoing edge. Otherwise, $\zeta_{G,r}(v) = \infty$. If $\zeta_{G,r}(v) > 0$, then the agent has no motivation to continue the project and abandons. Conversely, if $\zeta_{G,r}(v) \leq 0$, the agent traverses an edge (v, w) for which $c_{G,r}(v, w) + \beta d_{G,r}(w) = \zeta_{G,r}(v)$. Ties are broken arbitrarily. Note that the agent could take more than one path from s to t. A project is called motivating if the agent successfully reaches t along all such paths. To simplify our notation, we will omit G and r in the index of c, r, d and ζ whenever the graph and reward configuration is clear from context.

To illustrate the model, we consider the car wash problem from Sect. 1 once more. Assume that Alice's car must be washed during the next m days with $m > 50$. The task graph G is depicted in Fig. 1. For each day i with $1 \leq i \leq m$ there is a node v_i. Let v_1 be the source. There is an edge (v_i, t) of cost $i/50$ that

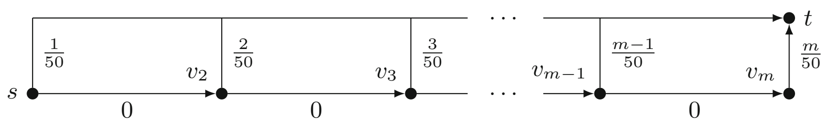


Fig. 1. Task graph of the car wash problem

represents the task of washing the car on day i. To keep the drawing simple, the edges (v_i, t) merge in Fig. 1. Furthermore, for every $i < m$ there is an edge (v_i, v_{i+1}) of cost 0 to postpone the task to the next day. Assume for now that Bob is located at some v_i with $i < m$. His perceived cost for procrastinating is at least $\beta(i + 1)/50$. This bound is tight if he plans to traverse (v_i, v_{i+1}) and then (v_{i+1}, t). Alternatively, his perceived cost for using (v_i, t) and washing the car on day i is $i/50$. Recall that Alice offers Bob a single reward $r(t) = 1$ upon completing the car wash. Furthermore, $\beta = 1/3$. As a result, the minimum perceived net cost is $\zeta(v_i) = \beta(i + 1)/50 - \beta$. We conclude that Bob always prefers to wash the car on the next day instead of doing it right away. Moreover, for $i < 50$ it holds true that $\zeta(v_i) \leq 0$. This means that during the first 49 days, Bob moves along $(v_i, i + 1)$ believing that he will finish the project the next day. However, once Bob reaches v_{50} he suddenly realizes that $\zeta(v_{50}) > 0$ and abandons. Therefore, the car wash problem in its current form is not motivating.

Next, assume that (v_{16}, v_{17}) is deleted from G. This can be thought of as a deadline set by Alice at day $i = 16$. Let G' be the resulting subgraph. When Bob reaches v_{16}, he cannot procrastinate anymore but must wash the car to get a reward. The perceived net cost is $\zeta_{G'}(v_{16}) = 16/50 - \beta = -1/75$. Because this is less than 0, he washes the car. This makes G' a motivating subgraph. It is interesting to note that no reward configuration in G is motivating for a budget of $b < (m/50)/\beta$. This is because no matter how much reward is placed at any of the nodes, Bob prefers to procrastinate until the very last day.

To illustrate the power of reward configurations, we will consider a second scenario. Suppose that Alice offers Bob a new job. If he first washes her car, which by now incurs a cost of 1, and afterwards also mows her lawn, which has a cost of 6, he receives a reward of 10. What Bob is unaware of is that Alice does not care about the lawn. Instead, she tries to get Bob to wash the car for free. We model this project with a task graph G consisting of a path from s to t via the intermediate node v and another path from v to t via w. The edge (s, v) corresponds to the car wash and has a cost of 1. Furthermore, (v, w) corresponds to mowing of the lawn and has a cost of 6. The edges (v, t) and (w, t) are of cost 0. Assuming that $\beta = 1/3$, there is a reward configuration r for which Bob will wash the car but will not claim a reward. Suppose Alice sets $r(w) = 10$. In this case, Bob traverses (s, v) with a minimum perceived net cost of $\zeta(s) = -1/3$ along the path s, v, w, t. When at v, Bob reevaluates the net cost for traversing (v, w) to $8/3$. In contrast, finishing the project right away along (v, t) has cost 0. As a result, Bob changes his plan and moves to t without collecting the reward, although he already washed the car.

3 Finding Motivating Subgraphs

In this Section, we assume that the project designer may only place a single reward at t. This way, no exploitative reward configurations are possible. We first argue that the problem of finding a motivating subgraph can be solved in polynomial-time if $\beta = 0$ or $\beta = 1$. Although this claim might seem intuitive, we will be able to generalize the idea to show the existence of an $(1+\sqrt{n})$-approximation algorithm for general $\beta \in (0,1)$ in Sect. 4.

Proposition 1. *If $\beta = 0$ or $\beta = 1$, it is possible to find a motivating subgraph in polynomial-time for arbitrary $r(t) \geq 0$.*

Proof. First, assume $\beta = 0$. Because the agent has no value for future rewards, it must walk along a path of cost 0. Otherwise, it would abandon once it encounters an edge of positive cost. If such a path exists, it itself is a motivating subgraph. Conversely, if no such path exists, no subgraph can be motivating. Next, assume $\beta = 1$. In this case, the agent behaves time-consistent and follows a cheapest path from s to t. Therefore, G contains a motivating subgraph if and only if there is a path from s to t with a total edge cost less or equal to $r(t)$. Any subgraph containing such a path is motivating. Clearly, if a motivating subgraph exists, it can be found efficiently in both scenarios, i.e. $\beta = 0$ and $\beta = 1$. □

Unfortunately, computing motivating subgraphs for general $\beta \in (0,1)$ is more challenging. We will give evidence for this in Theorem 1 by showing that the corresponding decision problem, which we name MOTIVATING SUBGRAPH (MS), is NP-complete for general $\beta \in (0,1)$.

Definition 1. *Given a task graph G, a reward $r(t) \geq 0$ and a bias factor $\beta \in [0,1]$, decide the existence of a motivating subgraph of G.*

To prove NP-completeness of MS, we must first show that MS is contained in NP. For this purpose we will argue that it can be decided in polynomial-time whether a task graph is motivating for a given reward configuration. Note that a naive approach that simply simulates the agents walk through G must fail as the agent might take more than one path whenever it is indifferent between two options. A possible solution that preserves polynomial-time bounds is presented in the following proposition.

Proposition 2. *For any task graph G, reward configuration r and bias factor $\beta \in [0,1]$, it can be decided in polynomial-time if G is motivating.*

Proof. We modify G in the following way. For each node v we calculate the lowest perceived net cost $\zeta_{G,r}(v)$. Next, we take a copy of G, say G', in which we remove all edges (v,w) for which $\zeta_{G,r}(v) < c_G(v,w) + \beta d_{G,r}(w)$ or $\zeta_{G,r}(v) > 0$. In other words, we remove all edges from G' that do not minimize the agent's perceived net cost or are not motivating. Let V' be the set of all nodes that can be reached from s in G'. Observe that V' contains exactly those nodes that might be visited by the agent in G. Clearly, G is motivating if and only if the agent can reach t from all nodes of V' via some path in G'. This condition can be checked in polynomial-time. □

To show NP-hardness, we will use a reduction from k DISJOINT CONNECTING PATHS (k-DCP), which is defined as follows [3]:

Definition 2. *Given a graph H and k disjoint node pairs $(s_1, t_1), \dots, (s_k, t_k)$, decide if H contains k mutually node-disjoint paths, one connecting every s_i to the corresponding t_i.*

Lynch showed that k-DCP is NP-complete if H is undirected [8]. A simple modification of Lynch's reduction, which can be found in the full version of this paper, implies that k-DCP is also NP-complete if H is directed and acyclic.

Before we finally tackle Theorem 1, we want to draw attention to a useful price structure that is common to all reductions presented in this paper. Imagine a directed path along $k+1$ edges, such that the i-th edge has a cost of $(1-\beta)^{k+1-i}$. According to the following Lemma, the agent's perceived cost for following the path to its end is 1 at every node except for the last.

Lemma 1. *For every positive integer k and bias factor $\beta \in [0, 1]$ it holds that:*

$$(1-\beta)^k + \beta\left(\sum_{i=0}^{k-1}(1-\beta)^i\right) = 1.$$

The proof of Lemma 1 can be found in the full version of this paper. We are now ready to show NP-completeness of MS.

Theorem 1. *MS is NP-complete for any bias factor $\beta \in (0, 1)$.*

Proof. By Proposition 2, any motivating subgraph G' serves as a certificate for a "yes"-instance of MS. Consequently, MS is in NP. To complete the proof, we will establishes NP-hardness via a polynomial reduction from k-DCP.

Consider an instance $\mathcal{I}$ of k-DCP consisting of a directed acyclic graph H and k disjoint node pairs $(s_1, t_1), \dots, (s_k, t_k)$. We will embed H into the task graph G of an MS instance $\mathcal{J}$ such that G has a motivating subgraph if and only if H has k disjoint connecting paths. For this purpose we assume that the encoding length of $\beta \in (0, 1)$ is polynomial in that of $\mathcal{I}$ and set $r(t) = 1/\beta$. The task graph G, which is illustrated in Fig. 2, is constructed as follows:

To get from s to t, the agent must follow the so called *main path* along intermediate nodes $v_1, \dots, v_{k+3}$. The first $k+1$ edges of this main path each have a cost of $(1-\beta)^3 - \varepsilon$, with ε being a positive constant satisfying

$$\varepsilon < \min\left\{\beta\frac{1-\beta}{k+1}, \beta\frac{(1-\beta)^3}{1+\beta}\right\}.$$

The last three edges have a cost of $(1-\beta)^2$, $1-\beta$ and 1, respectively. To keep the agent motivated, we introduce k *shortcuts* that connect every v_i with $1 \le i \le k$ to t via the embedding of H. More formally, the i-th shortcut starts at v_i and is routed through a distinct node w_i via an edge of cost $(1-\beta)^2$. Node w_i is then connected to s_i via an edge of cost $(k+1-i)(1-\beta)/(k+1)$. Finally, t_i is connected to t via an edge of cost $i(1-\beta)/(k+1) + 1$. To keep Fig. 2 simple, the edges (t_i, t) are merged and their cost is depicted as two terms, namely $i(1-\beta)/(k+1)$ and $+1$. Note that the prices of (w_i, s_i) and (t_i, t) complement

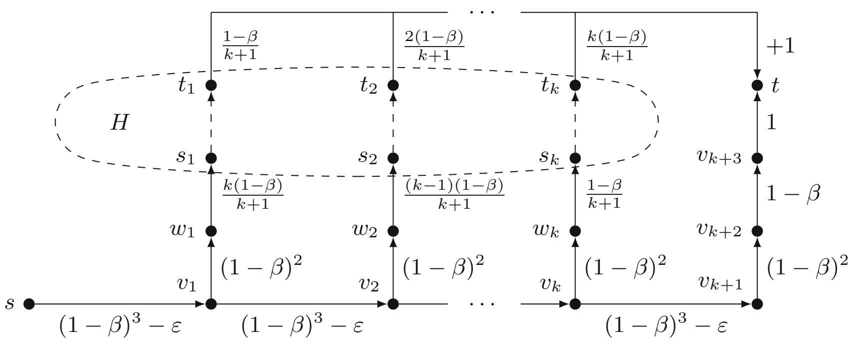


Fig. 2. Reduction from a general k-DCP instance: H

each other, i.e. they sum to $(1-\beta)+1$. The edges of H all have a cost of 0. The resulting graph G is acyclic and its encoding length polynomial in $\mathcal{I}$.

It remains to show, that $\mathcal{J}$ has a solution if and only if $\mathcal{I}$ has one. ($\Rightarrow$) First, suppose $\mathcal{I}$ has a solution, i.e. there exist k node-disjoint connecting paths. Let G' be a subgraph of G obtained by deleting all edges of H that are not part of one of these paths. Furthermore, let $s = v_0$ and assume the agent is located at v_i with $0 \leq i \leq k$. According to Lemma 1, the agent perceives a net cost of $-\varepsilon$ for taking the $(i+1)$-st shortcut or if $i = k$ for following the main path. In contrast, if $0 < i \leq k$, the perceived net cost of the i-th shortcut is 0. As a result, the agent follows the main path to v_{k+1} and then for lack of other options continues to t. We conclude that G' is a motivating subgraph of G.

($\Leftarrow$) Due to space constraints, we only sketch this direction of the proof. A thorough analysis can be found in the full version of this paper. Suppose $\mathcal{I}$ has no k node-disjoint connecting paths. Consequently, any subgraph of G must contain at least one shortcut i such that the cheapest path from v_i to t via w_i is different from $(1-\beta)^2 + (1-\beta) + 1$. We call such a shortcut *degenerate*. We distinguish between two scenarios: either the degenerate shortcuts are too expensive, and the agent looses motivation on the main path, or a degenerate shortcut is so cheap that the agent enters it. In the first case, the agent clearly abandons. In the latter case, the agent must traverse one of the edges (t_i, t) to reach t. However, since the price of (t_i, t) is greater than 1, a reward of $r(t) = 1/\beta$ is not sufficiently motivating for the agent to take this step. Again, it abandons. Therefore, no subgraph can be motivating. □

4 Approximating Reward Optimal Subgraphs

Considering that the decision problem MS is NP-hard, the next and arguably natural question is whether good approximation algorithms exist. Therefore, we restate MS as an optimization problem that we call MS-OPT.

Definition 3. *Given a task graph G and a bias factor $\beta \in (0, 1)$, determine the minimum reward $r(t)$ such that G contains a motivating subgraph.*

We will present two simple approximation algorithms: one that performs well for small values of β and one that leads to good solutions for large β. The algorithms return a reward $r(t)$ as well as a corresponding motivating subgraph G'. Combining both algorithms eventually yields a general approximation algorithm with a ratio of $(1 + \sqrt{n})$ for any $\beta \in (0, 1)$.

First, we assume that β is small. Because the agent is highly oblivious to the future, it is sensible to guide it along a path with minimal maximum edge cost. Paths with this property are called *minmax paths*. A minmax path can be computed easily in polynomial-time. For instance, starting with an empty subgraph, the edges of G can be inserted in non-decreasing order of cost until s and t become connected for the first time. Any path from s to t in the resulting subgraph is a minmax path. Our first algorithm, called MinmaxPathApprox, computes a minmax path P from s to t and returns a subgraph G' whose edges are that of P. Furthermore, $r(t)$ is chosen such that $\max\{\zeta_{G',r}(v) \mid v \in P\} = 0$. Clearly, this reward is sufficient to make G' motivating.

Proposition 3. MinmaxPathApprox *has an approximation ratio of* $1 + \beta n$.

Proof. Let c denote the maximum cost among the edges of the minmax path P computed by MinmaxPathApprox. By definition of P, the agent must encounter an edge of cost c or more in any subgraph that connects s with t. Thus the optimal reward is lower bounded by c/β. Conversely, the cost of every edge in P, of which there are at most $n - 1$, is c or less. This means that the reward returned by MinmaxPathApprox is upper bounded by $r(t) \leq c/\beta + (n-2)c \leq c/\beta + nc$. From this the desired approximation ratio of $1 + \beta n$ follows immediately. □

Next, suppose that β is large and the agent is hardly present-biased at all. Our second algorithm, called CheapestPathApprox, simply computes a path P of minimum cost from s to t and returns a subgraph G' containing the edges of P. Again, the algorithm chooses $r(t)$ such that $\max\{\zeta_{G',r}(v) \mid v \in P\} = 0$.

Proposition 4. CheapestPathApprox *has an approximation ratio of* $1/\beta$.

Proof. Let P be the path computed by CheapestPathApprox and c the total cost of P. At any node v of P the agent's perceived net cost is at most $d_{G',r}(v) - \beta r(t)$, which is less than $c - \beta r(t)$. The reward returned by CheapestPathApprox is therefore at most c/β. Conversely, when located at s, the agent perceives a cost of at least βc in any subgraph of G, including the optimal one. Consequently, a reward of c or more is required to motivate the agent. This establishes the approximation ratio of $1/\beta$. □

It is interesting to see how MinmaxPathApprox and CheapestPathApprox generalize the algorithmic ideas of Proposition 1. If we combine the two and use MinmaxPathApprox whenever $\beta \leq 1/\sqrt{n}$ and CheapestPathApprox otherwise, we obtain a general approximation algorithm called CombinedApprox. Propositions 3 and 4 directly imply the following result.

Theorem 2. CombinedApprox *has an approximation ratio of* $1+\sqrt{n}$.

Although the algorithmic techniques of CombinedApprox are simple, the following Theorem implies that asymptotically the approximation ratio is the best we can hope for in polynomial-time.

Theorem 3. *MS-OPT is NP-hard to approximate within a ratio less than* $\sqrt{n}/3$.

Proof. To establish hardness of approximation, we will use another reduction from k-DCP. Let $\mathcal{I}$ be an instance of k-DCP that consists of a directed acyclic graph H and k disjoint node pairs $(s_1,t_1),\dots,(s_k,t_k)$. Furthermore, let ϱ be an arbitrary positive integer. The best choice of ϱ will be determined later. We will construct an instance $\mathcal{J}$ of MS-OPT that consists of a task graph G and has the following two properties: (a) If $\mathcal{I}$ has a solution, then G has a subgraph that is motivating for a reward of $r(t)=1/\beta$. (b) If $\mathcal{I}$ does not have a solution, then no subgraph of G is motivating for a reward of $r(t)=\varrho/\beta$ or less. Consequently, any algorithm achieving an approximation ratio of ϱ or better must solve $\mathcal{I}$.

Unlike Theorem 1, the bias factor cannot be chosen arbitrarily anymore. Considering that Proposition 4 gives a $(1/\beta)$-approximation, β must be less than $1/\varrho$. For convenience, we set $\beta=1/(3\varrho+3)$. From a structural point of view, the task graph G consists of two units: the *embedding unit* and the *amplification unit*. The first unit contains an embedding of H, while the second unit amplifies approximation errors occurring in the embedding unit.

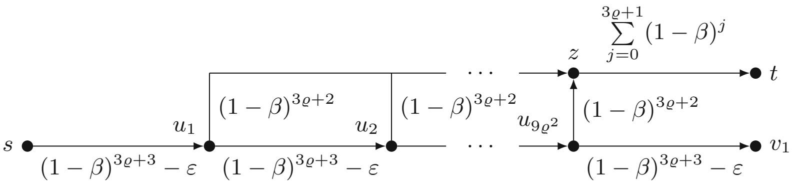
The overall structure of the embedding unit is similar to the task graph of Theorem 1. There exists a *main path* and k *shortcuts* that link to the embedding of H. However, there are some differences. First, the main path starts at the last node of the amplification unit $u_{9\varrho^2}$ and passes $k+3\varrho+3$ intermediate nodes $v_1,\dots,v_{k+3\varrho+3}$ before it ends in t. The first $k+1$ edges of the main path each have a cost of $(1-\beta)^{3\varrho+3}-\varepsilon$, where ε is a positive value satisfying

$$\varepsilon<\min\left\{\beta\frac{(1-\beta)^{3\varrho+1}}{k+1},\beta\frac{(1-\beta)^{3\varrho+3}}{1+\beta},\frac{1}{1+\varrho},(1-\beta)^{3\varrho+3}-\frac{1}{3}\right\}.$$

The remaining edges (v_i,v_{i+1}) of the main path, with $k<i\le k+3\varrho+3$ and $t=v_{k+3\varrho+3+1}$, have an increasing cost of $(1-\beta)^{k+3\varrho+3-i}$. Furthermore, the initial edge (v_i,w_i) of each shortcut has a cost of $(1-\beta)^{3\varrho+2}$, while the edges (w_i,s_i) and (t_i,t) have complementing prices of $(k+1-i)(1-\beta)^{3\varrho+1}/(k+1)$ and $i(1-\beta)^{3\varrho+1}/(k+1)+\sum_{j=0}^{3\varrho}(1-\beta)^j$. All edges of H are again free of charge. As a result, the cost of each shortcut sums up to $\sum_{j=0}^{3\varrho+2}(1-\beta)^j$.

The amplification unit, which is shown in Fig. 3, consists of an *amplification path* connecting s to $u_{9\varrho^2}$ along the intermediate nodes $u_1,\dots,u_{9\varrho^2-1}$. Each edge of the amplification path has a cost of $(1-\beta)^{3\varrho+3}-\varepsilon$. From every u_i there is also an edge of cost $(1-\beta)^{3\varrho+2}$ to a common node z. Node z is in turn connected to t via an edge of cost $\sum_{j=0}^{3\varrho+1}(1-\beta)^j$.

To conclude the proof, we must show that our construction satisfies properties (a) and (b) stated above. We start with (a) and assume that k node-disjoint



$\sum_{j=0}^{3\varrho+1}(1-\beta)^j$
z
t
v_1
s
u_1
u_2
$u_{9\varrho^2}$
$(1-\beta)^{3\varrho+2}$
$(1-\beta)^{3\varrho+3}-\varepsilon$

Fig. 3. Amplification unit

paths exist in H. Let G' be a subgraph of G obtained by deleting all edges of H that are not part of such a path. Furthermore, we set $r(t) = 1/\beta$ and $s = u_0$. When located at u_i with $0 \leq i \leq 9\varrho^2$, Lemma 1 suggests that the agent perceives a net cost of $-\varepsilon$ for traversing (u_i, u_{i+1}) and then following (u_{i+1}, z) or the first shortcut of the embedding unit if $i = 9\varrho^2$. Conversely, if $i > 0$, the agent evaluates the net cost of walking along (u_i, z) to 0. As a result, the agent follows the amplification path until it reaches v_1. From this point on it travels along the main path of the embedding unit until it eventually arrives at t for the same reasons given in Theorem 1. This means that G' is a motivating subgraph for a reward of $r(t) = 1/\beta$.

Due to space constraints, we refer to the full version of this paper for a thorough proof of statement (b). At this point we will confine ourselves to a brief sketch of the main ideas. Assume that $\mathcal{I}$ has no k node-disjoint paths. As argued before in Theorem 1, at least one *degenerate shortcut* must exist in any subgraph of G whose minimum cost is different from the target value of $\sum_{j=0}^{3\varrho+2}(1-\beta)^j$. As a result, two scenarios are conceivable. First, the perceived cost of some shortcut is so low that the agent diverts from the main path. However, in this case the agent must pass one of the edges (t_i, t), whose cost is greater than ϱ, to reach t. Consequently, no reward $r(t) \leq \varrho/\beta$ can be motivating. If the first case does not apply, one can argue that the perceived net cost of the main path at $u_{9\varrho^2}$ is greater than $1 - \beta r(t)$. To prevent the agent from moving to z, which results in cost greater than ϱ at (z, t), all edges (u_i, z) must be removed from the amplification unit. However, in this case the agent perceives a net cost greater than $\varrho - \beta r(t)$ at s. Again, no reward $r(t) \leq \varrho/\beta$ is sufficiently motivating. If ϱ is the number of nodes in H, then our lower bound on the approximability of MS-OPT converges to $\sqrt{n}/3$ as the size of H increases. □

5 Motivation Through Intermediate Rewards

In this section, we study the complexity of motivating agents through the strategic placement of rewards. In this scenario, the task graph must not be pruned. The goal is to minimize the total value of the rewards along the agent's walk from s to t. Similar to the previous setting of Sects. 3 and 4, a motivating reward configuration within a given budget b can be computed in polynomial-time if $\beta = 0$ or $\beta = 1$.

Proposition 5. *A motivating reward configuration within budget b can be computed in polynomial-time for $\beta = 0$ or $\beta = 1$.*

Proof. First, suppose that $\beta = 0$. In this case, the agent does not care for any future rewards and only traverses edges of cost 0. Let V' be the set of nodes that can be reached from s for cost 0. Note that V' contains exactly those nodes that might be visited by the agent independent of the specific reward configuration. As a result, G has a motivating reward configuration if and only if t can be reached from every node of V' for a cost of 0. Because no rewards need to be placed in this scenario, the budget constraint is always satisfied. Next, assume that $\beta = 1$. In this case the agent is time-consistent. Let c be the cost of a cheapest path from s to t. Setting $r(t) = c$ yields a motivating and also optimal reward configuration. The required budget is c. Clearly both cases, $\beta = 0$ and $\beta = 1$ can be solved in polynomial time. □

As before, the problem becomes much harder for general $\beta \in (0, 1)$. In particular, the corresponding decision problem MOTIVATING REWARD CONFIGURATION (MRC), which we define below, is NP-hard.

Definition 4. *Given a task graph G, a budget b and a bias factor $\beta \in [0, 1]$, decide the existence of a motivating reward configuration r such that the total reward collected on any walk of the agent is at most b.*

The following proposition establishes membership of MRC in NP.

Proposition 6. *For any task graph G, reward configuration r and bias factor $\beta \in [0, 1]$, it is possible to decide in polynomial-time if r is motivating within a given budget b.*

A proof of Proposition 6 can be found in the full version of this paper. To show NP-hardness of MRC, we will use a reduction form SET PACKING (SP), cf. [3]. For convenience, the definition of SP is stated below.

Definition 5. *Given a collection of finite sets $S_1, \ldots, S_\ell$ and an integer $k \leq \ell$, decide if at least k of these sets are mutually disjoint.*

We are now ready to prove NP-completeness of MRC. Note that the problem remains hard even if the budget is 0.

Theorem 4. *MRC is NP-complete for any bias factor $\beta \in (0, 1)$, even if $b = 0$.*

Proof. By Proposition 6, we can use any motivating reward configuration r within budge b as certificate for a "yes"-instance of MRC. This establishes membership of MRC in NP. To prove NP-hardness we will present a polynomial-time reduction from SP to MRC. We focus on the case that $b = 0$. A modified reduction for budgets $b > 0$ can be found in the full version of this paper.

Let $\mathcal{I}$ be an instance of SP consisting of finite sets $S_1, \ldots, S_\ell$ and an integer $k \leq \ell$. We start by constructing an MRC instance $\mathcal{J}$ that has a motivating reward configuration within a budget of $b = 0$ if and only if $\mathcal{I}$ has a solution.

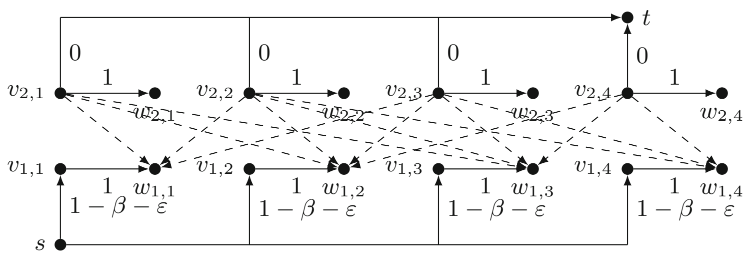


Fig. 4. Reduction from the SP instance: $S_1 = \{a, c, d\}$, $S_2 = \{a, b\}$, $S_3 = \{b, c, e\}$, $S_4 = \{b, e\}$ and $k = 2$

Figure 4 depicts the task graph G for a small sample instance of SP. In general, G consists of a source s, a target t and $1 \leq i \leq k$ levels of nodes $v_{i,j}$ with $1 \leq j \leq \ell$. For every $v_{i,j}$ with $i < k$ there is a so called *upward edge* to every node $v_{i+1,j'}$ on the next level. To maintain readability, upward edges are omitted in Fig. 4. In addition to the upward edges, there is an edge from s to every node $v_{1,j}$ on the bottom level and an edge towards t from every node $v_{k,j}$ on the top level. The idea behind this construction is that the agent walks along the upward edges from s to t in such a way that the nodes $v_{1,j}, \ldots, v_{k,j'}$ on its path correspond to a collection of k mutually disjoint sets $S_j, \ldots, S_{j'}$. The cost of the initial edges $(s, v_{1,j})$ and all upward edges $(v_{i,j}, v_{i+1,j'})$ is $1-\beta-\varepsilon$. Note that $\beta \in (0,1)$ might be an arbitrary value with an encoding length that is polynomial in that of $\mathcal{I}$. Moreover ε is a positive value satisfying

$$\varepsilon < \min\left\{\frac{(1-\beta)^2}{k}, \frac{\beta-\beta^2}{k-1+\beta}\right\}.$$

The cost of the edges $(v_{k,j}, t)$ is 0.

In order to motivate the agent, we add *shortcuts* to G that connect every $v_{i,j}$ to t via an intermediate node $w_{i,j}$. The first edge $(v_{i,j}, w_{i,j})$ has cost 1 and the second edge $(w_{i,j}, t)$ has cost 0. In Fig. 4 the second edges are omitted for the sake of readability. Note that a reward of value less than $1/\beta$ can be placed on $w_{i,j}$ without the agent claiming it. Furthermore, if the reward is at least $(1-\varepsilon)/\beta$, all edges $(v_{i-1,j'}, v_{i,j})$, or $(s, v_{i,j})$ if $i = 1$, become motivating.

We finish our construction by connecting each node $v_{i,j}$ with all nodes $w_{i',j'}$ for which $i' < i$ and $S_j \cap S_{j'} \neq \emptyset$ via a *downward path*. Each downward path consists of two edges: the first one is of cost 0 and the second one is of cost $(1-\beta-k\varepsilon)/(\beta-\beta^2)$. In Fig. 4, downward paths are drawn as single dashed edges. The idea behind these paths is to enforce the disjointness constraint of $\mathcal{I}$. In the next paragraph we will address this in more detail. But first note that G is an acyclic graph that is polynomial in the size of $\mathcal{I}$. It remains to show that $\mathcal{J}$ has a solution if and only if $\mathcal{I}$ has one.

Due to space constraints, we refer to the full version of this paper for a detailed proof. To offer some more insight, we briefly present a sketch of the main ideas. We first observe that the agent cannot enter any shortcut or downward path on its way from s to t. The reason for this is that a positive reward must be placed onto such a path for the agent to enter it. However, once the agent enters a shortcut or downward path it either collects the reward or abandons. In both cases the given reward configuration is not motivating for a budget of 0. Consequently, the agent must climb from level to level until it reaches t. As a motivation, rewards need to be placed on selected nodes $w_{i,j}$. If the reward is chosen correctly, for instance $r(w_{i,j}) = (1-\varepsilon)/\beta$, this is sufficiently motivating for the agent to move from any node on level $i-1$ to $v_{i,j}$, but not motivating enough for the agent to enter the shortcut from $v_{i,j}$ to t. Next, assume that the agent is located at some node $v_{i,j}$. To prevent it from taking the downward path, no substantial rewards may be placed on any node $w_{i',j'}$ with $i' < i$ and $S_j \cap S_{j'} \neq \emptyset$. By construction of G such a reward configuration is possible if and only if $\mathcal{I}$ has a feasible solution. □

Finally, we look at the optimization variant of MRC called MRC-OPT.

Definition 6. *Given a task graph G and a bias factor $\beta \in (0,1)$, determine the infimum of all budgets b for which there exists a reward configuration r such that the total reward collected on any of the agent's walks is at most b.*

The fact that MRC is NP-complete for $b = 0$ immediately implies that MRC-OPT does not permit any efficient approximation algorithm unless P = NP.

Corollary 1. *MRC-OPT is NP-hard to approximate within any ratio greater or equal to* 1.

References

1. Akerlof, G.A.: Procrastination and obedience. Am. Econ. Rev. **81**, 1–19 (1991)
2. Bryan, G., Karlan, D., Nelson, S.: Commitment devices. Ann. Rev. Econ. **2**, 671–698 (2010)
3. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., New York (1979)
4. Gravin, N., Immorlica, N., Lucier, B., Pountourakis, E.: Procrastination with variable present bias. In: 17th ACM Conference on Economics and Computation, pp. 361–361. ACM, New York (2016)
5. Kleinberg, J., Oren, S.: Time-inconsistent planning: a computational problem in behavioral economics. In: 15th ACM Conference on Economics and Computation, pp. 547–564. ACM, New York (2014)
6. Kleinberg, J., Sigal, O., Manish, R.: Planning problems for sophisticated agents with present bias. In: 17th ACM Conference on Economics and Computation, pp. 343–360. ACM, New York (2016)
7. Laibson, D.: Golden eggs and hyperbolic discounting. Q. J. Econ. **112**, 443–477 (1997)
8. Lynch, J.F.: The equivalence of theorem proving and the interconnection problem. SIGDA Newsl. **5**, 31–36 (1975)
9. O'Donoghue, T., Rabin, M.: Doing it now or later. Am. Econ. Rev. **89**, 103–124 (1999)

10. O'Donoghue, T., Rabin, M.: Incentives and self control. In: Advances in Economics and Econometrics: Theory and Application 9th World Congress, vol. 2, pp. 215–245. Cambridge University Press, Cambridge (2006)
11. O'Donoghue, T., Rabin, M.: Procrastination on long-term projects. J. Econ. Behav. Organ. **66**, 161–175 (2008)
12. Tang, P., Teng, Y., Wang, Z., Xiao, S., Xu, Y.: Computational issues in time-inconsistent planning (2015)