On the Value of Penalties in Time-Inconsistent Planning*†

Susanne Albers[1] and Dennis Kraft[2]

1 **Department of Computer Science, Technical University of Munich, Munich, Germany**
albers@in.tum.de
2 **Department of Computer Science, Technical University of Munich, Munich, Germany**
kraftd@in.tum.de

Abstract

People tend to behave inconsistently over time due to an inherent present bias. As this may impair performance, social and economic settings need to be adapted accordingly. Common tools to reduce the impact of time-inconsistent behavior are penalties and prohibition. Such tools are called commitment devices. In recent work Kleinberg and Oren [5] connect the design of a prohibition-based commitment device to a combinatorial problem in which edges are removed from a task graph G with n nodes. However, this problem is NP-hard to approximate within a ratio less than $\sqrt{n}/3$ [2]. To address this issue, we propose a penalty-based commitment device that does not delete edges, but raises their cost. The benefits of our approach are twofold. On the conceptual side, we show that penalties are up to $1/\beta$ times more efficient than prohibition, where $\beta \in (0, 1]$ parameterizes the present bias. On the computational side, we improve approximability by presenting a 2-approximation algorithm for allocating penalties. To complement this result, we prove that optimal penalties are NP-hard to approximate within a ratio of 1.08192.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases approximation algorithms, behavioral economics, commitment devices, computational complexity, time-inconsistent preferences

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.10

1 Introduction

Most people make long term plans. They intend to eat healthy, save money, prepare for exams, exercise regularly and so on. Curiously, the same people often change their plans at a later point in time. They indulge in fast food, squander their money, fail to study and skip workouts. Although change may be necessary due to unforeseen events, people often change their plans even if the circumstances stay the same. This type of *time-inconsistent behavior* is a well-known phenomenon in behavioral economics and might impair a person's performance in social or economic domains [1, 8].

A sensible explanation for time-inconsistent behavior is that people are *present biased* and assign disproportionately greater value to the present than to the future. Consider, for instance, a scenario in which a student named Alice attends a course over several weeks. To pass the course, Alice either needs to solve a homework exercise each week or give a

* The full version can be found at arXiv:1702.01677 [cs.DS], https://arxiv.org/abs/1702.01677.
† Work supported by the ERC, Grant Agreement No. 691672.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 10; pp. 10:1–10:12


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

presentation once. The presentation incurs a onetime effort of 3, whereas each homework exercise incurs an effort of 1. Furthermore, we assume that Alice immediately fails if she misses a homework assignment before she has given a presentation. If the course lasts for more than 3 weeks, she clearly minimizes her effort by giving a presentation in the first week. Paradoxically, if Alice is present biased, she may solve all homework exercises instead. The reason for this is the following:

Suppose Alice perceives present effort accurately, but discounts future effort by a factor of $\beta = 1/3$. In the first week Alice must decide between solving the homework exercise or giving a presentation. Remember that she automatically fails if she does neither of the two. Clearly, the homework incurs less immediate effort than the presentation. Furthermore, Alice can still give a presentation later on. Her perceived effort for doing the homework this week and giving the presentation the week after is $1 + \beta 3 = 2$. To Alice this plan appears more convenient than giving the presentation right away. Consequently, she does the homework. However, come next week she changes this plan and postpones the presentation once more. Her reasoning is the same as in the first week. Due to her time-inconsistent behavior, Alice continues to postpone the presentation and ends up doing all the homework assignments.

Previous Work. Time-inconsistent behavior has been studied extensively in behavioral economics. For an introduction to the topic refer for example to [1]. Alice's scenario demonstrates how time-inconsistency arises whenever people are present biased. Alice evaluates her preferences based on a well-established discounting model called *quasi-hyperbolic-discounting* [7]. As her story shows, quasi-hyperbolic-discounting tempts people to make poor decisions. To prevent poor decisions, social and economic settings need to be adapted accordingly. Depending on the domain, such adaptations might be implemented by governments, companies, teachers or people themselves. We call these entities *designers* and their motivation can be benevolent or self-serving. In either case, the designer's objective is to commit people to a certain goal. Their tools are called *commitment devices* and may include rewards, penalty fees and strict prohibition [3, 9].

Until recently, the study of time-inconsistent behavior lacked a unifying and expressive framework. However, groundbreaking work by Kleinberg and Oren closed this gap by reducing the behavior of a quasi-hyperbolic-discounting person to a simple planning problem in task graphs [5]. Their framework has helped to identify various structural properties of social and economic settings that affect the performance of present biased individuals [5, 10]. It has also been extended to people whose present bias varies over time [4] as well as people who are aware of their present bias and act accordingly [6]. We will formally introduce the framework in Section 2. A significant part of Kleinberg and Oren's work is concerned with the study of a simple yet powerful commitment device based on prohibition [5]. In particular, they demonstrate how performance can be improved by removing a strategically chosen set of edges from the task graph. The drawback of their approach is its computational complexity. As it turns out, an optimal commitment device is NP-hard to approximate within a ratio less than $\sqrt{n}/3$, where n denotes the number nodes in the task graph [2]. Currently, the only known polynomial-time algorithms achieve approximation ratios of $1/\beta$ and $1 + \beta/n$, which in combination yields a $1 + \sqrt{n}$ approximation [2]. It should be mentioned that Kleinberg and Oren's framework has also been used to analyze reward-based commitment devices [2, 10]. Unfortunately, the computational complexity of such commitment devices does not permit a polynomial-time approximation within a finite ratio unless P = NP [2].

Our Contribution. To circumvent the theoretical limitations mentioned above, we propose a natural generalization of Kleinberg and Oren's work. Instead of prohibition, our commitment

device is based on penalty fees, a standard tool in the economic literature [3, 9]. This means that the designer is free to raise the cost of arbitrary edges in the task graph. We call such an assignment of penalties a *cost configuration*. The designer's objective is to construct cost configurations that are as efficient as possible.

In Section 3 we conduct a quantitative comparison between the efficiency of penalty fees and prohibition. Assuming that optimal commitment devices can be computed, we show that penalties are strictly more powerful than prohibition. In particular, we show that penalties may outperform prohibition by a factor of $1/\beta$ where β parameterizes the present bias. This result is tight. In Section 4 we investigate the computational complexity of our commitment device. Using a reduction from 3-SAT, we argue that the construction of an efficient cost configuration is NP-hard when posed as a decision problem. A generalization of this reduction proves NP-hardness for approximations within a ratio of 1.08192. Unless P = NP, this dismisses the existence of a polynomial-time approximation scheme. While analyzing the complexity of our commitment device we also point to a remarkable structural property. More specifically, we show that every cost configuration admits another cost configuration of comparable efficiency that assigns its cost entirely along a single path. Assuming that the path is known in advance, we provide an algorithm for constructing such a cost configuration in polynomial-time. This result is important for the design of exact algorithms as it reduces the search space to the set of paths through the task graph. Finally, Section 5 introduces a 2-approximation algorithm for our commitment device. This is our main result and a considerable improvement to the $\sqrt{n}/3$ approximability of the prohibition-based commitment device [2]. It is interesting to note that the $1/\beta$ approximation algorithm of [2] can also be applied to penalty fees. As a result, our penalty-based approach is particularly interesting for highly present biased agents with $\beta < 1/2$.

2 The Formal Framework

In the following, we introduce Kleinberg and Oren's framework [5]. Let $G = (V, E)$ be a directed acyclic graph with n nodes that models a given long-term project. The edges of G correspond to the tasks of the project and the nodes represent the states. In particular, there exists a start state s and a target state t. Each path from s to t corresponds to a valid sequence of tasks to complete the project. The effort of a specific task is captured by a non-negative cost $c(e)$ assigned to the associated edge e.

To complete the project, an agent with a *present bias* of $\beta \in (0, 1]$ incrementally constructs a path from s to t as follows: At any node v different from t, the agent evaluates her *lowest perceived cost*. For this purpose she considers all paths P leading from v to t. However, she only anticipates the cost of the first edge of P correctly; all other edges of P are discounted by her present bias. More formally, let $d(w)$ denote the cost of a cheapest path from node w to t. The agent's lowest perceived cost at v is defined as $\zeta(v) = \min\{c(v, w) + \beta d(w) \mid (v, w) \in E\}$. We assume that she only traverses edges (v, w) that minimize her anticipated cost, i.e. edges for which $c(v, w) + \beta d(w) = \zeta(v)$. Ties are broken arbitrarily. For convenience, we define the perceived cost of (v, w) as $\eta(v, w) = c(v, w) + \beta d(w)$. The agent is motivated by an intrinsic or extrinsic reward r collected at t. As she receives this reward in the future, she perceives its value as βr at each node different from t. When located at v, she compares her lowest perceived cost to the anticipated reward and continues moving if and only if $\zeta(v) \leq \beta r$. Otherwise, if $\zeta(v) > \beta r$, we assume she abandons the project. We call G *motivating* if she does not abandon while constructing her path from s to t. Note that in some graphs the agent can take several paths from s to t due to ties between incident edges. In this case, G is considered motivating if she does not abandon on any of these paths.

For the sake of a clear presentation, we will assume throughout this work that each node of G is located on a path from s to t. This assumption is sensible for the following reason: Clearly, the agent can only visit nodes that are reachable from s. Furthermore, she is not willing to enter nodes that do not lead to the reward. Consequently, only nodes that are on a path from s to t are relevant to her behavior. Note that all nodes that do not satisfy this property can be removed from G in a simple preprocessing step.

To illustrate the model, we revisit Alice's scenario from Section 1. Assume that the course takes m weeks. We represent each week i by a distinct node v_i and set $s = v_1$. Furthermore, we introduce a target node t that marks the passing of the course. Each week $i < m$ Alice can either give a presentation or proceed with the homework. We model the first case by an edge (v_i, t) of cost 3 and the latter case by an edge (v_i, v_{i+1}) of cost 1. In the last week, i.e. $i = m$, Alice's only sensible choice is to do the homework. Therefore, edge (v_m, t) is of cost 1. Recall that Alice's present bias is $\beta = 1/3$. Moreover, assume that her intrinsic reward for passing is $r = 6$. For $i < m$ her perceived cost of the edges (v_i, t) is $\eta(v_i, t) = c(v_i, t) = 3$. As this is less than her perceived reward, which is $\beta 6 = 2$, she is never motivated to give a presentation right away. However, her perceived cost of the edges (v_i, v_{i+1}) is at most $\eta(v_i, v_{i+1}) \leq c(v_i, v_{i+1}) + \beta c(v_{i+1}, t) \leq 2$. This matches her perceived reward. As a result, she walks from v_1 to v_m along the edge (v_i, v_{i+1}). Once she reaches v_m she traverses the only remaining edge for a perceived cost of $\eta(v_m, t) = c(v_m, t) = 1$ and passes the course. This matches our analysis from Section 1.

3 Prohibition versus Penalties

In this section we demonstrate how the *designer* can modify a given project to help the agent reach t. For this purpose, the designer may have several commitment devices at her disposal. A straightforward approach is to increase the reward that the agent collects at t. Although this may keep the agent from abandoning the project prematurely, it has no influence on the path taken by the agent. Furthermore, increasing the reward may be costly for the designer. As a result, the designer has two conflicting objectives. On the one hand, she must ensure that the agent reaches t. On the other hand, she needs to minimize the resources spent. To deal with this dilemma, Kleinberg and Oren allow the designer to prohibit a strategically chosen set of tasks [5]. This commitment device is readily implemented in their framework. In fact, it is sufficient to remove all edges of prohibited tasks. The result is a subgraph G' that may significantly reduce the reward required to motivate the agent. Unfortunately, an optimal subgraph G' is NP-hard to approximate within a ratio less than $\sqrt{n}/3$ [2].

To circumvent this theoretical limitation, we propose a different approach. Instead of prohibiting certain tasks we allow the designer to charge penalty fees. Such fees could be implemented in several ways; for instance in the form of donations to charity. Our only assumption is that the designer does not benefit from the fees, i.e. there is no incentive to maximize the fees payed by the agent. Similar to the prohibition-based commitment device, our commitment device is readily implemented in Kleinberg and Oren's framework. The designer simply assigns a positive extra cost $\tilde{c}(e)$ to the desired edges e. The new cost of e is equal to $c(e) + \tilde{c}(e)$. We call $\tilde{c}$ a *cost configuration*. Applying a cost configuration to G yields a new task graph with increased edge cost. All concepts of the original framework carry over immediately. Sometimes it will be necessary to compare different commitment devices with each other. To clarify which commitment device we are talking about, we use the following notation whenever necessary: If we consider a subgraph G', we write $d_{G'}$, $\eta_{G'}$ and $\zeta_{G'}$. Similarly, if we consider a cost configuration $\tilde{c}$, we write $d_{\tilde{c}}$, $\eta_{\tilde{c}}$ and $\zeta_{\tilde{c}}$. Moreover, we denote the trivial cost configuration, i.e. the one that assigns no extra cost, by $\tilde{0}$.

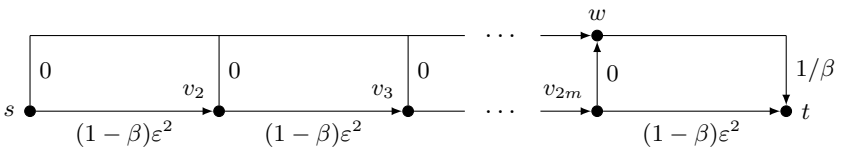

Figure 1 Graph maximizing the ratio between the efficiency of subgraphs and cost configurations.

It is interesting to think of penalty fees as a natural generalization of prohibition. This becomes particularly apparent in the context of Kleinberg and Oren's framework as we can recreate the properties of any subgraph G' by a cost configuration $\tilde{c}$. For this purpose, it is sufficient to assign an extra cost of $\tilde{c}(e) = r + 1$ to any edge e not contained in G'. As a result, the agent's perceived cost of paths along e certainly exceeds her perceived reward. However, this means that e is irrelevant to the agent's planning and could be deleted from G altogether. Consequently, penalties are at least as powerful as prohibition. But how much more efficient are penalties in the best case? As the following theorem suggests, cost configurations may outperform subgraphs by a factor of almost $1/\beta$.

▶ **Theorem 1.** *The ratio between the minimum reward r that admits a motivating subgraph and the reward q of a motivating cost configuration is at most $1/\beta$. This bound is tight.*

Proof. To see that $r/q \leq 1/\beta$, let G be an arbitrary task graph and consider a subgraph G' whose only edges are those of a cheapest path P from s to t. Recall that $d(s)$ denotes the cost of P. In G' the agent's only choice is to follow P. Because her perceived cost is a discounted version of the actual cost, she never perceives a cost greater than $d(s)$ in G'. Consequently, $d(s)/\beta$ is an upper bound on r. Next, consider an arbitrary cost configuration $\tilde{c}$. As $\tilde{c}$ only increases edge cost, the agent's lowest perceived cost at s is at least $\beta d(s)$. We conclude that q must be at least $d(s)$ to be motivating. This yields the desired ratio.

It remains to show the tightness of the result. For this purpose, we construct a task graph G such that: (a) The minimum reward that admits a motivating subgraph is $1/\beta^2$. (b) There exists a cost configuration that is motivating for a reward of $(1+\varepsilon)/\beta$, where ε is a positive value strictly less than 1. Our construction is a modified version of Alice's task graph. Let $m = \lceil \beta^{-2}(1-\beta)^{-1}\varepsilon^{-2} \rceil$ and assume that G contains a path $v_1, \ldots, v_{2m+1}$ whose edges are all of cost $(1-\beta)\varepsilon^2$. We call this the *main path* and set $s = v_1$ and $t = v_{2m+1}$. In addition to the main path, each v_i with $i \leq 2m$ has a *shortcut* to t via a common node w. The edges (v_i, w) are free, whereas (w, t) is of cost $1/\beta$. Figure 1 illustrates the structure of G. Note that the drawing merges some of the edges (v_i, w) for a concise representation.

We proceed to argue that G satisfies (a). For the sake of contradiction, assume the existence of a subgraph G' that is motivating for a reward $r < 1/\beta^2$. In this case the agent must not take shortcuts as her perceived cost at w exceeds her perceived reward. Therefore, she must follow the main path. In particular, she must visit each node v_i on the first half of the path, i.e. $i \leq m+1$. At each of these nodes, her lowest perceived cost is realized along the edge (v_i, v_{i+1}). Essentially, there are two ways she can come up with this cost. First, she might plan to take a shortcut at a later point in time. As a result, we get $\eta_{G'}(v_i, v_{i+1}) \geq c(v_i, v_{i+1}) + \beta c(w, t) > 1$. Secondly, she might plan to stay on the main path. In this case she must traverse at least m edges, each of which contributes $\beta(1-\beta)\varepsilon^2$ or more to $\eta_{G'}(v_i, v_{i+1})$. Consequently, we get $\eta_{G'}(v_i, v_{i+1}) \geq m\beta(1-\beta)\varepsilon^2 \geq 1/\beta \geq 1$. Either way her perceived cost for taking the main path is at least 1. As this tempts her to take the shortcut at v_i, all of the first $m+1$ shortcuts must be interrupted in G'. This means she must walk along at least m edges of the main path before taking the first shortcut. As a

Algorithm 1: PathAndFence

Input: Task graph G, present bias β, path $P = v_1, \dots, v_m$, positive value ε
Output: Cost configuration $\tilde{c}$

```
1 c̃ ← 0̃;
2 for i from m − 1 to 1 do
3     foreach w ∈ {w′ | (v_i, w′) ∈ E} do
4         if w ≠ v_{i+1} then c̃(v_i, w) ← max{0, η_c̃(v_i, v_{i+1}) − η_c̃(v_i, w) + βε/(m − 2)};
5 return c̃;
```

result, her lowest perceived cost at v_1 is at least $\zeta_{G'}(v_1) \geq m\beta(1-\beta)\varepsilon^2 \geq 1/\beta$. This is a contradiction to the assumption that r is motivating.

Next we show how to construct a cost configuration $\tilde{c}$ that satisfies (b). For this purpose it is sufficient to add an extra cost of ε to all edges (v_i, w). To upper bound the agent's perceived cost of (v_i, v_{i+1}), assume she plans to take a shortcut in the next step, i.e. at v_{i+1}. For $i < 2m$ we get $\eta_{\tilde{c}}(v_i, v_{i+1}) \leq c(v_i, v_{i+1}) + \beta(\tilde{c}(v_{i+1}, w) + c(w, t)) = (1-\beta)\varepsilon^2 + \beta\varepsilon + 1 < 1 + \varepsilon$. In the special case of $i = 2m$, the inequality $\eta_{\tilde{c}}(v_i, v_{i+1}) < 1 + \varepsilon$ is still satisfied, this time via the direct edge (v_{2m}, t). In contrast, the agent's perceived cost of an immediate shortcut is $\eta_{\tilde{c}}(v_i, t) = \varepsilon + \beta c(w, t) = 1 + \varepsilon$ for all $i \leq 2m$. Therefore, she is never tempted to divert from the main path. Furthermore, a reward of $q = (1+\varepsilon)/\beta$ is sufficient to keep her motivated. ◀

4 Computing Motivating Cost Configurations

We now turn our attention to the computational aspects of designing efficient penalty fees. In this section, we assume that the agent's reward is fixed to some value $r > 0$. Our goal is to compute cost configurations that are motivating for r whenever they exist. Similar to the prohibition-based commitment device [2], this task is NP-hard whenever the agent is present biased, i.e. $\beta \neq 1$. We will prove this claim at the end of the section. But first, assume that we already have partial knowledge of the solution. More precisely, assume we know one of the paths the agent might take in a motivating cost configuration provided a motivating cost configuration exists. We call this path P. Based on P, Algorithm 1 constructs a cost configuration $\tilde{c}$ that is motivating for a slightly larger reward $r + \varepsilon$.

The basic idea of Algorithm 1 is simple. Starting with v_{m-1}, it considers all nodes v_i of P in reverse order. For each v_i it assigns an extra cost of $\max\{0, \eta_{\tilde{c}}(v_i, v_{i+1}) - \eta_{\tilde{c}}(v_i, w) + \beta\varepsilon/(m-2)\}$ to the edges (v_i, w) that leave P, i.e. edges different from (v_i, v_{i+1}). As a result, the agent's perceived cost of (v_i, w) is greater than that of (v_i, v_{i+1}) by at least $\beta\varepsilon/(m-2)$. Consequently, she has no incentive to divert from P at v_i. Since the algorithm runs in reverse order, extra cost assigned in iteration i has no effect on the agent's behavior at later nodes, i.e. nodes v_j with $j > i$. Figuratively speaking, the algorithm builds a fence of penalty fees along P preventing the agent from leaving P. For this reason, we call the algorithm PathAndFence. As the next proposition suggests, cost configurations of this particular fence structure can achieve almost the same efficiency as any other cost configuration. Due to space constraints, refer to the full version of this work for a proof.

▶ **Proposition 2.** *Let P be the agent's path from s to t with respect to a cost configuration $\tilde{c}^*$ that is motivating for a reward r.* PathAndFence *constructs a cost configuration $\tilde{c}$ that is motivating for a reward of $r + \varepsilon$, where ε is an arbitrary small but positive quantity.*

Proposition 2 has some interesting implications. The first one is of conceptual nature.

Note that PathAndFence constructs a cost configuration that never actually charges the agent any extra cost. This suggests the existence of an efficient penalty-based commitment device that does not require the designer to enforce penalties. The mere threat of repercussions appears to be sufficient. The second implication is computational. Clearly, PathAndFence runs in polynomial-time with respect to n. In particular, the number of iterations does not depend on the choice of ε. Consequently, PathAndFence can be combined with an exhaustive search algorithm that considers all paths from s to t to search for a motivating cost configuration. Although the number of such paths can be exponential in n, this approach still reduces the size of the search space considerably. Finally, it should be noted that a similar result for the prohibition-based commitment device is unlikely to exist. The reason is that subgraphs remain hard to approximate even if the agent's optimal path is known [2], indicating a favorable computational complexity for the design of penalty fees. Of course there is another potential source of hardness: the computation of P. To prove that this is a limiting factor, we introduce the decision problem MOTIVATING COST CONFIGURATION:

▶ **Definition 3** (MCC)**.** Given a task graph G, a reward $r > 0$ and a present bias $\beta \in (0, 1]$, decide the existence of a motivating cost configuration.

We propose a reduction from 3-SAT to show that MCC is NP-complete for arbitrary $\beta \in (0, 1)$. At a later point we will use the same reduction to establish a hardness of approximation result.

▶ **Theorem 4.** *MCC is NP-complete for any present bias $\beta \in (0, 1)$.*

Proof. According to [2], whether or not a given task graph is motivating for a fixed reward can be verified in polynomial-time. Of course, this remains valid if the edges are assigned extra cost. Consequently, any motivating cost configuration is a suitable certificate for a "yes"-instance of MCC. We conclude that MCC is in NP. In the following, we present a reduction from 3-SAT to show that MCC is also NP-hard. This establishes the theorem.

Let $\mathcal{I}$ be an arbitrary instance of 3-SAT consisting of ℓ clauses $c_1, \ldots, c_\ell$ over m variables $x_1, \ldots, x_m$. We construct a MCC instance $\mathcal{J}$ such that its task graph G admits a motivating cost configuration for a reward of $r = 1/\beta$ if and only if $\mathcal{I}$ has a satisfying variable assignment. Figure 2 depicts G for a small sample instance of $\mathcal{I}$. In general, G consists of a source s, a target t and five nodes $u_1, \ldots, u_5$. Depending on $\mathcal{I}$, G also contains some extra nodes. For each variable x_k, there are two *variable nodes* $w_{k,T}$ and $w_{k,F}$. The idea is to interpret x_k as true whenever the agent visits $w_{k,T}$ and as false whenever she visits $w_{k,F}$. As a result, the agent's walk through G yields a variable assignment τ. Furthermore, for each clause c_i there is a *literal node* $v_{i,j}$ corresponding to the j-th literal of c_i. Our goal is to construct G in such a way that every motivating cost configuration guides the agent along literal nodes that are satisfied with respect to τ.

All nodes $v_{i,j}$ and $w_{k,y}$ are connected via so-called *forward edges*. More specifically, for all $1 \leq i < \ell$ and $1 \leq j, j' \leq 3$ there is a forward edge from $v_{i,j}$ to $v_{i+1,j'}$. Similarly, there is a forward edge from $w_{k,y}$ to $w_{k+1,y'}$ for all $1 \leq k < m$ and $y, y' \in \{T, F\}$. We also have forward edges from s to each $v_{1,j}$, from each $v_{\ell,j}$ to u_1, from u_2 to each $w_{1,y}$ and from each $w_{m,y}$ to u_3. For the sake of readability, some forward edges are merged in Figure 2. The price of each forward edge is $(1-\beta)^3 - \varepsilon$, where the encoding length of β is assumed to be polynomial in $\mathcal{I}$. Furthermore, ε denotes a small but positive quantity such that

$$\varepsilon < \min\left\{(1-\beta)^2, \frac{\beta(1-\beta)^3}{1+\beta}, \frac{\beta(1-\beta)^2}{1+\beta}\right\}.$$

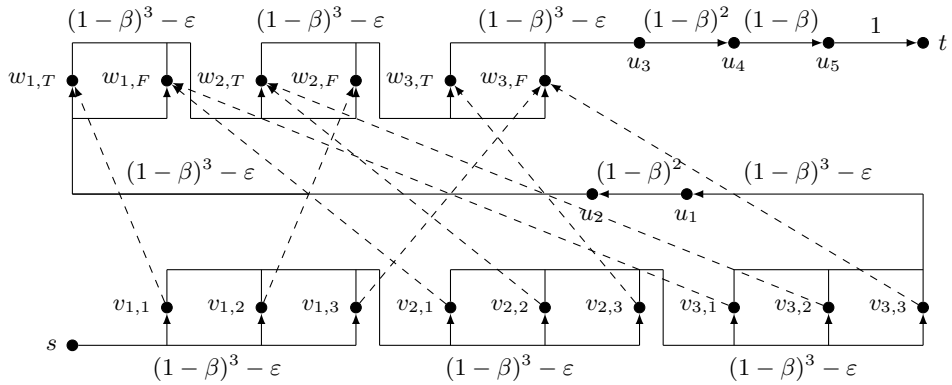

Figure 2 Reduction from the 3-SAT instance: $(\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$.

In addition to the forward edges, there are three types of *shortcuts*. The first type, which is depicted as dashed edges in Figure 2, connects each literal node $v_{i,j}$ to a distinct variable node via a single edge of cost $(1-\beta)^2$. If the j-th literal of c_i is equal to x_k, the shortcut goes to $w_{k,F}$. Otherwise, if the literal is negated, i.e. $\bar{x}_k$, the shortcut goes to $w_{k,T}$. The second type of shortcut goes from u_2 to t along a single edge of cost $2-\beta$. For clear representation, this shortcut is omitted in Figure 2. The third type of shortcut connects each variable node $w_{k,y}$ to t via a distinct intermediate node. The first edge is free while the second costs $2-\beta$. Again, shortcuts of this type are omitted in Figure 2 to keep the drawing simple. Finally, there are four more edges (u_1, u_2), (u_3, u_4), (u_4, u_5) and (u_5, t) of cost $(1-\beta)^2$, $(1-\beta)^2$, $1-\beta$ and 1 respectively. Note that G is acyclic and its encoding length is polynomial in $\mathcal{I}$.

To establish the theorem, we must show that $\mathcal{I}$ has a satisfying variable assignment if and only if $\mathcal{J}$ has a motivating cost configuration. A detailed argument is described in the full version of this work. At this point we only sketch the main ideas. For this purpose let $\tilde{c}$ be a cost configuration that is motivating for a reward of $1/\beta$ and let P be the agent's path through G with respect to $\tilde{c}$. Note that P cannot contain shortcuts of the second or third type as their edges are too expensive. Furthermore, P cannot contain a shortcut of the first type because the agent either perceives it as too expensive or is tempted to enter a shortcut of the third type immediately afterwards. As a result, P contains exactly one of the two nodes $w_{k,T}$ and $w_{k,F}$ for each variable x_k. Let $\tau : \{x_1, \dots, x_m\} \to \{T, F\}$ be the corresponding variable assignment. To keep the agent on P, $\tilde{c}$ must assign extra cost to all shortcuts that start at a variable node satisfied by τ. However, this raises the perceived cost of all paths via literal nodes not satisfied by τ to values that are not motivating. Consequently, P cannot contain such literal nodes. But P must contain exactly one literal node of each clause because P takes no shortcuts. This means that τ satisfies at least one literal in each clause and is therefore a feasible solution of $\mathcal{I}$. Conversely, whenever $\mathcal{I}$ has a feasible solution τ, we can construct a motivating cost configuration $\tilde{c}$ as follows: First, assign an appropriate extra cost, e.g. $(1-\beta)^2$, to the shortcuts of type three starting at the variable nodes $w_{k,\tau(x_k)}$. Secondly, block the forward edges into the variable nodes $w_{k,\tau(\bar{x}_k)}$ with high extra cost of e.g. 1. ◀

5 Approximating Motivating Cost Configurations

The previous section suggests that an optimal assignment of penalty fees is NP-hard to compute. This section therefore focuses on an optimization version of the problem. Our goal is to construct cost configurations that require the designer to raise the reward at t as little

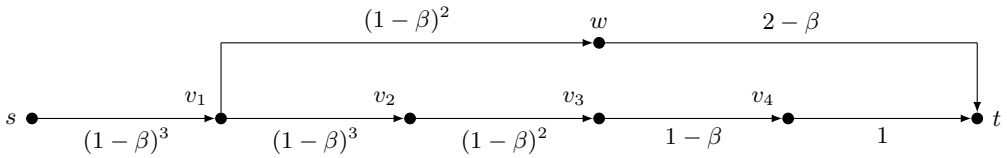

Figure 3 Task graph with no optimal cost configuration.

Algorithm 2: MinMaxPathApprox

Input: Task graph G, present bias β
Output: Cost configuration $\tilde{c}$

```
P ← minmax path from s to t with respect to η_{c̃0};
ϱ ← max{η_{c̃0}(e) | e ∈ P};
foreach v ∈ V \ {t} do
    ς(v) ← successor node of v on a cheapest path from v to t;
foreach (v, w) ∈ E do
    if (v, w) ∈ P ∨ (ς(v) = w ∧ v ∉ P) then c̃(v, w) ← 0;
    else if (v, w) ≠ P ∧ ς(v) ≠ w then c̃(v, w) ← 3ϱ/β;
    else
        P' ← v, ς(v), ς(ς(v)), ..., t;
        u ← first node of P' different from v that is also a node of P;
        e ← most expensive edge of P', between v and u;
        c̃(v, w) ← c(e);
return c̃;
```

as possible. However, before we provide a formal definition of the problem we should consider a curious technical detail; namely, not all task graphs admit an optimal cost configuration.

Consider, for instance, the task graph in Figure 3. At v_1 the agent is indifferent between the edges (v_1, v_2) and (v_1, w). In both cases her perceived cost is 1. If she chooses (v_1, w), she faces a perceived cost of $2 - \beta$ at w. Conversely, if she chooses (v_1, v_2), she perceives a cost of 1 at v_2, v_3 and v_4. Assuming that $\beta < 1$, (v_1, v_2) is the better choice. To break the tie between (v_1, w) and (v_1, v_2) we must place a positive extra cost of ε onto the upper path. However, when located at s the agent's perceived cost of the upper path is $1 + \beta\varepsilon$. In contrast, her perceived cost of the lower path is $1 + \beta(1-\beta)^3$. Assuming that $\varepsilon < (1-\beta)^3$, she prefers the upper path. Consequently, we can construct a cost configuration that is motivating for a reward arbitrarily close to $1/\beta$, but no cost configuration is motivating for a reward of exactly $1/\beta$. To account for the potential lack of an optimal solution, we compare our results to the infimum of all rewards that admit a motivating cost configuration. The optimization problem MCC-OPT is defined accordingly:

▶ **Definition 5** (MCC-OPT). Given a task graph G and a present bias $\beta \in (0, 1)$, determine the infimum of all rewards for which a motivating cost configuration exists.

We are now ready to introduce Algorithm 2. This algorithm enables us to construct cost configurations that approximate MCC-OPT within a factor of 2. At a high level, the algorithm proceeds in two phases. First, it computes a value ϱ such that ϱ/β is a lower bound for any reward that admits a motivating cost configuration. Secondly, it constructs a cost configuration $\tilde{c}$ that is motivating for a reward of $2\varrho/\beta$. This yields the promised approximation ratio of 2.

For a more detailed discussion of Algorithm 2 assume that each edge e is labeled with its perceived cost $\eta_{\tilde{0}}(e)$. Furthermore, let $\tilde{c}'$ be an arbitrary cost configuration and P' the agent's corresponding path from s to t. Our goal is to lower bound the minimum reward that is motivating for $\tilde{c}'$ by some value ϱ/β. For this purpose, it is instructive to observe that any motivating reward must be at least $\max\{\eta_{\tilde{c}'}(e) \mid e \in P'\}/\beta \geq \max\{\eta_{\tilde{0}}(e) \mid e \in P'\}/\beta$. Since P' can be an arbitrary path from s to t, we set

$$\varrho = \min\big\{\max\{\eta_{\tilde{0}}(e) \mid e \in P\} \;\big|\; P \text{ is a path from } s \text{ to } t\big\}.$$

In other words, ϱ is the maximum edge cost of a *minmax path* P from s to t with respect to $\eta_{\tilde{0}}$. Note that P can be computed in polynomial-time by adding the edges of G in non-decreasing order of perceived cost to an initially empty set E' until s and t become connected for the first time. Any path from s to t that only uses edges of E' is a suitable minmax path.

We continue with the construction of $\tilde{c}$. To facilitate this task, Algorithm 2 sets up a cheapest path successor relation ς. More precisely, it assigns a distinct successor node $\varsigma(v)$ to each $v \in V \setminus \{t\}$. The successor is chosen in such a way that $(v, \varsigma(v))$ is the initial edge of a cheapest path from v to t. Since we may assume that t is reachable from each node of G, all $v \neq t$ must have at least one suitable successor. By construction of ς, any path $P' = v, \varsigma(v), \varsigma(\varsigma(v)), \ldots, t$ is a cheapest path from v to t. We call P' the *ς-path* of v.

Once ς has been created, Algorithm 2 starts to assign an appropriate extra cost to all edges of G. The idea behind this assignment is to either keep the agent on P or guide her along a suitable ς-path. For this reason, we also call the algorithm MinMaxPathApprox. While iterating through the edges (v, w) of G the algorithm distinguishes between three types of edges: First, (v, w) might be an edge of P or an edge of a ς-path. In the latter case v must not be a node of P. Any (v, w) that satisfies these requirements is an edge we want the agent to traverse or use in her plans. Consequently, (v, w) is assigned no extra cost. Secondly, (v, w) might neither be an edge of P nor of a ς-path. Since we do not want the agent to traverse or plan along such an edge, the algorithm assigns an extra cost of $3\varrho/\beta$ to (v, w). This is sufficiently expensive for the agent to lose interest in (v, w) provided that the reward is $2\varrho/\beta$. Thirdly, (v, w) might not be an edge of P but of a ς-path such that v is a node of P. This is the most involved case. To find an appropriate cost for (v, w), the algorithm considers the ς-path P' of v. Let u be the first common node between P and P' that is different from v. Because P and P' both end in t, such a node must exist. Moreover, let e be the most expensive edge of P' between v and u. The algorithm assigns an extra cost of $c(e)$ to (v, w). As we will show in Theorem 6, this cost is either high enough to keep the agent on P or she travels to u along P' without encountering edges that are too expensive.

Clearly, Algorithm 2 can be implemented to run in polynomial-time with respect to the size of G. It remains to show that the algorithm returns a cost configuration $\tilde{c}$ that approximates MCC-OPT within a factor of 2.

▶ **Theorem 6.** MinMaxPathApprox *has an approximation ratio of* 2*.*

Proof. Recall that ϱ denotes the maximum perceived edge cost along the minmax path P. From the above description of MinMaxPathApprox, it should be evident that ϱ/β is a lower bound on the minimum motivating reward of any cost configuration. To prove the theorem, we need to show that the algorithm returns a cost configuration $\tilde{c}$ that is motivating for a reward of $2\varrho/\beta$.

As our first step we argue that the cost of a cheapest path from any node v to t with respect to $\tilde{c}$ is at most twice the cost of a cheapest path with respect to $\tilde{0}$. More formally, we prove that $d_{\tilde{c}}(v) \leq 2d_{\tilde{0}}(v)$. For this purpose let P' be the ς-path of v. By construction

of ς, P' is a cheapest path from v to t. It is crucial to observe that MinMaxApprox only assigns extra cost to an edge $(v', \varsigma(v'))$ of P' if v' is located on P. Consequently, there is at most one edge with extra cost between any two consecutive intersections of P and P'. Furthermore, this extra cost is equal to the cost of an edge on P' between v' and the next intersection of P and P'. Therefore, each edge of P' can contribute at most once to the total extra cost assigned to P'. This means that the price of P' with respect to $\tilde{c}$ is at most twice the price of P' with respect to $\tilde{0}$. Because the price of P' is an upper bound for $d_{\tilde{c}}(v)$, we have shown that $d_{\tilde{c}}(v) \leq 2d_{\tilde{0}}(v)$.

We proceed to investigate the agent's walk through G. Our goal is to show that her lowest perceived cost is at most 2ϱ at every node v on her way. This establishes the theorem. Our analysis is based on the following case distinction: First, assume that v is located on P. The immediate successor of v on P is denoted by w. Remember that $\tilde{c}$ assigns no extra cost to (v, w). Using the result from the previous paragraph, we get

$$\begin{aligned}\zeta_{\tilde{c}}(v) &\leq \eta_{\tilde{c}}(v,w) = c(v,w) + \beta d_{\tilde{c}}(w) \leq c(v,w) + \beta 2 d_{\tilde{0}}(w) \leq 2\big(c(v,w) + \beta d_{\tilde{0}}(w)\big) \\ &= 2\eta_{\tilde{0}}(v,w) \leq 2\varrho.\end{aligned}$$

The last inequality is valid by definition of ϱ.

Secondly, assume that v is not located on P and consider the last node v' on P the agent visited before v. Because she traversed $(v', \varsigma(v'))$ to get to v, we know that $\eta_{\tilde{c}}(v', \varsigma(v')) \leq 2\varrho$ and $d_{\tilde{c}}(\varsigma(v')) \leq 2\varrho/\beta$. We also know that she faces an extra cost of $3\varrho/\beta$ whenever she tries to leave the ς-path P' of v' before the next intersection of P and P'. Since she is not willing to pay this much, v must be located on P'. In particular, all paths from $\varsigma(v')$ to t either visit $\varsigma(v)$ or cross an edge that charges an extra cost of $3\varrho/\beta$. Consequently, a cheapest path from $\varsigma(v')$ to t with respect to $\tilde{c}$ costs at least $d_{\tilde{c}}(\varsigma(v')) \geq \min\{3\varrho/\beta, d_{\tilde{c}}(\varsigma(v))\}$. As $d_{\tilde{c}}(\varsigma(v')) \leq 2\varrho/\beta$, this implies that $d_{\tilde{c}}(\varsigma(v')) \geq d_{\tilde{c}}(\varsigma(v))$. Our proof is almost complete. For the final part, recall that $(v, \varsigma(v))$ is located on P' between v' and the next intersection of P and P'. By construction of $\tilde{c}$ we have $\tilde{c}(v', \varsigma(v')) \geq c(v, \varsigma(v))$. Furthermore, $(v, \varsigma(v))$ has no extra cost. Putting all the pieces together we get

$$\begin{aligned}\zeta_{\tilde{c}}(v) &\leq \eta_{\tilde{c}}(v, \varsigma(v)) = c(v, \varsigma(v)) + \beta d_{\tilde{c}}(\varsigma(v)) \leq \tilde{c}(v', \varsigma(v')) + \beta d_{\tilde{c}}(\varsigma(v')) \\ &\leq c(v', \varsigma(v')) + \tilde{c}(v', \varsigma(v')) + \beta d_{\tilde{c}}(\varsigma(v')) = \eta_{\tilde{c}}(v', \varsigma(v')) \leq 2\varrho.\end{aligned}$$

◀

To complement this result, we argue that MCC-OPT is NP-hard to approximate within any ratio of $1 + \beta(1-\beta)^4$ or less. Choosing $\beta = 1/5$ maximizes this term and yields inapproximability for constant ratios of 1.08192 or less. In particular, assuming that P $\neq$ NP this rules out the existence of a polynomial-time approximation scheme.

▶ **Theorem 7.** *MCC-OPT is NP-hard to approximate within a ratio less or equal to* 1.08192.

Proof. To establish the theorem, a reduction similar to the one from Theorem 4 can be used. In fact, given a 3-SAT instance $\mathcal{I}$ we can construct the corresponding MCC-OPT instance $\mathcal{J}$ the same way as in the proof of Theorem 4. The only difference is that our choice of ε is slightly more restrictive as we require

$$\varepsilon < \min\Big\{\beta(1-\beta)^3, \beta(1-\beta)^2(2-\beta), \frac{\beta^2(1-\beta)^3}{1+\beta}, \frac{\beta^2(1-\beta)^2(2-\beta)}{1+\beta}\Big\}.$$

The proof can be structured around the following properties of $\mathcal{J}$: (a) If $\mathcal{I}$ has a solution, $\mathcal{J}$ admits a motivating cost configuration for a reward of $1/\beta$. (b) If $\mathcal{I}$ has no solution, $\mathcal{J}$ admits no motivating cost configuration for a reward of $(1+\beta(1-\beta)^4)/\beta$ or less. Consequently,

any algorithm that approximates MCC-OPT within a ratio of $1+\beta(1-\beta)^4$ or less must also solve $\mathcal{I}$. To maximize this ratio we choose $\beta = 1/5$ and obtain the desired approximability bound, namely $1 + (1 - 1/5)^4/5 = 1.08192$. All that remains to show is that $\mathcal{J}$ indeed satisfies (a) and (b). The correctness of (a) is an immediate consequence of the proof of Theorem 4. A detailed proof of (b) can be found in the full version of this work. ◀

6 Conclusion

In this work we have used Kleinberg and Oren's graph theoretic framework [5] to provide a systematic analysis of a penalty-based commitment device. We have shown that penalty fees are strictly more powerful than prohibition. In particular, we have shown that penalties may outperform prohibition by a factor of up to $1/\beta$. We have also been able to obtain some of the first positive computational results for the algorithmic design of commitment devices. We have given a polynomial-time algorithm to construct penalty fees that match an optimal solution by a factor of 2. This is significant progress when compared to the prohibition-based commitment device, whose approximation is known to be NP-hard within a factor less than $\sqrt{n}/3$ [2]. Due to its versatility, expressiveness and favorable computational properties, we believe that our penalty-based commitment device will prove to be a valuable tool for addressing time-inconsistent behavior in complex social and economic settings.

References

1 George A. Akerlof. Procrastination and obedience. *The American Economic Review*, 81(2):1–19, 1991.

2 Susanne Albers and Dennis Kraft. Motivating time-inconsistent agents: A computational approach. In *Proceedings of the 12th Conference on Web and Internet Economics*, pages 309–323. Springer, 2016.

3 Gharad Bryan, Dean Karlan, and Scott Nelson. Commitment devices. *Annual Review of Economics*, 2:671–698, 2010.

4 Nick Gravin, Nicole Immorlica, Brendan Lucier, and Emmanouil Pountourakis. Procrastination with variable present bias. In *Proceedings of the 17th ACM Conference on Economics and Computation*, pages 361–361, New York, NY, USA, 2016. ACM.

5 Jon Kleinberg and Sigal Oren. Time-inconsistent planning: A computational problem in behavioral economics. In *Proceedings of the 15th ACM Conference on Economics and Computation*, pages 547–564, New York, NY, USA, 2014. ACM.

6 Jon Kleinberg, Sigal Oren, and Manish Raghavan. Planning problems for sophisticated agents with present bias. In *Proceedings of the 17th ACM Conference on Economics and Computation*, pages 343–360, New York, NY, USA, 2016. ACM.

7 David Laibson. Golden eggs and hyperbolic discounting. *The Quarterly Journal of Economics*, pages 443–477, 1997.

8 Ted O'Donoghue and Matthew Rabin. Doing it now or later. *The American Economic Review*, 89:103–124, 1999.

9 Ted O'Donoghue and Matthew Rabin. Incentives and self control. *Advances in Economics and Econometrics: The 9th World Congress*, 2:215–245, 2006.

10 Pingzhong Tang, Yifeng Teng, Zihe Wang, Shenke Xiao, and Yichong Xu. Computational issues in time-inconsistent planning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017. To appear.