Time-Inconsistent Planning: Simple Motivation Is Hard to Find

Fedor V. Fomin,[1] **Torstein J. F. Strømme**[1]
[1]University of Bergen, Norway

Abstract

People sometimes act differently when making decisions affecting the present moment versus decisions affecting the future only. This is referred to as time-inconsistent behaviour, and can be modeled as agents exhibiting *present bias*. A resulting phenomenon is abandonment, which is when an agent initially pursues a task, but ultimately gives up before reaping the rewards.

With the introduction of the graph-theoretic *time-inconsistent planning model* due to Kleinberg and Oren, it has been possible to investigate the computational complexity of how a task designer best can support a present-biased agent in completing the task. In this paper, we study the complexity of finding a *choice reduction* for the agent; that is, how to remove edges and vertices from the task graph such that a present-biased agent will remain motivated to reach his target even for a limited reward. While this problem is NP-complete in general, this is not necessarily true for instances which occur in practice, or for solutions which are of interest to task designers. For instance, a task designer may desire to find the best task graph which is not too complicated.

We therefore investigate the problem of finding *simple* motivating subgraphs. These are structures where the agent will modify his plan at most k times along the way. We quantify this simplicity in the time-inconsistency model as a structural parameter: The number of branching vertices (vertices with out-degree at least 2) in a minimal motivating subgraph.

Our results are as follows: We give a linear algorithm for finding an optimal motivating path, i. e. when $k = 0$. On the negative side, we show that finding a simple motivating subgraph is NP-complete even if we allow only a single branching vertex — revealing that simple motivating subgraphs are indeed hard to find. However, we give a pseudo-polynomial algorithm for the case when k is fixed and edge weights are rationals, which might be a reasonable assumption in practice.

Introduction

Time-inconsistent behavior is a theme attracting great attention in behavioral economics and psychology. The field investigates questions such as why people let their bills go to debt collection, or buy gym memberships without actually using them. More generally, inconsistent behavior over time occurs when an agent makes a multi-phase plan, but does not follow through on his initial intentions despite circumstances remaining essentially unchanged. Resulting phenomenons include procrastination and abandonment.

A common explanation for time-inconsistent behavior is the notion of *present bias*, which states that agents give undue salience to events that are close in time and/or space. This idea was described mathematically already in 1930's when (Samuelson 1937) introduced the discounted-utility model, which has since been refined in different versions (Laibson 1994).

George Akerlof describes in his lecture (Akerlof 1991) an even simpler mathematical model; here, the agent simply has a *salience factor* causing immediate events to be emphasized more than future ones. He goes on to show how even a very small salience factor in combination with many repeated decisions can lead to arbitrary large extra costs for the agent. This salience factor also has support from psychology, where (McClure et al. 2004) showed by using brain imaging that separate neural systems are in play when humans value immediate and delayed rewards.[1]

In 2014, (Kleinberg and Oren 2018) introduced a graph-theoretic model which elegantly captures the salience factor and scenarios of Akerlof. In this framework, where the agent is maneuvering through a weighted directed acyclic graph, it is possible to model many interesting situations. We will provide an example here.

Example

The student Bob is planning his studies. He considers taking a week-long course whose passing grade is a reward he quantifies to be worth $r = 59$. And indeed, Bob discovers that he can actually complete the course incurring costs and effort he quantifies to be only 46 — if he works evenly throughout the week (the upper path in Figure 1). Bob will reevaluate the cost every day, and as long as he perceives the cost to be at most equal to the reward, he will follow the path he finds to have the lowest cost.

[1]We remark that quasi-hyperbolic discounting (discussed in (Laibson 1994; McClure et al. 2004)) can be seen as a generalization of both the discounted-continuity model (Samuelson 1937) and the salience factor (Akerlof 1991). There has been some empirical support for this model; however there are also many known psychological phenomena about time-inconsistent behavior it does not capture (Frederick, Loewenstein, and O'Donoghue 2002).

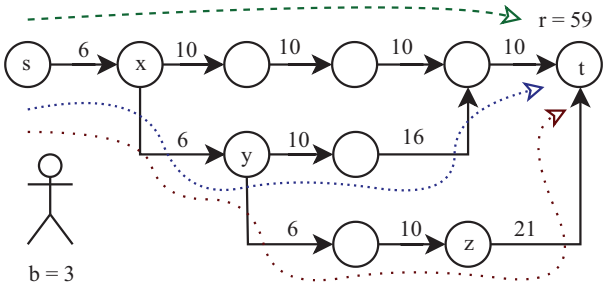

Figure 1: Acyclic digraph illustrating the ways in which Bob can distribute his efforts in order to complete the course. The upper path (green dashed line) is Bob's initial plan requiring the least total effort. The middle path (blue, narrowly dotted line) is the plan which appears better when at vertex x, and lower path (red, widely dotted line) is the plan he ultimately changes to at vertex y.

The first day of studies incurs a cost of 6 for Bob due to some mandatory tasks he needs to do that day. But because Bob has a salience factor $b = 3$, he actually perceives the cost of that day's work to be 18, and of the course as a whole to be 58 ($18+10+10+10+10$). The reward is even greater, though, so Bob persists to the next day.

When the second day of studies is about to start, Bob quasi-subconsciously makes the incorrect judgment that reducing his studies slightly now is the better strategy. He then changes his plan to the middle path in Figure 1. In terms of our model, the agent Bob standing at vertex x reevaluates the original plan (the upper path) to now cost $3 \cdot 10 + 10 + 10 + 10 = 60$, whereas the middle path is evaluated to only cost $3 \cdot 6 + 10 + 16 + 10 = 54$. He therefore chooses to go on with the plan that postpones some work to later, incurring a small extra cost to be paid at that time.

On the third day Bob finds himself at vertex y, and is yet again faced with a choice. The salience factor, as before, cause him to do less work in the present moment at the expense of more work in the future. He thus changes his plan to the lower path of Figure 1. However, it turns out that the choice was fatal — on the last day of the course (at vertex z), Bob is facing what he perceives to be a mountain of work so tall that it feels unjustified to complete the course; he evaluates the cost to be $3 \cdot 21 = 63$, strictly larger than the reward. He gives up and drops the course.

Because Bob abandons the task in our example above, we say that the graph in Figure 1 is not *motivating*. A natural question is to ask what we can do in order to make it so.

An easy solution for making a model motivating is to simply increase the reward. By simulating the process, it is also straightforward to calculate the minimum required reward to obtain this. However, it might be costly if we are the ones responsible for purchasing the reward, or even impossible if the reward is not for us to decide. A more appealing strategy might therefore be to allow the agent to only move around in a subgraph of the whole graph; for instance, if the lower path did not exist in our example above, then the graph would actually be motivating for Bob.[2] Finding such a subgraph is a form of *choice reduction*, and can be obtained by introducing a set of rules the agent must follow; for instance deadlines.

The aim of the current paper is not, however, to delve into the details of any particular scenario, but rather to investigate the formal underlying graph-theoretic framework. In the same spirit, (Kleinberg and Oren 2018) showed that the structure of a minimal motivating subgraph is actually quite restricted, and ask whether there is an efficient algorithm finding such subgraphs. Unfortunately, (Tang et al. 2017) and (Albers and Kraft 2019) independently proved that this problem is NP-complete in the general case. However, this does not exclude the existence of polynomial time algorithms for more restricted classes of graphs, or algorithms where the exponential blow-up occurs in parameters which in practical instances are small. This is what we investigate in the current paper; specifically, we look at restricting the number of *branching vertices* (vertices with out-degree at least 2) in a minimal motivating subgraph. This parameter can also be understood as the number of times a present-biased agent changes his plan.

Before we present our results, let us introduce the model more formally.

Formal model

We here present the model due to (Kleinberg and Oren 2018). Formally, an instance of the *time-inconsistent planning model* is a 6-tuple $M = (G, w, s, t, r, b)$ where:

- $G = (V(G), E(G))$ is an acyclic digraph called a *task graph*. $V(G)$ is a set of elements called *vertices*, and $E(G) \subseteq V(G) \times V(G)$ is a set of directed *edges*. The graph is *acyclic*, which means that there exists an ordering of the vertices called a *topological order* such that, for each edge, its first endpoint comes strictly before its second endpoint in the ordering. Informally speaking, vertices represent states of intermediate progress, whereas edges represent possible actions that transitions an agent between states.
- $w : E(G) \to \mathbb{R}_{\geq 0}$ is a function assigning non-negative weight to each edge. Informally speaking, this is the cost incurred by performing a certain action.
- $s \in V(G)$ is the start vertex.
- $t \in V(G)$ is the target vertex.
- $r \in \mathbb{R}_{\geq 0}$ is the reward.
- $b \in \mathbb{R}_{\geq 1}$ is the agent's salience factor.[3]

[2] Removing edges and/or vertices that destroys the lower path is also the *only* option for how to make the example graph motivating; the upper path must be kept in its entirety, otherwise the agent will not be motivated to move from s to x; the middle path must also be kept in its entirety, otherwise the agent will give up when at x.

[3] While we in the current paper use b for the salience factor as introduced in (Kleinberg and Oren 2018), the literature about time-inconsistent planning commonly use the term $\beta = b^{-1}$ instead, which (while slightly more convoluted to work with for our purposes) seamlessly integrates with the quasi-hyperbolic discounting model. An artifact of our current definition is that the reward

An agent with salience factor b is initially at vertex s and can move in the graph along edges in their designated direction. The agent's task is to reach the target t, at which point the agent can pick up a reward worth r. We can usually assume that there is at least one path from s to each vertex, and at least one path from each vertex to t, as otherwise these vertices are of no interest to the agent.

When standing at a vertex u, the agent evaluates (with a present bias) all possible paths from u to t. In particular, a u-t path $P \subseteq G$ with edges $e_1, e_2, \ldots, e_p$ is evaluated by the agent standing at u to cost $\zeta_M(P) = b \cdot w(e_1) + \sum_{i=2}^{p} w(e_i)$. We refer to this as the *perceived* cost of the path. For a vertex u, its perceived cost to the target is the minimum perceived cost of any path to t, $\zeta_M(u) = \min\{\zeta_M(P) \mid P \text{ is a } u\text{-}t \text{ path}\}$. If the perceived cost from vertex u to the target is strictly larger than the reward, $\zeta_M(u) > r$, then an agent standing there *abandons* the task. Otherwise, he will (non-deterministically) pick one of the paths which minimize perceived cost of reaching t, and traverse its first edge. This repeats until the agent either reach t or abandons the task.

If every possible route chosen by the agent will lead him to t, then we say that the model instance is *motivating*. If the model instance is clear from the context, we take that the *graph* is motivating to mean the same thing, and we may drop the subscript $_M$ in the notation.

Definition 1 (Motivating subgraph)**.** *If G' is a subgraph of G belonging to a time-inconsistent planning model $M = (G, w, s, t, r, b)$, then we call G' a* motivating subgraph *if G' contains s and t and $M' = (G', w|_{E(G')}, s, t, r, b)$ is motivating.*

In the current paper, we investigate the problem of finding a simple motivating subgraph. In order to quantify what we mean by *simple*, we first provide the definition of a *branching vertex*:

Definition 2 (Branching vertex)**.** *The* out-degree *of a vertex u in a digraph G is the number of edges in G that have u as its first endpoint. We say that u is a* branching vertex*, if its out-degree is at least two.*

In its most general form, we will investigate the following problem:

SIMPLE MOTIVATING SUBGRAPH
Input: A time-inconsistent planning model $M = (G, w, s, t, r, b)$, and a non-negative integer $k \in \mathbb{Z}_{\geq 0}$.
Question: Does there exist a motivating subgraph $G' \subseteq G$ with at most k branching vertices?

Previous work

Time-inconsistent behavior is a field with a long history in behavioral economics, see (Akerlof 1991; Frederick, Loewenstein, and O'Donoghue 2002; O'Donoghue and Rabin 1999). The model used in this paper was introduced by (Kleinberg and Oren 2018), and they also give structural results concerning how much extra cost the salience factor can incur for an agent, and how many values the salience factor can take which will lead the agent to follow distinct paths. They raise the issue of motivating subgraphs, and give a characterization of the minimal among them which we will see later.

(Tang et al. 2017) refine the structural results concerning extra costs caused by present bias. Furthermore, they show that finding motivating subgraphs is NP-complete in the general case by a reduction from 3-SAT. They also investigate a few variations of the problem where intermediate rewards can be placed on vertices.

(Albers and Kraft 2019) independently show that finding motivating subgraphs is NP-complete, by a reduction from the ℓ-LINKAGE problem in acyclic digraphs. Furthermore, they show that the approximation version of the problem (finding the smallest r such that a motivating subgraph exists) cannot be approximated in polynomial time to a ratio of $\sqrt{n}/3$ unless P = NP; but a $1+\sqrt{n}$ -approximation algorithm exists. They also explore another variation of the problem with intermediate rewards.

There have been work on variations on the model and problem where the designer is free to raise edge costs (Albers and Kraft 2017a), where the agents are more sophisticated (Kleinberg, Oren, and Raghavan 2016), exhibit multiple biases simultaneously (Kleinberg, Oren, and Raghavan 2017), or where the salience factor varies (Albers and Kraft 2017b; Gravin et al. 2016).

A_φ	For a set A and a constraint $\varphi : A \to \{\mathtt{T}, \mathtt{F}\}$, the set A *constrained to* φ denotes the elements of A that satisfy φ, i. e. $\{a \in A \mid \varphi(a) = \mathtt{T}\}$. For example, $\mathbb{Z}_{\geq 0}$ indicates the set of all non-negative integers.
$[x]$	For $x \in \mathbb{Z}_{\geq 1}$, $[x]$ is the set $\{1, 2, \ldots, x\}$.
$\subseteq$	For sets, $\subseteq$ is the standard subset notation. For graphs G and H, H is a *subgraph* of G, denoted $H \subseteq G$, if $V(H) \subseteq V(G)$, and $E(H) \subseteq E(G)$.
$f\|_A$	For a function $f : B \to C$ and a set $A \subseteq B$, the function f *restricted to* A is a function $f\|_A : A \to C$ such that for every $a \in A$, $f\|_A(a) = f(a)$.
$\mathrm{C}_G(u, v)$	For a graph G and vertices $u, v \in V(G)$, the (true) *cost* from u to v is the minimum sum of weights for a path from u to v in G.
$G[A]$	For a directed graph G and vertex set $A \subseteq V(G)$, the *induced subgraph* $G[A]$ is the graph where $V(G[A]) = A$ and $E(G[A]) = E(G) \cap A \times A$.

Table 1: Summary of notation.

Our contribution

We prove two main results about the complexity of SIMPLE MOTIVATING SUBGRAPH. First, we show that the problem is solvable in linear time when $k = 0$, and is NP-complete otherwise. Secondly, we show that when the costs are bounded

is not scaled by b when the agent is one leg away; however, both algorithms and hardness proofs can be adapted to account for this technical difference.

by some polynomial in the size of the task graph, then it is solvable in polynomial time for every fixed k. More precisely, we show the following.

Theorem 3. SIMPLE MOTIVATING SUBGRAPH *is solvable in polynomial time for $k = 0$, and is NP-complete for any $k \geq 1$.*

The reduction we use to prove NP-completeness of the problem in Theorem 3 is from SUBSET SUM, which is weakly NP-complete. Indeed, our hardness reduction strongly exploits constructions with exponentially large (or small) edge weights; the parameter W in our reduction — the sum of all edge weights when scaled to integer values — is exponential in the number of vertices in the graph. A natural question is hence whether SIMPLE MOTIVATING SUBGRAPH can be solved by a pseudo-polynomial algorithm, i. e. an algorithm which runs in polynomial time when W is bounded by some polynomial of the size of the graph. Unfortunately, this is highly unlikely. A closer look at the NP-hardness proof of (Tang et al. 2017) for finding a motivating subgraph, reveals that instances created in the reduction from 3-SAT have weights that depend only on the constant b.

On the other hand, in the reduction of (Tang et al. 2017) the number of branchings in their potential solutions grows linearly with the number of clauses in the 3-SAT instance. This leaves a possibility that when W is bounded by a polynomial of the size of the input graph G and k is a constant, then SIMPLE MOTIVATING SUBGRAPH is solvable in polynomial time. Theorem 4 confirms that this is exactly the case. (In this theorem we assume that all edge weights are integers — but since scaling the reward and all edge weights by a common constant is trivially allowed, it also works for rationals.)

Theorem 4. SIMPLE MOTIVATING SUBGRAPH *is solvable in time $(|V(G)| \cdot W)^{\mathcal{O}(k)}$ whenever all edge weights are integers and W is the sum of all weights.*

Theorem 4 naturally leads to another question, whether SIMPLE MOTIVATING SUBGRAPH is solvable in time $(|V(G)| \cdot W)^{\mathcal{O}(1)} \cdot f(k)$ for some function f of k only. Or in other words, whether the problem is fixed-parameter tractable parameterized by k when W is encoded unary? We observe that the hardness proof of (Albers and Kraft 2019), reducing from the ℓ-LINKAGE problem in acyclic digraphs, combined with hardness result for the ℓ-LINKAGE problem by (Slivkins 2010), implies the following theorem.

Theorem 5. *Unless* FPT $=$ W[1]*, there is no algorithm solving* SIMPLE MOTIVATING SUBGRAPH *in time $(|V(G)| \cdot W)^{\mathcal{O}(1)} \cdot f(k)$ for any function f of k only.*

A dichotomy (proof of Theorem 3)

In this section we prove Theorem 3. Recall that the theorem states that SIMPLE MOTIVATING SUBGRAPH is solvable in polynomial time for $k = 0$, and is NP-complete for any $k \geq 1$. We split the proof into two subsections. The first subsection contains a polynomial time algorithm solving SIMPLE MOTIVATING SUBGRAPH for $k = 0$ (Lemma 7). The second subsection proves that the problem is NP-complete for every $k \geq 1$ (Lemma 8).

A linear-time algorithm for motivating paths

Any connected graph without branching vertices is a path. Thus, we refer to the variant of SIMPLE MOTIVATING SUBGRAPH with $k = 0$ as to the MOTIVATING PATH problem. This algorithm is essentially a variation on the classical linear time algorithm computing a shortest path in an acyclic digraph.

We give an algorithm which solves the MOTIVATING PATH problem in $\mathcal{O}(|V(G)| + |E(G)|)$ time. In fact, our algorithm will solve the problem of finding the minimum cost such path, if one exists.

Algorithm 6: MOTIVATING PATH

Input: An instance of MOTIVATING PATH
Output: The cost of a cheapest motivating s-t path witnessing a yes-instance; or ∞ if no motivating path exists.

```
For each vertex u ∈ V(G), let d_u ← ∞
For the target t, let d_t ← 0
for each vertex u ∈ V(G) in rev. topological order do
    for each out-neighbor v ∈ N(u) do
        if b · w(uv) + d_v ≤ r and w(uv) + d_v < d_u
         then
            d_u ← w(uv) + d_v
        end
    end
end
return d_s
```

Lemma 7. *The* MOTIVATING PATH *problem can be solved in linear time.*

Proof. We prove that Algorithm 6 is correct. We assume that every vertex has a path to t in G (otherwise we can simply remove it), hence t will come last in the topological order. For every $u \in V(G)$, we claim that d_u holds the minimum cost of a motivating path from u to t. We observe that our base, $u = t$, is correct since $d_t = 0$.

Consider some vertex u. Because the vertices are visited in reverse topological order, all out-neighbors of u are already processed, and hold by the induction hypothesis the correct value. An agent standing at vertex u is motivated to move to the next vertex v in a path P if $b \cdot w(uv) + \mathrm{C}_P(v, t) \leq r$. Hence, if the condition holds, prepending a motivating path from v to t with u will also yield a motivating path. By choosing the minimum total cost among all feasible candidates for the next step, the final value is in accordance with our claim.

Finally, we observe that the runtime is correct. Assuming an adjacency list representation of the graph, a topological sort can be done in linear time. The algorithm then process each vertex once and touches each edge once. □

We remark that the graph produced by Algorithm 6 is a $(1 + \sqrt{n})$ -approximation to the general motivating subgraph problem. This follows because the approximation algorithm of (Albers and Kraft 2019) with the stated approximation ratio always produce a path; Algorithm 6 will on the other hand find the optimal path, and is hence at least as good.

Hardness of allowing branching vertices

Deciding whether there exists a *path* which will motivate a biased agent to reach the target turns out to be easy, but we already know by the results of (Tang et al. 2017) and (Albers and Kraft 2019) that finding a motivating *subgraph* in general is NP-hard. Both aforementioned results give hardness reductions where the feasible solutions to the reduced instance have a "complicated" structure in the sense that they contain many branching vertices. A natural question is whether it could be easier to find motivating subgraphs whose structure is simpler, as in the case of the path.

A first question might be whether SIMPLE MOTIVATING SUBGRAPH is *fixed parameter tractable* (FPT) parameterized by k, the number of branching vertices. Unfortunately, this is already ruled out by the reduction of Albers and Kraft, since they reduce from the W[1]-hard ℓ-LINKAGE problem for acyclic digraphs — the number of branchings in their feasible solutions is linear in ℓ.

As the ℓ-LINKAGE problem can be solved in time $n^{f(\ell)}$ for acyclic digraphs (Bang-Jensen and Gutin 2008), their reduction does not rule out an XP-algorithm. In this section we show that SIMPLE MOTIVATING SUBGRAPH is actually NP-hard even for $k = 1$, and is thus an even harder problem.

We show this by a reduction from the NP-hard SUBSET SUM problem. The reduction is provided in the form of Algorithm 9, whereby someone who wants to solve SUBSET SUM can give their problem instance as input, and receive a SIMPLE MOTIVATING SUBGRAPH-instance with $k = 1$ as output. This can then be piped into an (imaginary) algorithm which solves SIMPLE MOTIVATING SUBGRAPH efficiently for $k = 1$. If said imaginary algorithm runs in polynomial time, this implies P=NP.

Lemma 8. *The* SIMPLE MOTIVATING SUBGRAPH *problem is NP-complete for every* $k \geq 1$.

Proof. Deciding whether a given graph is motivating can be done in polynomial time by simply checking whether it is possible for the agent to reach a vertex where the perceived cost is greater than the reward. This implies the membership of SIMPLE MOTIVATING SUBGRAPH in NP.

To prove NP-completeness, we reduce from the classical NP-complete problem SUBSET SUM (Karp 1972).

> SUBSET SUM
> **Input:** A set of integers $X = \{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{Z}_{\geq 0}$ and a target $W \in \mathbb{Z}_{\geq 0}$.
> **Question:** Does there exists a subset $X' \subseteq X$ such that its elements sums to W?

The reduction is described in the form of Algorithm 9 and an example is given in Figure 2. Soundness of the reduction is proved in the following claim.

Claim 10. *Algorithm 9 is safe. Given as input an instance I of* SUBSET SUM *and a salience factor $b \in \mathbb{R}_{>0}$, the output instance I' of* SIMPLE MOTIVATING SUBGRAPH *is a yes-instance if and only if I is a yes-instance.*

Proof of claim. Before we begin the proof, we will adapt the following notation: For a vertex u and subgraph $H \subseteq G$ containing at least one path from u to t, we let $\zeta_H(u)$ denote the perceived cost from vertex u to t in H. In other words, $\zeta_H(u) = \min\{\zeta(P) \mid P \subseteq H \text{ is a } u\text{-}t \text{ path}\}$.

Algorithm 9: Reduction from SUBSET SUM to SIMPLE MOTIVATING SUBGRAPH ($k = 1$)

Input: An instance $I = (X = \{x_1, x_2, \ldots, x_n\}, W)$ of SUBSET SUM; and any salience factor $b \in \mathbb{R}_{>1}$.
Output: An instance $I' = (G, w, s, t, b, r, k)$ of SIMPLE MOTIVATING SUBGRAPH with $k = 1$ and salience factor b.

$$V(G) \leftarrow \{s, a_0, a_1, a_2, a_3, t\} \cup \{c_1, c_1^*, c_2, c_2^*, \ldots, c_n, c_n^*\} \cup \{c_{n+1}, c_{n+2}\}$$

$$E(G) \leftarrow \{sa_0, a_0a_1, a_1a_2, a_2a_3, a_3t\} \cup \{c_ic_i^*, c_ic_{i+1}, c_i^*c_{i+1} \mid i \in [n]\} \cup \{a_0c_1, c_{n+1}c_{n+2}, c_{n+2}t\}$$

$$w \leftarrow \left\{ \begin{array}{rll} a_3t \to & \frac{1}{b} & \\ a_2a_3 \to & \frac{1-w(a_3t)}{b} & \\ a_1a_2 \to & \frac{1-w(a_2a_3)-w(a_3t)}{b} & \\ a_0a_1 \to & \frac{1-w(a_1a_2)-w(a_2a_3)-w(a_3t)}{b} & \\ sa_0 \to & \frac{1-(\sum_{i=0}^{2} w(a_ia_{i+1}))-w(a_3t)}{b} + \frac{\epsilon}{b} & \\ c_{n+2}t \to & w(a_3t) + \epsilon & \\ c_{n+1}c_{n+2} \to & w(a_2a_3) - 2\epsilon - \frac{2\epsilon}{b-1} & \\ c_ic_{i+1} \to & \frac{x_i \cdot w(a_1a_2)}{W} & \text{for } i \in [n] \\ a_0c_1 \to & w(a_0a_1) + \frac{2\epsilon}{b-1} & \\ \to & 0 & \text{otherwise} \end{array} \right\}$$

return $(G, w, s, t, b, r = 1, k = 1)$

For the forward direction of the proof, assume that I is a yes-instance and let $X' \subseteq X$ be a witness to this. Let $G' \subseteq G$ be the graph where for each $i \in [n]$ the vertex c_i^* and incident edges are removed if $x_i \in X'$, and the edge c_ic_{i+1} is removed if $x_i \notin X'$. We make a series of step-wise observations which shows that G' is motivating:

1. G' contains a single vertex with out-degree at least 2, namely a_0. There are two possible paths from s to t: The a-path $P_a = [s, a_0, a_1, a_2, a_3, t]$; and the c-path $P_c = [s, a_0, c_1, (c_1^*), c_2, (c_2^*), \ldots, c_n, (c_n^*), c_{n+1}, c_{n+2}, t]$ (where for each $i \in [n]$, P_c only include vertex c_i^* when $x_i \notin X'$).
2. In the path P_a, the agent will by construction perceive the cost of moving towards t to be 1, regardless of which vertex he resides on. The only exception to this is when the agent is at s; then the perceived cost of following the path P_a is $1 + \epsilon$. In other words, $\zeta(P_a) = 1 + \epsilon$ and for each of $i \in \{0, 1, 2, 3\}$, $\zeta_{P_a}(a_i) = 1$.
3. The cost from c_1 to c_{n+1} in G' is $\sum_{x_i \in X'} \frac{x_i \cdot w(a_1a_2)}{W}$, which simplifies to exactly $w(a_1a_2)$. This holds because X' sums to W by the initial assumption. In other words, $\mathrm{C}_{P_c}(c_1, c_{n+1}) = w(a_1a_2)$.
4. The cost from a_0 to t is ϵ shorter in the c-path compared to the a-path, $\mathrm{C}_{P_c}(a_0, t) = \mathrm{C}_{P_a}(a_0, t) - \epsilon$. This follows from (3) and how the remaining weights in the c-path are defined.

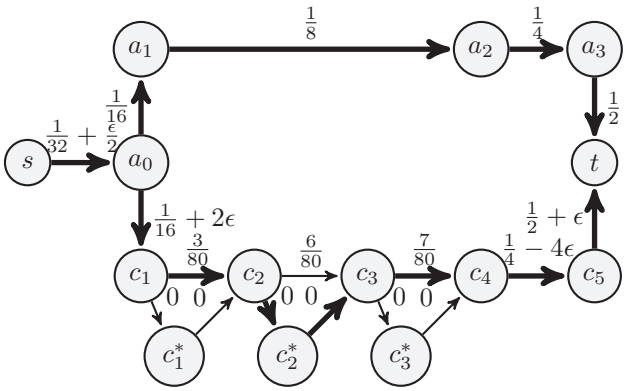

Figure 2: The graph G constructed by Algorithm 9 on input $X = \{3, 6, 7\}, W = 10, b = 2$. The solution to SUBSET SUM is $X' = \{3, 7\}$ and the corresponding motivating subgraph G' is formed by thick arcs. In graph G the agent is tempted to pursue the lower path $P_c = [s, a_0, c_1, c_1^*, c_2, c_2^*, c_3, c_3^*, c_4, c_5, t]$ but will lose motivation at node c_5. In graph G', at node s, the agent's plan will be to follow the lower path P_c' but at node a_0 the agent will switch to the upper path P_a. At every step on this path the agent is motivated to reach t.

5. The perceived cost from s to t in P_c is 1, $\zeta(P_c) = 1$ (follows by combining (2) and (4), and observing that the two paths share their first leg). The agent is hence motivated to move from s to a_0 with a plan of following P_c towards the target.
6. The perceived cost from a_0 to t in P_c is ϵ more than the perceived cost from a_0 to t in P_a. This follows from the following chain of substitutions:

$$\begin{aligned}
\zeta_{P_c}(a_0) &= b \cdot w(a_0c_1) + \mathbf{C}_{P_c}(c_1, t) \\
&\quad + w(c_{n+1}c_{n+2}) + w(c_{n+2}t) \\
&= b \cdot w(a_0a_1) + b \cdot \frac{2\epsilon}{b-1} + w(a_1a_2) \\
&\quad + w(a_2a_3) - 2\epsilon - \frac{2\epsilon}{b-1} + w(a_3t) + \epsilon \\
&= b \cdot w(a_0a_1) + w(a_1a_2) \\
&\quad + w(a_2a_3) + w(a_3t) + \epsilon \\
&= b \cdot w(a_0a_1) + \mathbf{C}_{P_a}(a_1, t) + \epsilon \\
&= \zeta_{P_a}(a_0) + \epsilon
\end{aligned}$$

It follows from (6) that an agent standing at a_0 is *not* willing to walk along P_c, but change his plan to walking along P_a instead. The agent will stay motivated on the a-path, and will reach the reward (2). Hence, the graph G' is motivating.

For the reverse direction of the proof, assume there is a motivating subgraph $G' \subseteq G$ that motivates the agent to reach t. We notice that G' must contain the a-path, since any agent moving out of the a-path will by the construction of G' need to walk past c_{n+2} — however, an agent standing at this vertex will by the construction always give up. We also notice that P_a itself is not motivating, hence G' must contain at least one other path from s to t. Let P_c denote the *cheapest* s-t path that is different from P_a.

We begin by observing that P_c must start at s, include a_0 and c_1,, a path from c_i to c_{i+1} for every $i \in [n]$, as well as the vertices c_{n+1}, c_{n+2}, and t.

There are some cost requirements that P_c needs to fulfill in order for G' to be motivating. In order to motivate an agent at s to move to a_0, the cost from a_0 to t in P_c can be at most $\mathbf{C}_{P_a}(a_0, t) - \epsilon$. This implies that $\mathbf{C}_{P_c}(c_1, c_{n+1}) \leq w(a_1a_2)$.

However, P_c can not have so low cost that it tempts the agent to move off the a-path. In particular, an agent standing at a_0 must perceive the a-path to be strictly cheaper than walking along P_c. We obtain the following inequality:

$$\begin{aligned}
\zeta_{P_a}(a_0) &< \zeta_{P_c}(a_0) \\
b \cdot w(a_0a_1) + \mathbf{C}_{P_a}(a_1, t) &< b \cdot w(a_0c_1) + \mathbf{C}_{P_c}(c_1, t) \\
\mathbf{C}_{P_a}(a_1, t) &< b \cdot \frac{2\epsilon}{b-1} + \mathbf{C}_{P_c}(c_1, t) \\
w(a_1a_2) + w(a_2a_3) + w(a_3t) &< b \cdot \frac{2\epsilon}{b-1} \\
&\quad + \mathbf{C}_{P_c}(c_1, c_{n+1}) \\
&\quad + w(a_2a_3) - 2\epsilon - \frac{2\epsilon}{b-1} \\
&\quad + w(a_3t) + \epsilon \\
w(a_1a_2) &< (b-1) \cdot \frac{2\epsilon}{b-1} \\
&\quad + \mathbf{C}_{P_c}(c_1, c_{n+1}) - \epsilon \\
w(a_1a_2) - \epsilon &< \mathbf{C}_{P_c}(c_1, c_{n+1})
\end{aligned}$$

We now construct a solution $X' \subseteq X$ to SUBSET SUM by including x_i in X' if P_c use the edge c_ic_{i+1}. Notice that the sum of edge weights for these edges will make up the cost $\mathbf{C}_{P_c}(c_1, c_{n+1})$. We can lift the bounds for that cost to the sum of elements in X':

$$\begin{aligned}
w(a_1a_2) - \epsilon < \mathbf{C}_{P_c}(c_1, c_{n+1}) &\leq w(a_1a_2) \\
w(a_1a_2) - \epsilon < \sum_{x_i \in X'} \frac{x_i \cdot w(a_1a_2)}{W} &\leq w(a_1a_2) \\
W - \epsilon \cdot \frac{W}{w(a_1a_2)} < \sum_{x_i \in P_c} x_i &\leq W
\end{aligned}$$

By choosing ϵ strictly smaller than $\frac{w(a_1a_2)}{W}$, we guarantee that the set X' has value exactly W (note: this bound on ϵ is a function of b and W). This concludes the proof of the soundness of the reduction. ◇

By the claim, SIMPLE MOTIVATING SUBGRAPH is NP-complete for $k = 1$ and hence is also for every $k \geq 1$. □

A pseudo-polynomial algorithm (proof of Theorem 4)

The reduction in the previous section shows that restricting the number of branchings in the motivating structures we look for does not make the task of finding them significantly

easier. However, we notice that the weights used in the graph constructed in the reduction can be exponentially small, and depend on W as well as on b. On the other hand, in the hardness proof for MOTIVATING SUBGRAPH by (Tang et al. 2017), the instances created in the reduction from 3-SAT have weights that depend only on the constant b; but the number of branchings in their potential solutions grows linearly with the number of clauses in the 3-SAT instance.

Hence, MOTIVATING SUBGRAPH is hard even if the number of branchings in the solution we are looking for is bounded, and it is also hard if the input instance only use integer weights bounded by a constant. But what if we impose both restrictions simultaneously? In this section we prove that if the input instance has bounded integer weights, then we can quickly determine whether it contains a motivating subgraph with few branchings.

We will make a use of an auxiliary problem which is a variation of the exact ℓ-LINKAGE problem in acyclic digraphs, but which also impose restrictions on the links — requiring them to be motivating for biased agents. We define the problem:

EXACT MOTIVATING k-LINKAGE IN DAG (EMKL)
Input: An acyclic digraph G; edge weights $w : E(G) \to \mathbb{Z}_{\geq 0}$ s. t. $\sum_{e \in E(G)} w(e) = W$; sources $s_1, s_2, \ldots, s_k \in V(G)$; sinks $t_1, t_2, \ldots, t_k \in V(G)$; target link weights $\ell_1, \ell_2, \ldots, \ell_k \in Z_{\geq 0}$; salience factors $b_1, b_2, \ldots, b_k \in \mathbb{R}_{\geq 1}$; and rewards $r_1, r_2, \ldots, r_k \in \mathbb{R}_{\geq 0}$.
Question: Does there exist (internally) vertex disjoint paths $P_1, P_2, \ldots, P_k$ such that for each $i \in [k]$, P_i starts in s_i, ends in t_i, has weight ℓ_i, and is such that an agent with salience factor b_i will be motivated by a reward r_i to move from s_i to t_i?

We solve the EXACT MOTIVATING k-LINKAGE IN DAG problem using dynamic programming, inspired by the solution of (Fortune, Hopcroft, and Wyllie 1980) for the ℓ-LINKAGE problem in acyclic digraphs (see also (Bang-Jensen and Gutin 2008)).

Lemma 11. EXACT MOTIVATING k-LINKAGE IN DAG *can be solved in time* $\mathcal{O}(kn^{k+1}W^k)$.

Proof sketch. We solve the problem by dynamic programming using a Boolean table dp of size $\mathcal{O}(n^k W^k)$. The table is indexed by vertices $u_i \in V(G)$ and weights $d_i \in [W]$ for $i \in [k]$. We define a cell $dp[u_1, u_2, \ldots, u_k, d_1, d_2, \ldots, d_k]$ to be TRUE if and only if there exists vertex disjoint paths $P_1, P_3, \ldots, P_k$, such that for each $i \in [k]$ the following holds:

- P_i starts in u_i and ends in t_i, and
- P_i has cost d_i, and
- an agent with salience factor b_i is motivated by a reward r_i to move from u_i to t_i in P_i.

Observe that the final answer to the EMKL instance will by this definition be found in $dp[s_1, s_2, \ldots, s_k, \ell_1, \ell_2, \ldots, \ell_k]$. In the recurrence, try every neighbor of every vertex in the current index, to see whether it is possible to find a partial solution where one of the k paths is one leg shorter, and then extend that partial solution with one vertex. □

Returning to our problem SIMPLE MOTIVATING SUBGRAPH with integer weights, we make use of the following observations that enables us to give an algorithm.

Proposition 12 ((Kleinberg and Oren 2018, Theorem 5.1))**.** *If G' is a minimal motivating subgraph, then it contains a unique s-t path $P \subseteq G'$ that the agent will follow. Moreover, every node of G' has at most one outgoing edge that does not lie on P.*

Note that a consequence of Proposition 12 is that all branching vertices of a minimal motivating subgraph are on the path P. We can further observe that the number of vertices with in-degree two or more is bounded by the number of branching vertices. We are now prepared for the algorithm that proves Theorem 4. Due to space constraints we only present a short sketch.

Proof sketch of Theorem 4. We give an algorithm sketch for SIMPLE MOTIVATING SUBGRAPH. First, we guess all "interesting" vertices of a solution $G' \subseteq G$, namely those which are either branching vertices, their immediate out-neighbors, or vertices of in-degree at least two. By Proposition 12, there are at most $\mathcal{O}(k)$ such vertices.

After guessing the interesting vertices, we guess how they are interconnected, and specifically which cost the paths between them have in G'. This gives rise to an instance of EMKL, where the sources, sinks and target link costs corresponds to the interesting vertices and how they are guessed to be interconnected. The salience factor is given as either b or 1 depending on where the agent is guessed to move, and the rewards are calculated based on the guessed distance to the target. In total, there will be created at most $(|V(G)| \cdot W)^{\mathcal{O}(k)}$ instances of EMKL, each of which will run in time $(|V(G)| \cdot W)^{\mathcal{O}(k)}$. The total runtime is hence $(|V(G)| \cdot W)^{\mathcal{O}(k)}$. □

Conclusion

We have shown that the SIMPLE MOTIVATING SUBGRAPH problem is polynomial-time solvable when $k = 0$, and NP-complete otherwise (Theorem 3). However, when edge weights are (scaled to) integers and their sum is bounded by W, we gave a pseudo-polynomial algorithm which solves the problem in time $(|V(G)| \cdot W)^{\mathcal{O}(k)}$ (Theorem 4). Finally, we observed that an algorithm with run-time $(|V(G)| \cdot W)^{\mathcal{O}(1)} \cdot f(k)$ where f is a function of k only is not possible unless FPT=W[1] (Theorem 5).

We end the paper with an open question and a reflection for further research. First, a question; in Theorem 5, a more careful analysis reveals that the statement holds even when $W \in \mathcal{O}(k^2)$. But is it also the case when W is a constant?

Finally, a reflection. In Theorem 4 we give a pseudo-polynomial algorithm for SIMPLE MOTIVATING SUBGRAPH when the edge weights are scaled to integer values and the number of branchings is constant. While we can reasonably assume that edge weights are represented as fractions or integers after storing them in a computer, this suggests that

the values might have been quantized or approximated in some way. However, this rounding could potentially alter the solution space. In particular when the cost of two paths are perceived to be almost equal at the branching point — then rounding values even a tiny bit can have big consequences. To overcome this, one might change the model such that the perceived difference of costs must exceed an epsilon for the agent's choice to be unequivocal.

Acknowledgments

This work has been supported by the Research Council of Norway via the project "MULTIVAL". We also thank the AAAI reviewers for valuable suggestions.

References

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

Albers, S., and Kraft, D. 2017a. On the value of penalties in time-inconsistent planning. In Chatzigiannakis, I.; Indyk, P.; Kuhn, F.; and Muscholl, A., eds., *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 10:1–10:12. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Albers, S., and Kraft, D. 2017b. The price of uncertainty in present-biased planning. In *International Conference on Web and Internet Economics*, 325–339. Springer.

Albers, S., and Kraft, D. 2019. Motivating time-inconsistent agents: A computational approach. *Theory of Computing Systems* 63(3):466–487.

Bang-Jensen, J., and Gutin, G. 2008. *Digraphs: Theory, Algorithms and Applications*. Springer Monographs in Mathematics. Springer London.

Fortune, S.; Hopcroft, J.; and Wyllie, J. 1980. The directed subgraph homeomorphism problem. *Theoretical Computer Science* 10(2):111 – 121.

Frederick, S.; Loewenstein, G.; and O'Donoghue, T. 2002. Time discounting and time preference: A critical review. *Journal of Economic Literature* 40(2):351–401.

Gravin, N.; Immorlica, N.; Lucier, B.; and Pountourakis, E. 2016. Procrastination with variable present bias. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, 361–361. ACM.

Karp, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. New York: Plenum Press. 85–103.

Kleinberg, J., and Oren, S. 2018. Time-inconsistent planning: A computational problem in behavioral economics. *Communications of the ACM* 61(3):99–107.

Kleinberg, J.; Oren, S.; and Raghavan, M. 2016. Planning problems for sophisticated agents with present bias. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, 343–360. ACM.

Kleinberg, J.; Oren, S.; and Raghavan, M. 2017. Planning with multiple biases. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, 567–584. ACM.

Laibson, D. I. 1994. *Hyperbolic Discounting and Consumption*. Ph.D. Dissertation, Massachusetts Institute of Technology, Department of Economics.

McClure, S. M.; Laibson, D. I.; Loewenstein, G.; and Cohen, J. D. 2004. Separate neural systems value immediate and delayed monetary rewards. *Science* 306(5695):503–507.

O'Donoghue, T., and Rabin, M. 1999. Doing it now or later. *American Economic Review* 89(1):103–124.

Samuelson, P. A. 1937. A note on measurement of utility. *The Review of Economic Studies* 4(2):155–161.

Slivkins, A. 2010. Parameterized tractability of edge-disjoint paths on directed acyclic graphs. *SIAM Journal on Discrete Mathematics* 24(1):146–157.

Tang, P.; Teng, Y.; Wang, Z.; Xiao, S.; and Xu, Y. 2017. Computational issues in time-inconsistent planning. In *Proceedings of the 31st t AAAI Conference on Artificial Intelligence (AAAI)*.