Computational Issues in Time-Inconsistent Planning

Pingzhong Tang,* **Yifeng Teng,**† **Zihe Wang,**‡ **Shenke Xiao,*** **Yichong Xu**♯
∗Tsinghua University, † University of Wisconsin-Madison,
‡Shanghai University of Finance and Economics, ♯Carnegie Mellon University

Abstract

Time-inconsistency refers to a paradox in decision making where agents exhibit inconsistent behaviors over time. Examples are *procrastination* where agents tend to postpone easy tasks, and *abandonments* where agents start a plan and quit in the middle. To capture such behaviors and to quantify inefficiency caused by such behaviors, Kleinberg and Oren (2014) propose a graph model with a certain cost structure and initiate the study of several interesting computation problems: 1) cost ratio: the worst ratio between the actual cost of the agent and the optimal cost, over all the graph instances; 2) motivating subgraph: how to motivate the agent to reach the goal by deleting nodes and edges; 3) Intermediate rewards: how to incentivize agents to reach the goal by placing intermediate rewards. Kleinberg and Oren give partial answers to these questions, but the main problems are open.

In this paper, we give answers to all three open problems. First, we show a tight upper bound of cost ratio for graphs, and confirm the conjecture by Kleinberg and Oren that Akerlof's structure is indeed the worst case for cost ratio. Second, we prove that finding a motivating subgraph is NP-hard, showing that it is generally inefficient to motivate agents by deleting nodes and edges in the graph. Last but not least, we show that computing a strategy to place minimum amount of total reward is also NP-hard and we provide a 2n-approximation algorithm.

1 Introduction

In behavioral economics, an important theme has been to understand individual behaviors that are inconsistent over time. There are at least two types of inconsistencies investigated in the literature. The first type is *procrastination* (Akerlof 1991; O'Donoghue and Rabin 1999; Kleinberg and Oren 2014): agents tend to postpone costly actions even though such delay may incur further cost. The second type is *abandonment* (O'Donoghue and Rabin 2008): agents plan for a multi-phase task (usually with rewards in the end), spend efforts in the initial phases and decide to quit in the middle.

Both types of behaviors have been widely observed in reality. Akerlof (1991) describes a story of procrastination where an agent must ship a package within the next a few days, incurs an immediate cost for shipping the package or some additional daily cost for not shipping the package. Clearly, the *optimal* strategy for the agent is to ship the package right away, avoiding any additional daily cost. However, as the story goes, the agent chooses to procrastinate and to send the package in one of the last few days. Similar examples abound, ranging from golf club members that never play golf (abandonment) to investors that rent an apartment for years before making a purchase (procrastination).

The interpretation to all these phenomena lies in that agents value current cost more than the cost in the future. Researchers in the literature have developed various models to capture this observation and to interpret the inconsistencies (Strotz 1955; Akerlof 1991; Laibson 1997; Frederick, Loewenstein, and O'donoghue 2002; Ockenfels and Roth 2002; Osogami and Morimura 2012). In what follows, we describe the coined *time-inconsistent planning* model in (Kleinberg and Oren 2014).

1.1 The time-inconsistent planning model

Roughly put, the time-inconsistent planning model is no different from the standard planning model (Pollak 1968; Russell and Norvig 2003), except for a slight twist on the cost structure. The standard planning model is a directed graph (aka. task graph) where each node in the graph represents a state, each directed edge denotes an action that transits one state to another and each action incurs a certain cost, marked as the weight on the edge. The planner's goal is to find a shortest (min-cost) path between the initial state and target state. While the time-inconsistent planning model redefines the cost of a path: instead of summing the costs of all edges on that path: $\sum_{i=1} c(e_i)$ where $c(e_i)$ is the cost of the i-th edge on the path, a time-inconsistent agent applies a multiplicative factor $0 < \beta < 1$ to the costs of all the edges except for the first edge in the path $c(e_1)+\beta \sum_{i=2} c(e_i)$. The interpretation is that the time-inconsistent agent evaluates the current actions by its true cost, while discounts the costs (rewards) of all future actions by β. The time-inconsistent planning model is defined as a time-inconsistent agent who looks for a discounted shortest path at any state in the task graph. The cost model above is also a special case of the *quasi-hyperbolic discounting* model (Laibson 1997).

The model is general enough to capture a range of interesting time-inconsistent behaviors. In particular, Kleinberg and Oren (2014) show that:

- It can easily capture Akerlof's example (Figure 1): time-inconsistency agents may follow a sub-optimal path.
- The model can be easily extended to a model to include reward. It can be further used to capture the phenomenon of abandonment: agents may find it desirable to follow the optimal discounted path at first but then find it not beneficial when evaluating at some following node.
- The model can be used to model the interesting fact of *choice reduction*: agents can be better motivated to reach the target by deleting certain middle nodes and edges.

The time-inconsistent model can also be served for studying "Badge design" of online communities and social media sites (Anderson et al. 2013; 2014; Easley and Ghosh 2013), where one basic problem is how to optimally place rewards on the intermediate nodes (i.e., the third open problem). The time-inconsistent planning is also a bounded rationality behavior (Wang and Tang 2015; Zuo and Tang 2015).

After the first draft of this paper, several extensions of the original model have been proposed. Kleinberg, Oren, and Raghavan (2016) consider a sophisticated agent who make choices based on the belief that himself will procrastinate in the future. Such an agent has a bounded cost ratio of β^{-1} (the present bias). Gravin et al. (2016) consider the present bias as a random variable. For any distribution of present bias, the graph that maximizes the cost ratio is still similar to the Akerlof example, and distribution with high tails is likely to cause high cost ratios. As a follow-up work of our unpublished manuscript, Albers and Kraft (2016) solve several closely related complexity problems and give approximation algorithms, which we will describe later.

1.2 Results and contributions

1. Cost ratio. Cost ratio is defined as the ratio between the cost of the path found by the agent and that of the min-cost path. Kleinberg and Oren (2014) give a partial characterization of cost ratio: roughly, any graph with a sufficiently high cost ratio must contain Figure 1 (denoted as $\mathcal{F}_k$) as a graph minor and k is at least a constant fraction of n. In light of this, an important open problem is: when the graph does not contain a $\mathcal{F}_k$-minor, how bad can the cost ratio be? This question is particularly important since it concerns whether $\mathcal{F}_k$ is indeed the worst case instance.

   We solve this problem by proving that, for any graph that does not contain a $\mathcal{F}_k$-minor, the cost ratio can be at most β^{2-k}, where β is the discount factor. Therefore, we confirm that $\mathcal{F}_k$ is indeed the worst case instance for cost ratio and the bound proved by Kleinberg and Oren is tight.

   We put forward a few technical insights in order to prove this upper bound. First of all, we identify a set of *shortcut nodes* where the agent's min-cost choice and time-inconsistent choice differs. Then we focus on shortcut nodes and the nodes around them. Finally, we conduct a refined analysis to the costs of paths according to the structure of shortcut nodes.

2. Minimal Motivating Subgraphs. We consider the case when there is a reward at the target node. If the agent's cost is less or equal to the reward, then the agent moves on, otherwise, the agent stops and doesn't reach the target. A motivating subgraph is a subgraph of the original task graph and a time-inconsistent agent can reach the target in this subgraph. It is minimal if none of its proper subgraphs is motivating. Clearly, motivating subgraph is closely related to the previously mentioned economic problem of choice reduction. Kleinberg and Oren prove a relatively complex property that says the minimal motivating subgraphs are necessarily *sparse*. Here, we ask a natural complexity question: what is the computational complexity of finding minimal motivating subgraph?

   We answer this question by showing that the problem is NP-hard. More generally, we show that finding any (e.g., minimal or maximal) motivating subgraph is NP-hard. As a follow-up of our work, Albers and Kraft (2016) give a $(1 + \sqrt{n})$ approximation of computing a motivating subgraph, and shows the NP-completeness of approximating the problem by any factor smaller than $\sqrt{n}/3$.

3. Cost of incentivizing agents by placing intermediate rewards. Instead of motivating agents to reach their target by choice reduction (i.e., via motivating subgraph), an alternative way in the literature is to place rewards on intermediate nodes. The question (the third open problem by Kleinberg and Oren) is: what is the minimum total reward needed to motivate an agent to reach the target?

   We consider various versions of this problem, including whether we can use negative rewards and whether all rewards put should be claimed at last. We prove that, all the problems are NP-hard. After that, we give a $2n$-approximation algorithm. As a follow-up, Albers and Kraft (2016) give NP-completeness of a slight variant of our problem, where the user only gives positive reward and only cares for what is actually claimed.

To sum up, we solve all three open problems in (Kleinberg and Oren 2014).

2 Formal description of the model

A task graph is an acyclic directed graph G with a start node s and a target node t, where each edge (u, v) has a non-negative cost $c(u, v)$. For any pair of nodes (u, v), we denote by $d(u, v)$ the minimum total cost from u to v,

$$d(u, v) = \min_{P \in \mathcal{P}(u,v)} \sum_{e \in P} c(e),$$

where $\mathcal{P}(u, v)$ is the set of all possible paths from u to v. For simplicity, let $d(v) = d(v, t)$ for any node v. Denote the discount parameter by $\beta \in [0, 1]$. An agent starts from s and travels towards t. In each step, the agent at node u chooses an out-neighbor v that minimizes $c(u, v) + \beta d(v)$ (if more than one node minimizes this value, the agent chooses one arbitrarily). Clearly, the agent cares less about the future for smaller β. Suppose P is the s-t path that the agent chooses, the *cost ratio* is defined as $\sum_{e \in P} c(e)/d(s)$, i.e., the ratio of actual cost to the optimal cost.

Given two undirected graphs H and K, we say that H contains a K*-minor* if we can map each node κ in K to a connected subgraph S_κ in H, with the properties that (i) S_κ

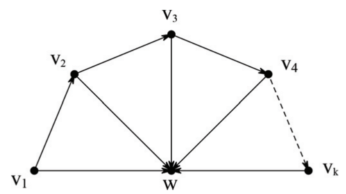

Figure 1: $\mathcal{F}_k$

and $S_{\kappa'}$ are disjoint for every two nodes κ, κ' in K, and (ii) if (κ, κ') is an edge in K, then there is some edge connecting a node in S_κ with a node in $S_{\kappa'}$.

Moreover, let $\sigma(G)$ denote the *skeleton* of G, the undirected graph obtained by removing the directions on the edges in G. Let $\mathcal{F}_k$ denote the directed graph with nodes $v_1, v_2, \dots, v_k$, and w, and edges (v_i, v_{i+1}) for $i = 1, \dots, k-1$, and (v_i, w) for $i = 1, \dots, k$. Figure 1 depicts $\mathcal{F}_k$. $\mathcal{F}_k$ is a special structure in this setting. It vividly illustrates the Akerlof story. An agent starts from node v_1 with the target node w. The optimal strategy is going from v_1 to w directly. However, the agent will choose the path $v_1, v_2, ..., v_k, w$ and cause procrastination. Furthermore, Kleinberg and Oren (2014) have proved the following theorem:

Theorem 2.1 *For every $\lambda > 1$ there exist $n_0 > 0$ and $\varepsilon > 0$ such that if $n \geq n_0$ and cost ratio $r > \lambda^n$, then $\sigma(G)$ contains an $\mathcal{F}_k$-minor for some $k \geq \varepsilon n$.*

3 Maximum Cost Ratio

In light of the result above, Kleinberg and Oren propose the following open question: what is the maximum cost ratio (if exists) if $\sigma(G)$ does not contain an $\mathcal{F}_k$-minor. In other words, the problem asks, without $\mathcal{F}_k$, how much waste can be resulted from time-inconsistency. The question is essential in understanding cost ratio since it is closely concerned whether $\mathcal{F}_k$ is the worst case instance.

Note that an edge is exactly $\mathcal{F}_1$. Assume $k > 1$, we have the following theorem.

Theorem 3.1 *For any $k > 1$, if $\sigma(G)$ does not contain an $\mathcal{F}_k$-minor, the cost ratio is at most β^{2-k}. This bound is tight and can be achieved by $\mathcal{F}_{k-1}$.*

3.1 Proof Sketch

To analyze the cost ratio of any graph, our first observation is to focus on the set of *shortcut nodes*. Roughly, a shortcut node is a node that is on the path chosen by the agent and at this node the agent's min-cost choice differs from his actual choice. Clearly, if there were no such nodes, i.e., the agent's actual path and min-cost path coincide, we would end up in the ideal case where the cost ratio is 1.

For each shortcut node u, we obtain a lower bound for $d(u)$, which is a sum of $d(\cdot)$'s (may be multiplied by some coefficients) for nodes lying after u on the actual path. An important intuition here is that each appearance of such a shortcut node contributes a factor of β to the cost ratio, and k such appearances (to be formally defined) would lead to the worst case of β^k and we show that this happens only when $\mathcal{F}_k$-minor exists.

To upper bound the cost ratio, we need to carefully expand the cost formula $d(s)$ as a linear combination of costs on edges $c(e)$'s. This is complicated, again, by the existence of shortcut nodes, since the recursive formula that defines $d(u_i)$ (where u_i is some shortcut node) introduces two new terms $d(u_i')$, the cost from the next node on the actual path and $d(w_i)$, the cost where the current min-cost path merges with the actual path (See Figure 2). Our strategy then is to fix $d(w_i)$ and carefully expand $d(u_i')$.

A key step of our proof is that two different cases of w_i (immediately after some shortcut nodes on the actual path, or not) are considered and different relaxations are tailored for each case. The absence of this refined analysis would lead to a loose bound as in the Kleinberg-Oren paper.

Continue the expansion of $d(u_i)$ until the right hand side contains $c(e)$'s only, i.e., representing $d(u_i)$ as a linear combination of $c(e)$'s, we obtain the final result by bounding the coefficients of the linear combination.

3.2 Formal Proof

We now formally prove Theorem 3.1. During the proof, we introduce a number of lemmas, which are proved in the full version.

Proof. Let P be the path that the agent actually travels through. Recall that, $c(e)$ is the cost of edge e and $d(u, v)$ is the minimum total cost from u to v, and $d(v) = d(v, t)$. The main idea of the proof is to obtain an inequality with the form $d(s) \geq \sum_{e \in P} \alpha(e)c(e)$ where α's are positive coefficients, and then give the upper bound in terms of the minimal coefficient, i.e. $\sum_{e \in P} c(e)/d(s) \leq \min_{e \in P} \alpha(e)$.

First of all, we need a basic inequality. Here we introduce some notations. For any node u on P, denote by u' the first node after u in P; for any pair of nodes (u, v) on P, denote by $c(u, v)$ the total cost of edges between u and v in P.

Definition 3.2 *A node u on P is called shortcut node if the second node in the min-cost path from u to t is not u'. In other words, at a shortcut node, the agent's min-cost choice differs from his actual choice on P.*

If node u on P is not a shortcut node, we have $d(u) = c(u, u') + d(u')$. So if there is no shortcut node, we would have $d(s) = c(s, s') + d(s') = c(s, s') + c(s', s'') + d(s'') = \cdots = \sum_{e \in P} c(e)$, resulting in a cost ratio of 1.

Suppose there are n shortcut nodes: $u_1, u_2, \dots, u_n$ in the order of appearance on P. For $i = 1, 2, \dots, n$, denote by P_i the min-cost path from u_i to t, and note for any two nodes u, v in the order of appearance on P_i, $d(u, v)$ is exactly the sum of the cost of edges connecting them on P_i, so that for any three nodes u, v, w in order on P_i, we have $d(u, w) = d(u, v) + d(v, w)$; denote by w_i the second crossing point (note that the first node is u_i) of P and P_i (if there are more than one min-cost path, arbitrarily choose one); denote by v_i the first node after u_i on P_i. Figure 2 describes the notations.

By the definition of time-inconsistency, the agent at u_i chooses P over P_i, we have

$$c(u_i, v_i) + \beta d(v_i) \geq c(u_i, u_i') + \beta d(u_i'). \tag{1}$$

Intuitively, we do not want any node that is not on P to show up. Note $d(v_i) = d(v_i, w_i) + d(w_i)$, we can expand (1) as

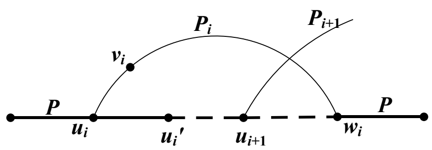

Figure 2: Relationship between nodes.

$$c(u_i, v_i) + \beta d(v_i, w_i) + \beta d(w_i) \geq c(u_i, u_i') + \beta d(u_i'). \quad (2)$$

Then note $d(u_i, w_i) = c(u_i, v_i) + d(v_i, w_i) \geq c(u_i, v_i) + \beta d(v_i, w_i)$, combining (2), we have

$$d(u_i, w_i) + \beta d(w_i) \geq c(u_i, u_i') + \beta d(u_i'). \quad (3)$$

Add $(1-\beta)d(w_i)$ to both sides of (3) and use $d(u_i') = c(u_i', u_{i+1}) + d(u_{i+1})$, we have

$$\begin{aligned} d(u_i) &\geq c(u_i, u_i') + \beta d(u_i') + (1-\beta)d(w_i) \\ &= c(u_i, u_i') + \beta c(u_i', u_{i+1}) + \beta d(u_{i+1}) \\ &\quad +(1-\beta)d(w_i). \end{aligned} \quad (4)$$

Second, we use Formula (4) iteratively to obtain the full decomposition of $d(s)$. The righthand side of the inequality consists of three parts: one cost term c and one distance term $d(w)$ and one distance term $d(u)$. Our idea is to expand the $d(u)$ term iteratively. For concreteness, we list below the first few steps of the expansion:

$$\begin{aligned} d(s) &= c(s, u_1) + d(u_1) \\ &\geq c(s, u_1) + c(u_1, u_1') + \beta d(u_1') + (1-\beta)d(w_1) \\ &= c(s, u_1) + c(u_1, u_1') + \beta c(u_1', u_2) + \beta d(u_2) \\ &\quad +(1-\beta)d(w_1) \\ &\geq c(s, u_1) + c(u_1, u_1') + \beta c(u_1', u_2) + \beta c(u_2, u_2') \\ &\quad +\beta^2 d(u_2') + \beta(1-\beta)d(w_2) + (1-\beta)d(w_1) \\ &\geq \cdots. \end{aligned}$$

The next lemma shows the expansion after i steps. Before that, we give some notations for ease of presentation.

Definition 3.3 *S_i is the set of j's such that $j < i$ and w_j lies after u_{i-1}' (excluding) on P.*

Informally, S_i denotes the indexes of shortcut nodes that might make up of an $\mathcal{F}_k$-minor with node u_{i-1}'. We can immediately obtain that for all i, $|S_i| \leq k-2$, because of the lack of $\mathcal{F}_k$-minor.

Definition 3.4 *For $i = 1, 2, \ldots, n$, if there exists j such that $w_i = u_j'$, then let $t_i = j + 0.5$; otherwise let t_i be the smallest index such that u_{t_i} lies after w_i (including) on P (if no such index exists, $t_i = n+1$).*

Since w_i must lie after u_i' on P, we can obtain a trivial property that $t_i \geq i+1$. Now by an easy analysis we can obtain that S_i can be represented by $\{j | 1 \leq j < i, t_j \geq i\}$.

Now we are ready to present our expansion lemma.

Lemma 3.5 *For $i = 1, 2, \ldots, n$,*

$$\begin{aligned} d(s) &\geq \sum_{j=1}^{i} \left(a_j c(u_{j-1}', u_j) + b_j c(u_j, u_j')\right) + a_{i+1} d(u_i') \\ &\quad + \sum_{j \in S_{i+1}} (1-\beta) b_j d(w_j), \end{aligned} \quad (5)$$

where $\{a_i, b_i, i = 1, 2, \ldots\}$ are determined as

$$a_1 = 1, a_i = \beta b_{i-1} + \sum_{j: i-1 < t_j < i} (1-\beta) b_j$$

$$b_i = a_i + \sum_{j: t_j = i} (1-\beta) b_j.$$

Here $u_0' = s$ and $u_{n+1} = t$.

Set $i = n$ in (5), we have

$$\begin{aligned} d(s) &\geq \sum_{j=1}^{n} \left(a_j c\left(u_{j-1}', u_j\right) + b_j c\left(u_j, u_j'\right)\right) \\ &\quad + a_{n+1} c(u_n', t). \end{aligned} \quad (6)$$

Third, we show lower bounds for coefficients a_i and b_i.

Lemma 3.6 $b_i \geq a_i \geq \beta^{|S_i|}$.

It means that coefficient b_i and a_i cannot be too small, then by $d(s) / \sum_{e \in P} c(e) \geq \min_i\{b_i, a_i\}$, we can get an upper bound for cost ratio. In the proof, we first find an interesting relationship of a_i, b_i and S_i: $a_m + \sum_{j \in S_m} (1-\beta) b_j = 1$, and then prove the bounds for a_i and b_i by induction.

For contradiction, assume $|S_i| \geq k-1$ for some i. Choose $k-1$ elements from S_i, say $j_1, j_2, \ldots, j_{k-1}$ where $j_1 < j_2 < \cdots < j_{k-1} < i$. Consider $u_{j_1}, u_{j_2}, \ldots, u_{j_{k-1}}, u_{i-1}'$. The following lemma states that if a graph has such a structure, it must contain an $\mathcal{F}_k$-minor.

Lemma 3.7 *Let P be a path of G, and $u_1, u_2, \ldots, u_k$ are nodes on P in order of appearance. If for $i = 1, 2, \ldots, k-1$, there exists a path P_i such that (i) it starts at u_i, and (ii) the second crossing point of P_i and P (certainly the first one is u_i) exists and lies after u_k (excluding) on P, then $\sigma(G)$ contains an $\mathcal{F}_k$-minor.*

By this lemma we conclude that $\sigma(G)$ contains an $\mathcal{F}_k$-minor, a contradiction. Thus $|S_i| \leq k-2$. Hence by (6) and Lemma 3.6 we have

$$\begin{aligned} d(s) &\geq \sum_{j=1}^{n} \left(\beta^{|S_j|} c\left(u_{j-1}', u_j\right) + \beta^{|S_j|} c\left(u_j, u_j'\right)\right) \\ &\quad + \beta^{|S_{n+1}|} c\left(u_n', t\right) \\ &\geq \beta^{k-2} c(s, t) \end{aligned}$$

which implies $c(s,t)/d(s) \leq \beta^{2-k}$. □

3.3 Tightness of the Bound

So far we have obtained an upper bound for cost ratio r, now we provide an example to show that the bound is achievable. We use the example mentioned in (Kleinberg and Oren 2014) to show a graph with exponential cost ratio, i.e. the graph obtained from $\mathcal{F}_{k-1}$ by adding the corresponding weights (see Figure 3). The cost ratio of the graph is exactly β^{2-k}, which proves the tightness of our upper bound.

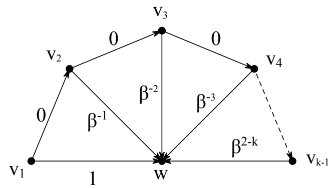

Figure 3: Akerlof's example

4 Hardness of Finding minimal motivating subgraphs

As mentioned, the basic model introduced in Section 2 can be easily extended to capture abandonment, by placing a reward at the target node. Formally, suppose the reward is r and the agent is in node u, if $\min_v c(u,v) + \beta d(v) > \beta r$, i.e., the discounted cost is less than the discounted reward, the agent will abandon the plan.

A natural question in this extended model is whether a time-inconsistent agent can reach the target, and if not, can we delete some nodes and edges to help it reach the target. The first question is easy to check. To formally investigate the second question, define *motivating subgraph* as a subgraph of the original task graph such that the agent can reach the target in the subgraph. A motivating subgraph is minimal if none of its proper subgraph is motivating. We are interested in the following computational question concerning (minimal) motivating subgraph (the second the open question): *is there a polynomial time algorithm that finds a (minimal) motivating subgraph?*

In what follows, we answer this question negatively (unless NP = P) with the following theorem.

Definition 4.1 MOTIVATING-SUBGRAPH*: for an acyclic graph G with n nodes, given reward r on target node and bias factor β, find a motivating subgraph of G.*

Theorem 4.2 MOTIVATING-SUBGRAPH *is NP-hard.*

Before the proof, we first consider an easier complexity problem related to minimal motivating subgraphs. We can show that finding a minimal motivating subgraph is hard.

Definition 4.3 MINIMAL-MOTIVATING-SUBGRAPH*: for an acyclic graph G with n nodes, given reward r on target node and present bias β, find a minimal motivating subgraph of G.*

Theorem 4.4 MINIMAL-MOTIVATING-SUBGRAPH *is NP-hard.*

Proof. [Proof Sketch] We show that finding valid assignments for 3-SAT instances is polynomial-time reducible to MINIMAL-MOTIVATING-SUBGRAPH. For any 3-CNF with n variables $x_1, x_2, ..., x_n$ and m clauses $C_1, C_2, ..., C_m$, construct a task graph G based on the structure of the formula. Here we depict the graph generated by $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4)$ and $\beta = 0.9$. Each u_i corresponds to clause C_i, while each v_i and v'_i correspond to variable x_i; u_i and v_j are adjacent if x_j is in clause C_i; w_i are functional nodes that allow the agent to travel through the top path in the graph. The idea is to construct valid assignment for each variable from its corresponding nodes' neighborhood structure in a minimal motivating subgraph.

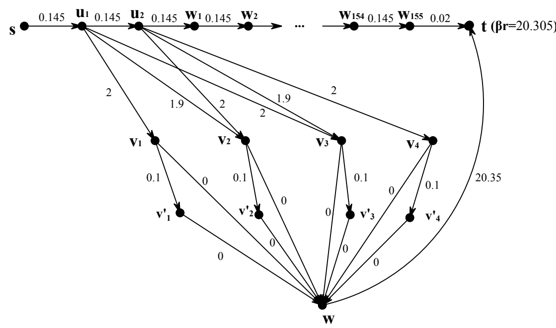

Figure 4: Corresponding graph of $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4)$ with $\beta = 0.9$.

Given any valid assignment for the original formula, we build a minimal motivating subgraph of G in polynomial time as follows. For each $1 \leq i \leq n$, if x_i is assigned true, remove v'_i and adjacent edges; otherwise remove edge (v_i, w). For $1 \leq i \leq m$, let $C_i = y_{i_1} \vee y_{i_2} \vee y_{i_3}$, here y_{i_j} denotes x_{i_j} or $\neg x_{i_j}$. Assume WLOG. that y_{i_j} is assigned true, then preserve only (u_i, v_{i_j}) among (u_i, v_{i_1}), (u_i, v_{i_2}) and (u_i, v_{i_3}), eliminate the other two edges from G. Finally, we remove all nodes with no in-degree. The resulting graph is a minimal motivating subgraph of G.

Given G' being a minimal motivating subgraph of G, we construct a valid assignment for the original boolean formula in polynomial time as follows. For each $1 \leq i \leq n$, if path $v_i \to v_{i'} \to w$ remains in G', assign x_i to be false, otherwise assign x_i to be true. Together with the previous paragraph we prove the correctness of the reduction. □

Now we prove Theorem 4.2: it is hard to find any motivating subgraph, not just minimal ones.

Proof. Assume we have an algorithm $\mathcal{A}$ solving MOTIVATING-SUBGRAPH. Notice that we can use $\mathcal{A}$ to check whether a graph has a motivating subgraph. Now we propose the following algorithm that solves MINIMAL-MOTIVATING-SUBGRAPH by calling $\mathcal{A}$.

For a given graph G, check whether G contains a motivating subgraph. If not, we reject the input. Repeatedly remove an edge from G such that the remaining graph still includes a motivating subgraph, until no edge can be removed.

The correctness of the algorithm is straightforward, while the running time of the algorithm is polynomial of the size of the graph. From Theorem 4.4 we know that MOTIVATING-SUBGRAPH is NP-hard. □

5 Hardness of motivating agents by placing intermediate reward

In this section, we consider the question of whether we can place intermediate reward on internal nodes to motivate the agent. Denote by $r(v)$ the reward on node v; the agent at

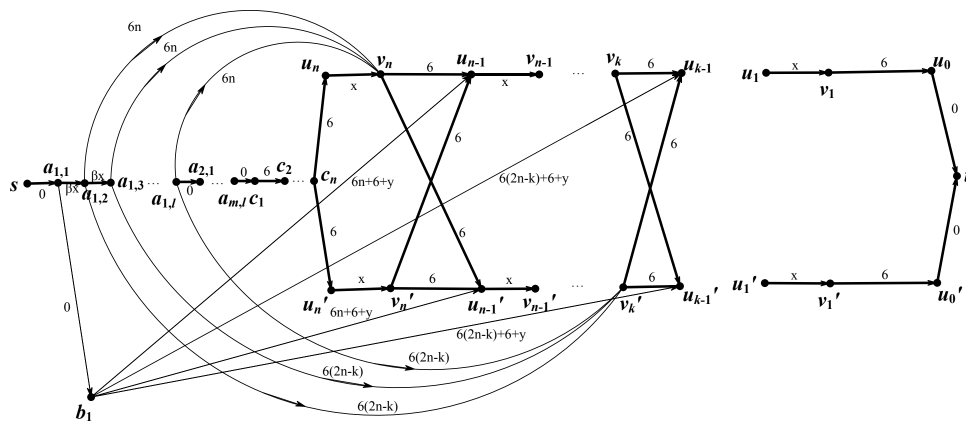

Figure 5: Example figure for proving Theorem 5.2. Edges that might be traveled by the agent are in bold lines.

node u will continue to move if and only if there exists a path P from u to t such that

$$c'(P) = c(u, u') + \beta \sum_{v \in P, v \neq u, t} (c(v, v') - r(v)) \leq 0,$$

where u' denotes the first node after u in P and v' denotes the node after v on P. The agent chooses the path that minimize $c'(P)$, which is the equivalent cost (i.e., cost of edges minus rewards on the path) of path P. We are interested in finding the configuration with the smallest sum of reward that motivate the agent to reach t (the third open problem).

5.1 Problem statement

As suggested by Kleinberg and Oren (2014), multiple versions of this problem are considered. In the first two versions, we restrict attention to positive rewards. In the first version, we might put rewards that are never claimed. We avoid this in the second version: all rewards that we put must be claimed. In the third version, we consider the possibility of placing negative rewards, which can be used to prevent the agent from entering bad paths. In this version, we want to minimize the sum of absolute values of all rewards.

We formally define this problem:

Definition 5.1 *Minimum Total Rewards with bias factor β (*MINIMAL-TOTAL-REWARD$_\beta$*): Given a weighted acyclic graph G, node s, t, and a real number R, decide whether there exists a reward configuration $r(v), \forall v \in G$ such that the agent is motivated to reach the target, and that $\sum_{v \in G} |r(v)| \leq R$.*

There are three versions of the problem, depending on the constraints on the rewards:
MINIMAL-TOTAL-REWARD$_\beta$ *I: $r(v) \geq 0, \forall v$.*
MINIMAL-TOTAL-REWARD$_\beta$ *II: $r(v) \geq 0, \forall v \in P; r(v) = 0, \forall v \notin P$, where P is the path actually taken by the agent.*
MINIMAL-TOTAL-REWARD$_\beta$ *III: $r(v) \in \mathbb{R}$.*

5.2 Main Theorem

We show that all versions of this problem are NP-hard:

Theorem 5.2 MINIMAL-TOTAL-REWARD$_\beta$ *I,II,III are NP-hard for all $\beta < 1$.*

Proof. [Proof Sketch] We prove the theorem by reducing 3-SAT to MINIMAL-TOTAL-REWARD$_\beta$. The reduction graph is depicted in Figure 5 (detailed description is in the full version). The proof idea is as below: We construct a long graph body with two long strings ($u_n v_n u_{n-1} v_{n-1} \cdots u_0$ and $u'_n v'_n u'_{n-1} v'_{n-1} \cdots u'_0$), and set weight in the graph such that there is reward on exact one of v_i and v'_i in the minimum reward setting; also, the one with reward depends on the 3-SAT clause so as to construct a valid assignment. □

5.3 Approximation Algorithm

Considering the problem is NP-hard, we provide an approximation algorithm which works for all three versions.

Algorithm 1 The shortest good path algorithm for MINIMAL-TOTAL-REWARD$_\beta$

for each edge (v_i, v_j) **do**
  Suppose the agent currently lies at node v_i and wants to go to node v_j, if the agent would go through edge (v_i, v_j) directly, we call this edge "good".
end for
Find the shortest path which only consists of good edges: $s(= v_{k(0)}), v_{k(1)}, v_{k(2)}, ..., v_{k(h)}, t(= v_{k(h+1)})$.
for each node $k(i), i = 1, ..., h$ **do**
  $r(v_{k(i)}) = c(v_{k(i-1)}, v_{k(i)}) * \beta^{-1} + c(v_{k(i)}, v_{k(i+1)})$
end for
$r(t) = c(v_{k(h)}, t) * \beta^{-1}$
r(any other node)=0.

The idea of good edge (v_i, v_j) is, when the agent is at node v_i, as long as $r(v_j)$ is large enough, the agent will go to v_j directly. For the edge that is not good, the agent will never pass edge (v_i, v_j) in any reward configuration. Algorithm 1 runs in polynomial time and we have the following claim.

Theorem 5.3 *Alg. 1 has an approximation ratio of 2n.*

6 Acknowledgments

This work was supported by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the Natural Science Foundation of China Grant 61033001, 61361136003, 61303077, 61561146398, a Tsinghua Initiative Scientific Research Grant and a China Youth 1000-talent program.

References

Akerlof, G. A. 1991. Procrastination and obedience. *The American Economic Review* 1–19.

Albers, S., and Kraft, D. 2016. Motivating time-inconsistent agents: A computational approach. *arXiv preprint arXiv:1601.00479*.

Anderson, A.; Huttenlocher, D.; Kleinberg, J.; and Leskovec, J. 2013. Steering user behavior with badges. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, 95–106. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee.

Anderson, A.; Huttenlocher, D.; Kleinberg, J.; and Leskovec, J. 2014. Engaging with massive online courses. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, 687–698. New York, NY, USA: ACM.

Easley, D., and Ghosh, A. 2013. Incentives, gamification, and game theory: An economic approach to badge design. In *Proceedings of the Fourteenth ACM Conference on Electronic Commerce*, EC '13, 359–376. New York, NY, USA: ACM.

Frederick, S.; Loewenstein, G.; and O'donoghue, T. 2002. Time discounting and time preference: A critical review. *Journal of economic literature* 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*, EC '16, 361–361. New York, NY, USA: ACM.

Kleinberg, J., and Oren, S. 2014. Time-inconsistent planning: a computational problem in behavioral economics. In *Proceedings of the fifteenth ACM conference on Economics and computation*, 547–564. ACM.

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*, EC '16, 343–360. New York, NY, USA: ACM.

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

Ockenfels, A., and Roth, A. E. 2002. The timing of bids in internet auctions: Market design, bidder behavior, and artificial agents. *AI magazine* 23(3):79.

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

O'Donoghue, T., and Rabin, M. 2008. Procrastination on long-term projects. *Journal of Economic Behavior & Organization* 66(2):161–175.

Osogami, T., and Morimura, T. 2012. Time-consistency of optimization problems. In *AAAI*.

Pollak, R. A. 1968. Consistent planning. *The Review of Economic Studies* 201–208.

Russell, S. J., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition.

Strotz, R. H. 1955. Myopia and inconsistency in dynamic utility maximization. *The Review of Economic Studies* 165–180.

Wang, Z., and Tang, P. 2015. Optimal auctions for partially rational bidders. In *Proceedings of the 24th International Conference on Artificial Intelligence*, 118–124. AAAI Press.

Zuo, S., and Tang, P. 2015. Optimal machine strategy to commit to in two-person repeated games.