Inconsistent Planning: Task Graph Modification

Artur Ignatiev

Supervisor: Fedor Fomin

Master thesis

Saint Petersburg
2024

Contents

Abstract

The present bias is a well-documented behavioral trait that significantly influences human decision-making, with present-biased agents often prioritizing immediate rewards over long-term benefits, leading to suboptimal outcomes in various real-world scenarios. Kleinberg and Oren (2014) proposed a popular graph-theoretical model of inconsistent planning to capture the behavior of present-biased agents. In this model, a multi-step project is represented by a weighted directed acyclic task graph, where the agent traverses the graph based on present-biased preferences.

We use the model of Kleinberg and Oren to address the principal-agent problem, where a principal, fully aware of the agent's present bias, aims to modify an existing project by adding or deleting tasks. The challenge is to create a modified project that satisfies two somewhat contradictory conditions. On one hand, the present-biased agent should select specific tasks deemed important by the principal. On the other hand, if the anticipated costs in the modified project become too high for the agent, there is a risk of the agent abandoning the entire project, which is not in the principal's interest.

To tackle this issue, we leverage the tools of parameterized complexity to investigate whether the principal's strategy can be efficiently identified. We provide algorithms and complexity bounds for this problem.

1 Introduction

The notion of *present bias* is a standard assumption in behavioral economics used to explain the gap between long-term intention and short-term human decision-making. A present-biased agent prioritizes immediate rewards over long-term benefits, leading to suboptimal outcomes in real-world scenarios. The present bias is one of the reasons for time-inconsistent behavior of an agent changing his optimal plans in the short run without new circumstances [OR99, TG15]. Some examples of human time-inconsistent behavior include indulging in unhealthy eating, procrastination on essential tasks and responsibilities, spending on immediate desires instead of saving, addiction abuse despite being aware of the negative consequences or neglecting the immediate efforts in environmental conservation.

While originating in behavioral economics, inconsistent planning is related to AI in several ways. In *Model of Human Behavior*, AI systems are often designed to interact with and assist humans. Understanding human behavior, including time inconsistency, is crucial for creating AI systems that can adapt to and predict human actions and preferences. AI models that consider time inconsistency provide more accurate recommendations or assistance [ESG16]. In *Personalization and Recommendations*, recommendation systems rely on understanding and predicting user preferences. If users exhibit time inconsistency in their preferences, AI systems may need to adapt their recommendations accordingly [DM22]. Finally, in *Reinforcement Learning*, agents make decisions to maximize cumulative rewards over time. Time inconsistency can affect an AI agent's ability to make optimal decisions, as it may need to evaluate future rewards and penalties accurately [LSP22].

Our work builds on Akerlof's model [Ake91], in which the *salience factor* causes the agent to prioritize immediate events over the future, with the cost of future tasks assumed to be $1/\beta$ times smaller than their actual costs for some present-bias parameter $\beta < 1$. Even a tiny salience factor could result in significant additional charges for the agent.

Kleinberg and Oren [KO14, KO18] introduced an elegant graph-theoretic model that incorporates the salience factor and scenarios from Akerlof. In this model, an agent traverses from a source s to a target t in a directed edge-weighted graph G. We will provide the formal description shortly and begin with an illustrative example.

Kleinberg-Oren model example. Alice is a PhD student, and she has to accomplish several research projects to obtain her PhD. After discussing with her advisor Bob, they agree on several possible scenarios, see Fig. 1. Every arc of the task graph corresponds to a project, and the cost of an arc is the expected cost required to finish this task. The node s is the starting position of Alice, and the node t is the final node she wants to reach. Thus Alice has three possible options to pursue, corresponding to the three paths in the graph, namely, $P_1 = sabct$, $P_2 = sadt$, and $P_3 = sadet$. She always wants to use the less costly option. To estimate the costs, Alice uses the present-bias parameter $\beta = 1/3$—when estimating the cost of a path; she estimates the cost of the first arc correctly. However, she underestimates the costs of all further arcs of the path by factor β. Thus standing in s, Alice estimates the cost of P_1 as $6 + (2 + 2 + 2)/3 = 8$, the cost of P_2 as $6 + (1 + 6)/3 = 8\frac{1}{3}$, and P_3 as $6 + (1 + 3 + 7)/3 = 9\frac{2}{3}$. She plans to pursue P_1. By accomplishing the task sa, Alice re-evaluates the remaining costs. The cost of the remaining part of P_1 is now $2 + (2 + 2)/3 = 3\frac{1}{3}$, which is more than the cost of the remaining part of P_2, that is, $1 + 6/3 = 3$. This impacts Alice's plans and now she decides to follow P_2. However, after arriving at d, she compares the remaining costs of P_2, which is 6 and P_3, which is $5\frac{1}{3}$. Alice changes her plans again and switches to P_3.

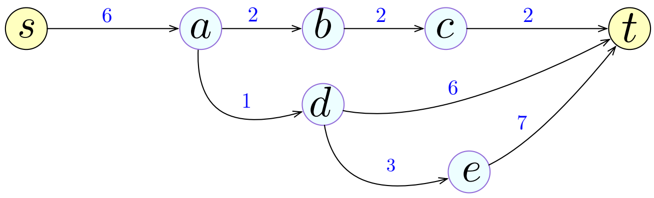

Figure 1: For $\beta = 1/3$, the agent will follow the path $sadet$ instead of selecting the shortest path $sabct$.

In this work, we use the model of Kleinberg and Oren to study a variant of the principal-agent problem, where the principal could reduce the choices to guarantee that the agent will accomplish some selected tasks. We continue with our example.

Motivating by reducing choices. We continue with the example Fig. 1. To explain the phenomenon of abandonment, Kleinberg and Oren use the reward model. We assume that Alice expects a reward of r for obtaining her PhD. At every step, she evaluates the cost of completing the path, and if this cost exceeds $\beta \cdot r$ (reward is also discounted by β), she abandons the whole project. For this example, we put $r = 24$. While Bob, the doctoral advisor of Alice, wants her to finish her study, he has additional interests too. To Bob, the task corresponding to the arc dt is the most exciting part of the whole project. However, if Alice proceeds according to the present bias protocol, she will go through P_3 and never accomplish the task so important to Bob. The first thing that comes to Bob's mind—to leave only the tasks of the path P_2 available to Alice—does not work. For Alice standing in s, the estimated biased cost of path P_2 is $8\frac{1}{3} > \beta \cdot r = 1/3 \cdot 24 = 8$. Thus, if Bob leaves P_2 as the only choice for Alice, she will abandon her studies. This brings us to the question that is the main motivation for our study. *Is it possible to reduce choices to make both Alice and Bob happy?* That is, Alice will get PhD while working on the tasks that are most interesting to Bob. In our example, the solution is easy—Bob has to delete the task de—but in general, as we will see, this question brings interesting algorithmic challenges. See Fig. 2.

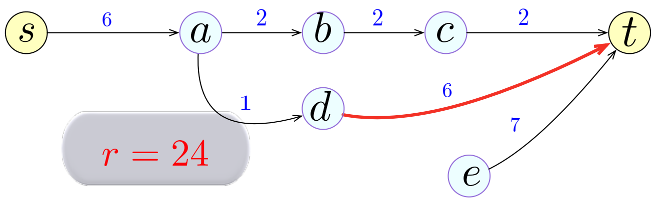

Figure 2: Let $P_1 = sabct$ and $P_2 = sadt$. For $\beta = 1/3$, the agent will follow the path P_2. Indeed, in node s, the estimated cost is $6 + 1/3(2 + 2 + 2) = 8$, which is exactly the value $1/3 \cdot r$ of discounted reward, so the agent proceeds to a. When standing in a, the estimated cost of the remaining part of P_1 is now $3\frac{1}{3}$ and of P_2 is 3. Both costs are less than the discounted reward, so the agent follows P_2.

We proceed with the formal description of the Kleinberg-Oren's model.

Definition 1.1 (Kleinberg-Oren's Model [KO14])

An instance of the *time-inconsistent planning model* is a 6-tuple $M = (G, w, s, t, \beta, r)$ where:

- $G = (V(G), E(G))$ is a directed acyclic n-vertex graph 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 *arcs* (directed edges). Vertices of G represent states of intermediate progress, whereas edges represent possible actions that transition an agent between states.
- $w : E(G) \to \mathbb{N}_0$ is a weight function representing the costs of transitions between states. The transition of the agent from state u to state v along arc $uv \in E(G)$ is of cost $w(uv)$.
- The agent starts from the start vertex $s \in V(G)$.
- $t \in V(G)$ is the target vertex.
- The rational $\beta \leq 1$ is the agent's present-bias parameter.
- $r \in \mathbb{Q}_{\geq 0}$ is the reward the agent receives by reaching t.

An agent is initially at vertex s and can move along arcs in their designated directions. The agent's task is to reach the target t. The agent moves according to the following rule. When standing at a vertex v, the agent evaluates (with a present bias) all possible paths from v to t. In particular, a v-t path $P \subseteq G$ with edges $e_1, e_2, \ldots, e_p$ is evaluated by the agent standing at v to cost

$$\zeta_M(P) = w(e_1) + \beta \cdot \sum_{i=2}^{p} w(e_i).$$

We refer to this as the *perceived* cost of the path P. For a vertex v, its *perceived cost to the target* is the minimum perceived cost of any path to t,

$$\zeta_M(v) = \min\{\zeta_M(P) \mid P \text{ is a } v\text{-}t \text{ path}\}.$$

We refer to an v-t path P with perceived cost $\zeta_M(v)$ as to a *perceived path*. If for the agent in vertex v the perceived cost $\zeta_M(v)$ exceeds $\beta \cdot r$, the value of the reward evaluated in the light of the present bias, the agent abandons the whole project. Thus when in vertex v, the agent picks one of the perceived paths[1] and traverses its first edge, say vu. After arriving at the new vertex u, the agent computes the perceived cost to the target $\zeta_M(u)$, selects a perceived u-t path, and traverses its first edge. This repeats until the agent either abandons the project or reaches t.

1.1 Guiding through specified arcs

We are interested in the variant of the principal-agent problem where the principle wants the present-biased agent to perform certain tasks. Using the Kleinberg-Oren model, we model this problem as the following graph modification problem.

[1] If there are several paths of minimum perceived cost, we assume that an agent uses a consistent tie-breaking rule, like selecting the node that is earlier in a fixed topological ordering of G.

For a set of arcs $T \subseteq E(G)$, we say that an s-t path P is a T-path if P contains all arcs of T. Our work addresses the following question.

For a given set of prescribed tasks T, is it possible to modify the time-inconsistent planning model by deleting (or adding) a few tasks such that in the modified model, the present-biased agent will reach t by following a T-path?

Formally, we study the following algorithmic problems. The first problem models the situation when the principal wants to guide the agent through the project by reducing the available options. The second problem models the situation when, instead of reducing the choice, the principal could add more choices from a selected family of tasks. In this case, we assume that we will not create directed cycles when we add arcs.

T-path-Deletion

Input: Time-inconsistent planning model $M = (G, w, s, t, \beta, r)$, integer k and a set of arcs $T \subseteq E(G)$.
Task: Find a subset of arcs $D \subseteq E(G)$ of size at most k (or prove that no such set exists), such that after removing D from M, the present-biased agent will follow a T-path.

T-path-Addition

Input: Time-inconsistent planning model $M = (G, w, s, t, \beta, r)$, integer k, a set of arcs $T \subseteq E(G)$, and a set of additional weighted arcs $A \subset V \times V$.
Task: Find a set S of at most k arcs from A (or prove that no such set exists), such that after adding these arcs to G the agent will follow a T-path.

1.2 Reducing agent costs to achieve the task

The second class of tasks that we are interested in is the problem of reducing the agent's costs. As task designers, we would like the agent to achieve the goal with minimal or limited costs, for example, an agent has a limited budget and cannot spend more than it. First, we introduce an optimization version of this problem.

We consider that there is no reward in these algorithmic problems, and the agent is always moving forward. In the case of equal estimates at the vertices, we use a probabilistic model Section 3 and [DFI22] in which, in the case of ambiguity, the agent uses a distribution to select a path.

Optimal Reducing Expectation by Deletion (Opt-REC-Deletion)

Input: Time-inconsistent planning model $M = (G, w, s, t, p, \beta)$, integer k.
Task: Find the minimum value of $\mathbf{E}(C_\beta)$, which can be achieved by removing no more than k arcs from the graph G.

Optimal Reducing Expectation by Addition (Opt-REC-Addition)

Input: Time-inconsistent planning model $M = (G, w, s, t, p, \beta)$, integer k and a set of additional weighted arcs $A \subset V \times V$.
Task: Find the minimum value of $\mathbf{E}(C_\beta)$, which can be achieved by addition no more than k arcs from A to the graph G.

We also study decision problems Reducing Expectation of Cost by Deletion and Reducing Expectation of Cost by Addition.

Reducing Expectation of Cost by Deletion (REC-Deletion)

Input: Time-inconsistent planning model $M = (G, w, s, t, p, \beta)$, integers k, ℓ.
Task: Is there a $S \subseteq E, |S| \leq k$ such that in $G - S$ math expectation $\mathbf{E}(C_\beta) \leq \ell$.

Reducing Expectation of Cost by Addition (REC-Addition)

Input: Time-inconsistent planning model $M = (G, w, s, t, p, \beta)$, integers k, ℓ and a set of additional weighted arcs $A \subset V \times V$.
Task: Is there a set S of at most k arcs from A, such that after adding these arcs to G the agent expectation $\mathbf{E}(C_\beta) \leq \ell$.

The process of adding and deleting arcs could be simulated by changing the weights of the arcs. This more general model, where the principal could change the weights of the arcs in order to motivate the agent, is much more algorithmically challenging. We consider that it is possible to increase and decrease the weight on the edges, as well as create new edges within the budget. All weights in the resulting graph should be in $\mathbb{N}_0$. Thus, we formulate the following more general problem.

Reducing Expectation of Cost with the Budget (REC-Budget)

Input: Time-inconsistent planning model $M = (G, w, s, t, p, \beta)$, integers ℓ and budget B.
Task: Is there a reassignment of the weights on the edges in the graph G for a given budget B so that $\mathbf{E}(C_\beta) \leq \ell$.

1.3 Our contribution

Motivation to complete tasks. Our work extends the current understanding and offers a nuanced perspective on the interplay between computation tractability, graph theory, and decision-making scenarios involving present-biased agents. In our algorithmic study of T-path-Deletion and T-path-Addition, we use the tools of parameterized complexity.

We start with establishing the hardness of the T-path-Deletion problem parameterized by k. The problem trivially belongs to the class XP. An algorithm of running time $|E(G)|^k \cdot poly(|M|)$ is to try all subsets of at most k arcs and simulate in polynomial time the actions of the agent on the graph, resulting in the removal of each of the subset. In Theorem 4.1, we show that T-path-Deletion is W[1]-hard parameterized by k even when T consists of a single arc. This shows that designing an algorithm of running time $f(k) \cdot poly(|M|)$ is highly unlikely for any function f of k only. We refine this result in Theorem 4.2 by establishing that T-path-Deletion problem is Para-NP-hard with various parameters. In particular, the problem is NP-hard when the maximum cost of the T-path is 6, when the reward $r = 48$, there exists only T-path, or when the input graph G has only one heavy arc, and all its other arcs are of weight 1. Thus, Theorem 4.2 refute the existence of parameterized algorithms for many natural parameters of the time-inconsistent model.

The intractability results of T-path-Deletion lead us to contemplate the following question: *although deriving efficient algorithms for general scenarios seems unlikely, could certain structural properties of the instance be algorithmically exploited?* In other words, while the overall problem may be inherently challenging, there may be specific properties within certain instances that could be leveraged to develop more efficient algorithms. We introduce two such structural properties, shedding light on potential avenues for algorithmic improvement in the context of T-path-Deletion.

The first structural property that we exploit algorithmically is the following. Suppose that in the input graph G, any path from s to t contains at most m edges. This corresponds to the situation when any sequence of tasks, either taken or anticipated by the agent, contains at most m steps. In Theorem 4.4, we give an FPT algorithm parameterized by k and m. Our second parameterization concerns the situation when the underlying undirected graph has a small number of edge-disjoint cycles. Every such cycle could potentially force the agent to change the decision. Thus this parameter is related to the number of nodes where the time-inconsistent agent could change his mind. The main result here is Theorem 4.5, which establishes the possibility of compressing the instance when the underlying graph of G has a small number of edge-disjoint cycles. More precisely, a feedback edge set of an undirected graph is a set of edges whose removal turns the graph into a forest. Informally, Theorem 4.5 proves that there is a polynomial time algorithm that, for any instance of the problem, constructs an equivalent instance whose size is bounded by a polynomial of the minimum feedback edge set of the underlying undirected graph. In other words, T-path-Deletion admits a polynomial kernel parameterized by the size of a feedback edge set of the underlying undirected graph. In particular, this implies that the problem is FPT parameterized by the size of a feedback edge set.

Finally, we provide several algorithmic results for the T-path-Addition problem. We consider the case when the set of prescribed arcs T form a path P containing all vertices of the graph. In terms of principal-agent problem, this corresponds to the following interesting scenario. The principal already decided on the sequence of steps the agent should perform. However, in order the agent to move along this path, the anticipated cost of the proposed path needs to be lowered. Coming back to our example with Alice and Bob, Bob already knows what work Alice has to perform but Alice is too scared by anticipated amount of time she has to spend on these tasks. Could Bob add some tasks (shortcuts to the path) such that Alice at the end will do all the tasks from T? As we will see in Theorem 4.6, even in this case, T-path-Addition remains intractable. On the positive side, in Theorem 4.7, we prove that for a wide class of problems with a well-separable properties of additional tasks, the problem becomes FPT.

Reducing the agent's path. We start with $(\frac{1}{\beta})^n$-approximation for the Opt-REC-Deletion problem. On the other hand, we refine this result in Theorem 5.1 by establishing that there is no $(\frac{1}{\beta} - \varepsilon)^n$–approximate FPT algorithm with parameter k under the assumption $\mathsf{W}[1] \neq \mathsf{FPT}$ for any $\varepsilon > 0$. In Theorem 5.2 and corollary 5.2, we show that REC-Deletion problem is NP-hard, as well as W[1]-hard with respect to the parameter k and several other parameters that naturally arise in this setting.

For the setting, when the principal wants to add arcs to reduce agent costs, in Theorem 5.4, we show that for Opt-REC-Addition problem there is polynomial algorithm when the input graph of the problem is a path on n vertices and set A is detours on the path. But Theorem 5.5 shows that the problem is NP-hard in general case.

Finally, we consider the case when the process of graph modification can be represented as a change in the weight of arcs. In Theorem 5.6, we establish that REC-Budget problem is NP-hard.

1.4 Related work

The mathematical ideas of present bias go back to the 1930s when [Sam37] introduced the discounted-utility model. It has developed into the hyperbolic discounting model, one of the cornerstones of behavioral economics [Lai94, MLLC04]. The model of time-inconsistent planning that we adopt for our work is due to Kleinberg and Oren [KO14, KO18]. It could be seen as a special case of the quasi-hyperbolic discounting model (see e.g. [Lai94, MLLC04]), which also generalizes both Samuelson's discounted-continuity model [Sam37] and Akerlof's salience factor [Ake91]. While there is a lot of empirical support for this model, there are also known psychological phenomena about time-inconsistent behavior it does not capture [FLO02].

There is a significant amount of follow-up work on the model of Kleinberg and Oren, see e.g. [DFI22, FS20, HS23, GILP16, KOR16, KOR17, MPS22]. In particular, the following two problems are most relevant to our model.

The first problem is of finding a motivating subgraph. In our model, this corresponds to the situation when the set of prescribed arcs T is empty. [TTW$^+$17] show that finding motivating subgraphs is NP-complete. They also investigate a few variations of the problem where intermediate rewards can be placed on vertices. [AK19] independently show that finding a motivating subgraph is NP-complete. 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. Still, a $1+\sqrt{n}$ -approximation algorithm exists. They also explore another variation of the problem with intermediate rewards. [FS20] studied the parameterized complexity of computing a simple motivating subgraph. [AK17] study a variation on the model where the designer is free to raise arc costs.

The second problem related to our work is the P-motivating subgraph problem of [OS19]. In this variant of the principal-agent problem with a present-biased agent, the principal identifies an s-t path P in the task graph G. Then the question is whether there is a subgraph of G, such that in this subgraph, the agent will follow along P. In our model, this corresponds to the situation when the prescribed arcs T form the edge set of P. Also, the difference with our model is that [OS19] looks for any P-motivating subgraph, while in our model, we are interested in a subgraph from the original graph by a small number of arc deletions/additions. [OS19] prove that the P-motivating subgraph problem is NP-complete even when there are only two different costs of arcs. In the same scenario of two costs, [OS19] gave an algorithm that runs in polynomial time when the number of light arcs in the path P is a constant.

Finally, in graph algorithms, a prevalent subject of interest revolves around graph modifications, wherein the objective is to alter a graph by modifying adjacencies or deleting vertices to achieve a graph with predefined properties. For comprehensive insights into this topic, we direct readers to surveys such as [BBD06, CDFG23, NSS01]. Our contribution can be viewed as an augmentation to the existing body of literature within this vibrant research domain.

2 Preliminaries

2.1 Parameterized complexity

We briefly recap the main definitions of parameterized complexity.

Definition 2.1

A parameterized problem is a language $Q \subseteq \Sigma^* \times \mathbb{N}$, where Σ^* is the set of strings over a finite alphabet Σ. Respectively, an input of Q is a pair (I, k) where $I \in \Sigma^*$ and $k \in \mathbb{N}$; k is the *parameter* of the problem.

We define the size of an instance (x, k) of a parameterized problem as $|x| + k$. One interpretation of this convention is that, when given to the algorithm on the input, the parameter k is encoded in unary.

Definition 2.2

A parameterized problem Q is *fixed-parameter tractable* (FPT) if it can be decided whether $(I, k) \in Q$ in time $f(k) \cdot |I|^{\mathcal{O}(1)}$ for some computable function f that depends of the parameter k only. Respectively, the parameterized complexity class FPT is composed of fixed-parameter tractable problems.

For technical reasons, it will be convenient to assume, from now on, that f is also nondecreasing. We now define the complexity class XP.

Definition 2.3

A parameterized problem Q is *slice-wise polynomial* (XP) if it can be decided whether $(I, k) \in Q$ in time $f(k) \cdot |I|^{g(k)}$ for some computable functions f, g that depends of the parameter k only. Respectively, the parameterized complexity class XP is composed of slice-wise polynomial problems.

The W-hierarchy is a collection of computational complexity classes: we omit the technical definitions here. The following relation is known amongst the classes in the W-hierarchy: $\mathsf{FPT} = \mathsf{W}[0] \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{W}[P]$. It is widely believed that $\mathsf{FPT} \neq \mathsf{W}[1]$, and hence if a problem is hard for the class $\mathsf{W}[i]$ (for any $i \geq 1$) then it is considered to be fixed-parameter intractable. For our purposes, to prove that a problem is W[1]-hard it is sufficient to show that an FPT algorithm for this problem yields an FPT algorithm for some W[1]-complete problem. We also use notation Para-NP-hard with parameter k that means NP-hard for a constant value of the parameter k. We refer to [CFK+15] for an introduction to parameterized complexity.

2.2 Kernelization

Preprocessing, or data reduction, is a widely used technique in almost all practical computer implementations that aim to solve NP-hard problems. The goal of the preprocessing step is to efficiently solve the easier parts of a problem instance and reduce (or shrink) it to its more difficult core structure, which is called the problem kernel of that instance. In other words, this method aims to reduce (but does not necessarily solve) a given problem to an equivalent, smaller instance in time that is polynomial in the size of the input. A slower, exact algorithm can then be used to solve this smaller instance.

Recall that a *kernelization* algorithm, given an instance (I,k) of some parameterized problem, runs in polynomial time and outputs an equivalent instance (x',k') of the same problem such that $|I'|, k' \le f(k)$, for some computable function f. This instance (I',k') is called the *kernel*, and function f is called the size of the kernel. Kernelization is one of the fundamental tools for parameterized algorithms, and it is well known that a problem admits a kernel from parameter k if and only if there exists an FPT algorithm for it from this parameter. We refer to books [CFK+15, FLSZ19] for further expositions of kernelization.

2.3 Time-inconsistent planning

Let $P_\beta(s,t)$ be a s-t path followed by an agent with present-bias β and let $c_\beta(s,t)$ be the cost of this path. Let $d(s,t)$ be the distance, that is the cost of a shortest s-t path. Then Kleinberg and Oren defined the measure describing the "*price of irrationality*" of the system.

Definition 2.4

The *cost of the irrationality* of the model $M = (G, w, s, t, \beta)$ is

$$\frac{c_\beta(s,t)}{d(s,t)}.$$

Is there some bounds for the cost of irrationality? To begin with, it can be exponential relative to the number of vertices in the graph.

Example 2.1 ([KO14])

There is the graph with n vertices in which cost of irrationality equals μ^n, where $\mu < 1/\beta$.

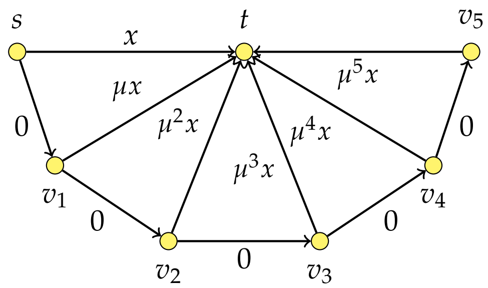

Figure 3: Akerlof example.

On the other hand, each graph with a high cost of irrationality has a specific structure. We describe this structure in the following theorem. Let $\sigma(G)$ denote the skeleton of G, the undirected graph obtained by removing the directions on the edges of G. Let $\mathcal{F}_k$ denote the graph with nodes $v_1, v_2, \ldots, v_k$, and w, and edges (v_i, v_{i+1}) for $i = 1, \ldots, k-1$, and (v_i, w) for $i = 1, \ldots, k$. We refer to $\mathcal{F}_k$ as the k-fan.

Theorem 2.1 ([KO14])

For every $\lambda > 1$ there exist $n_0 > 0$ and $\varepsilon > 0$ such that if $n > n_0$ and $c_\beta(s,t)/d(s,t) > \lambda^n$, then $\sigma(G)$ contains an $\mathcal{F}_k$-minor for some $k \ge \varepsilon n$.

3 Probabilistic Model

The results of this section have been published in [DFI22]. For a more detailed presentation of the results, you can check the paper.

One omitted detail in the definition makes the meaning of the cost of irrationality ambiguous. It could be that several paths with minimum perceived cost $\zeta_M(v)$ lead from v. In this situation an agent in the state v might be indifferent between several arcs leaving v—they both evaluate to equal perceived costs. While for the agent standing in a vertex v the perceived costs of all perceived paths are the same, the actual costs of feasible paths could be different.

Because of that, we revisit the model of Kleinberg and Oren in [KO14] and redefine the cost of irrationality. Our approach is natural—*when in doubt, toss a coin!* When several paths of minimum prescribed cost lead from v, the agent selects one of them with some probability and traverses the first arc of this path.

Thus the instance of the *inconsistent planning model* is a 6-tuple $M = (G, w, s, t, p, \beta)$, where for each edge uv of the task graph, we assign the probability $p(u, v)$ of transition $u \to v$. Here for every $u \in V(G)$, $\sum_{uv \in E(G)} p(u, v) = 1$.

Moreover, the probability can be positive only for edges that could serve for transitions of the agent. In other words, $p(u, v) > 0$ only if there is a u-t path P of perceived cost $\zeta_M(u)$ whose first edge is uv. The selection of probability p corresponds to some predictions or future preferences in breaking the ties. For example, when the agent at stage u faces ℓ u-t paths of minimum perceived cost and has no preferences over any of them, it would be natural to assign each transition from u the probability $1/\ell$. On the other hand, if the agent has preferences in selecting from paths of equal costs, this can be controlled by a different selection of p. With these settings, we call an s-t path P *feasible*, if with a non-zero probability the present-biased agent will follow P.

Now we can define the cost of the agent with present-bias β as discrete random variable C_β with $\Pr(C_\beta = W)$ being the probability that the path traversed by the agent is of cost W. Then we can redefine the cost of irrationality as follows.

Definition 3.1 ([DFI22])

The *cost of the irrationality* of the model $M = (G, w, s, t, p, \beta)$ is

$$X_\beta = \frac{C_\beta}{d(s, t)}.$$

Let us note that when no ties occur, our definition coincides with the definition of Kleinberg and Oren. Estimating the cost of irrationality X_β could help the task-designer to evaluate the chances of *abandonment*, the situation when an agent realizes that accomplishing the task takes much more effort than he presumed initially, and thus ultimately gives up.

3.1 Contribution

We introduce the randomized version of the cost of irrationality and initiate its study from the computational perspective. To support our point of view on the cost of irrationality, we start from the combinatorial result, showing that there are time-inconsistent planning models with exponentially (in n) many feasible paths of different costs. It yields that in the deterministic model of Kleinberg and Oren (Definition 2.4) there could be exponentially many different costs of irrationality.

To study the cost of irrationality X_β, we define the following computational problem.

Estimating the Cost of Irrationality (ECI)

Input: A time-inconsistent planning model $M = (G, w, s, t, p, \beta)$, and $W \geq 0$.
Task: Compute $\Pr(X_\beta \leq W)$.

We show that ECI is #P-hard. Thus computationally, ECI is not easier than counting Hamiltonian cycles, counting perfect matching, satidying assignments, and all other #P-complete problems. Our hardness proof strongly exploits the fact that the edge weight w of the model are exponential in the n, the number of vertices of G. We show that when the edge weights are bounded by some polynomial of n, then ECI is solvable in polynomial time. We also obtain polynomial time algorithms, even for exponential weights, for the important "border" cases: minimum, maximum, and average. More precisely, we prove that each of the following tasks

(a) finding the minimum value W such that $\Pr(X_\beta \leq W)$ is positive and computing $\Pr(X_\beta \leq W)$,

(b) finding the minimum value W such that $\Pr(X_\beta \leq W) = 1$, and

(c) computing $\mathbf{E}(X_\beta)$,

can be done in polynomial time.

We also take a look at ECI from the perspective of structural parameterized complexity. Structural parameterized complexity is the common tool in graph algorithms for analyzing intractable problems. We prove the following

- ECI is #P-hard even when in the inconsistent planning model $M = (G, w, s, t, p, \beta)$, we have $\mathrm{tw}(G) = 2$.
- ECI is W[1]-hard parameterized by $\mathrm{fvs}(G)$ and by $\mathrm{vc}(G)$. On the other hand, ECI is solvable in times $n^{\mathcal{O}(\mathrm{fvs}(G))}$ and $n^{\mathcal{O}(\mathrm{vc}(G))}$.

On the other hand, when parameterized by the minimum size of the feedback edge set of the underlying graph, $\mathrm{fes}(G)$, that is the set of edges whose removal makes the graph acyclic, the problem becomes fixed-parameter tractable.

4 Motivation to Complete Tasks

4.1 Motivate by deletion

In this section we study the complexity of the T-path-Deletion problem. We show that it is NP-hard, as well as W$[1]$-hard with respect to the parameter k and several other parameters that naturally arise in this setting. Also in Theorems 3 and 4 we show that the problem admits an FPT algorithm with respect to the structural parameter fes, and also that by adding a new parameter—the maximum edge length of the path, one can obtain an efficient parameterized algorithm.

To prove hardness, we will reduce the NP-hard problem Shortest Path Most Vital Edges [BFN+19] to our problem. In [GT11] authors prove that the SP-MVE problem is

W[1]-hard with parameter k and polynomial weights on the arcs. The original formulation of the SP-MVE problem assumes an undirected graph, but all the results are preserved for the case of an directed acyclic graph.

Shortest Path Most Vital Edges (SP-MVE)

Input: A directed acyclic graph $G = (V, E)$ with positive arcs lengths, two vertices $s, t \in G$, and integers $k, \ell \in \mathbb{N}$.
Task: Is there an arc subset $S \subseteq E, |S| \leq k$, such that the length of a shortest s-t path in $G - S$ is at least ℓ?

The following theorem rules out algorithms with a running time of $f(k) \cdot |V(G)|^{O(1)}$ for T-path-Deletion, for any function f of k only.

Theorem 4.1

T-path-Deletion is W[1]-hard parameterized by k for any $\beta \leq 1$ even when T consists of a single arc and the weights of arcs are polynomial in $|V(G)|$.

Proof. We construct a parameterized reduction from the SP-MVE problem to the T-path-Deletion problem. Let (G, s, t, k, ℓ) be an input of SP-MVE such that weights of arcs of G are bounded by a polynomial in the number of vertices of the G. We construct an instance of T-path-Deletion $M = (G', w, s', t', \beta, r)$, integer k' and a set of arcs $T \subseteq E(G)$ such that in G' at most k arcs can be removed to motivate the agent to pass along the T-path if and only if in G it is possible to remove at most k arcs so that the shortest path between s and t is at least ℓ.

We construct graph G' from G as follows. We start the construction of G' by multiplying all the arcs' weights of G by 2. Then we add new vertices s', v_1, t' and arcs with the following weights: $w(s'v_1) = 0$, $w(v_1t') = 2\ell - 1$, $w(s's) = 0$, and $w(tt') = 0$, see Fig. 4. We put one prescribed arc $T = \{s'v_1\}$, parameter $k' = k$, and reward $r = \frac{2\ell}{\beta}$. Finally, we make arc $s's$ and tt' of multiplicity $k + 1$. Thus G' has $|V(G)| + 3$ vertices and $|E(G)| + 2k + 4$ arcs.

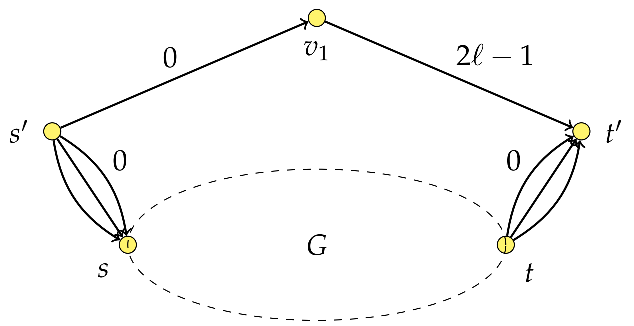

Figure 4: The construction of the graph G' for Theorem 4.1.

Such a construction could clearly be done in time polynomial in k and $|V(G)|$. Thus, to prove that this is an FPT-reduction, it remains to show that the reduction transforms an instance of T-path-Deletion into an equivalent instance of SP-MVE. In other words, we have to prove that $(M = (G', w, s', t', \beta, r), k', T)$ is a yes-instance of T-path-Deletion if and only if (G, s, t, k, ℓ) is a yes-instance of SP-MVE.

First, the principal wants the agent to pass through $T = s'v_1$ and thus through the path $s'v_1t'$. Hence, none of the arcs of this path could be removed. Second, in G' arcs $s's$ and tt' are of multiplicity $k+1$, but the principal could remove at most $k' = k$ arcs. Therefore, it is safe to assume that in every solution to SP-MVE, none of these arcs is removed. This allows us to conclude that the only arcs the principal could remove to reach his goal are from G. Let D be the set of k arcs deleted by the principal from G.

The agent starts at vertex s'. Currently, the perceived cost of the upper path $s'v_1t'$ is $\beta(2\ell - 1)$. If the agent moves to v_1, he will follow the path $s'v_1t'$ because the perceived reward $\beta \cdot r$ is always more than the perceived costs along this path at each step. The only reason why the agent decides not to follow this path is that there is another path in $G' - D$ with a smaller estimated cost, which should be at most $\beta(2\ell - 2)$. The arcs $s's$ and tt' are of zero costs, and the principal reaches his goal if and only if graph $G' - D$ has no path from s to t of cost at most $2\ell - 2$. Since in G' the weights of the arcs taken from G are twice their original weights in G, it means that the agent will move to v_1 instead of s' in $G' - D$ (and thus will follow the plans of the principal) if and only if the length of a shortest s-t path in $G - D$ is at least ℓ. □

Corollary 4.1

T-path-Deletion problem is W[1]-hard parameterized by the number of light arcs (set of arcs that have the minimum weight in the instance).

Proof. We use reduction from Theorem 4.1. We turn all arcs of cost more than 1 into a sequence of arcs of weight 1 in graph G' Fig. 4. While the size of the graph increases, it is still bounded by a polynomial of the size of the original graph. In the new graph, we have arcs of only two weights, and the number of light arcs equals $2k + 3$. □

Theorem 4.1 can also be generalized to the case of any constant $|T| \geq 1$, we can split the arc $s'v_1$ into path $s'u_1 \dots u_h v_1$ with zero arcs where h is a constant and set $T = \{s'u_1, u_1u_2, \dots, u_hv_1\}$. It means that T-path-Deletion is W[1]-hard parameterized by k with any constant $|T| \geq 1$. On the other hand, the problem is also W[1]-hard parameterized by k with an empty set T, see Theorem 4.3.

The lower bound established in Theorem 4.1 immediately questions whether there is a potential for more refined parameterizations to yield parameterized tractability. Unfortunately, the problem is Para-NP-hard for many natural parameters like the value of the reward r or the cost of an T-path. That is, the problem remains NP-hard even when these parameters are constants. We summarize these results in the following theorem.

Theorem 4.2

T-path-Deletion problem remains NP-hard even when one of the following conditions holds.

1. The costs of any T-paths in M does not exceed $C \leq 6$.
2. The reward r is a constant that does not exceed 48.
3. There is a unique T-path in G.
4. All arcs in G but one are of weight 1.
5. Any path from s to t contains at most $m = 8$ arcs.

Proof. The proof is similar to the Theorem 4.1 except for some parameters. We construct a parameterized reduction from the SP-MVE problem to the T-path-Deletion problem. Let (G, s, t, k, ℓ) be an input of SP-MVE such that weights of arcs of G are bounded by a polynomial in the number of vertices of the G. We construct an instance of T-path-Deletion $M = (G', w, s', t', \beta, r)$, integer k' and a set of arcs $T \subseteq E(G)$ such that in G' at most k arcs can be removed to motivate the agent to pass along the T-path if and only if in G it is possible to remove at most k arcs so that the shortest path between s and t is at least ℓ.

We construct graph G' from G as follows. We add new vertices s', v_1, v_2, t' and arcs with the following weights: $w(s'v_1) = \ell/2$, $w(v_1t') = 1$, $w(s'v_2) = 1$, $w(v_2s) = \ell$ and $w(tt') = 1$, see Fig. 5. We put one prescribed arc $T = \{s'v_1\}$, parameter $k' = k$, and reward $r = \frac{\ell}{\beta}$. Finally, we make arc $s'v_2$, v_2s and tt' of multiplicity $k+1$. Thus G' has $|V(G)| + 4$ vertices and $|E(G)| + 3k + 5$ arcs.

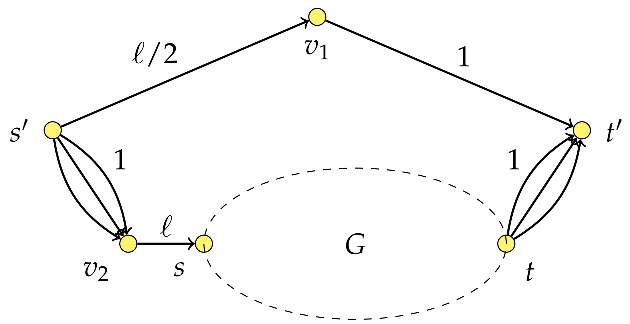

Figure 5: The construction of the graph G' for Theorem 2.

Note that the agent will not remain at the vertex s' since the anticipated cost of the path $s'v_1t'$ is less than βr. Let's set the agent's estimates at the vertex s':

$$1 + \beta\ell + \beta(\ell - 1) + \beta < \frac{\ell}{2} + \beta < 1 + \beta\ell + \beta\ell + \beta.$$

Then $\dfrac{\ell - 2}{4\ell} < \beta < \dfrac{\ell - 2}{4\ell - 2}$, we set β any rational number in this interval.

We now show that the answer to the SP-MVE problem is positive if and only if the answer to the T-path-Deletion problem is positive. Thus, the agent will choose to go to graph G if and only if there is a path between s and t of length no more than $\ell - 1$. Hence, in graph G', at most k edges can be removed to motivate the agent to pass along the T-path if and only if in a graph G, it is possible to remove at most k edges so that the shortest path between s and t is at least ℓ.

Let's look at the complexity of the problem with in terms of natural parameters.

1. It is proven in [BFN+19] that Shortest Path Most Vital Edges problem is NP-hard with $\ell \geq 9$ for undirected graphs. It can be easily seen that the result also holds for directed acyclic graphs. Hence, in the reduction we solve a NP-hard problem with the value of the parameter $\ell = 10$ using the solution of the T-path-Deletion problem with parameter cost of the T-path P: $\ell' = \ell/2 + 1 = 6$.

2. We know that $\frac{\ell-2}{4\ell} < \beta < \frac{\ell-2}{4\ell-2}$, so $\frac{(4\ell-2)\cdot\ell}{\ell-2} < r < \frac{\ell\cdot 4\ell}{\ell-2}$, since $r = \frac{\ell}{\beta}$. When $\ell = 10$ we have $47\frac{1}{2} < r < 50$. Thus, the T-path-Deletion problem is NP-hard with $r = 48$.

3. In graph G' there is only one T-path sv_1t.

4. According to [BFN+19], Shortest Path Most Vital Edges remains NP-hard on directed graphs with edges of unit wights. In this way, all arcs in graph G will be of unit weight, and the arc v_2s is split into ℓ unit arcs with multiplicities of $k+1$. In the proof of Theorem 2 we have all arcs except for one of weight 1.

5. Shortest Path Most Vital Edges problem is NP-hard on the graphs in which any path from s to t contains at most 5 arcs [BFN+19]. Thus, in graph G', any path from s' to t' contains no more than 8 arcs.

□

Theorem 4.3

T-path-Deletion is W[1]-hard parameterized by k even when T is empty set.

Proof. We use the same reduction from Theorem 2 except for the set $T = \emptyset$. Let's look at the agent's decision at the vertex v_2, the agent evaluates any path $v_2s\ldots tt'$ at least as $\ell+\beta$, which is more than expected reward $\beta r = \ell$, so if the agent goes to the vertex v_2, then he will never reach the vertex t'. □

The lower bounds of Theorem 4.1 and Theorem 4.2 create an impression that no efficient algorithms for T-path-Deletion could exist for any reasonable scenario. Despite that, we can identify two interesting parameterizations that make the problem computationally tractable. The first parameter models the natural situation when any sequence of tasks, either taken or anticipated by the agent, contains a bounded number of steps m. In other words, in this model we assume that in the input graph G, any path from s to t contains at most m edges. Although our problem is Para-NP-hard for the parameter m and W[1]-hard parameterized by k. Our next theorem provides an FPT algorithm parameterized by k and m.

Theorem 4.4

T-path-Deletion problem is solvable in time $\mathcal{O}(m^{2k}) \cdot poly(|M|)$.

Proof. To prove the theorem, we employ the classic technique of parameterized algorithms, namely branching. The idea is to identify a subgraph H of G with at most m^2 arcs such that if the principal can motivate the present-biased agent to move over edges of T by removing a set D of at most k arcs, then at least one arc of D should be from H.

Consider how the present-biased agent navigates from s to t in graph G. If the agent's path includes all arcs from T, there is no need for the principal to delete any arc from G, so we set $D = \emptyset$. Otherwise, we construct a subgraph H of G as follows.

Let $P_0 = sv_1v_2\cdots v_p$, $p \le m$, be the path along which the present-biased agent traverses in G from s to t (perhaps not reaching the vertex t if the agent abandons the project at vertex v_p). When standing at a vertex v_i, $1 \le i \le p-1$, the agent evaluates (with a present bias) all possible paths from v_i to t. We pick up a path P_i of the minimum perceived cost $\zeta_M(P_i)$ from v_i to t. Then we define the graph H as the union of paths $H = \cup_{i=0}^{p-1} P_i$. For every $0 \le i \le p-1$, path P_i has at most $m-i$ arcs. Thus, the number of arcs in subgraph H does not exceed $\sum_{i=0}^{m}(m-i) \le m^2$. Since computing the perceived cost of a path could be done in polynomial time [KO14], the time required to construct graph H is polynomial in the input size.

Let $D \neq \emptyset$, $|D| \le k$, be the arcs the principal deletes to achieve his goals. We claim that at least one arc of D is from H. Indeed, if this is not the case, then the minimum value $\zeta_M(v_i)$ for each vertex v_i in graph $G - D$ does not change. Hence, if none of the arcs of H are deleted, the agent will traverse $G - D$ along the path P_0 and thus will not traverse all arcs from T.

This suggests the following branching algorithm. We go through all arcs of H. By the above arguments, we know that at least one of the arcs, say e, is in D. Thus, for the correct guess of the arc e, we have that $M = (G, w, s, t, \beta, r)$ with parameter k is a yes-instance if and only if $M' = (G - e, w, s, t, \beta, r)$ with parameter $k - 1$ is a yes-instance. In other words, we employ the following branching algorithm:

(i) Compute graph H and branch into $|E(H)| = \mathcal{O}(m^2)$ subproblems, corresponding to removing an arc from G and reducing the parameter by 1. (ii) Repeat the procedure recursively. That is, in polynomial time, we find a new path of the agent P' in graph $G' := G - e$ and check if it contains all the selected arcs. If yes, then we stop; otherwise, go to step (i).

To analyze the running time of the algorithm, we obtain a branching tree of depth k and arity at most m^2, and thus with $\mathcal{O}(m^{2k})$ nodes. For each tree node, we compute graph H, which is done in time polynomial in $|M|$. Thus, the running time of the algorithm is $m^{2k} \cdot \mathrm{poly}(|M|)$. $\square$

Our second algorithmic result about T-path-Deletion concerns the limited number of situations when an agent could change a decision. Let us note that the agent could change his mind only when he is on a vertex of some cycle of the underlying undirected graph. The following parameterization concerns the situation when the underlying undirected graph has few edge-disjoint cycles.

A *feedback edge set* of an undeirected graph G is the set of edges whose removal makes G acyclic. For a directed graph G, we use $\mathrm{fes}(G)$ to denote the minimum size of a feedback edge set of the underlying undirected graph of G. Equivalently, $\mathrm{fes}(G)$ is the *cyclomatic* number of the underlying graph. Note that if G is weakly connected, that is, the underlying graph of G is connected, then $\mathrm{fes}(G) = |E(G)| - |V(G)| + 1$.

We consider kernelization for T-path-Deletion parameterized by $\mathrm{fes}(G)$. Recall that a *kernelization* algorithm, given an instance (x, k) of some parameterized problem, runs in polynomial time and outputs an equivalent instance (x', k') of the same problem such that $|x'|, k' \leq f(k)$, for some function f. This instance (x', k') is called the *kernel*, and function f is called the size of the kernel. Kernelization is one of the fundamental tools for parameterized algorithms, and it is well known that a problem admits a kernel from parameter k if and only if there exists an FPT algorithm for it from this parameter. We refer to books [CFK+15, FLSZ19] for further expositions of kernelization.

It is convenient for us to consider the more general variant of our problem, called T-path-Deletion with False Promises, where promised rewards for distinct vertices may be distinct. Formally, we consider time-inconsistent planning models $M = (G, w, s, t, \beta, r)$ where $r\colon V(G) \to \mathbb{R}_{\geq 0}$. In this variant of the model, the agent occupying a vertex v abandons the project if $\zeta_M(v) > \beta \cdot r(v)$. Note that T-path-Deletion is a special case of T-path-Deletion with False Promises where $r(v) = r$.

To obtain a kernel, we have to compress weights and rewards. For this, we use the approach proposed by Etscheid et al. [EKMR17] that is based on the result of Frank and Tardos [FT87].

Proposition 4.1 ([FT87]). *There is an algorithm that, given a vector* $\mathsf{w} \in \mathbb{Q}^d$ *and an integer* N*, in polynomial time finds a vector* $\overline{\mathsf{w}} \in \mathbb{Z}^d$ *with* $\|\overline{\mathsf{w}}\|_\infty \leq 2^{4d^3} N^{d(d+2)}$ *such that* $\mathrm{sign}(\mathsf{w} \cdot b) = \mathrm{sign}(\overline{\mathsf{w}} \cdot b)$ *for all vectors* $b \in \mathbb{Z}^d$ *with* $\|b\|_1 \leq N - 1$.

Theorem 4.5

There is a polynomial-time algorithm that, given an instance of T-path-Deletion with False Promises, outputs an equivalent instance where the graph has at most $8\,\mathrm{fes}(G) + 3$ vertices and at most $9\,\mathrm{fes}(G) + 2$ arcs. Moreover, if the weights and rewards are rational, and β is a rational constant which is not a part of the input, then the T-path-Deletion with False Promises problem admits a polynomial kernel when parameterized by the size of a feedback edge set of the input graph.

Proof. Let (M, k, T) be an instance of T-path-Deletion with False Promises with $M = (G, w, s, t, \beta, r)$. Let also $f = \mathrm{fes}(G)$. We apply the following reduction rules.
Rule 1. If there is $v \in V(G) \setminus \{s, t\}$ with $d_{in}(v) = 0$ or $d_{out}(v) = 0$, then set $G := G - v$. Furthermore, if v is incident to an arc from T, then stop and return a trivial no-instance of T-path-Deletion with False Promises.

The rule is *safe*, that is, it returns an equivalent instance of the problem because v with $d_{in}(v) = 0$ or $d_{out}(v) = 0$ cannot be involved in any s-t path or agent's evaluation. We apply Rule 1 exhaustively. The next rule is trivially safe.
Rule 2. If t is not reachable from s then stop and return a trivial no-instance.

Notice that if we did not stop after applying the rules then s and t are unique source and target, respectively, of G. In particular, for each vertex v, v is reachable from s and t is reachable from v. The next rule is crucial for kernelization.
Rule 3. If G has a path xyz such that $d_{out}(x) = 1$ and $d_{in}(y) = d_{out}(y) = 1$ then

- delete y and add an arc xz,
- set $w(xz) := w(xy) + w(yz)$,
- if $T \cap \{xy, yz\} \neq \emptyset$ then set $T := (T \setminus \{xy, yz\}) \cup \{xz\}$,
- set $r(x) := \min\{r(x) + \frac{1-\beta}{\beta} w(yz), r(y) + \frac{1}{\beta} w(xy)\}$.

To argue that the rule is safe, assume that the instance (M', k, T') is obtained by the application of the rule from (M, k, T) and denote by w' and r' the obtained weight and reward functions. We claim that the instances are equivalent.

For the forward direction, assume that (M, k, T) is a yes-instance. Then there is a set of arcs D of size at most k such that after removing D from G, the present-biased agent follows a T-path P. We define $D' = (D \setminus \{xy, yz\}) \cup \{xz\}$ if $\{xy, yz\} \cap D \neq \emptyset$, and we set $D' = D$ otherwise. Note that $|D'| \leq |D| \leq k$. We claim that D' is a solution to (M', k, T'). The claim is trivial if P does not contain x because in this case $xy, yz \notin E(P)$. Assume that this is not the case and $x \in V(P)$. Let Q be the x-t subpath of P. Because $d_{out}(x) = 1$ and $d_{in}(y) = d_{out}(y) = 1$, xyz is a prefix of Q. Because the agent does not abandon the project, $\zeta_M(x) \leq \beta \cdot r(x)$ and $\zeta_M(y) \leq \beta \cdot r(y)$. Suppose that $r'(x) = r(x) + \frac{1-\beta}{\beta} w(yz)$. Then $\zeta_{M'}(x) = \zeta_M(x) + (1-\beta) w(yz) \leq \beta \cdot r'(x)$. If $r'(x) = r(y) + \frac{1}{\beta} w(yz)$ then $\zeta_{M'}(x) = \zeta_M(y) + w(xy) \leq \beta \cdot r'(x)$. Therefore, the agent occupying x would not abandon the project in the modified graph. Thus, the agent would follow the path Q' obtained from Q by the replacement of xyz by xz. This implies that the path P' obtained from P by the replacement of xyz by xz is a T'-path in $G' - D'$ in the modified instance and the agent should follow it. We conclude that (M', k, T') is a yes-instance.

For the opposite direction, assume that (M', k, T') is a yes-instance and denote by D' a set of arcs of G' of size at most k such that after removing D' from G', the present-biased agent follows a T'-path P'. If $xz \in D'$, we set $D = (D' \setminus \{xz\}) \cup \{xy\}$, and we set

$D = D'$ otherwise. By the definition, $|D| = |D'| = k$. We claim that D is a solution to (M,k,T). Similarly to the proof for the forward direction, the claim is trivial if P' does not contain x. Let assume that $x \in V(P')$. Then $xz \in E(P')$. Denote by Q' the x-t subpath of P'. Since the agent follows Q', $\zeta_{M'}(x) \leq \beta \cdot r'(x)$. Because $r'(x) \leq r(x) + \frac{1-\beta}{\beta} w(yz)$, $\zeta_M(x) = \zeta_{M'}(x) - (1-\beta)w(yz) \leq \beta \cdot r(x)$. Hence, the agent occupying x in G would not abandon the project and go to y. Further, we have that $\zeta_M(y) = \zeta_{M'}(x) - w(xy)$. Because $r'(x) \leq r(y) + \frac{1}{\beta} w(xy)$, $\zeta_M(y) \leq \beta \cdot r(y)$. Therefore,the agent occupying y in G would go to z. We obtain that the agent occupying x in G would follow the path obtained from Q' by replacing of xz by xyz. This implies that the path P obtained from P' by the replacement of xz by xyz is a T-path in $G - D$ and the agent should follow it. Thus, (M,k,T) is a yes-instance. This concludes the safeness proof.

Rule 3 is applied exhaustively whenever possible. Assume from now that Rules 1, 2, and 3 cannot be applied to (M,k,T) with $M = (G,w,s,t,\beta,r)$. Observe that the rules cannot increase the feedback edge set of the underlying graph, that is, $\mathsf{fes}(G) \leq f$. We show the following claim.

Claim 4.1

$|V(G)| \leq 8\,\mathsf{fes}(G) + 3$ and $|E(G)| \leq 9\,\mathsf{fes}(G) + 2$

Proof of Claim 4.1. Denote by H the underlying undirected graph of G.

We observe that H has no adjacent vertices of degree two in $V(H) \setminus \{s,t\}$. To see this, assume that that x and y are adjacent vertices of degree two distinct from s and t. We assume without loss of generality that $xy \in E(G)$. Notice that it cannot happen that $d_{in}(x) = 0$ in G because of Rule 1. Hence, $d_{out}(x) = 1$. Similarly, $d_{out}(y) = 1$. Because G is acyclic and $y \neq t$, y has a neighbor z distinct from x. However, this means that we would be able to apply Rule 3 for the path xyz contradicting our assumptions that the rules cannot be applied.

Let F be a set of edges of size $\mathsf{fes}(G)$ such that $R = H - F$ is acyclic. Because G is weakly connected, R is a tree. Denote by X the set containing s, t, and the endpoints of the edges of F. Note that $|X| \leq 2\,\mathsf{fes}(G) + 2$. Observe that all the leaves of R are in X because of Rule 1. It is a folklore observation that a tree with ℓ leaves has at most $\ell - 2$ vertices of degree at least three. Thus, R has at most $2\,\mathsf{fes}(G)$ vertices $v \in V(H) \setminus X$ of degree at least three. The degrees of vertices in $V(H) \setminus X$ are the same in H and R. Hence, H has at most $2\,\mathsf{fes}(G)$ vertices of degree at least three outside X. By our observation that H has no adjacent vertices of degree two distinct from s and t, we obtain that H has at most $4\,\mathsf{fes}(G) + 1$ vertices $v \in V(H) \setminus X$ of degree two because R is a tree. Therefore, the total number of vertices of G is at most $8\,\mathsf{fes}(G) + 3$. Because R has at most $8\,\mathsf{fes}(G) + 2$ edges, the number of arcs of G is at most $9\,\mathsf{fes}(G) + 2$. This concludes the proof. □

Since Rules 1, 2, and 3 can be applied in polynomial time, Claim 4.1 concludes the proof of the first part of the theorem.

To show the second claim, assume that the weights and rewards are rational and $\beta = p/q$ is a constant. Consider the vector $\mathsf{w} \in \mathbb{Q}^d$ for $d = |V(G)| + |E(G)|$ whose elements are the values of the reward function r for the vertices of G and the weights of arcs. We define $N = d \max\{p,q\} - 1$. Then we apply the algorithm of Frank and Tardos from Proposition 4.1. The algorithm outputs a vector $\overline{\mathsf{w}} \in \mathbb{Z}^d$ and we replace the rewards and the weights by the corresponding values of the elements of $\overline{\mathsf{w}}$. We have that $\mathsf{sign}(\mathsf{w} \cdot b) = \mathsf{sign}(\overline{\mathsf{w}} \cdot b)$ for all vectors $b \in \mathbb{Z}^d$ with $\|b\|_1 \leq N - 1$. In particular, the equality holds for vectors b whose elements are $0, \pm p, q$. This implies that the replacements of the rewards and weights creates an equivalent instance. Because the rewards and weights are upper-bounded by $2^{4d^3} N^{d(d+2)}$ and $d = \mathcal{O}(\text{fes}(G))$, we obtain that each numerical parameter can be encoded by a string of length $\mathcal{O}(\text{fes}(G)^3)$. We conclude that the algorithm outputs an instance of T-path-Deletion with False Promises of size $\mathcal{O}(\text{fes}(G)^4)$. This means that we have a polynomial kernel. This completes the proof. $\square$

In the second part of Theorem 4.5, we assume that β is a rational constant which is not a part of the input. However, it can be observed that the claim holds if $\beta = p/q$ for integers $p, q \leq 2^{\text{fes}(G)^c}$ for some constant c. Also, we note that because T-path-Deletion is NP-complete for rational weights and any rational positive constant $\beta < 1$, any problem from NP can be reduced to T-path-Deletion in polynomial time. This implies the following corollary.

Corollary 4.2

If the weights are rational and β is a rational constant which is not a part of the input, then T-path-Deletion admits a polynomial kernel when parameterized by the size of a feedback edge set of the input graph.

Also, we can solve T-path-Deletion in FPT time using the algorithm from Theorem 4.5—we reduce an instance of T-path-Deletion to an equivalent instance of T-path-Deletion with False Promises with a graph of bounded size and guess a solution.

Corollary 4.3

T-path-Deletion can be solved in $2^{\mathcal{O}(\text{fes}(G))} \cdot n^{\mathcal{O}(1)}$ time.

4.2 Motivate by addition

In this section, we show that the T-path-Addition problem is computationally hard with respect to the number of edges added even on the simplest type of instances when the initial graph is a path whose edges form T and only detours are allowed to be added—edges whose start and end belong to the path. In this case, we assume that all the arcs we add go from left to right. We will call such inputs a *path with detours*. To prove this result, we need the following problem:

Modified k-Sum

Input: Sets of positive integers $X_1, X_2, \ldots, X_k$ and integer Z.
Task: Decide whether there is $x_1 \in X_1$, $x_2 \in X_2$, $\ldots$, $x_k \in X_k$ such that $x_1 + \cdots + x_k = Z$.

It is known [DFI22] that this problem is W[1]-hard with respect to the parameter k.

Theorem 4.6

T-path-Addition problem on the path with detours instances is W[1]-hard parameterized by k.

Proof. We construct a parameterized reduction of the Modified k-Sum problem to the T-path-Addition problem. Let $X_1, X_2, \ldots, X_k$ and Z be an instance of the Modified k-Sum problem. We transform the input such that all elements $x_j \in X_i$ are in the interval $[b, 2b]$. For that we assign $b = \max_{i=1,\dots,k} \max_{x_j \in X_i} x_j$, add b to all elements x_j, and set $Z := Z + kb$.

Then we construct an instance of T-path-Addition.

- Parameter k is unchanged.
- Graph G is the path on $2k+4$ vertices $v_1 v_2 v_3 \dots v_{2k+4}$. We assign a reward r to the vertex $t = v_{2k+4}$ (see Fig. 6).
- We set additional arcs $A = \bigcup_{i=1}^{k} X_i' \bigcup \{v_2v_4, v_3t\}$, where X_i' is the set of multiple arcs $v_{2i+2}v_{2i+4}$, and the weights of these arcs are numbers from the set X_i. The weight of arc v_2v_4 is 1 and the weight of arc v_3t is $Z + \frac{1}{\beta} - 1$.

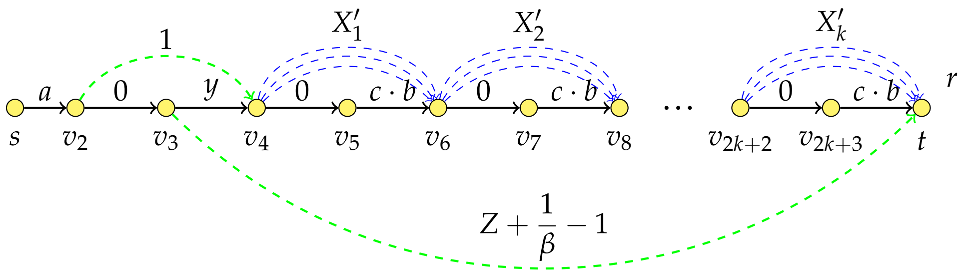

Figure 6: Construction of the graph for reduction in Theorem 4.6.

To make it easier to describe, let us first assume that our input graph G is not just a path but a path with two additional arcs $\{v_2v_4, v_3t\}$ (colored in green in the Fig. 6). We will show how to build a reduction for this type of input, and then we will explain why it is possible not to add these two arcs to the path, but to give them in a set of additional arcs A. Note that we take the values of k, b and Z from the input of the Modified k-Sum problem. We will select the parameters a, c, β, r, y during the proof.

Initially, we want that the agent in graph G is not motivated even to leave the start vertex $s = v_1$. For this we put

$$\begin{cases} a + \beta \cdot (1 + k \cdot c \cdot b) > \beta \cdot r, \\ a + \beta \cdot (y + k \cdot c \cdot b) > \beta \cdot r, \\ a + \beta \cdot (Z + \dfrac{1}{\beta} - 1) > \beta \cdot r. \end{cases} \tag{1}$$

For our purposes, we will consider only the values of $y \geq \frac{1}{\beta} > 1$. Therefore, in (1), it is sufficient to satisfy only the first and the third inequalities. To motivate the agent to move from vertex s, it is necessary to add some path from the vertex v_4 to t along the arcs from $\{X_i\}_{i=1}^{k}$. We denote this path by $\widehat{P}$ and its cost, that is the sum of the costs of its arcs, by $w(\widehat{P})$. For an agent to decide to move from the vertex s, it is necessary that his initial estimate be no more than the expected reward, namely:

$$a + \beta \cdot (1 + w(\widehat{P})) \leq \beta \cdot r. \tag{2}$$

But on the other hand, when the agent reaches vertex v_2, we need him to continue his path to vertex v_3, and not immediately turn into the green arc that leads to vertex v_4.

From here, we impose the following constraint on the cost of the added path:

$$1 + \beta \cdot w(\widehat{P}) > \beta \cdot (Z + \frac{1}{\beta} - 1). \tag{3}$$

Similarly, the condition that the agent does not leave the black path destined for him at vertex v_3 is that

$$y + \beta \cdot w(\widehat{P}) < Z + \frac{1}{\beta} - 1. \tag{4}$$

We put $c = 2k$, $a = k \cdot c \cdot b + 1 = 2k^2 b + 1$, $r = Z + \frac{a}{\beta} + 2 - \varepsilon$, where ε—rational number between 0 and 1. Then it is easy to check that all inequalities from system (1) are satisfied. We also should select the values of y and $w(\widehat{P})$ to satisfy the remaining inequalities (2)–(4). Note that (2) is equivalent to $w(\widehat{P}) \leq r - 1 - \frac{a}{\beta} = Z + 1 - \varepsilon$ and (3) is equivalent to $w(\widehat{P}) > (Z + \frac{1}{\beta} - 1) - \frac{1}{\beta} = Z - 1$.

Because the weights of all edges in our construction are integers, we have that under the already existing restrictions, if the agent could go along the black path, then we added the path from v_4 to t along the arcs from $\{X_i\}_{i=1}^{k}$ of cost exactly Z. In each X_i the numbers are in the interval from $b > 0$ to $2b$, hence the value Z does not exceed $2bk$. It follows that if we have found the v_4-t path of cost Z, then we have added at least one of the proposed arcs in each gadget since the initial arcs have costs $c \cdot b = 2bk \geq Z$.

The inequality (4) is equivalent to $y < (Z - \beta \cdot w(\widehat{P})) + \frac{1}{\beta} - 1$. Now let us analyze what will happen after the agent reaches vertex v_4. It is obvious that at all subsequent vertices, he will be motivated to move on (the estimate of his remaining path will not exceed the expected reward $\beta \cdot r$, since the estimate of the path does not exceed his estimate at vertex s, from which he decided to move on). We only need to make sure that in each gadget the agent does not turn into a blue arc, but continues along the black ones. Since the minimum value of an arc in any X_i is b, then the following inequality guarantees that the agent will make the right choice in each gadget:

$$\forall i \geq 4 \quad 0 + \beta \cdot (c \cdot b + w(v_{i+2}\text{-}t \text{ path})) < b + \beta \cdot w(v_{i+2}\text{-}t \text{ path}).$$

$$\beta < \frac{1}{c} = \frac{1}{2k}.$$

Thus, if we take a present bias coefficient $\beta < 1/2k$ into the input of the T-path-Addition problem, then the agent will follow the black path if and only if it is possible to assemble a path of cost exactly Z from the arcs $x_1 \in X_1, x_2 \in X_2, \ldots, x_k \in X_k$, or, which is the same when the answer to the Modified k-Sum problem is "yes".

Now we will show how to fine-tune the parameters so that we do not have to force the addition of green arcs to the graph but provide them within the addition set, thus leaving only the path as the input.

In order to add arc v_2v_4, it is necessary that even with the shortest path from vertex v_3 to vertex t, the agent is not motivated to move from s. In other words,

$$a + \beta \cdot (y + k \cdot b) > \beta \cdot r.$$

$$y > r - \frac{a}{\beta} - kb = Z - kb + 2 - \varepsilon.$$

To add arc v_3t, we need that at vertex v_2, the agent turns to the already added arc v_2v_4. That is

$$0 + \beta \cdot y > 1.$$

Finally, we set

$$\begin{cases} c = 2k, \\ a = 2k^2b + 1, \\ \beta < \frac{1}{2k}, \\ r = Z + 2 + \frac{a}{\beta} - \varepsilon, \\ Z - kb + 2 - \varepsilon < y \le Z - \beta \cdot w(\widehat{P}) + \frac{1}{\beta} - 1. \end{cases}$$

The last double inequality has a solution for y, since the left hand side is less than the right hand side:

$$Z - kb + 2 - \varepsilon < Z - \beta \cdot w(\widehat{P}) + \frac{1}{\beta} - 1.$$

$$\beta w(\widehat{P}) - \frac{1}{\beta} + 3 < kb + \varepsilon.$$

We show that starting from some k the desired inequalities hold.

$$\beta w(\widehat{P}) - \frac{1}{\beta} + 3 < \frac{w(\widehat{P})}{2k} - 2k + 3 \le b - 2k + 3 < kb + \varepsilon.$$

□

By Theorem 4.6, T-path-Addition is difficult even in the particular case when the set of selected tasks is a Hamiltonian path. On the other hand, the problem is solvable in time $2^{|A|}n^{\mathcal{O}(1)}$ by going through all potential solutions $S \subseteq A$, $|S| \le k$, and checking in polynomial time whether S is a solution of the problem. Our next theorem generalizes this observation to the situation when the set A has a special structure.

Let us start with an example. Let $c \in [n]$, $V_1 = \{v_1, \dots, v_c\}$, $V_2 = \{v_c, \dots, v_n\}$, and let A do not contain any arc (v_i, v_j) such that $i < c < j$. Let us partition A into two *intersection components* $A_1 = A \cap (V_1 \times V_1)$ and $A_2 \cap (V_2 \times V_2)$, see Fig. 7. We want to show that to solve the problem, we then can solve it on $G[V_1]$ and $G[V_2]$ separately in total time $(2^{|A_1|} + 2^{|A_2|}) \cdot n^{\mathcal{O}(1)}$.

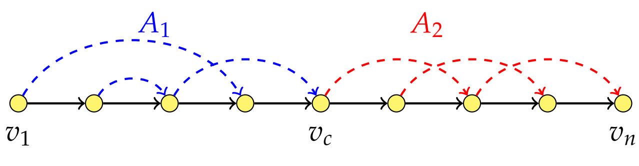

Figure 7: Intersection components of set A for the T-path-Addition problem.

Let $S \subseteq A$, $S_1 = S \cap A_1$, $S_2 = S \cap A_2$, $k_1 = |S_1|$, $k_2 = |S_2|$. Consider the agent's path in $G \cup S$ and also divide it into two parts going through V_1 and V_2, respectively. Notice that in case the agent gets to V_2, the second part of the path is exactly the agent's path in $G[V_2] \cup S_2$. Now let us consider the first part of the path. Notice that for every vertex $v_i \in V_1$, any perceived path induces one of the shortest paths in $G[V_2] \cup S_2$. That means that the agent's decisions depend only on $G[V_1] \cup S_1$ and $dist_{G[V_2] \cup S_2}(v_c, v_n)$, where by $dist_G(s,t)$ we denote weight of the shortest path in G from s to t. Moreover, $dist_{G[V_2] \cup S_2}(v_c, v_n)$ takes part only in comparison of a perceived cost with $\beta \cdot r$ that can be replaced with comparison of the perceived cost in $G[V_1] \cup S_1$ with $\beta(r - dist_{G[V_2]\cup S_2}(v_c, v_n))$, so the first part of the agent's path is exactly the agent's path in $G[V_1] \cup S_1$ with reward $r - dist_{G[V_2]\cup S_2}(v_c, v_n)$.

Hence, there exists a solution S for the initial problem if and only if there exist $k_1, k_2 : k_1 + k_2 \leq k$, $S_1 \subseteq A_1$ of size k_1 and $S_2 \subseteq A_2$ of size k_2 such that in $G[V_2] \cup S_2$ the agent follows path $v_c \dots v_n$ with reward r, and in $G[V_1] \cup S_1$ the agent follows path $v_1 \dots v_c$ with reward $r - dist_{G[V_2]\cup S_2}(v_c, v_n)$. We also notice that for every such (k_1, k_2, S_1, S_2), the above also holds for any (k_1, k_2, S_1, S_2') where $S_2' \subseteq A_2$ of size k_2 such that the agent takes path $v_c, \dots, v_n$ in $G[V_1] \cup S_2'$, and $dist_{G[V_2]\cup S_2'}(v_c, v_n) \leq dist_{G[V_2]\cup S_2}(v_c, v_n)$. That means, that it is sufficient to consider only S_2 that minimizes $dist_{G[V_2]\cup S_2}(v_c, v_n)$.

Now we can solve T-path-Addition in time $(2^{|A_1|} + 2^{|A_2|}) \cdot n^{\mathcal{O}(1)}$ in the following way. For every $k_2 \leq k$ we compute $d[k_2] = \{dist_{G[V_2]\cup S_2}(v_c, v_n) \mid S_2 \subseteq A_2, |S_2| \leq k_2$, the agent follows path $v_c \dots v_n$ in $G[V_2] \cup S_2$ with reward $r\}$. That can be done in time $2^{|A_2|} n^{\mathcal{O}(1)}$ by going through all $S_2 \subseteq A_2$. Then, for every $k_1 \leq k$ we go through all $S_1 \subseteq A_1$, $|S_1| = k_1$ and check whether the agent follows path $v_1 \dots v_c$ in $G[V_1] \cup S_1$ with reward $r - d[k - k_1]$. That can be done in time $2^{|A_1|} n^{\mathcal{O}(1)}$.

Let us now generalize the result. Let $1 = c_1 < c_2 < \dots < c_{m+1} = n$ be such indices that for every $2 \leq \ell \leq m$ there is no edge (v_i, v_j) such that $i < c_\ell < j$. For every $1 \leq \ell \leq m$, let $V_\ell = \{v_{c_\ell}, \dots, v_{c_{\ell+1}}\}$, and let us partition A into intersection components $A_\ell = A \cap (V_\ell \times V_\ell)$. Then we show that the following theorem holds.

Theorem 4.7

T-path-Addition problem on paths with detours can be solved in time $2^\tau n^{\mathcal{O}(1)}$, where τ is the size of the maximum intersection component of A.

Proof. For every $1 \leq \ell \leq m$, let $A_\ell^\cup = A_\ell \cup \dots \cup A_m$ and let $d[\ell][\kappa] = \min\{dist_{G[A_\ell^\cup]\cup S}(v_{c_\ell}, v_n) \mid S \subseteq A_\ell^\cup, |S| \leq \kappa$, the agent follows path $v_{c_\ell} \dots v_n$ with reward $r\}$. If there is no such S then $d[\ell][\kappa]$ is assigned to $+\infty$. Then, to solve the problem it is sufficient to compute all $d[\ell][\kappa]$ and check whether $d[1][k]$ is not equal to $+\infty$.

We compute $d[\ell][\kappa]$ using dynamic programming technique. Using the same argument as in case of $m = 2$ above, we notice that $d[\ell][\kappa] = \min\{dist_{G[A_\ell]\cup S}(v_{c_\ell}, v_{c_{\ell+1}}) + d[\ell+1][\kappa - \kappa'] \mid S \subseteq A_\ell, \kappa' \leq \kappa$, the agent follows path $v_{c_\ell} \dots v_{c_{\ell+1}}$ in $G[A_\ell] \cup S$ with reward $r - d[\ell + 1][\kappa - \kappa']\}$. We start with $\ell = m$ and iteratively decrease it after exhausting all possible $\kappa \leq k$. Then, we can compute all $d[\ell][\kappa]$ and solve the problem in time $2^\tau n^{\mathcal{O}(1)}$. □

5 Cost Reduction

5.1 Modification by deletion

In this section we study the complexity of the Opt-REC-Deletion and REC-Deletion problem. We show that it is NP-hard, as well as W[1]-hard with respect to the parameter k and several other parameters that naturally arise in this setting. Also for the optimization version of the problem, we show that there is no $(\frac{1}{\beta} - \varepsilon)^n$–approximate FPT algorithm with parameter k under the assumption $\mathsf{W}[1] \neq \mathsf{FPT}$ for any $\varepsilon > 0$.

5.1.1 Approximation

Lemma 5.1

An empty set gives $(\frac{1}{\beta})^n$-approximation for Opt-REC-Deletion.

Proof. Let us prove that the path chosen by the agent has weight at most $(\frac{1}{\beta})^n \cdot \mathrm{OPT}$, where OPT is the weight of the shortest path from s to t. Then, since the optimal agent's path cannot be shorter than the shortest path in the graph, we get the approximation.

Let us consider the agent's path $(s = v_0, v_1, \dots, v_{h-1}, v_h = t)$. For each $1 \le i \le h$, let a_i be the weight of the edge (v_{i-1}, v_i), and for each $0 \le i \le h$, let OPT_i be the weight of the shortest path from v_i to t. Our goal is to prove that $a_1 + \cdots + a_h \le (\frac{1}{\beta})^n \cdot \mathrm{OPT}_0$. We prove a stronger inequality $a_1 + \dots a_i + \mathrm{OPT}_i \le (\frac{1}{\beta})^i \cdot \mathrm{OPT}_0$ for every i using induction.

For $i = 0$, the inequality holds. For $i \ge 0$, we notice that

$$a_{i+1} + \beta \cdot \mathrm{OPT}_{i+1} \le \mathrm{OPT}_i.$$

Hence,

$$a_{i+1} + \mathrm{OPT}_{i+1} \le \frac{1}{\beta}\mathrm{OPT}_i \Rightarrow$$

$$\sum_{j=1}^{i+1} a_j + \mathrm{OPT}_{i+1} \le \sum_{j=1}^{i} a_j + \frac{1}{\beta} \cdot \mathrm{OPT}_i \le$$

$$\frac{1}{\beta}(\sum_{j=1}^{i} a_j + \mathrm{OPT}_i) \le (\frac{1}{\beta})^{i+1}\mathrm{OPT}_0.$$

□

Corollary 5.1

The Opt-REC-Deletion problem admits $(\frac{1}{\beta})^n$-approximation algorithm working in polynomial time.

Now we show that there is no better approximation factor than $(\frac{1}{\beta})^n$ if $\mathsf{W}[1] \ne \mathsf{FPT}$ or $\mathsf{P} \ne \mathsf{NP}$.

Theorem 5.1

The Opt-REC-Deletion problem is $\mathsf{W}[1]$-hard with respect to the parameter k for any β and there is no $(\frac{1}{\beta} - \varepsilon)^n$–approximate FPT algorithm with parameter k under the assumption $\mathsf{W}[1] \ne \mathsf{FPT}$ for any $\varepsilon > 0$.

Proof. We construct a parameterized reduction from the Shortest Path Most Vital Edges problem to the Opt-REC-Deletion problem.

Let G be the graph given at the input of the problem Shortest Path Most Vital Edges. We construct graph G' as shown in the Fig. 8, add new vertexes s, v_1, v_2, t and edges $w(s, v_1) = a, w(s, v_2) = 0, w(v_2, \tilde{s}) = \ell, w(v_2, t) = 0, w(\tilde{t}, t) = 0$, where vertexes $\tilde{s}, \tilde{t}$ are initial and final vertices of the graph G. Also multiply all the weights in graph G by c.

Let's set the agent's estimates at the vertex s: $\beta\ell + \beta c\ell < a < \beta\ell + \beta c\ell + \beta c$. Let's take $c = \frac{2}{\beta}$, so we may take a as integer. Thus, the agent will choose to go to graph G iff

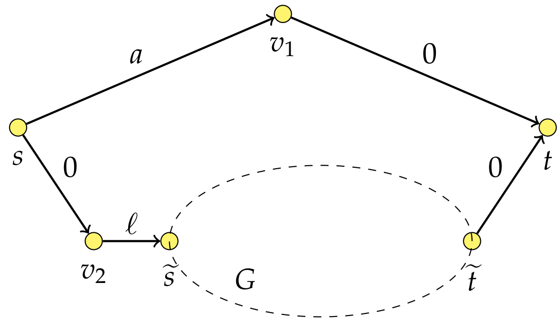

Figure 8: Graph G'.

there is a path between $\widetilde{s}$ and $\widetilde{t}$ of length no more than $c\ell$. For the edges $(v_2, \widetilde{s})$ we add the following gadget shown in the Fig. 9 and set μ so that $\ell > 0 + \beta\mu\ell$ so $\mu < \frac{1}{\beta}$. Therefore, if the agent goes in graph G, then instead of the edge of cost ℓ, she will pass along the edge of cost $\mu^n \ell$.

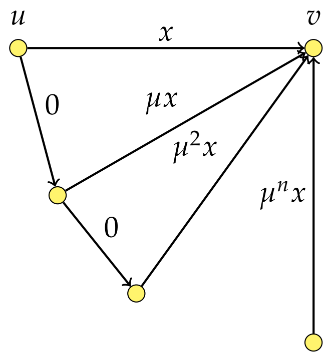

Figure 9: Graph G'.

Thus, if an agent goes through graph G, his path will be at least $\mu^n \ell$, otherwise his path is equal to a and $a < \beta\ell(1+c) + \beta c < \beta\ell + 2\ell + 2 < 3\ell + 2 \leq 5\ell << \mu^n l$. Hence, in graph G' no more than k edges can be removed so that the agent's path becomes no more than a iff in graph G, it is possible to remove at most k edges so that the shortest path between $\widetilde{s}$ and $\widetilde{t}$ is at least $l+1$. There is also no $\dfrac{\mu^n \ell}{5\ell} = (\dfrac{1}{\beta} - \varepsilon)^n$–approximate algorithm for the problem. □

5.1.2 Decision problem

Theorem 5.2

REC-Deletion problem is NP-hard with two weights on the edges.

Proof. The proof is similar to the Theorem 5.1 except for the weights on the edges.

Let's set the agent's estimates at the vertex s:

$$1 + 2\beta l < \frac{1}{2}l + 1 < 1 + \beta l + \beta(l+1).$$

Then $\frac{l-2}{4l} < \beta < \frac{1/2l-1}{2l-1} = \frac{l-2}{4l-2}$. Thus, the agent will choose to go to graph G if and only if there is a path between $\widetilde{s}$ and $\widetilde{t}$ of length no more than l. The shortest path in graph G' is $1/2l+1$. Hence, in graph G' no more than k edges can be removed so that the agent's path becomes no more than $1/2l+1$ iff in graph G, it is possible to remove at most k edges so that the shortest path between $\widetilde{s}$ and $\widetilde{t}$ is at least $l+1$. □

Corollary 5.2

The REC-Deletion problem is Para-NP-hard with various parameters:

1. Agent cost $\ell = 5$.
2. Number of different weights on the edges (in Theorem 5.2 equals 2).
3. Number of heavy edges.
4. Number of vertices at which the agent can evaluate paths via β.

Corollary 5.3

There is no $2^{o(n)}$ algorithm for REC-Deletion under ETH.

Proof. The Shortest Path Most Vital Edges problem does not have $2^{o(n)}$ algorithm under ETH, because the authors [BFN+18] have reduced the vertex cover problem to a problem with $O(n)$ vertices. □

It turns out that if ℓ is small, then the problem can be solved in polynomial time.

Claim 5.1

REC-Deletion can be solved in polynomial time when ℓ equals 1 or 2.

Our next result works for the standard Kleinberg-Oren's model without probabilities. The problem can be reformulated as follows.

Reducing Cost of the Path by Deletion (RCP-Deletion)

Input: Time-inconsistent planning model $M = (G, w, s, t, \beta)$, integer k.
Task: Find a set of at most k edges of the graph G, removing which reduces the cost of the agent's path $c_\beta(s,t)$.

Theorem 5.3

RCP-Deletion problem is FPT from a pair of parameters (k, m) with complexity $m^{2k} \cdot poly(n)$.

Proof. The proof of this Theorem is similar to the Theorem 4.4. □

5.2 Modification by adding

In this section we study the complexity of the Opt-REC-Addition and REC-Addition problem, when the input graph of the problem is a path on n vertices and set A is detours on the path (*path with detours*).

Theorem 5.4

The Opt-REC-Addition problem on paths with detours is in P.

Proof. Let graph G be a path $v_1, \dots, v_n$, and let $E_{add} \subseteq [n] \times [n]$ be a set of edges that can be added.

We use dynamic programming to solve this problem. Let $dp[i][k]$ be the minimum cost of the agent's path from v_i to v_n in $G[\{v_i, v_{i+1}, \dots, v_n\}]$ with at most k added edges from E_{add}.

Consider the minimal set $S \subseteq E_{add}$ of size at most k, that leads to such optimal agent's path from v_i to v_n. It either does not contain any edge from v_i or contains an edge (v_i, v_j) for some $j > i$. Let us consider the latter case and notice that: (i) if agent does not use (v_i, v_j) in its path, then $S \setminus \{(v_i, v_j)\}$ is also a solution, and S is not minimal, so (v_i, v_j) belongs to the agent's path; (ii) if S contains an edge $(v_{i'}, v_{j'})$, where $i \le i' < j$, then $(v_{i'}, v_{j'})$ does not belong to the agent's path and does not motivate the agent at any point, so $S \setminus \{(v_{i'}, v_{j'})\}$ is also a solution, and S is not minimal.

That means that the minimal solution S either does not contain any edge from v_i, or contains exactly one edge (v_i, v_j) such that $w(v_i, v_j) < w(v_i, v_{i+1}) + \beta \sum_{h=i+1}^{j-1} w(v_h, v_{h+1})$, and $S \setminus \{(v_, v_j)\} \subseteq \{v_j, v_{j+1}, \dots, v_n\}^2$. That makes us consider the following dynamic programming:

$$dp[i][k] = \min \begin{cases} dp[i+1][k] + w(v_i, v_{i+1}) \\ \min\limits_{j>i} \{dp[j][k-1] + w(v_i, v_j) \mid (v_i, v_j) \in E_{add}, \\ w(v_i, v_j) < \beta \sum\limits_{h=i+1}^{j-1} w(v_h, v_{h+1})\} \end{cases}$$

Such dynamic programming can be computed in polynomial time, and the answer to opt-REC-by-adding is $dp[1][k]$. □

Corollary 5.4

REC-Addition problem on paths is in P.

Next, we show that the problem is NP-hard for the case when there can be no more than two paths from s to t in the resulting graph.

Theorem 5.5

REC-Addition problem is NP-hard.

5.3 Modification with a budget

In this section, we will explore the complexity of the Reducing Expectation of Cost with the Budget problem. We will be considering a scenario in which we are given a dedicated budget for modifying the weight of edges. We may increase or decrease the weight of these edges until they reach zero weight within our budget. The change in the weight of the edges can only be natural. There are also two settings, in one if the edge has reached zero weight, we leave it, in the other if the edge becomes zero, then it disappears from the graph.

We prove that the REC-Budget problem is NP-hard in two settings with zero arcs. To prove this, we reduce classical NP-hard knapsack problem.

Knapsack

Input: Sets of items with a integer weight $w_1, \ldots, w_n$ and integer value $\ell_1, \ldots, \ell_n$, and limit on the total weight L, and integer W.
Task: Decide whether there is subset of items with total weight no more than L and total value at least W.

Theorem 5.6

REC-Budget problem is NP-hard.

6 Conclusion

In this work, we use the graph-theoretical model of Kleinberg and Oren to introduce the principal-agent problem, where the principal could reduce the choices to guarantee that the agent will accomplish some selected tasks. We conclude with directions for further research and some concrete open problems. While we consider only the scenario of deleting and adding arcs, several other natural models would be exciting to explore. The process of adding and deleting arcs could be simulated by changing the weights of the arcs. This more general model, where the principal could change the weights of the arcs in order to motivate the agent, is much more algorithmically challenging. Another attractive model is where the principal motivates the agent by putting rewards for accomplishing some intermediate tasks like in [AK17].

As a concrete open question, for the T-path-Deletion problem, we obtained a kernel whose size is polynomial in the size of a feedback edge set of G. We do not know if a kernel whose size is bounded by a size (even exponential) of a vertex cover of G exists.

6.1 List of papers

This thesis is based on the following papers.

(i) **Inconsistent planning: When in doubt, toss a coin!**

Yuriy Dementiev, Fedor Fomin, and Artur Ignatiev. AAAI 2022. Conference on Artificial Intelligence (AAAI).

(ii) **How to guide a present-biased agent through prescribed tasks?**

Tatiana Belova, Yuriy Dementiev, Fedor Fomin, Petr Golovach, Artur Ignatiev. Preprint. Under review in ECAI 2024.

References

[AK17] Susanne Albers and Dennis Kraft. On the value of penalties in time-inconsistent planning. In *44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 10:1–10:12, 2017.

[AK19] Susanne Albers and Dennis Kraft. Motivating time-inconsistent agents: A computational approach. *Theory Comput. Syst.*, 63(3):466–487, 2019.

[Ake91] George A. Akerlof. Procrastination and obedience. *American Economic Review: Papers and Proceedings*, 81(2):1–19, 1991.

[BBD06] Pablo Burzyn, Flavia Bonomo, and Guillermo Durán. NP-completeness results for edge modification problems. *Discrete Applied Mathematics*, 154(13):1824–1844, 2006.

[BFN$^+$18] Cristina Bazgan, Till Fluschnik, André Nichterlein, Rolf Niedermeier, and Maximilian Stahlberg. A more fine-grained complexity analysis of finding the most vital edges for undirected shortest paths, 2018.

[BFN$^+$19] Cristina Bazgan, Till Fluschnik, André Nichterlein, Rolf Niedermeier, and Maximilian Stahlberg. A more fine-grained complexity analysis of finding the most vital edges for undirected shortest paths. *Networks*, 73(1):23–37, 2019.

[CDFG23] Christophe Crespelle, Pål Grønås Drange, Fedor V. Fomin, and Petr A. Golovach. A survey of parameterized algorithms and the complexity of edge modification. *Comput. Sci. Rev.*, 48:100556, 2023.

[CFK$^+$15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[DFI22] Yuriy Dementiev, Fedor Fomin, and Artur Ignatiev. Inconsistent planning: When in doubt, toss a coin! *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*, 36(9):9724–9731, Jun. 2022.

[DM22] Sarah Dean and Jamie Morgenstern. Preference dynamics under personalized recommendations. In *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, pages 795–816. ACM, 2022.

[EKMR17] Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems. *J. Comput. Syst. Sci.*, 84:1–10, 2017.

[ESG16] Owain Evans, Andreas Stuhlmüller, and Noah Goodman. Learning the preferences of ignorant, inconsistent agents. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 30, 2016.

[FLO02] Shane Frederick, George Loewenstein, and Ted O'Donoghue. Time discounting and time preference: A critical review. *Journal of Economic Literature*, 40(2):351–401, 2002.

[FLSZ19] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization. Theory of Parameterized Preprocessing.* Cambridge University Press, 2019.

[FS20] Fedor V. Fomin and Torstein J. F. Strømme. Time-inconsistent planning: Simple motivation is hard to find. In *Proceeding of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, pages 9843–9850. AAAI Press, 2020.

[FT87] András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Comb.*, 7(1):49–65, 1987.

[GILP16] Nick Gravin, Nicole Immorlica, Brendan Lucier, and Emmanouil Pountourakis. Procrastination with variable present bias. In *ACM Conference on Economics and Computation (EC)*, page 361, 2016.

[GT11] Petr A. Golovach and Dimitrios M. Thilikos. Paths of bounded length and their cuts: Parameterized complexity and algorithms. *Discret. Optim.*, 8(1):72–86, 2011.

[HS23] Joseph Y Halpern and Aditya Saraf. Chunking tasks for present-biased agents. In *Proceedings of the 24th ACM Conference on Economics and Computation (EC)*, pages 853–884, 2023.

[KO14] Jon M. Kleinberg and Sigal Oren. Time-inconsistent planning: a computational problem in behavioral economics. In *ACM Conference on Economics and Computation (EC)*, pages 547–564, 2014.

[KO18] Jon M. Kleinberg and Sigal Oren. Time-inconsistent planning: a computational problem in behavioral economics. *Commun. ACM*, 61(3):99–107, 2018.

[KOR16] Jon M. Kleinberg, Sigal Oren, and Manish Raghavan. Planning problems for sophisticated agents with present bias. In *ACM Conference on Economics and Computation (EC)*, pages 343–360, 2016.

[KOR17] Jon M. Kleinberg, Sigal Oren, and Manish Raghavan. Planning with multiple biases. In *ACM Conference on Economics and Computation (EC)*, pages 567–584, 2017.

[Lai94] David I. Laibson. *Hyperbolic Discounting and Consumption*. PhD thesis, Massachusetts Institute of Technology, Department of Economics, 1994.

[LSP22] Nixie S. Lesmana, Huangyuan Su, and Chi Seng Pun. Reinventing policy iteration under time inconsistency. *Trans. Mach. Learn. Res.*, 2022, 2022.

[MLLC04] Samuel M. McClure, David I. Laibson, George Loewenstein, and Jonathan D. Cohen. Separate neural systems value immediate and delayed monetary rewards. *Science*, 306(5695):503–507, 2004.

[MPS22] Seth A Meyer, Jessica Pomplun, and Joshua Schill. Present bias in partially sophisticated and assisted agents. *Mathematical Social Sciences*, 118:36–47, 2022.

[NSS01] Assaf Natanzon, Ron Shamir, and Roded Sharan. Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, 113(1):109–128, 2001.

[OR99] Ted O'Donoghue and Matthew Rabin. Doing it now or later. *American economic review*, 89(1):103–124, 1999.

[OS19] Sigal Oren and Dolav Soker. Principal-agent problems with present-biased agents. In *Proceedings of the 12th International Symposium on Algorithmic Game Theory (SAGT)*, pages 237–251. Springer, 2019.

[Sam37] Paul A. Samuelson. A note on measurement of utility. *The Review of Economic Studies*, 4(2):155–161, 02 1937.

[TG15] Richard H Thaler and LJ Ganser. Misbehaving: The making of behavioral economics. 2015.

[TTW+17] Pingzhong Tang, Yifeng Teng, Zihe Wang, Shenke Xiao, and Yichong Xu. Computational issues in time-inconsistent planning. In *Proceedings of the 31st Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2017.