Optimal world design in video games

Yifu Li
International Institute of Finance, School of Management, University of Science and Technology of China, yifuli@ustc.edu.cn

Christopher Thomas Ryan
UBC Sauder School of Business, University of British Columbia, chris.ryan@sauder.ubc.ca

Lifei Sheng
College of Business, University of Houston-Clear Lake, sheng@uhcl.edu

Benny Wong
Datadog,benny.wong0623@gmail.com

Spending time in virtual spaces is a growing part of the human experience. We study the design of virtual spaces in a video game context, with an emphasis on understanding how people spend more or less time enjoying these spaces. People enjoy spending time immersed in a video game world but also want a sense of achievement. When deciding how to chart a meaningful path through a virtual world, game players confront a series of choices. An effective design of a virtual world must balance two things. First, the world should be flexible to differing time budgets of players. Second, complex designs can overwhelm players with decision fatigue. We model virtual world design as a graph design problem. We find a polynomial-time algorithm when decision fatigue depends only on the number of vertices and paths in the graph. The algorithm uses an elegant optimality condition: optimal world maps have a "side-quest" tree structure that is amendable to an efficient inductive construction.

Key words: video games; virtual worlds; decision fatigue; graph design; service design
History: This version: October 30, 2023.

1. Introduction

Virtual worlds are becoming an increasingly more significant part of the human experience. News outlets talk of the "metaverse" as the next stage of evolution in technology.[1] Lockdowns during the COVID-19 pandemic only accelerated interest in virtual worlds. Technology and entertainment companies are investing to learn how to design this new frontier.[2]

But none of this is entirely new. The video game industry has explored the design of virtual "worlds" for decades. Game developers have designed interactive worlds that capture the sustained attention of players. This is one of the fundamental goals of effective video game design.[3]

[1] https://connectedworld.com/metaverse-the-evolution-of-the-internet/

[2] https://www.makeuseof.com/companies-investing-in-metaverse/

[3] See, for example, Schell (2019) for a textbook treatment of video game design principles.

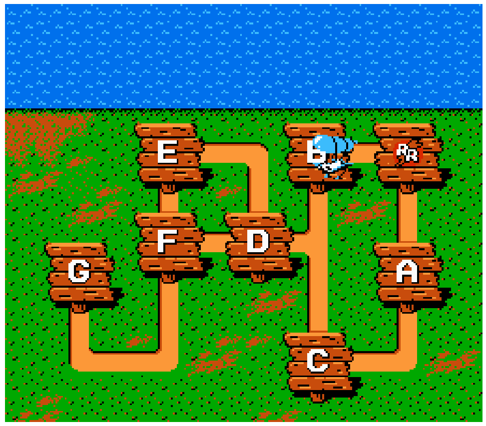


Figure 1 **The world map of the video game Chip n' Dale Rescue Rangers released in 1990 by Capcom for the Nintendo Entertainment System.**

fig:chip-n-dale

Designing interactive worlds is complex. It involves computer programming, artistic design, storytelling, economics, sociology, and psychology (Schell 2019, Hiwiller 2015, Hodent 2020, Kremers 2009, Totten 2017). A comprehensive treatment is beyond the scope of any one mathematical model. Here, we study a stylized problem that may serve as a foundation for further research. We state the problem now.

A game design team has prepared a set of game elements (levels, encounters, puzzles, etc.). Our focal design question is how to arrange the game elements into a "map".

Let's make things concrete. Consider the world map in Figure 1 from the classic game *Chip n' Dale's Rescue Rangers* released by Capcom in 1990 on the Nintendo Entertainment System.[4] Here, the game elements are levels labeled A through G. Players can pass through the world along different paths: ACDFG, BDFG, ACDEFG, etc. The arrangement of levels in this map raises many questions. Why must players tackle level A before C but can go to level B directly? Why make level E optional? Why give the players so many options for paths to level G?

Chip n' Dale's is an example of a *nonlinear* world map, in the terminology of game developers (see, for instance, Schell (2019)). Nonlinear maps give players choices on how to proceed through the game. In a graph encoding, a nonlinear world map is one with more than one path for players to navigate the world. *Linear gameplay* is when the player has no choice, corresponding to a linear graph. While linear gameplay is common, nonlinearity is the mainstay of video game design.[5]

The degree of nonlinearity varies across different types of games. While *Chip n' Dale* offers choices to players, these choices are limited. At the far end of the spectrum of choice are *sandbox*

[4] We use this example to illustrate our setting vividly. There are, of course, many recent examples of video games with similar world maps. See, for example, *Bad North* released in 2018 on multiple platforms, and the *Plants versus Zombies* series released from 2009 until the present day on all major platforms.

[5] Of the top 100 most popular video games on IGN, upwards of 90 of those games have nonlinear designs. Accessed on 27 December 2022 at: https://www.ign.com/articles/the-best-100-video-games-of-all-time

games like the *Grand Theft Auto* series. Sandbox games allow players to explore an "open" world with very few restrictions. This yields a nearly limitless number of paths.

This raises a simple research question. What is the ideal "degree of nonlinearity" in a video game world? Should players have a lot of choice or little? How to arrange game elements to augment or limit choice? To answer these questions, we need to think carefully about the implications of nonlinearity.

A clear benefit of nonlinearity is flexibility for players. Nonlinearity offers many ways to interact with the game.[6] The traditional base of video game players was young people with a lot of free time. But games have grown to welcome a much more diverse player population. In particular, these players differ in the amount of time they have to play.[7] Nonlinearity allows different types of players to engage with different intensities, offering a multitude of ways to gain a sense of accomplishment. Linear games may demand a significant time investment to reach a satisfying conclusion. Players have no choice but to pass through all game elements. By contrast, players of nonlinear games may only interact with a subset of elements to reach a satisfying conclusion.

In this way, nonlinearity offers a stark contrast between games and other entertainment genres. A meaningful conclusion of most modern television series requires watching every episode in order. Watching a subset of episodes leaves the watcher confused or without closure. Video games offer both choice and a sense of completion.

Presented with choice, players need to assess what parts of the game to tackle, given their limited budget for playing time. Managing time is central to many service design problems. See, for example, Tong et al. (2017), Song et al. (2020), Ruiz-Meza and Montoya-Torres (2021). But managing time in video games has its own subtleties. Players get utility for their time playing the game. But playing too long without resolution builds impatience and frustration. An easy solution to this trade-off is to offer all available content in every possible order. Players can select as few or as many game elements as they like in navigating the world. This design is uncommon because offering all orderings gives rise to complex worlds. Imagine the *Chip n' Dale* map with every possible path to area G. It would be an eyesore.

This leads to our third consideration: decision fatigue. Overwhelmed with many decisions, players experience disutility from mental exertion. Weighing many possibilities leaves players exhausted,

[6] We should note that nonlinearity also encourages replayability. Different playthroughs unfold the game differently. This can increase the perceived value of a game. While these benefits are of interest in general, we do not model them. As the reader will see, the problem we focus on is already difficult to describe and model. More enhancements will be welcome improvements for future work.

[7] For more information about the diverse base of video game players, see the 2020 industry report of the Entertainment Software Association (ESA): https://www.theesa.com/wp-content/uploads/2021/03/Final-Edited-2020-ESA_Essential_facts.pdf. The ESA is the largest industry association of video game developers.

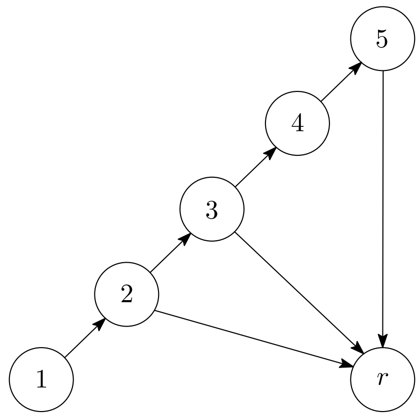


Figure 2 **A side-quest tree.**

fig:side-quest-tree-intro

especially as most players view gaming as a leisure activity. Decision fatigue is well-studied in psychology. See, for instance, Kahneman (2011), Vohs et al. (2018), Augenblick and Nicholson (2016), Ma et al. (2021)). To our knowledge, no prior research has developed a mathematical model for decision fatigue in the graphical context we study here.

Let's restate our research question in light of the above considerations:

Research question: How to design a map of a virtual world to maximize player enjoyment? The design must balance the utility of play, the disutility of playing too long without resolution, and decision fatigue. It must also consider different time budgets among players, a key differentiating factor among the evermore diverse gaming population.

We cast this question as a graph design problem. Vertices encode game elements, and edges encode precedences among elements. The resulting graph design problem is not standard. To our knowledge, the objective function—maximizing player enjoyment—has no precedence in the graph design literature.

The first nontrivial case of the world design problem one may consider is when decision fatigue is a function of the number of vertices and paths (and not edges). In this case, we establish a sufficient condition for the optimality of a world map called the *side-quest* tree structure illustrated in Figure 2.

A side-quest tree has the distinguishing feature of having many paths to the "concluding" game element (labeled in the figure by r), but these paths build on one another. The name "side-quest" refers to the fact that there is a "main" path to the world's conclusion (the shortest path to r), but there is an option connected set of "side quests" that can be done in addition to the "main" path. For the precise definition of a side-quest tree, see Definition 2.

The side-quest tree optimality structure allows us to compute optimal world maps in polynomial time in the case where decision fatigue is a function of the number of vertices in the world map and the number of paths. The polynomial-time algorithm leverages an "augmenting path"-like argument reminiscent of classical combinatorial optimization problems.

We also show that if we consider a decision fatigue function that also depends on the number of edges, side-quest trees are no longer optimal. We show this by giving an example of a non-side-quest tree graph that provides better expected utility than any side-quest tree. When decision fatigue depends on the number of edges, a structure with "interweaving side paths" (see Figure 6(b)).

We leave it for future research to discover the optimality structure for more general settings, but our initial attempts suggest this is challenging.

The rest of the paper is organized as follows. Section 2 describes related work in video game design, the design of experiential services, and service network design. Section 3 introduces our model across three subsections. In Section 3.1, we introduce the novel graph theoretical concept of "world maps", which is a building block for defining the optimization problem of players (in Section 3.2) given a world map, and the bilevel graph design problem of the game designer in Section 3.3. In Section 3.2, we describe another key novel feature of our optimization setup, the player utility functions, which involves utility from play, impatience, and decision fatigue.

Section 4 contains our analysis of side-quest trees (defined in Section 4.1). We offer a polynomial-time algorithm to compute optimal side-quest trees in Section 4.3. Section 5 is a brief section that illustrates that side-quest trees are no longer optimal in more general settings.

We should stress here that this paper raises many more questions about world design than it has scope to answer. We admit that the optimality of side-quest trees is not necessarily an immediately "actionable" practical design for a video game, but it nonetheless tells us something about the nature of optimality structures. If anything, we view our work as a stepping stone, rather than a definite conclusion, to the world design problem.

In this spirit, we provide an extensive list of possible extensions in our concluding section. We believe this raises an interesting possibility for a whole genre of optimization problems inspired by games— "combinatorial optimization problems for fun"—that take classical discrete optimization problems where the objective is no longer minimizing cost but maximizing the "fun" of finding and implementing an optimal solution. We believe this to be a new, and potentially exciting, area for operations researchers to explore. As the analysis in this paper illustrates, this direction is far from trivial to pursue.

2. Related Work

sec:lit-review

This paper contributes to the growing literature in operations management, information systems, and marketing on video game design (Chen et al. 2021b, Han et al. 2023, Li et al. 2023, Ryan et al. 2020, Vu et al. 2020, Huang et al. 2019, Ascarza et al. 2020, Guo et al. 2019a, Sheng et al. 2022, Guo et al. 2019b, Turner et al. 2011, Jiao et al. 2020, Meng et al. 2021, Huang et al. 2020, Appel et al. 2020, Runge et al. 2021, Chen et al. 2021a, Mai and Hu 2023). Of that literature, the one most closely related to ours is Li et al. (2023). In that paper, the authors are given a set of game elements with different levels of rewards and difficulties and sequence them into a linear design. That is, their study does not consider the possibility of nonlinear gameplay. This is the major point of departure in our study: we consider nonlinear gameplay and the possibility that different papers proceed through the game elements in different amounts of time. This is an orthogonal consideration to this earlier paper.

Our study also relates to the growing literature on the design of experiential services (see, for example, Das Gupta et al. (2016), Li et al. (2022), Roels (2019), Baucells and Sarin (2007)). These papers examine scenarios where a set of possible service interactions are arranged to maximize customer satisfaction. In the video game, these service interactions are our game elements. Most of the papers in that area examine "linear" scenarios similar to Li et al. (2023). Of these papers, the closest is Aouad et al. (2022), which studies the layout of museums from an optimization perspective. In both settings, we are interested in exploring the design of spaces for customers to explore, but our goals are different. In Aouad et al. (2022), their design problem is to maximize the expected lengths of visitor's paths using probabilistic data derived from real visitor sojourns to design expectations over random player behavior. Our setting has a different objective function (maximizing expected player utility). This yields a fundamentally different optimization setting.

Another area of research with many similarities to ours is trip design, initiated by (Tsiligirides 1984) with extensions studied until the present day (see, for instance, Yu et al. (2021), Gunawan et al. (2016), Song et al. (2020), Ruiz-Meza and Montoya-Torres (2021)). Similar to the museum design question of Aouad et al. (2022), these papers examine how to arrange stops on a tour in order to maximize the benefit to tourists without exceeding a time budget. Our major point of departure is that we allow players to self-select their route through the world map to maximize their own utility, something not considered in previous papers in this research stream. This makes our problem quite different than existing work, as it allows for the possibility of multiple paths taken by multiple players with differing objectives.

3. Model

sec:model

Our model is established across three subsections. First, we construct the "world map" concept, an ingredient in both the player's and game designer's problems. These two problems are the subjects of the following two subsections. The final optimization problem is a bilevel graph design problem incorporating the decisions of both the players and the game designer.

3.1 World maps

ss:world-map-setup

A game designer must arrange a given set of game elements (levels, encounters, puzzles, etc.) into a world map. We model a world map as a graph. Vertices correspond to game elements. Edges correspond to connections between game elements.

The set $\mathscr{V}$ denotes all available game elements. We use v or w to denote game elements as needs arise. There are two special vertices: 1 and r. The vertex 1 is a game element designated as the "start" of the game. Examples include opening cinematics, introductory puzzles, or character creation tools. The vertex r denotes the final game element. Examples of r include concluding cinematics, final boss fights, or challenging final puzzles. For concreteness, we set $\mathscr{V} := [N] \cup \{r\}$ where

eq:define-N

$$N := |\mathscr{V}| - 1 \tag{1}$$

and the notation $[N] := \{1, 2, \ldots, N\}$. In other words, the vertices in $\mathscr{V}$ have the labels $r, 1, 2, \ldots, N$.

After completing a game element v, players travel along an edge from v to another game element w. We may think of the edge (v, w) as a door or path. The interpretation depends on the fiction of the game. Edges are directed, establishing precedences between the game elements. We let $\mathcal{E}$ denote the set of all edges and use the notation e to denote a generic element of $\mathcal{E}$. We also use the notation (v, w) for edges to specify the starting vertex v and ending vertex w. The starting vertex 1 only has outgoing edges, while the ending vertex r only has incoming edges.

We let $\mathscr{U}$ denote the *universe* graph that has vertex set $\mathscr{V}$ and edge set $\mathscr{E}$. We take the universe graph to be the complete graph on $\mathscr{V}$. A *world map* $G = (V, E)$ is a subgraph of the universe selected by the game designer. That is, $V \subseteq \mathscr{V}$ and $E \subseteq \mathscr{E}$.

To be a world map, G must satisfy additional restrictions. First, G must be a directed acyclic graph (DAG) of $\mathscr{U}$. Under the DAG property, players cannot revisit game elements once completed. This is a common design mechanic. If each game element spends a given "time" in the fiction of the game, repeating an element can disrupt the storyline. Even when revisiting makes sense to the story, players are often averse to backtracking through known terrain.[8] Without too much loss, we restrict attention to acyclic world maps.

Second, in a world map, all edge choices by players lead to the end vertex r. In other words, there are no dead ends. To formalize this, we introduce some terminology. A *directed path* $p = (v_1, v_2, \ldots, v_\ell)$ (where ℓ is some arbitrary nonnegative integer) is a sequence of adjacent vertices (i.e., (v_i, v_{i+1}) for $i = 1, \ldots, \ell - 1$) where each vertex v_i is distinct (i.e., $v_i \neq v_j$ for all $i, j \in \{1, 2, \ldots, \ell\}$ with $i \neq j$). A *complete path* is a directed path that has starting vertex 1 and ending vertex r (i.e.,

[8] Backtracking is avidly debated among players. See, for example, the following webpage: `https://www.reddit.com/r/Games/comments/1wqet4/why_do_people_hate_backtracking_so_much/`

$v_1 = 1$ and $v_\ell = r$). A subgraph $G = (V, E)$ has *no dead ends* if every edge e in G lies on a complete path. That is, for all $e = (v, w) \in E$ there exists a complete path $p = (v_1, v_2, \ldots, v_\ell)$ in G with $v_i = v$ and $v_{i+1} = w$ for some $i \in \{1, 2, \ldots, \ell - 1\}$.

The "no dead ends" property ensures that the game has a single end. That is, the end vertex r is always the final game element of any path. Relatively few video games have multiple endings.[9] Many game designers shy away from multiple endings. These can compromise coherency and dilute resources across content players may never see. See Chapter 17 of Schell (2019) for more discussion.

We can now formally state our definition of a world map: def:world-map

Definition 1 (World Map). A subgraph G of the universe graph $\mathscr{U}$ is a *world map* if

(W1) G is a DAG (directed and acyclic), and

(W2) G has no dead ends (i.e., all edges in G lie on a complete path).

We will use the notation $\leq$ to denote the world map relationship: $G \leq \mathscr{U}$, meaning G is a world map in $\mathscr{U}$. ◄

The concept of a world map is central to this paper. In the next subsection, we define a player's utility as a function of a given world map. World maps are the decision variable of the designer's problem described in Section 3.3.

3.2 The player's problem

ss:players-problem

Given a world map G, players must decide on a path from the beginning vertex 1 to the end vertex r. In other words, a player must select a complete path in G. We do not allow for a player to "quit" a path before its completion.[10] As we shall see below, long paths may exert an impatience penalty for delaying a player's sense of resolution in the game.

The player's choice of path depends on their utility. Player utility is constructed from the following three components:

(U1) *utility from play*: the player earns utility proportional to the amount of time played, item:play

(U2) *impatience penalty*: the player suffers disutility when too much time elapses before reaching a satisfying resolution, and item:penalty

(U3) *decision fatigue*: the player suffers disutility associated with mental fatigue from having an overwhelming number of options (i.e., complete paths) to choose from. item:decision-fatigue

Each component is discussed in more detail below.

Components (U1) and (U2) both depend on the *game time* that a player needs to finish a path from 1 to r. Let us formalize this concept. First, we distinguish game time from "clock time".

[9] *Mass Effect 3* is a notorious example of an ineffective multiple-ending design. Despite offering players many choices, only three nearly identical endings were possible. This resulted in widespread outrage among players. For details, see https://www.inverse.com/gaming/mass-effect-3-ending-was-almost-completely-different.

[10] An extension that considers the possibility of quitting is an interesting prospect for future research.

Game time is the elapsed time that a player has been playing the game. For a mobile game, for example, this can be measured by the amount of time the game app is "open". This game time can be broken up across a much longer period of *clock time*. For example, consider a player who can only play for one hour a day. In this case, four hours of game time happens across four days of clock time. In our model, utility and disutility depend on game time and not clock time.

We assume that each game element takes a single unit of game time to complete. Thus, from a time perspective, the game elements are interchangeable. This is among the stricter assumptions of our model. Indeed, it is easy to imagine that some game elements take less time while others take more time. One idea is that longer game elements can be broken down into smaller-sized chunks that each takes one time unit. The difficulty here is that it may not make sense for these "chunks" to be separated along a path if they are linked thematically. Our model glosses over such subtleties and each game element as discrete and independent, each taking a single time unit to complete. As we shall see, even with this simplification, the problem is difficult to analyze.

We further assume that each game element does not have an intrinsic utility for completion. The derived utility is purely a function of how much additional gameplay the game element offers—namely, one additional time unit. Accordingly, the game elements are homogeneous from a utility perspective. Others have looked at how different reward and difficulty values for game elements give rise to much more complex forms of utility, see for instance Li et al. (2023). As discussed in the literature review section, papers in the tradition of Li et al. (2023) (starting with Das Gupta et al. (2016)) only consider linear graphs. The complexity we want to focus on here is offering branching paths and understanding this implication for service design. Accordingly, we have simplified the nuanced utility considerations of the game elements themselves as a matter of emphasis. Future work that brings together insights from papers like Li et al. (2023) to world design setting would be an exciting development.

With clarity about our focus on game time, we can now formally define a few useful concepts. Let P_G denote the set of complete paths in the world map G. Every element $p \in P_G$ has a duration $d(p)$ in $\mathbb{N}$—the set of nonnegative integers—equal to the number of *edges* that form the path. Concretely, if $p = (1, v_2, v_3, \ldots, v_{\ell-1}, r)$ then $d(p) = \ell - 1$. This can also be interpreted as assigning a duration to each vertex a unit duration except for the starting vertex 1. This is consistent with the fact that, in many games, the starting element is a "dummy" activity that just indicates where play is initiated. Given a world map G, its set of complete paths P_G determines the set $D_G = \{t \in \mathbb{N} : t = d(p) \text{ for some } p \in P_G\}$ of possible gameplay durations. It is easy to see that $D_G \subseteq [N]$ where N is defined in (1). Indeed, the longest path connects 1 to r via all N other vertices in $\mathscr{V}$ for a path length of N.

We now have all of the concepts and terminology needed to formalize the first two components of player utility (U1) and (U2).

3.2.1 Utility from play, (U1). To model utility from play, we define a function u from $\mathbb{N}$ to $\mathbb{R}_+$, where $u(t)$ is the utility from playing t units of time, and $\mathbb{R}_+$ is the set of nonnegative real numbers. Given a world map G, the set of possible utility values from play is $\{u(t) : t \in D_G\}$.

We assume that

–eq:play-utility-assumption˝

$$u : \mathbb{N} \to \mathbb{R}_+ \text{ is an increasing function,} \tag{2}$$

which is consistent with other time-based service design studies (see, for instance, Xu et al. (2015)). We will often assume u is a linear function of t; that is,

–eq:linear-utility˝

$$u(t) = \alpha t \tag{3}$$

where $\alpha > 0$ suffices to guarantee (2). The assumption of linearity is also common in the literature (see, for instance, Liao and Chen (2021)). This completes our discussion of the utility component (U1).

3.2.2 Impatience penalty, (U2). To define the impatience penalty, we need the notion of a *time budget*. Each player has a time budget b that represents the preferred amount of time investment to bring the game to a satisfying resolution. Impatience only starts to build after time budget b has been exhausted. Accordingly, the impatience penalty function has the form:

$$q(t \mid b) = \begin{cases} 0 & \text{if } t \leq b, \\ \phi(t-b) & \text{if } t > b, \end{cases}$$

where

–eq:phi-nondecreasing˝

$$\phi : \mathbb{N} \to \mathbb{R}_+ \text{ is an increasing function.} \tag{4}$$

Without loss, we assume b is a nonnegative integer expressed in the same time units as t.

Following LaGanga and Lawrence (2012), we will often assume that ϕ is a linear function $\phi(t-b) = \beta(t-b)$ where β is a positive constant, guaranteeing (4). This allows us to express the impatience penalty function as:

–eq:overtime-penalty˝

$$q(t \mid b) = \beta(t-b)\mathbb{1}[t > b] \tag{5}$$

where $\mathbb{1}[\cdot]$ is the indicator function that evaluates to 1 if the statement in its argument is true.

We assume without loss that $b \leq |\mathscr{V}| + 1$. That is, there are sufficiently many vertices in the universe to satisfy the player's time budget if all game elements are offered in a single long path. Indeed, otherwise, the structure of q would be such that the player never experiences any positive penalty, and so will always choose the path with the largest possible duration. This does not capture our tradeoff of interest, and so we remove this possibility from consideration.

ss:players-problem-decision-fatigue

3.2.3 Decision fatigue, (U3).

We assume that the players observe the whole world map G before deciding on a path. There are games where the world map is slowly "unlocked" as the player progresses, but such settings are beyond the scope of our inquiry.[11]

A world map G has n^v_G vertices, n^p_G complete paths, and n^e_G edges. We call the tuple (n^v_G, n^p_G, n^e_G) the *complexity* of the world map G. Players experience decision fatigue as a function of G's complexity. Given a world map G with complexity (n^v_G, n^p_G, n^e_G), the player experiences disutility $F(n^v_G, n^p_G, n^e_G)$ due to decision fatigue from pondering how to proceed through G. The function $F : \mathbb{N}^3 \to \mathbb{R}_+$ is called the *decision fatigue function*. We abuse notation to write $F(G) \triangleq F(n^v_G, n^p_G, n^e_G)$ when we want to suppress the detailed complexity notation.

We assume the following natural restriction:

–eqn:nondecreasing-complexity"

$$\frac{\partial F}{\partial n^v_G} \geq 0, \; \frac{\partial F}{\partial n^p_G} \geq 0, \text{ and } \frac{\partial F}{\partial n^e_G} \geq 0 \tag{6}$$

That is, F is nondecreasing in all of the components of complexity. Condition (6) is natural. Past research on decision fatigue confirms this. Augenblick and Nicholson (2016), Hirshleifer et al. (2019), Ma et al. (2021) examine how decision fatigue grows with the number of decisions to make. In our setting, the number of decisions depends on the number of vertices and edges in G. Players need to decide on a path, which is a sequence of vertices and edges. Vohs et al. (2018), Shah and Wolford (2007), Long et al. (2021) argue that decision fatigue also grows in the *number of options* for each decision. In our setting, this corresponds to the number of paths in G.

We will put another natural condition on decision fatigue that disciplines its growth with respect to the utility for play u. We want to argue that, all else being equal, the utility gained from additional play by extending a path exceeds the additional decision fatigue from extending that path. This is a relatively lighted-handed way to guarantee utility from play somewhat "dominates" decision fatigue disutility. While it is hard to make decisions about what path to take, the fatigue from doing so is outweighed, in a precise way, by the additional utility you get from playing.

It turns out that we only need to make this idea precise in the following specific setting. Let L_k denote the world map that is a line graph of length k. This is, it consists of a single path from start 1 to end r of length k. Then we assume the following:

ass:discipline-complexity-along-paths

Assumption 1. *The additional fatigue from extending a line graph by one more game element is less than the utility of playing that additional game element; that is,* $F(L_{k+1}) - F(L_k) \leq u(k+1) - u(k)$. ◀

[11] We share a few thoughts about this scenario in the concluding section.

3.2.4 A formal statement of the player's problem. We have all the terminology and notation to state the player's decision problem. The player chooses a path $p \in P_G$ to maximize her utility. It is easier to set up the problem as a play time decision. Recall that paths map to durations via the set D_G.

Let the world map G and time budget b be given. If a player selects a path with duration t, then we assume the player receives (total) utility

$$\pi(t|G,b) = u(t) - q(t|b) - F(G).^{12} \tag{7}$$

The set D_G contains the durations of all of the complete paths in G. Since a player selects a complete path in G, the set D_G contains all possible choices for the player's game time t. Accordingly, the player's decision problem is[13]

$$\begin{aligned} &\max_t \ \pi(t|G,b), \\ &s.t.\ t \in D_G. \end{aligned} \tag{$P|G,b$}$$

The notation $(P|G,b)$ underscores that the decision problem depends on the given world map G and budget b.

Understanding how optimal solutions to $(P|G,b)$ depend on changing G, and b is critical to later analysis. Luckily, this optimality structure is straightforward. Deriving it now will help us state a clean version of the game designer's problem.

We start by making the following innocuous assumption.

Assumption 2. *The following holds:*

$$u(t) - q(t|b) \text{ is a decreasing function of } t \text{ when } t \geq b. \tag{8}$$

When u and ϕ satisfy (3) and (5), it suffices that $\beta > \alpha$. ◀

This assumption ensures that the impatience penalty has "bite". After the time budget has been met, the impatience penalty $q(t|b)$ for additional play more than makes up for the additional utility from play $u(t)$.

To state the optimality structure of $t^*_{G,b}$, we use the following definitions:

$$\begin{aligned} \lfloor b \rfloor_G &= \max\{t | t \in D_G, t \leq b\}, \\ \lceil b \rceil_G &= \min\{t | t \in D_G, t \geq b\}, \\ \text{proj}_G(b) &= \arg\max\left\{\pi(t|G,b) : t \in \{\lfloor b \rfloor_G, \lceil b \rceil_G\}\right\}. \end{aligned} \tag{9}$$

[12] It is common to assume that total utility is the sum of its utility components. Since utilities are only defined up to an affine scaling, the form can be taken without loss assuming utility is a general affine function of its components. See, for example, Mas-Colell et al. (1995) for more details.

[13] The disutility term $-F(G)$ in $\pi(t|G,b)$ does not affect this optimization problem since it does not depend on t. We maintain this term in $\pi(t|G,b)$; it is crucial for understanding the optimal choice of G.

The "floor" and "ceiling" are with respect to the set D_G. This, of course, depends on G, and so the subscripts $\lfloor\cdot\rfloor_G$ and $\lceil\cdot\rceil_G$ are appropriate. The notation $\text{proj}_G(b)$ connotes the fact that we are projecting b on the "closest" element of G with the largest utility. Note that it is possible for $\text{proj}_G(b)$ to be the non-singleton set $\{\lfloor b\rfloor_G, \lceil b\rceil_G\}$ if $\pi(\lfloor b\rfloor_G|G,b) = \pi(\lceil b\rceil_G|G,b)$.

We can now characterize the optimal solutions of $(P|G,b)$.

theorem:game-duration

THEOREM 1 **(Optimality structure of $(P|G,b)$).** *Under Assumption 2,* $\text{proj}_G(b)$ *is the set of optimal solutions of* $(P|G,b)$.

To avoid the hassle of $\text{proj}_G(b)$ not being a singleton, we assume that when $\lfloor b\rfloor_G$ and $\lceil b\rceil_G$ yields the same value for π, the player chooses a path with the shorter duration $\lfloor b\rfloor_G$. Abusing notation, we will always take the unique (up to this tie break) optimal solution of $(P|G,b)$ to be:

$$t^*_{G,b} \triangleq \begin{cases} \lfloor b\rfloor_G & \text{if } \pi(\lfloor b\rfloor_G|G,b) = \pi(\lceil b\rceil_G|G,b) \\ \text{proj}_G(b) & \text{otherwise} \end{cases}. \tag{10}$$

eq:optimal-solution

This characterization proves very useful in our analysis of the game designer's problem.

rem:convert-back-to-paths

REMARK 1. In practice, players decide on paths rather than durations. The gameplay duration is decided by the game path indirectly. A player's utility $\bar{\pi}(p|G,b)$ for choosing path $p \in G$ under time budget b is

$$\bar{\pi}(p|G,b) = \pi\Big(\sum_{e\in E} \mathbb{1}[e \in p] | G, b\Big).$$

where $\sum_{e\in E} \mathbb{1}[e \in p]$ is a count of the edges in p. Phrased in terms of paths, the player's path decision problem over world map G and time budget b is

$$\max_{p\in P_G} \bar{\pi}(p|G,b). \quad \blacktriangleleft$$

3.3 Game designer's problem

ss:game-designers-problem

The game designer chooses the world map G in order to maximize player utility. One may ask why the designer does not optimize for revenue.[14] We are imagining a scenario where revenue is an increasing function of player utility. This is also not an unrealistic assumption. For premium games that are purchased with an upfront fee, the enjoyment that players experience drives word-of-mouth sales and purchases of sequels. For free-to-play mobile games, the more the players enjoy the game, the more likely they are to make in-app purchases that drive revenue. A more detailed model of the mapping between utility and revenue is beyond the scope of the current study.

[14] We do not consider any costs in this model. The game elements are given, and so we assume that the cost of their development is sunk. We are implicitly assuming that the task of "coding" the world map is the same irrespective of the design. This is also not an unreasonable assumption. The artifacts needed to render the "look" of the world are designed irrespective of how the artifacts are arranged. The cost of arranging artifacts is minuscule compared to generating the artifacts themselves. Consider, for instance, the *Chip N' Dale's* map in Figure 1. It is a straightforward task to move the sprites around the map to arrange them as preferred. This is not a cost worthy of consideration.

sss:single-player

3.3.1 Single player. Let's dispatch a special case of the game design problem when there is a single player with utility function π defined in ($P|G,b$) and with a given and known time budget b.[15] This special case serves as a building block for later analysis and motivates the setup of our most general model set up stated later in this subsection.

It turns out that this single-player case is trivial to solve. This is intuitively clear. In this case, it is optimal for the game designer to design the least complex world map G that offers a path with a duration equal to b. Namely, the line graph with starting vertex 1, ending vertex r, and $b-2$ intermediate vertices. This intuition is captured in the following result and its proof.

prop:single-player

THEOREM 2. *Suppose there is a single player whose utility function π satisfies Assumptions 1 and 2, and whose time budget b is known to the game designer. Then the world map G that maximizes player utility is the line graph consisting of a single path from 1 to r of length b.*

We want to remark that this result shows you that it was easier to design video games in a more homogeneous setting where players were very similar. Linear experiences were more common and were satisfying to the vast majority of players. As discussed in the introduction, however, players with very different levels of patience have started playing games as the industry has expanded. This motivates an investigation into the heterogeneity in time budgets, which we take up now.

3.3.2 Distribution of time budgets. Let's now look at the more realistic setting with multiple types of players who differ in time budgets. Suppose time budgets are distributed over the set $[N] = \{1, 2, \ldots, N\}$ with N finite. Let m denote the probability mass function of the distribution of budgets over $[N]$.[16] We may interpret $m(b)$ as the proportion of players with time budget b. Let B denote the discrete random variable that represents the time budget of a randomly chosen player. The expected time budget is thus $\mathbb{E}_B[B] = \sum_{b=1}^{N} bm(b)$.

Let's formally state the game designer's problem given a distribution of time budgets. The game designer's problem is to select a world map G in order to maximize the expected utility of players with distributed time budgets. Of course, the route chosen by the player is determined by their own optimization problem ($P|G,b$). Using the unique (up to our tie break rule) optimal solution $t^*_{G,b}$ to ($P|G,b$) defined in (10), we can state the game designer's objective function as:

eq:designer-objective

$$\Pi(G) := \mathbb{E}_B[\pi(t^*_{G,B}|G,B)] \tag{11}$$

[15] We describe this problem as if there is a single player. However, the same analysis holds if there are many players, but they all have the same utility function π and time budget b.

[16] We assume that the game designer has estimated this distribution using demographic information (e.g., young people prefer to play longer), previous gaming habits, and player game reviews.

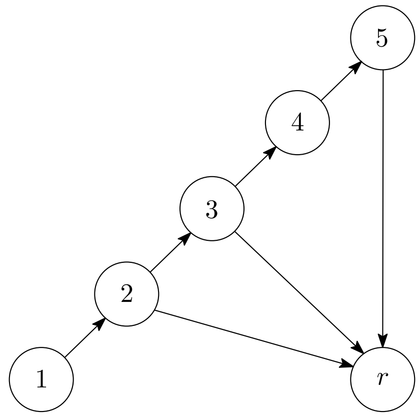


Figure 3 **The side-quest tree $T^{\{2,3,5\}}$.**

fig:side-quest-tree

where π is defined in (7), and the expectation is taken over the distribution m of the random time budget B. The game designer's world map problem (WMP) is

–eq:designer-problem˝

$$\max_{G \leq \mathscr{U}} \Pi(G), \tag{WMP}$$

where the notation $\leq$ (set in Definition 1) means that G is a world map selected from the universe $\mathscr{U}$. The rest of the paper takes up the challenge of solving (WMP).

4. Optimality of side quest trees

s:optimality-side-quest-trees

In this section, we establish the optimality structure of the world design problem (WMP) when the decision fatigue function F does not depend on the number of edges n_G^e in the chosen world map G. In optimality structure is the notion of a side-quest tree introduced in Section 4.1 and proven to be optimal in Section 4.2. Using this optimality structure, we can define an algorithm for computing optimal world maps when the fatigue function F is a function of the number of vertices and paths of a world map.

4.1 Side-quest trees

ss:define-side-quest-tree

We begin by defining the notion of a side-quest tree.

def:side-quest-tree

Definition 2 (side-quest tree). Let $D \subseteq [N]$ be a subset of durations (recall that $[N]$ is the set of all possible durations in a world map). Then, the *side-quest tree* T^D is the graph consisting of the path $(1, 2, \ldots, \hat{d})$ (where $\hat{d} = \max D$ is the largest duration in D) with appending the additional edges (v, r) for all $v \in D$.

Figure 3 illustrates the side-quest tree T^D with $D = \{2, 3, 5\}$.[17] The following lemma shows that side-quest trees are world maps, and thus feasible choices in (WMP).

[17] It is important to note that the phrase "tree" does not imply that T^D is a tree in the underlying undirected graph, where it may contain undirected cycles.

lemma:side-quest-trees-are-world-maps

LEMMA 1. *Let $D \subseteq [N]$ be a subset of durations and let T^D be the side-quest tree as constructed in Definition 2. Then T^D is a world map.*

The next section shows that there exists an optimal world map in the family of side-quest trees. In exploring the optimality of side-quest trees, it will prove useful later to have a count of the vertices and paths in a side-quest tree. This is captured in the following lemma.

lemma:counts-for-side-quest-trees

LEMMA 2. *Let $D \subseteq [N]$ be a subset of durations and let T^D be the side-quest tree as constructed in Definition 2. Then, T^D has exactly one complete path for each duration $d \in D$ (implying $D_{T_D} = D$) $1 + \hat{d}$ vertices and $\hat{d} - 1 + |D|$ edges, where $\hat{d} = \max D$.*

The result is easy to verify by inspecting Figure 3, but a formal proof is found in the appendix.

4.2 Optimality properties

ss:optimality-of-side-quest-trees

At the outset of the section, we spoke of restricting attention to fatigue functions that depend only on the number of paths and vertices. The next result helps us in taking advantage of this context.

lemma:complexity-minimum-paths-and-vertices

LEMMA 3. *Let $D \subseteq [N]$ be a set of durations. If the set D_G of durations of world map G contains D (i.e., $D \subseteq D_G$), then G has a minimum of $|D|$ complete paths and a minimum of $1 + \hat{d}$ vertices, where $\hat{d}$ is the largest element of D.*

Observe that for any duration set D, the number of paths and vertices of the side-quest tree T^D (coming from Lemma 2 above) match the minima in Lemma 3 and so yield the smallest possible fatigue among graphs that cover duration set D. Combined with the fact (Lemma 1) that all side-quest trees are feasible to (WMP), this can help establish the following result.

thm:paths-and-vertices-T-N-subgraph-best

THEOREM 3. *Suppose the fatigue function depends only on the number of paths n_G^p and the number of vertices n_G^v. Then, there exists an optimal solution to (WMP) that is a side-quest tree.*

From an analytical perspective, this result already says a lot. The game designer can restrict attention to world designs that look like the graph in Figure 3; namely, there is a single path of intermediate game elements (i.e., the path $(1, 2, \ldots, k)$) that the player can progress through, with occasional "exits" to the final game element r (i.e., along the edges (v, r) for v in a subset $[N]$ of possible durations). These are "quick exits" in the sense that they immediately lead to the final game element. For example, this means that the designer can be thinking about building a narrative for linear progress of game elements from 1 to k, with ways to abort this progression by uncovering a "shortcut" to the final boss. We see this in the classical "warp" mechanic in the early 8-bit era of home consoles. All the designer needs to decide is the length of the linear path (i.e., choose k), and where to place the occasional "exits" (i.e., choose D with k the largest element of D).

Of course, this insight does not offer an efficient computational approach for finding the optimal side-quest tree. Indeed, there are many possible choices for the length of the long path and many possible places to choose exits. A brute force enumeration of all "side-quest trees" is still exponential work.

In the next section, we discuss an algorithmic approach to enumerating side-quest trees in polynomial time when utility from play u is linear (i.e., satisfies (3)) and the impatience penalty function is piecewise linear (i.e., satisfies (5)).

But before turning to an algorithmic approach, we are interested in analytical ways of restricting our search among the set of all side-quest trees for an optimal side-quest tree. An immediate restriction that comes to mind is examining if we can restrict the set D of possible durations that we might consider.

A natural candidate for D is the set of possible time budgets b held by the players. Recall that $m(b)$ is the proportion of players with time budget b, where m is a probability mass function. Let

–eq:budget-set˝

$$\mathcal{B} := \{b \in [N] : m(b) > 0\} \tag{12}$$

denote the set of *budget set* of supported time budgets.

A natural thing to hope for is that there exists an optimal side-quest tree whose duration set is a subset of $\mathcal{B}$ (and maybe even equal to $\mathcal{B}$). Unfortunately, this is too good to be true, as the following counter-example illustrates.

ex:an-optimal-world-map-with-weird-durations

Example 1 (An optimal world map with duration set is not a subset of $\mathcal{B}$). Let $F(G) = 5(n_G^p)^2 + (n_G^v)^2$. Consider the probability mass function of time budgets to be m with $m(1) = \frac{1}{2}$ and $m(3) = \frac{1}{2}$ and 0 otherwise. That is, $\mathcal{B} = \{1, 3\}$. Also, suppose u is linear (i.e. satisfies (3)) with $\alpha = 13$ and $q(t|b)$ satisfies (5) with $\beta = 14$. From Theorem 3 there exists an optimal side-quest tree. The possible side-quest trees are $T^{\{1\}}$, $T^{\{2\}}$, $T^{\{3\}}$, $T^{\{1,2\}}$, $T^{\{1,3\}}$, $T^{\{2,3\}}$, and $T^{\{1,2,3\}}$. Straightforward calculations yields: $\Pi(T^{\{1\}}) = 4$, $\Pi(T^{\{2\}}) = 5$, $\Pi(T^{\{3\}}) = 4$, $\Pi(T^{\{1,2\}}) = -9.5$, $\Pi(T^{\{1,3\}}) = -10$, $\Pi(T^{\{2,3\}}) = -10.5$, and $\Pi(T^{\{1,2,3\}}) = -35$. Thus, $T^{\{2\}}$ is the optimal side-quest tree but $\{2\} \not\subseteq \mathcal{B} = \{1, 3\}$. ◀

Fortunately, all is not lost on the connection between budget set $\mathcal{B}$ and the structure of the optimal side-quest tree. We recover the following result.

prop:optimal-complexity-has-at-most-D-paths

Proposition 1. *Suppose the fatigue function depends only on the number of vertices and paths. Then, there exists an optimal solution to (WMP) that is a side-quest tree T^D with $|D| \leq |\mathcal{B}|$. That is, the duration set of an optimal side-quest tree has no more elements than the budget set $\mathcal{B}$.*

This result is more powerful when $\mathcal{B}$ is small. Indeed, consider the extreme case where $\mathcal{B}$ is a singleton. In other words, the game designer knows that all players have exactly the same time budget b, for some $b \in [N]$. This case was studied earlier in Theorem 2, where we showed that there exists an optimal line graph. Indeed, if $\mathcal{B}$ is a singleton, then there exists an optimal side-quest tree T^D where D is a singleton. But, if D is the singleton set $\{v\}$ then T^D is nothing more than the line graph consisting of the single path $(1, 2, \ldots, v-1, v, r)$. Thus, Proposition 1 can be seen as a type of generalization of Theorem 2 to more players, under the restriction that fatigue only depends on the number of vertices and paths.

Despite its power, for small Proposition 1, it does not preclude the situation we saw in Example 1 where the duration set D is *not* a subset of $\mathcal{B}$. To get the more expected condition (that $D \subseteq \mathcal{B}$), we make one further assumption.

prop:optimal-linear-complexity-has-at-most-D-paths

PROPOSITION 2. *Suppose the fatigue function*

(i) depends only on the number of paths n_G^p and the number of vertices n_G^v, and

(ii) is a linear *function of both n_G^p and n_G^v.*

u and ϕ are linear. Then, there exists an optimal solution to (WMP) *that is a side-quest tree T^D with $D \subseteq \mathcal{B}$. That is, the duration set of an optimal side-quest tree is a subset of the budget set $\mathcal{B}$.*

The result that the duration set of an optimal side-quest tree is a subset of the budget set is quite reassuring. However, this does not yield an immediately obvious efficient algorithm to find the optimal spanning tree. Indeed, $\mathcal{B}$ could still be large. We take up the challenge of computing an optimal side-quest tree in the next subsection.

4.3 An algorithm for computing optimal side-quest trees ss:algorithms-for-computing-side-quest-trees

In the previous subsection, we proved that when the decision fatigue only depends on the number of vertices and the number of paths, there exists an optimal graph to (WMP) that is a side-quest tree. In this subsection, we show how to compute the optimal side-quest tree efficiently.

The idea behind this computation is as follows. We say the *length* of a side-quest tree T^D is the length of the longest complete path in T^D. It is straightforward to see that the length of T^D is $\max D$. We say the *capacity* of a side-quest tree T^D is the number of complete paths in T^D. It is straightforward to see that the capacity of T^D is $|D|$. For every $i \in [N]$, among all the side-quest trees with length i and capacity $\mu \in [i]$, we find the best one that generates the highest expected utility. We denote it as $T^*_{i,\mu}$. Then the optimal side-quest tree must be an element of $\{T^*_{i,\mu} | i \in [N], \mu \in [i]\}$. Hence, if we compare the expected utilities generated by those $T^*_{i,\mu}$ $(i \in [N], \mu \in [i])$, the one with the highest expected utility will be the optimal side-quest tree.

The difficulty lies in how we could generate all those $\{T^*_{i,\mu}\}$ $(i \in [N], \mu \in [i])$ in an efficient way. We are able to develop an induction algorithm that works in polynomial time. This algorithm is built on the operations of *single-path transformation* mappings that map a side-quest tree to new side-quest tree with one additional complete path.

Below, we begin by giving the definition of single-path transformation. Then we discuss how this single-path transformation influences the total expected utility. Followed by that, we examine a key property of the single-path transformation that is it preserves optimality. Finally, we present our algorithm and prove it's in polynomial time.

subsubsec:SPT-definition

4.3.1 Definition of single-path transformation Let $\mathcal{S}$ denote the set of all side-quest trees in the universe graph $\mathscr{U}$. We can partition the set $\mathcal{S}$ into subsets of the same length. Let $\mathcal{S}_{i,\mu}$ denote the set of side-quest trees with length i and capacity μ. That is, the side-quest tree T^D is in $\mathcal{S}_{i,\mu}$ if and only if the length of T^D is i and the capacity of T^D is μ. Equivalently, $T^D \in \mathcal{S}_{i,\mu}$ if and only if i is the largest element in D and μ equals the number of elements in D. We call an element of $\mathcal{S}_{i,\mu}$ an *(i,μ)-side-quest tree (or (i,μ)-SQT for short)* and denote an arbitrary element of $\mathcal{S}_{i,\mu}$ by $T_{i,\mu}$.

Next, let $\mathcal{S}^*_{i,\mu}$ denote the set of optimal solutions to the following problem:

–eq:designer-problem-restricted-general-v-p”

$$\max_{T \in \mathcal{S}_{i,\mu}} \Pi(T), \qquad (\mathsf{WMP}_{i,\mu})$$

This is our main problem (WMP) where the designer is restricted to selecting from side-quest trees of length i and capacity μ. We call an element of $\mathcal{S}^*_{i,\mu}$ an *optimal (i,μ)-SQT* and denote an arbitrary element of $\mathcal{S}^*_{i,\mu}$ by $T^*_{i,\mu}$. Observe that if ($\mathsf{WMP}_{i,\mu}$) has a unique solution, $\mathcal{S}^*_{i,\mu}$ will be a singleton. Otherwise, $\mathcal{S}^*_{i,\mu}$ contains multiple elements that share the same expected utility.

When the fatigue function depends only on the number of vertices and paths, the optimal side-quest tree is an optimal solution to (WMP) and can be found in the sets of $\mathcal{S}^*_{i,\mu}$ for $i \in [N]$ and $\mu \in [i]$. We state the result in the lemma below.

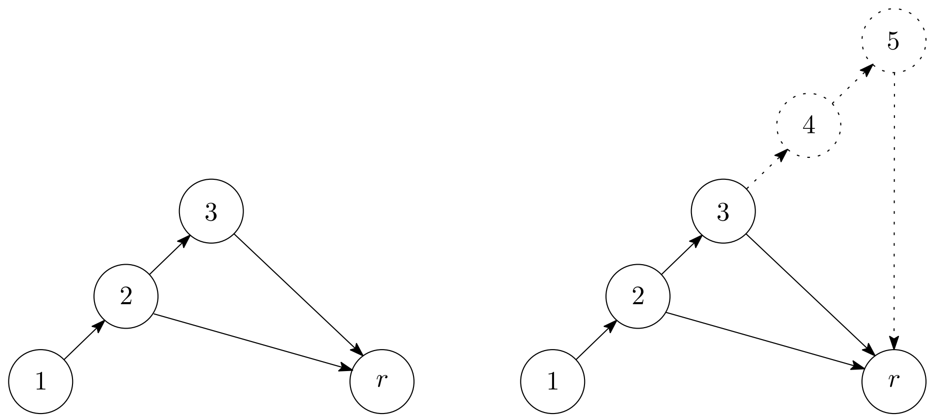
lemma:opt-tree-Sj-general-v-p

Lemma 4. *Optimal side-quest trees are contained in the union of sets of $\mathcal{S}^*_{i,\mu}$ over all $i \in [N]$ and $\mu \in [\min\{|\mathcal{B}|, i\}]$.*

Lemma 4 suggests that to find the optimal side-quest tree, it suffices to construct the sets $\mathcal{S}^*_{i,\mu}$ for every $i \in [N]$ and $\mu \in [\min\{|\mathcal{B}|, i\}]$ and then find the one that results in the largest expected utility. We will show later in Section 4.3.3 that the construction will be done through a graph operation named "single-path transformation" (defined below) that appends additional vertices and edges to a side-quest tree to form a new side-quest tree.

definition:SPT-general-v-p

Definition 3 (Single-path transformation). For $i, j, \mu \in [N]$, $i < j$ and $1 < \mu \le j$, define the function $\psi_{ij} : \mathcal{S}_{i,\mu-1} \to \mathcal{S}_{j,\mu}$ where $(i,\mu-1)$-SQT $T_{i,\mu-1}$ maps to the (j,μ)-SQT $T_{j,\mu}$ (i.e., $\psi_{ij}(T_{i,\mu-1}) = T_{j,\mu}$) where $T_{j,\mu}$ is the side-quest tree that results from extending the path $(1, 2, \ldots, i)$ in $T_{i,\mu-1}$ to path $(1, 2, \ldots, i, i+1, \ldots, j)$ (by adding $j - i$ more vertices and edges) and appending the edge (j, r). ◀



(a) an element $T_{3,2}$ of $\mathcal{S}_{3,2}$ (b) The element $\psi_{3,5}(T_{3,2})$ of $\mathcal{S}_{5,3}$

Figure 4 An illustration of the single-path transformation $\psi_{3,5}$. Appended vertices and edges are dotted.

fig:single-path-transformation

The transformation $\psi_{3,5}$ is illustrated in Figure 4. Observe that if T^D is in $\mathcal{S}_{i,\mu-1}$ then $\psi_{ij}(T^D)$ has duration set $D \cup \{j\}$. Observe also that $\psi_{ij}(T^D)$ has $j-i$ more vertices than T^D but only one more complete path. It is straightforward to verify that $\psi_{ij}(T^D)$ is a side-quest tree $\mathcal{S}_{j,\mu}$, and so the mappings ψ_{ij} are well-defined.

subsubsec:SPT-utility

4.3.2 Impact of single-path transformation on expected utilities As defined, the single-path transformation ψ_{ij} generates a (j,μ)-SQT from an $(i,\mu-1)$-SQT for $i,j,\mu \in [N]$, $i<j$ and $1<\mu \le j$. In this subsection, we are interested in tracking the designer's objective value as we undertake single-path transformation. Suppose we start with an $(i,\mu-1)$-SQT $T_{i,\mu-1}$ with an expected utility $\Pi(T_{i,\mu-1})$. We want to characterize the difference $\Pi(\psi_{ij}(T_{i,\mu-1})) - \Pi(T_{i,\mu-1})$.

The following result provides insights into the difference $\Pi(\psi_{ij}(T_{i,\mu-1})) - \Pi(T_{i,\mu-1})$.

lemma:learn-the-delta-general-v-p

Lemma 5. *Suppose u satisfies (3), q satisfies (5), and the fatigue function depends on the number of vertices n^v and complete paths n^p. Let $T_{i,\mu}$ be the element of $\mathcal{S}_{i,\mu}$. Then, for $i,j,\mu \in [N]$, $i<j$ and $1<\mu \le j$, we have:*

–eq:change-in-objective˝

$$\Pi(\psi_{ij}(T_{i,\mu-1})) - \Pi(T_{i,\mu-1}) = \Delta U_{ij} - \Delta F_{ij}(\mu-1) \tag{13}$$

where

–eq:change-in-utility-part˝

$$\Delta U_{ij} := \sum_{b=j}^{N} \alpha(j-i)m(b) + \sum_{b=\bar{b}}^{j-1} ((\alpha-\beta)j + \beta b - \alpha i)m(b) \tag{14}$$

with

–eq:cut-off˝

$$\bar{b} = \max\{\lceil \frac{\alpha i - (\alpha-\beta)j}{\beta} \rceil, i+1\}, \tag{15}$$

and

–eq:change-in-complexity-part˝

$$\Delta F_{ij}(\mu-1) := F(j+1,\mu) - F(i+1,\mu-1) \tag{16}$$

where $\lceil \frac{\alpha i-(\alpha-\beta)j}{\beta} \rceil$ indicates the smallest integer that is not smaller than $\frac{\alpha i-(\alpha-\beta)j}{\beta}$.

Recall (from (11)), that $\Pi(G) = E_B[\pi(t^*_{G,B}|G,B)]$, and (from (7)) $\pi(t^*_{G,b}|G_b) = u(t^*_{G,b}) - q(t^*_{G,b}|b) - F(G)$ and $t^*_{G,b}$ is the optimal duration of a time budget b player (specified in (10)). Notice that the $-F(G)$ term in $\pi(t^*_{G,b}|G_b)$ does not depend on the random variable B, and so we can express $\Pi(G)$ in two terms:

–fun:profit-restructure˝

$$\Pi(G) = U(G) - F(G) \tag{17}$$

where

$$U(G) := \mathbb{E}_B[u(t^*_{G,B}) - q(t^*_{G,B}|B)]. \tag{18}$$

The designer's objective consists of two terms: $U(G)$ is the expected utility from play minus the impatience penalty, and $F(G)$ indicates the disutility from decision fatigue. As a result, the difference in designer objective under an single-path transformation (expressed in (13)) also comes in two terms, ΔU_{ij} and $\Delta F_{ij}(\mu-1)$.

The second term ΔF_{ij} is easy to interpret. Decision fatigue depends only on the number of vertices and paths. The original side-quest tree $T_{i,\mu-1}$ has $i+1$ vertices and $\mu-1$ complete paths while the new side-quest tree $\psi_{ij}(T_{i,\mu-1})$ has $j+1$ vertices and μ complete paths. Thus, we have $\Delta F_{ij}(\mu-1) := F(j+1,\mu) - F(i+1,\mu-1)$.

More interesting is the expression for the first term ΔU_{ij} in (14). The two components in the expression

–eq:change-in-utility-two-terms˝

$$\underbrace{\sum_{b=j}^{N} \alpha(j-i)m(b)}_{\text{(i) players with time budget } \geq j} + \underbrace{\sum_{b=\bar{b}}^{j-1} ((\alpha-\beta)j + \beta b - \alpha i)m(b)}_{\text{(ii) players with time budget in } (i,j)} \tag{19}$$

arise from two groups of players: (i) players with time budgets of at least j and (ii) players with time budgets strictly between i and j. To interpret (19), let's consider the change of players' decisions after adding a new j-length path to the original side-quest tree $T_{i,\mu-1}$ with $i<j$.

(i) For players with time budgets of at least j, they selected the longest path with length i under the original side-quest tree $T_{i,\mu-1}$. Now given the new j-length path, those players will all switch to this new path, because it gives them higher utility from play and does not incur any impatience penalty. Thus, the change of the expected utility is equal to $\sum_{b=j}^{N} \alpha(j-i)m(b)$.

(ii) For players with time budgets strictly between i and j, they selected the longest path with length i under the original side-quest tree $T_{i,\mu-1}$. Given the new side-quest tree, they must decide between choosing the i-length path that $\psi_{ij}(T_{i,\mu-1})$ inherits from $T_{i,\mu-1}$ or the new path j-length path added by the single-path transformation. If a player with budget b chooses the original i-length path, his utility from play is αi and the impatience penalty is 0. If a player with budget b chooses the new j-length path, he earns the utility from play αj but pays the impatience penalty $\beta(j-b)$, resulting in a difference of $\alpha j - \beta(j-b)$. Thus, the player compares the two utilities

αi and $\alpha j - \beta(j-b)$ and will select the path which gives him the higher utility. Clearly, there exists a break-even point $\bar{b}$ expressed in (15). Only those players with time budget $b \in [\bar{b}, j-1]$ will switch to the new j-length path. The rest of the players with time budget $b \in [i+1, \bar{b})$ will stay in the original i-length path. Therefore, the change of the expected utility is computed by $\sum_{b=\bar{b}}^{j-1}(\alpha j - \beta(j-b) - \alpha i)m(b) = \sum_{b=\bar{b}}^{j-1}((\alpha-\beta)j + \beta b - \alpha i)m(b)$.

Observe that there is no term for players with time budgets at most i. This is because these players will not change their decision of optimal path. Note that the single-path transformation ψ_{ij} only adds one single path, which is the j-length path. Suppose those players with time budgets at most i switch to this new path, their utility from play increases, and so does the impatience penalty. However, we assume the growth in impatience penalty is greater than the growth in utility from play (Assumption 2). Then switching to the new longer path will make those players worse off. Therefore, in both side-quest trees, $T_{i,\mu-1}$ and $\psi_{ij}(T_{i,\mu-1})$, players with time budgets at most i will choose the same optimal path, therefore there is no change in players' utility.

Finally, we remark that ΔU_{ij} and $\Delta F_{ij}(\mu-1)$ *do not* depend on the structure of $T_{i,\mu-1}$. This invariant streamlines the inductive algorithm presented in Section 4.3.4.

subsubsec:SPT-preserve-optimality

4.3.3 Key property of single-path transformation The following lemma illustrates a key property of single-path transformations that is they preserve optimality in a precise sense.

lemma:nested-optimalty-general-v-p

Lemma 6. *Suppose u satisfies (3), q satisfies (5), and the fatigue function depends on the number of vertices and paths. For $j, \mu \in [N], 1 < \mu \leq j$, the following properties hold:*

(i) For every element $T^*_{j,\mu}$ of $\mathcal{S}^*_{j,\mu}$ there exists $i \in [j-1]$ such that $T_{j,\mu} = \psi_{ij}(T_{i,\mu-1})$ for some $T^*_{i,\mu-1} \in \mathcal{S}^*_{i,\mu-1}$. In other words, every optimal (j,μ)-SQT arises from a single-path transformation of some optimal $(i,\mu-1)$-SQT for $i \in [j-1]$.

(ii) Suppose $\psi_{ij}(T^*_{i,\mu-1}) \in \mathcal{S}^*_{j,\mu}$ for some $T^*_{i,\mu-1} \in \mathcal{S}^*_{i,\mu-1}$ ($i \in [j-1]$), then $\psi_{ij}(\hat{T}^*_{i,\mu-1}) \in \mathcal{S}^*_{j,\mu}$ and $\Pi(\psi_{ij}(T_{i,\mu-1})) = \Pi(\psi_{ij}(\hat{T}^*_{i,\mu-1}))$ for any $\hat{T}^*_{i,\mu-1} \in \mathcal{S}^*_{i,\mu-1}$. In other words, suppose an optimal (j,μ)-SQT arises from some optimal $(i,\mu-1)$-SQT, then the (j,μ)-SQT arises from any optimal $(i,\mu-1)$-SQT is an optimal (j,μ)-SQT sharing the same optimal value for $i \in [j-1]$.

(iii) The graph in $\{\psi_{i,j}(T^*_{i,\mu-1}) | i \in [j-1]\}$ with largest Π value is an element of $\mathcal{S}^*_{j,\mu}$.

Lemma 6 has important implications. First, (i) indicates that every optimal (j,μ)-SQT ($j, \mu \in [N], 1 < \mu \leq j$) can be generated from a smaller optimal $(i,\mu-1)$-SQT ($i \in [j-1]$) by a single-path transformation. Consequently, if all the optimal $(i,\mu-1)$-SQT ($i \in [j-1]$) are given, we can derive the set of all optimal (j,μ)-SQT. This sheds light on our inductive algorithm in Section 4.3.4. Roughly speaking, we will inductively construct the set $\mathcal{S}^*_{i,\mu-1}$ for all $j, \mu \in [N], 1 < \mu \leq j$ and the one with the largest expected utility will be an optimal solution to (WMP) (from Lemma 4).

(ii) further suggests that it is not necessary to construct the whole set $\mathcal{S}^*_{i,\mu-1}$ in every induction step. When the set $\mathcal{S}^*_{i,\mu-1}$ is not a singleton, it suffices to only select one optimal $(i,\mu-1)$-SQT in the set $\mathcal{S}^*_{i,\mu-1}$ as a representative. Because (ii) ensures that if an optimal (j,μ)-SQT can be derived from an optimal $(i,\mu-1)$-SQT for some $i \in [j-1]$, then the resulting graph from a single-path transformation of any optimal $(i,\mu-1)$-SQT in the set $\mathcal{S}^*_{i,\mu-1}$ must also be an optimal (j,μ)-SQT. As a result, when constructing an optimal (j,μ)-SQT through induction, we only need one optimal $(i,\mu-1)$-SQT for each $i \in [j-1]$.

What is more, (iii) points out how we find an optimal (j,μ)-SQT. Specifically, for each $i \in [j-1]$, we will apply a single-path transformation on the representative optimal $(i,\mu-1)$-SQT, resulting in a set of (j,μ)-SQT (i.e., the set $\{\psi_{i,j}(T^*_{i,\mu-1}) | i \in [j-1]\}$). Among this set of (j,μ)-SQT, the one with largest expected utility will be an optimal (j,μ)-SQT.

To conclude, Lemma 6 serves as the foundation of our inductive algorithm. It is straightforward to see that the optimal $(i,1)$-SQT is uniquely defined with $i+1$ vertices and a single length i complete path from vertex 1 to r. Starting from the optimal $(i,1)$-SQT $T^*_{i,1}$, we can inductively construct an optimal (j,μ)-SQT $T^*_{j,\mu}$ by conducting a series of single-path transformations on an optimal $(i,\mu-1)$-SQT $T^*_{i,\mu-1}$ for each $i \in [j-1]$ and $\mu \in [i]$. Finally, after we derive all the optimal (j,μ)-SQT and obtain the set $\{T^*_{j,\mu} | j \in [N], \mu \in [j]\}$, we compare the expected utilities associated with those $T^*_{j,\mu}$. The optimal side-quest tree will be the one with largest expected utility.

subsubsec:induction-algorithm

4.3.4 Induction algorithm Lemma 5 gave us closed-form formulas for how a single-path transformation changes the value of the designer's objective function. By Lemma 6 and the paragraphs that followed it, we learned that we can compute the optimal side quest tree by conducting a series of single-path transformations. These two ingredients come together in Algorithm 1. We illustrate idea of Algorithm 1 in Figure 5.

In the following, we present the framework of Algorithm 1 and explain how our algorithm works.

We start the induction that is what happens inside the `for` loop from line 1 to line 17. By Lemma 4, we only cares about the case where $j \in \{1,2,\dots,N\}$ and $\mu \in [\min\{|\mathcal{B}|, i\}]$.

For every $j \in \{1,2,\dots,N\}$, we construct the optimal $(j,1)$-SQTs from line 4 to 6, and the optimal (j,μ)-SQTs where $\mu > 1$ from line 8 to 14.

For the optimal $(j,1)$-SQTs, the algorithm constructs the optimal side quest tree $T^*_{j,1}$ with vertex and edge sets $V_{j,1}$ and $E_{j,1}$ from line 4 to 5, because it can be uniquely defined by Definition 2 and Lemma 2. The objective value is computed in line 6.

For the optimal (j,μ)-SQTs, the algorithm constructs the possible objective values of $\psi_{ij}(T_{i,\mu-1})$ arising from optimal $(i,\mu-1)$-SQTs where $i \in [j-1]$ using single-path transformations as illustrated in line 9. For sake of demonstration, we denote $\Pi^*_{i,\mu-1} = \Pi(T^*_{i,\mu-1})$ and $\Pi_{i,j,\mu} = \Pi(\psi_{ij}(T_{i,\mu-1}))$.

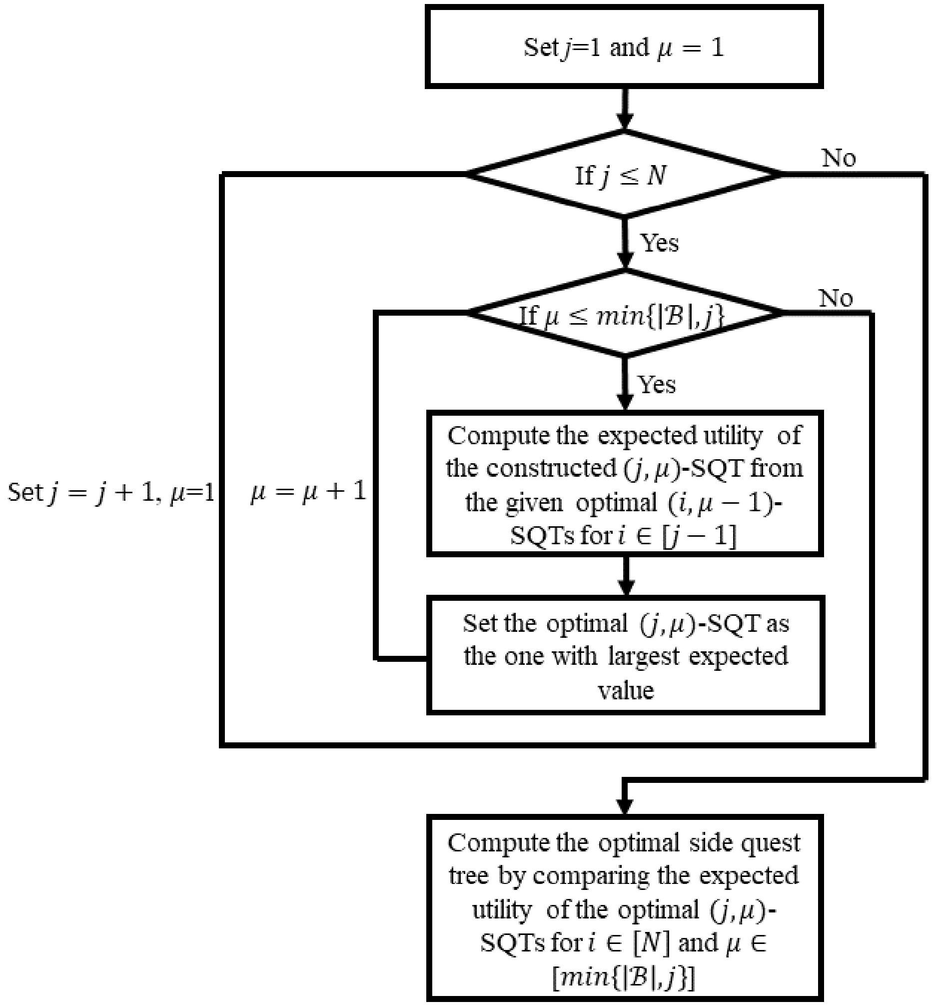


Figure 5 **The framework of Figure 5**

fig:algorithm-induction-general-v-p

Following Lemma 5, we have $\Pi_{i,j,\mu} = \Pi(\psi_{ij}(T^*_{i,\mu-1})) = \Pi^*_{i,\mu-1} + \Delta U_{ij} - \Delta F_{ij}(\mu - 1)$. In line 11, we find the largest expected utility among those $\Pi_{i,j,\mu}$ for all $i \in [j-1]$. According to Lemma 6 (iii), the (j,μ)-SQT $\psi_{ij}(T_{i,\mu-1})$ that arises from the optimal $(i^*,\mu-1)$-SQT must be an optimal (j,μ)-SQT. Then from line 12 to 13, we apply the single-path transformation to the optimal $(i^*,\mu-1)$-SQT and construct this optimal (j,μ)-SQT with vertex and edge sets $V_{j,\mu}$ and $E_{j,\mu}$. The algorithm constructs the optimal (j,μ)-SQT in line 14 for $j \in [N]$, and $\mu \in [\min\{|\mathcal{B}|, j\}]$.

Combining the cases where $\mu = 1$ and $\mu > 1$, the loop starting from line 1 terminates with a single optimal (j,μ)-SQT for each $j \in [N]$ and $\mu \in [\min\{|\mathcal{B}|, j\}]$ of the largest possible objective value. Lastly, in line 18, the algorithm selects the optimal (j,μ)-LSQT with the largest possible objective value and returns it as the optimal side-quest tree $T^* = T^*_{j^*,\mu^*}$.

The following theorem describes the optimally of the algorithm and its run-time complexity.

theorem:dp-general-v-p

Algorithm 1 An inductive algorithm to compute an optimal side-quest tree algorithm-induction-general-v-p

Input: Universe graph $\mathscr{U} = (\mathscr{V}, \mathscr{E})$, start and end vertices 1 and r, time budget set $\mathcal{B}$, probability mass function m of time budget random variable B, utility from play function u that satisfies (3), impatience penalty function q that satisfies (5), and fatigue function $F(n_G^v, n_G^p)$ that depends only on the vertex and path count of a world map $G \leq \mathscr{U}$;

Output: A side-quest tree that is an optimal solution to (WMP).

1: **for** $j \in [N]$ **do** algorithm:for-loop
2: **for** $\mu \in [\min\{|\mathcal{B}|, j\}]$ **do**
3: **if** $\mu = 1$ **then**
4: Let $V_{j,1} = \{1, 2, \ldots, j-1, j, r\}$; algorithm:start-construct-n=1
5: Set $E_{j,1} = \{(1,2), (2,3), \ldots, (j-2, j-1), (j-1, j), (j, r)\}$; algorithm:middle-construct-n=1
6: Let $T^*_{j,1} = (V_{j,1}, E_{j,1})$, and set $\Pi_{j,1} = \Pi(T^*_{j,1})$; algorithm:IA-recursion-n=1
7: **else**
8: **for** $i \in [j-1]$ **do** algorithm:for-loop-1
9: Let $\Pi_{i,j,\mu} = \Pi_{i,\mu-1} + \Delta U_{ij} - \Delta F_{ij}(\mu - 1)$; algorithm:IA-recursion
10: **end for**
11: Let $i^* = \arg\max\{\Pi_{i,j,\mu} | i \in [j-1]\}$, and set $\Pi_{j,\mu} = \Pi_{i^*,j,\mu}$; algorithm:IA-select-j-1
12: Let $V_{j,\mu} = V_{i^*,\mu-1} \cup \{i^*+1, i^*+2, \ldots, j-1, j\}$; algorithm:start-construct
13: Set $E_{j,\mu} = E_{i^*,\mu-1} \cup \{(i^*, i^*+1), (i^*+1, i^*+2) \ldots, (j-2, j-1), (j-1, j), (j, r)\}$; algorithm:middle-construct
14: Let $T^*_{j,\mu} = (V_{j,\mu}, E_{j,\mu})$; algorithm:end-construct
15: **end if**
16: **end for**
17: **end for** algorithm:end-for-loop
18: Let $(j^*, \mu^*) = \arg\max\{\Pi_{j,\mu} | j \in [N], \mu \in [\min\{|\mathcal{B}|, j\}]\}$, and set $\Pi^* = \Pi_{j^*,\mu^*}$ and $T^* = T^*_{j^*,\mu^*}$; algorithm:IA-select-optimal
19: **return** T^*.

THEOREM 4. *Suppose u satisfies (3), q satisfies (5), and the fatigue function depends on the vertex and path count of a world map G. Then Algorithm 1 produces an optimal solution to (WMP) has run-time complexity $O(N^2|\mathcal{B}|)$.*

Algorithm 1 has $O(N|\mathcal{B}|)$ stages. At each stage $j \in [N]$ and $\mu \in \{2, \ldots, \min\{|\mathcal{B}|, j\}\}$, the algorithm calculates the possible objective values $\Pi_{i,j,\mu}$ based on the optimal $(i, \mu-1)$-SQT where $i \in [j-1]$ and $\mu \in [\min\{|\mathcal{B}|, j\}]$ recursively in line 9. It finds the maximum objective value $\Pi^*_{j,\mu}$ of the optimal (j, μ)-SQT in line 11 by choosing the maximum $\Pi_{i,j,\mu}$ for $i \in [j-1]$. Thus, it takes $O(N)$ iterations to compute the possible objective value of the optimal (j, μ)-SQT.

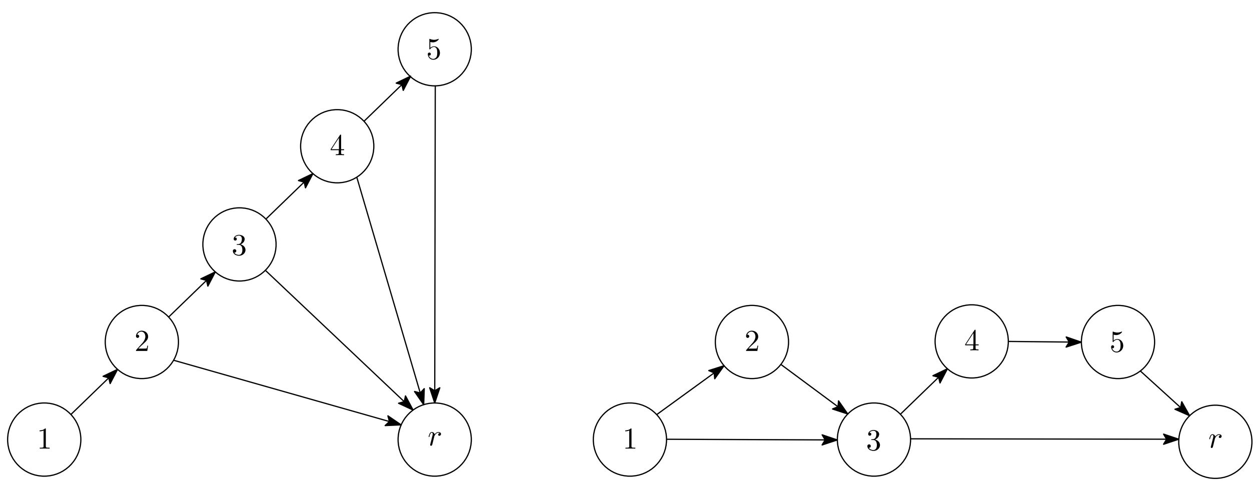
The algorithm then constructs the optimal (j,μ)-SQT from the $(i^*,\mu-1)$-SQT in line 14 for any optimal (j,μ)-SQT where $j \in [N]$ and $\mu \in [\min\{|\mathcal{B}|, j\}]$. Lastly, Algorithm 1 computes the optimal side quest tree T^* in line 18 by choosing the maximum $\Pi^*_{j,\mu}$ of the optimal (j,μ)-SQT for $j \in [N]$ and $\mu \in [\min\{|\mathcal{B}|, j\}]$. Hence, the total computational complexity of Algorithm 1 is $O\left(N^2|\mathcal{B}|\right)$, which is in polynomial time.

5. More general fatigue functions

s:more-general

While the analysis of the case where the fatigue function only depends on the vertices and paths is quite complete, it raises the question of whether side-quest trees remain optimal when the fatigue function depends on the number of edges. The following counter-example reveals that this need not be the case.

Example 2 (Side-quest tree is not optimal). Consider the setting with $N = 5$ and the fatigue function F is increasing in number of vertices, paths, and edges. This implies that when two world maps G and G' have the same number of vertices and paths—and the same duration set—but G has fewer edges than G', then $\Pi(G) > \Pi(G')$. Consider now the two world maps illustrated in Figure 6.



(a) $T^{\{2,3,4,5\}}$ (b) a non-side-quest tree

Figure 6 Non-optimality of side-quest trees.

fig:non-optimality-side-quest-tree

Both Figure 6(a) and Figure 6(b) have six vertices, four paths, and the same duration set $\{2,3,4,5\}$, but the world map has one fewer edge (eight versus seven). Accordingly, the side-quest tree will never be an optimal solution to (WMP) for any choice of u, q, and m. ◀

This example demonstrates that, in a sense, side-quest trees have *too many* edges in general. The world map in Figure 6(b) generates its duration set by two diversionary paths from the "main path" $(1,3,r)$, one taking a detour to vertex 2 and the second diverting to vertices 4 and 5. The path of duration 3 in Figure 6(b) taking vertices $\{3,4,5\}$ arises by electing *not* to go vertex 2, while in Figure 6(a), every journey to vertex 5 *must* take vertex 2. This lack of flexibility requires the additional edge $(3,r)$ to give Figure 6(a) a path of duration three.

In attempting to construct a class of optimal world maps for general fatigue functions, it was the possibility of adding more and more "flexibility" of this type that made it hard to find a more general optimality structure. Without a handle on additional optimality structure, it proved challenging to find optimal world maps for more general fatigue functions. Accordingly, we leave the investigation of additional structures for future work.

6. Conclusion

s:conclusion

In this paper, we introduced a novel graph design problem motivated by a problem of growing interest in practice—design virtual worlds. Our setting looked at the problem of designing a video game world map based on considerations of how players earn utility from play but incur disutility from impatience and decision fatigue.

There are numerous ways to extend the setting we studied to add even more realism. Each of these extensions, in our opinion, is nontrivial to pursue:

- What if the different game elements offer differing utilities and durations? In the current setting, all game elements offer unit durations and utilities to all players. This extension would abrogate a lot of the symmetry we use in our result, making analysis much more complex.
- What if there is "hard-coded" precedence between certain levels? For example, in the "Metroidvania" genre[18] players must backtrack to find new paths in previously explored areas as the player's avatar gains new abilities. In our setting, the underlying universe graph $\mathscr{U}$ was always complete, which was particularly useful when showing that side-quest trees were optimal. Indeed, at a minimum, we knew every side-quest tree was a feasible world map and was connected via the single path transformation property. A more restrictive universe graph would require a more careful accounting of feasibility.
- In a similar vein, we have assumed throughout that players have complete information about the nature of the world map and make decisions on how to traverse it in a static way. In many games, the world map is only revealed slowly as your progress through the various game elements. But a dynamic, "learn the map on the fly" analysis would add considerable complexity to the underlying path selection problem. The theory of stochastic or online combinatorial optimization would need to develop in order to

[18] https://en.wikipedia.org/wiki/Metroidvania

tackle this setting, and the analysis would itself become more approximate or in search of competitive ratios. Even if this is the ultimate goal, studying the full-information version of the problem is a pre-requisite, something we have started to explore in this paper.

- Extensions could add more player heterogeneity in terms of their utilities and speed of traversing the game elements. Our analysis only addresses heterogeneity in the time budget of players, which we have argued is a salient consideration given the changing nature of player demographics.
- Finally, our analysis assumed that the game designer has complete information about the player's payoff functions. This is not entirely realistic, and these utilities probably needed to be learned as players interact with designers. This "learning phase" is something that could be studied with models analogous to "demand learning" in our setting, but is much beyond the scope of what we study here.

Finally, beyond the world design problem, this research direction raises the possibility of a whole genre of research papers that have traditional combinatorial optimization problems with new objective functions related to player utility. Usually, combinatorial optimization problems have simple objective functions: minimize cost, minimize time, maximize flow, etc. What if our goal is to design optimization problems that maximize how "fun" they are to solve, in a sense related to notions explored in this paper and other video game papers.

References

Aouad A, Deshmane A, Martinez-de Albeniz V (2022) Designing layouts for sequential experiences: Application to cultural institutions .

Appel G, Libai B, Muller E, Shachar R (2020) On the monetization of mobile apps. *International Journal of Research in Marketing* 37(1):93–107.

Ascarza E, Netzer O, Runge J (2020) The twofold effect of customer retention in freemium settings. *Harvard Business School Working Paper 21-062* .

Augenblick N, Nicholson S (2016) Ballot position, choice fatigue, and voter behaviour. *The Review of Economic Studies* 83(2):460–480.

Baucells M, Sarin RK (2007) Satiation in discounted utility. *Operations Research* 55(1):170–181.

Chen M, Elmachtoub AN, Lei X (2021a) Matchmaking strategies for maximizing player engagement in video games. *Available at SSRN 3928966* .

Chen N, Elmachtoub AN, Hamilton ML, Lei X (2021b) Loot box pricing and design. *Management Science* 67(8):4809–4825.

Das Gupta A, Karmarkar US, Roels G (2016) The design of experiential services with acclimation and memory decay: Optimal sequence and duration. *Management Science* 62(5):1278–1296.

Gunawan A, Lau HC, Vansteenwegen P (2016) Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research* 255(2):315–332.

Guo H, Hao L, Mukhopadhyay T, Sun D (2019a) Selling virtual currency in digital games: Implications for gameplay and social welfare. *Information Systems Research* 30(2):430–446.

Guo H, Zhao X, Hao L, Liu D (2019b) Economic analysis of reward advertising. *Production and Operations Management* 28(10):2413–2430.

Han J, Ryan C, Tong XT (2023) Algorithms for loot box design. *Available at SSRN 4326724* .

Hirshleifer D, Levi Y, Lourie B, Teoh SH (2019) Decision fatigue and heuristic analyst forecasts. *Journal of Financial Economics* 133(1):83–98.

Hiwiller Z (2015) *Players Making Decisions: Game Design Essentials and the Art of Understanding Your Players* (New Riders).

Hodent C (2020) *The Psychology of Video Games* (Routledge).

Huang Y, Jasin S, Manchanda P (2019) "Level Up": Leveraging skill and engagement to maximize player game-play in online video games. *Information Systems Research* 30(3):927–947.

Huang Y, Lim KH, Lin Z (2020) Leveraging the numerosity effect to influence perceived expensiveness of virtual items. *Information Systems Research* 32(1):93–114.

Jiao Y, Tang CS, Wang J (2020) Opaque selling in player-vs-player games. *Available at SSRN 3558774* .

Kahneman D (2011) *Thinking, fast and slow* (Macmillan).

Kremers R (2009) *Level design: concept, theory, and practice* (CRC Press).

LaGanga LR, Lawrence SR (2012) Appointment overbooking in health care clinics to improve patient service and clinic performance. *Production and Operations Management* 21(5):874–888.

Li Y, Dai T, Qi X (2022) A theory of interior peaks: Activity sequencing and selection for service design. *Manufacturing & Service Operations Management* 24(2):993–1001.

Li Y, Ryan CT, Sheng L (2023) Optimal sequencing in single-player games. *Management Science (to appear)* .

Liao CN, Chen YJ (2021) Design of long-term conditional cash transfer program to encourage healthy habits. *Production and Operations Management* 30(11):3987–4003.

Long X, Sun J, Dai H, Zhang D, Zhang J, Chen Y, Hu H, Zhao B (2021) Choice overload with search cost and anticipated regret: Theoretical framework and field evidence. *Available at SSRN 3890056* .

Ma X, He J, Liao J (2021) Does decision fatigue affect institutional bidding behavior? evidence from chinese ipo market. *Economic Modelling* 98:1–12.

Mai Y, Hu B (2023) Optimizing free-to-play multiplayer games with premium subscription. *Management Science* 69(6):3437–3456.

Mas-Colell A, Whinston MD, Green JR (1995) *Microeconomic Theory* (Oxford University Press, New York).

Meng Z, Hao L, Tan Y (2021) Freemium pricing in digital games with virtual currency. *Information Systems Research* 32(2):481–496.

Roels G (2019) Optimal structure of experiential services: Review and extensions. *Handbook of Service Science, Volume II*, 105–146 (Springer).

Ruiz-Meza J, Montoya-Torres JR (2021) Tourist trip design with heterogeneous preferences, transport mode selection and environmental considerations. *Annals of Operations Research* 305(1):227–249.

Runge J, Nair H, Levav J (2021) Price promotions for "freemium" app monetization. *Available at SSRN 3357275* .

Ryan CT, Sheng L, Zhao X (2020) Strategic timing and pricing for selling bonus actions in video games. *Available at SSRN 3751523* .

Schell J (2019) *The Art of Game Design: A Book of Lenses* (CRC press).

Shah AM, Wolford G (2007) Buying behavior as a function of parametric variation of number of choices. *PSYCHOLOGICAL SCIENCE-CAMBRIDGE-* 18(5):369.

Sheng L, Ryan CT, Nagarajan M, Cheng Y, Tong C (2022) Incentivized actions in freemium games. *Manufacturing & Service Operations Management* 24(1):275–284.

Song Y, Ulmer MW, Thomas BW, Wallace SW (2020) Building trust in home services—stochastic team-orienteering with consistency constraints. *Transportation Science* 54(3):823–838.

Tong LC, Zhou L, Liu J, Zhou X (2017) Customized bus service design for jointly optimizing passenger-to-vehicle assignment and vehicle routing. *Transportation Research Part C: Emerging Technologies* 85:451–475.

Totten CW (2017) *Level design: Processes and experiences* (CRC Press).

Tsiligirides T (1984) Heuristic methods applied to orienteering. *Journal of the Operational Research Society* 35(9):797–809.

Turner J, Scheller-Wolf A, Tayur S (2011) Scheduling of dynamic in-game advertising. *Operations Research* 59(1):1–16.

Vohs KD, Baumeister RF, Schmeichel BJ, Twenge JM, Nelson NM, Tice DM (2018) Making choices impairs subsequent self-control: A limited-resource account of decision making, self-regulation, and active initiative. *Self-regulation and self-control*, 45–77 (Routledge).

Vu D, Zhao X, Stecke K (2020) Pay-to-win in video games: Microtransactions and fairness concerns. *Available at SSRN 3658537* .

Xu Y, Scheller-Wolf A, Sycara K (2015) The benefit of introducing variability in single-server queues with application to quality-based service domains. *Operations Research* 63(1):233–246.

Yu Q, Adulyasak Y, Rousseau LM, Zhu N, Ma S (2021) Team orienteering with time-varying profit. *INFORMS Journal on Computing* .

E-companion for "Optimal world design in video games"

Appendix A: Technical Proofs

sec:appendix-proof

A.1 Proof of Theorem 1

ss:proof-of-theorem-game-duration

Given G and b, the player chooses $t \in D_G$ to maximize his utility $\pi(t|G,b) = u(t) - q(t|b) - F(G)$. Note that the term $F(G)$ is unrelated to the player's decision. We claim that $\pi(t|G,b)$ is an unimodal function of t that achieves its unique maximum at $t = b$. Indeed, by (2), u is strictly increasing and $q(t|b) = 0$ on $0 \le t < b$. Thus, π is strictly increasing on $0 \le t < b$. By Assumption 2, $u(t) - q(t|b)$ is strictly decreasing on $t > b$, and thus so is π. This implies that π is unimodal and achieves its unique maximum at $t = b$.

Thus, if $b \in D_G$ then b is clearly the unique minimizer of $\max_{t \in D_G} \pi(t|G,b)$. If $b \notin D_G$ then one of either $\lfloor b \rfloor_G$ or $\lceil b \rceil_G$ is optimal. This follows since π is strictly increasing when $0 \le t < b$ and strictly decreasing on $t > b$. Whichever of $\lfloor b \rfloor_G$ or $\lceil b \rceil_G$ achieves the highest value in π is thus the optimal solution to $(P|G,b)$. In other words, $\text{proj}(b)$ is the set of optimal solutions of $(P|G,b)$. □

A.2 Proof of Theorem 2

ss:proof-of-prop-single-player

In this setting, the game designer's problem is:

$$\max_{G \le \mathscr{U}} \max_{t \in D_G} \pi(t|G,b).$$

Observe that the term $F(G)$ is a constant with respect to the choice of $t \in D_G$ and so we can rewrite this problem as:

$$\max_{G \le \mathscr{U}} \left[\max_{t \in D_G} (u(t) - q(t|b)) - F(G) \right].$$

Using the optimality structure established in (10) the problem amounts to solving:

$$\max_{G \le \mathscr{U}} \left[u(t^*_{G,b}) - q(t^*_{G,b}|b) - F(G) \right].$$

Now, by the optimality structure of $t^*_{G,b}$ we know that the player will select the path with duration either $\lfloor b \rfloor_G$ or $\lceil b \rceil_G$, where these values are defined in (9).

Notice that there is no utility gained by offering more than one path in a world map since a player will only select one of these paths anyway, and the game designer will know which one that is by the optimality structure of $t^*_{G,b}$. Thus, the choice of G is restricted to a line graph; that is, $G = L_k$ for some k. We only need to consider what length of path to choose.

When restricting our attention to a line graph $G = L_k$ for some k, we observe that $D_G = \{k\}$ is a singleton. Therefore, the player's game time must be $t^*_{G,b} = k$ and the resulting player utility is $u(k) - q(k|b) - F(L_k)$. Assumption 1 implies that the growth in fatigue as k increases is less than the growth in utility. Thus, when $k < b$, $q(k|b) = 0$ and $u(k) - q(k|b) - F(L_k)$ increases in $k < b$. When $k > b$, $u(k) - q(k|b)$ decreases in $k \geq b$ (by Assumption 2) and $F(L_k)$ is nondecreasing in k (by (6)). Thus, $u(k) - q(k|b) - F(L_k)$ decreases in $k > b$.

In conclusion, the function $u(k) - q(k|b) - F(L_k)$ increases in $k < b$ but decreases in $k > b$, implying that $k = b$ is optimal. □

A.3 Proof of Lemma 1

ss:property-of-side-quest-tree

We need to show T^D: (i) is acyclic, (ii) has no dead ends, and (iii) is unilaterally connected.

For (i) observe that all maximal directed paths reach vertex r, but r has no outgoing edges. This means the path cannot return to any of its earlier edges, and so there are no directed cycles.

For (ii), there are two types of edges: (a) edges on the path $(1, 2, \ldots, \hat{d})$ and (b) edges from that path to r. As $\hat{d} = \max D \in D$, the side-quest tree T^D must include the edge $(\hat{d}, r)$ following its definition. Then the path $(1, 2, \ldots, \hat{d}, r)$ is a complete path, and those edges in case (a) are contained in this complete path. Similarly, edges in case (b) are those additional edges (v, r) for all $v \in D$. The path $(1, 2, \ldots, v, r)$ is a complete path and contains the edge (v, r). □

A.4 Proof of Lemma 2

ss:counts-for-side-quest-trees

The vertex count is straightforward, by construction, the graph contains the vertices $\{1, 2, \ldots, \hat{d}\}$ along with the end vertex r. That is exactly $1 + \hat{d}$ vertices. The set of all complete paths of T^D is $\{p_d : d \in D\}$ where $p_d := (1, \ldots, d, r)$ for all $d \in D$. Clearly, p_d has duration d. This implies T^D has exactly one path for each duration $d \in D$. □

A.5 Proof of Lemma 3

proof-lemma:complexity-minimum-paths-and-vertices

Each complete path in a graph has a duration. Thus, each duration must be associated with at least one complete path. As $D \subseteq D_G$, the graph G has at least $|D|$ different durations in the duration set, implying at least $|D|$ number of complete paths.

Moreover, let $\hat{d} = \max D$ be the maximal duration in D. The complete path that provides this duration $\hat{d}$ must contain $1 + \hat{d}$ vertices, which offers a minimal value on the number of vertices. □

A.6 Proof of Theorem 3

proof-thm:paths-and-vertices-T-N-subgraph-best

We prove the result by contradiction. Suppose the optimal solution to (WMP) is not a side-quest tree. We denote this optimal world map as G^* and its resulting duration set as D^*.

Now we consider a side-quest tree T^{D^*}. By construction, T^{D^*} has the same duration set of G^*. Therefore, for all players, their play time, utility from play, and impatience penalty are identical in the two graphs T^{D^*} and G^*. Furthermore, Lemma 2 and Lemma 3 imply that the number of paths and vertices of the side-quest tree T^{D^*} match the minima and thereby field the smallest possible fatigues among graphs that cover duration set D^*. Thus, $F(T^{D^*}) \le F(G^*)$. We conclude $\Pi(T^{D^*}) \ge \Pi(G^*)$, that is the expected utility of players under the side-quest tree T^{D^*} is weakly higher than that under G^*.

If $\Pi(T^{D^*}) > \Pi(G^*)$, it contradicts the fact that G^* is the optimal world map. If $\Pi(T^{D^*}) = \Pi(G^*)$, meaning that T^{D^*} performs equally well as G^*, then D^* should also be an optimal world map. But this contradicts the initial assumption that the optimal solution to (WMP) is not a side-quest tree.

From above, we have proven that there must exist an optimal solution to (WMP) that is a side-quest tree. □

A.7 Proof of Proposition 1

proof-prop:optimal-linear-complexity-has-at-most-D-paths

Following Theorem 3, there exists an optimal solution to (WMP) that is a side-quest tree T^D. If $|D| \le |\mathcal{B}|$, we are done. If $|D| > |\mathcal{B}|$, it implies that there exists at least one complete path in the graph T^D that is not selected by players. We construct a new side-quest tree $T^{D'}$, where $D' = \{t^*_{T^D,b} | b \in \mathcal{B}\}$ and $t^*_{T^D,b}$ is the optimal play time of a player with time budget b under the graph T^D. We have $|D'| \le |\mathcal{B}|$.

Compared to the original side-quest tree T^D, we observe that the new side-quest tree $T^{D'}$ has all the durations that were in use. Therefore, utility from play and impatience penalty remain the same. Since $|D'| < |D|$, the side-quest tree $T^{D'}$ contains fewer paths and vertices than T^D, leading to a strictly smaller decision fatigues disutility. That is, $F(T^{D'}) < F(T^D)$. As a result, we conclude $\Pi(T^{D'}) > \Pi(T^D)$. However, this contradicts the fact that T^D is the optimal world map. As a result, we cannot have $|D| > |\mathcal{B}|$. We have proven Proposition 1. □

A.8 Proof of Proposition 2

proof-lemma:complexity-minimum-paths-and-vertices

Proposition 1 indicates that there exists an optimal solution to (WMP) that is a side-quest tree T^D with $|D| \le |\mathcal{B}|$. If $D \subseteq \mathcal{B}$, we are done. Otherwise, we claim that we can construct a new duration set D' with the following properties:

1. $(D' \setminus \mathcal{B}) \subset (D \setminus \mathcal{B})$. The new set D' has fewer elements that are not contained in $\mathcal{B}$.
2. $\Pi(T^D) \le \Pi(T^{D'})$. The new side-quest tree $T^{D'}$ does not lower the expected utility.

The construction works as below: We let $U = D \cup \mathcal{B}$ be the union of D and $\mathcal{B}$. Let $t \in U \setminus \mathcal{B}$ be any element of U not contained in $\mathcal{B}$, which exists because we assume $U \setminus \mathcal{B} = D \setminus \mathcal{B}$ is not empty. The new duration set D' is constructed by replacing t with a different value $d \in U$. In other words, $D' = D \cup \{d\} \setminus \{t\}$ for some $d \in U$.

By the above construction, the element $t \in D \setminus \mathcal{B}$ is removed in the new duration set D'. That is, we remove an element that is contained in D but not in $\mathcal{B}$. In addition, the replacing element d is either an element of $\mathcal{B}$ or an element of D, so we are not introducing any new element outside of D or $\mathcal{B}$. As a result, we conclude that $(D' \setminus \mathcal{B})$ is a proper subset of $(D \setminus \mathcal{B})$. The first property holds.

To prove the second property, we split the discussions into three cases.

Case 1: t is the greatest element in U.

Let $\hat{b} = \max \mathcal{B}$. Since we assume t is the greatest element in U and $t \notin \mathcal{B}$, we have $t > \hat{b}$. Assumption 2 implies that, for players who chose the path with duration t, a path with duration $\hat{b}$ will give them higher utility for play and lower impatience penalty. Therefore, we let $D' = D \cup \{\hat{b}\} \setminus \{t\}$. The resulting side-quest tree $T^{D'}$ increases the game entertainment (defined as utility from play minus impatience penalty) of those players who previously selected the path with duration t, while it does not affect the game entertainment of other players. Furthermore, the decision fatigue will decrease, because the number of vertices decreases and the number of paths is either the same or lower, depending on whether $\hat{b}$ was originally in D. In conclusion, the overall expected utility will not decrease, i.e., $\Pi(T^D) \le \Pi(T^{D'})$.

Case 2: t is not the greatest element in U, neither is it the greatest element in D.

It suffices to restrict our attention to those players who choose the path with duration t under the original side-quest tree T^D. Let Λ be the proportion of players choosing the path with duration t, i.e., $\Lambda = \sum_{b \text{ prefers } t} m(b) = \sum_{\{b: t^*_{T^D,b} = t\}} m(b)$. We define $GE(d)$ as the total game entertainment (utility from play minus impatience penalty) when all players who preferred the path with duration t are instead changed to use a path with duration d[19]. That is,

$$GE(d) = \Lambda \alpha d - \sum_{\{b: b<d, t^*_{T^D,b} = t\}} m(b)\beta(d-b).$$

Thus, we obtain

$$\begin{aligned} GE(d+1) - GE(d) &= \Lambda\alpha - \sum_{\{b: b<d+1, t^*_{T^D,b}=t\}} m(b)\beta(d+1-b) + \sum_{\{b: b<d, t^*_{T^D,b}=t\}} m(b)\beta(d-b) \\ &= \Lambda\alpha - \sum_{\{b: b\le d, t^*_{T^D,b}=t\}} m(b)\beta. \end{aligned}$$

In particular, we consider the case when $d = t$. Since $t \notin \mathcal{B}$, $m(t) = 0$. Thus,

$$\begin{aligned} GE(t+1) - GE(t) =& \Lambda\alpha - \sum_{\{b: b\le t, t^*_{T^D,b}=t\}} m(b)\beta \\ =& \Lambda\alpha - \sum_{\{b: b\le t-1, t^*_{T^D,b}=t\}} m(b)\beta \end{aligned}$$

[19] Here we force all players to choose the new duration d when the duration t were removed. If players are allowed to choose their optimal paths in the new side-quest tree, the total game entertainment will be even higher than $GE(d)$. So our proof will still hold.

$$=GE(t)-GE(t-1).$$

If $GE(t+1)-GE(t)>0$, then the path with duration $t+1$ results in a larger game entertainment. So we replace t with $t+1$. That is, $D'=D\cup\{t+1\}\setminus\{t\}$. If $GE(t+1)-GE(t)<0$, implying $GE(t)-GE(t-1)<0$, then the path with duration $t-1$ results in a larger game entertainment. So we replace t with $t-1$. That is, $D'=D\cup\{t-1\}\setminus\{t\}$. If $GE(t+1)-GE(t)=0$ and then $GE(t)-GE(t-1)=0$, all three durations provide equal game entertainment. So t can be replaced with either $t-1$ or $t+1$.

Since t is not the largest element in D, removing t will not change the number of vertices n_v. The number of paths n_p will either stay the same or decrease by 1, depending on whether or not the new duration is already in D. Therefore, the decision fatigue will decrease or remain unchanged, and the game entertainment will increase or remain unchanged. In conclusion, the overall expected utility will not decrease, i.e., $\Pi(T^D)\le\Pi(T^{D'})$.

Case 3: t is not the greatest element in U, but it is the greatest element in D.

This case follows the previous case. The only difference is that removing t will change the number of vertices n_v as t is the largest element in D. Since we assume the fatigue function is a linear function of the number of vertices and the number of paths, we denote w_v as the marginal fatigue caused by adding a vertex. We would consider the following differences:

$$\begin{aligned}GE(t+1)-GE(t)-w_v=&\Lambda\alpha-\sum_{\{b:b\le t,t^*_{T^D,b}=t\}}m(b)\beta-w_v\\=&\Lambda\alpha-\sum_{\{b:b\le t-1,t^*_{T^D,b}=t\}}m(b)\beta-w_v\\=&GE(t)-GE(t-1)-w_v.\end{aligned}$$

If $GE(t+1)-GE(t)-w_v>0$, then t can be replaced with $t+1$. So $D'=D\cup\{t+1\}\setminus\{t\}$. Under the new side-quest tree $T^{D'}$, the number of vertices is increased by 1 and the number of paths n_p is unchanged. Hence, the decision fatigue will increase. However, the gain of game entertainment surpasses the increase of decision fatigue. Therefore, the overall expected utility will not decrease.

If $GE(t+1)-GE(t)-w_v<0$, implying $GE(t)-GE(t-1)-w_v<0$, then t can be replaced with $t-1$. So $D'=D\cup\{t-1\}\setminus\{t\}$. In this case, the decision fatigue will decrease since the number of vertices n_v decreases and the number of paths n_p will either stay the same or decrease by 1, depending on whether or not the new duration $t-1$ is in D. The benefit from lowering the decision fatigue dominates the change of game entertainment. Thus, the overall expected utility will not decrease.

If $GE(t+1)-GE(t)-w_v = GE(t)-GE(t-1)-w_v = 0$, then t can be replaced with either $t-1$ or $t+1$. The number of paths n_p will either stay the same or decrease by 1, depending on whether or not the new duration is in D. Again, the overall expected utility will not decrease in this case.

From the above discussion, we show that we have constructed a new duration set D' which contains fewer elements that are not contained in $\mathcal{B}$ and does not decrease the expected utility. In other words, by the above construction, we are able to remove one element in $D \setminus \mathcal{B}$ without lowering the expected utility. Because the original duration set D has finitely many elements that are not contained in $\mathcal{B}$. We can repeat the above construction by a finite number of times and remove all elements in D that are not contained in $\mathcal{B}$, eventually resulting in a new duration set $D*$ such that $D^* \subseteq \mathcal{B}$. By the second property, we guarantee $\Pi(T^D) \le \Pi(T^{D^*})$. We have completed the proof. □

Appendix B: Proofs regarding Algorithm 1 for the optimal side quest tree

sec:ia-general-v-p

B.1 Proof of Lemma 4

By Theorem 3, there exists an optimal solution to (WMP) that is a side quest tree. We let T^* be the optimal side quest tree, i^* be the length of the longest complete path in T^*, and μ^* be the number of complete paths in T^*. Clearly, $i^* \in [N]$ and $\mu^* \in [i^*]$ by Lemma 2. Since T^* is an optimal solution to (WMP) and yields the maximum expected utility, it should also be an optimal solution to (WMP$_{i,\mu}$) with $i = i^*$ and $\mu = \mu^*$. Thus, T^* must be an optimal (i^*, μ^*)-SQT, which implies that the optimal side quest tree can be found in the sets of $\mathcal{S}^*_{i,\mu}$ for $i \in [N]$ and $\mu \in [i]$.

By Proposition 1, there are at most $|\mathcal{B}|$ different elements in the duration set of the optimal side quest tree. Therefore, the capacity of the optimal side quest tree is at most $|\mathcal{B}|$. Thus, the optimal side quest tree can be found in sets $\mathcal{S}^*_{i,\mu}$ for $i \in [N]$ and $\mu \in [|\mathcal{B}|]$. □

B.2 Proof of Lemma 5

Recall from (17), the designer's objective consists of two terms: $U(G)$ is the expected utility from play minus the impatience penalty (below we refer it as "game entertainment"), and $F(G)$ indicates the disutility from decision fatigue.

Consider the difference in the expected utility (i.e., the designer's objective) under a single-path transformation, for $i, j, \mu \in [N]$, $i < j$ and $1 < \mu \le j$, we have:

$$\begin{aligned}
\Pi(\psi_{ij}(T_{i,\mu-1})) - \Pi(T_{i,\mu-1}) =& [U(\psi_{ij}(T_{i,\mu-1})) - U(T_{i,\mu-1})] - [F(\psi_{ij}(T_{i,\mu-1})) - F(T_{i,\mu-1})] \\
=& \Delta U_{ij}(\mu-1) - \Delta F_{ij}(\mu-1).
\end{aligned}$$

We denote $\Delta U_{ij}(\mu-1) = U(\psi_{ij}(T_{i,\mu-1})) - U(T_{i,\mu-1})$ and $\Delta F_{ij}(\mu-1) = F(\psi_{ij}(T_{i,\mu-1})) - F(T_{i,\mu-1})$. In what follows, we investigate the two terms $\Delta U_{ij}(\mu-1)$ and $\Delta F_{ij}(\mu-1)$.

(1): Decision fatigue increment after single-path transformation $\Delta F_{ij}(\mu-1)$

By its definition, the single-path transformation ψ_{ij} that maps an $(i,\mu-1)$-SQT $T_{i,\mu-1}$ to an (j,μ)-SQT $T_{j,\mu}$ introduces $j-i$ additional vertices and 1 additional complete path. In particular, the original side-quest tree $T_{i,\mu-1}$ has $i+1$ vertices and $\mu-1$ complete paths, and the new side-quest tree $\psi_{ij}(T_i)$ has $j+1$ vertices and μ complete paths. Finally, the decision fatigue increment after the single-path transformation ψ_{ij} is equal to

$$\begin{aligned}\Delta F_{ij}(\mu-1) =&F(\psi_{ij}(T_{i,\mu-1}))-F(T_{i,\mu-1})\\ =&F(j+1,\mu)-F(i+1,\mu-1).\end{aligned}$$

(2): Game entertainment increment after single-path transformation $\Delta U_{ij}(\mu-1)$

Recall from earlier that game entertainment indicates players' utility from play minus impatience penalty. Specifically, given a graph G, and a player with budget b chooses his optimal duration $t^*_{G,b}$ (specified in (10)) and his game entertainment is defined as $u(t^*_{G,b})-q(t^*_{G,b}|b)$ where u satisfies (3) and q satisfies (5).

Consider the original side quest tree $T_{i,\mu-1}$ and the new side quest tree $\psi_{ij}(T_{i,\mu-1})$ resulting from the single-path transformation. Then,

$$\begin{aligned}\Delta U_{ij}(\mu-1) &= \mathbb{E}_B[u(t^*_{\psi_{ij}(T_{i,\mu-1}),B})-q(t^*_{\psi_{ij}(T_{i,\mu-1}),B}|B)]-\mathbb{E}_B[u(t^*_{T_{i,\mu-1},B})-q(t^*_{T_{i,\mu-1},B}|B)]\\ &=\sum_{b=1}^{N}\left\{\left[u(t^*_{\psi_{ij}(T_{i,\mu-1}),b})-u(t^*_{T_{i,\mu-1},b})\right]-\left[q(t^*_{\psi_{ij}(T_{i,\mu-1}),b}|b)-q(t^*_{T_{i,\mu-1},b}|b)\right]\right\}m(b).\end{aligned}$$

What impacts the difference $\Delta U_{ij}(\mu-1)$ are the players' optimal durations $t^*_{T_i,b}$ and $t^*_{\psi_{ij}(T_i),b}$ under the two side quest trees $T_{i,\mu-1}$ and $\psi_{ij}(T_{i,\mu-1})$. Thus, we need to explore how players will adjust their path decisions after the single-path transformation ψ_{ij}.

Observe that the single-path transformation ψ_{ij} adds one single path, which is the j-length path. Hence, given the new side quest tree $\psi_{ij}(T_{i,\mu-1})$, players only need to think about whether to stay on their original path or switch to the new j-length path. We make the following claims about players' path decisions after the single-path transformation ψ_{ij}.

(i) For players with time budgets at most i, their decisions of optimal path remain the unchanged. That is, if a player with budget $b\le i$ selected the path of length k (for some duration k) under the original side quest tree $T_{i,\mu-1}$, he will continue to select the same path under the new side quest tree $\psi_{ij}(T_{i,\mu-1})$.

Suppose those players with time budgets at most i switch to the new j-length path. Their utility from play increases, and so does the impatience penalty. However, we assume the growth in impatience penalty is greater than the growth in utility from play (Assumption 2). Then switching to the new longer path will make those players worse off. Thus, under the new side quest tree $\psi_{ij}(T_{i,\mu-1})$, players with time budgets at most i would still prefer the same optimal path as they did under the original side quest tree T_i.

As a result, for players with time budgets at most i, there is no change in their game entertainment after the single-path transformation, i.e.,

$$\sum_{b=1}^{i}\left\{\left[u(t^*_{\psi_{ij}(T_{i,\mu-1}),b})-u(t^*_{T_{i,\mu-1},b})\right]-\left[q(t^*_{\psi_{ij}(T_{i,\mu-1}),b}|b)-q(t^*_{T_{i,\mu-1},b}|b)\right]\right\}m(b)=0.$$

(ii) For players with time budgets of at least j, they selected the longest path with length i under the original side-quest tree $T_{i,\mu-1}$. Now given the new side quest tree $\psi_{ij}(T_{i,\mu-1})$ with the new j-length path, those players will all switch to this longer j-length path, because it gives them higher utility from play and does not incur any impatience penalty.

Thus, for players with time budget of at least j, the change of their game entertainment after the single-path transformation is equal to

$$\begin{aligned}&\sum_{b=j}^{N}\left\{\left[u(t^*_{\psi_{ij}(T_{i,\mu-1}),b})-u(t^*_{T_{i,\mu-1},b})\right]-\left[q(t^*_{\psi_{ij}(T_{i,\mu-1}),b}|b)-q(t^*_{T_{i,\mu-1},b}|b)\right]\right\}m(b)\\&=\sum_{b=j}^{N}\left\{[u(j)-u(i)]-[0-0]\right\}m(b)\\&=\sum_{b=j}^{N}\alpha(j-i)m(b).\end{aligned}$$

(iii) For players with time budgets strictly between i and j, they used to select the longest path with length i under the original side-quest tree $T_{i,\mu-1}$. Given the new side-quest tree, they must decide between choosing the i-length path that $\psi_{ij}(T_{i,\mu-1})$ inherits from $T_{i,\mu-1}$ or the new path j-length path added by the single-path transformation.

If a player with budget b chooses the original i-length path, his utility from play is αi and the impatience penalty is 0. If a player with budget b chooses the new j-length path, he earns the utility from play αj but pays the impatience penalty $\beta(j-b)$, resulting in a difference of $\alpha j-\beta(j-b)$.

Then we compare the two utilites αi and $\alpha j-\beta(j-b)$. We define $\bar{b}=\max\{\lceil\frac{\alpha i-(\alpha-\beta)j}{\beta}\rceil, i+1\}$ where $\frac{\alpha i-(\alpha-\beta)j}{\beta}$ is solved from the equation $\alpha i=\alpha j-\beta(j-b)$, and $\lceil\frac{\alpha i-(\alpha-\beta)j}{\beta}\rceil$ indicates the smallest integer that is not smaller than $\frac{\alpha i-(\alpha-\beta)j}{\beta}$. By its definition, we guarantee $i<\bar{b}<j$.

When $\bar{b} \le b \le j-1$, we have $\alpha i \le \alpha j - \beta(j-b)$, suggesting that players with time budget $b \in [\bar{b}, j-1]$ should switch to the new j-length path to get a higher game entertainment. When $i+1 \le b < \bar{b}$, we have $\alpha i > \alpha j - \beta(j-b)$, implying that players with time budget $b \in [i+1, \bar{b})$ should stay in the original i-length path.

Therefore, for players with time budgets strictly between i and j, the change of their game entertainment after the single-path transformation is computed by

$$\begin{aligned}
&\sum_{b=i+1}^{j-1} \left\{ \left[u(t^*_{\psi_{ij}(T_{i,\mu-1}),b}) - u(t^*_{T_{i,\mu-1},b}) \right] - \left[q(t^*_{\psi_{ij}(T_{i,\mu-1}),b}|b) - q(t^*_{T_{i,\mu-1},b}|b) \right] \right\} m(b) \\
=& \sum_{b=i+1}^{\bar{b}-1} \left\{ \left[u(t^*_{\psi_{ij}(T_{i,\mu-1}),b}) - u(t^*_{T_{i,\mu-1},b}) \right] - \left[q(t^*_{\psi_{ij}(T_{i,\mu-1}),b}|b) - q(t^*_{T_{i,\mu-1},b}|b) \right] \right\} m(b) \\
&+ \sum_{b=\bar{b}}^{j-1} \left\{ \left[u(t^*_{\psi_{ij}(T_{i,\mu-1}),b}) - u(t^*_{T_{i,\mu-1},b}) \right] - \left[q(t^*_{\psi_{ij}(T_{i,\mu-1}),b}|b) - q(t^*_{T_{i,\mu-1},b}|b) \right] \right\} m(b) \\
=& 0 + \sum_{b=\bar{b}}^{j-1} \left\{ [\alpha j - \alpha i] - [\beta(j-b) - 0] \right\} m(b) \\
=& \sum_{b=\bar{b}}^{j-1} ((\alpha-\beta)j + \beta b - \alpha i) m(b).
\end{aligned}$$

Following the above discussion, we conclude the game entertainment (i.e., utility from play minus impatience penalty) increment after the single-path transformation ψ_{ij} to be

$$\begin{aligned}
\Delta U_{ij}(\mu-1) &= \sum_{b=1}^{N} \left\{ \left[u(t^*_{\psi_{ij}(T_{i,\mu-1}),b}) - u(t^*_{T_{i,\mu-1},b}) \right] - \left[q(t^*_{\psi_{ij}(T_{i,\mu-1}),b}|b) - q(t^*_{T_{i,\mu-1},b}|b) \right] \right\} m(b) \\
&= \sum_{b=j}^{N} \alpha(j-i) m(b) + \sum_{b=\bar{b}}^{j-1} ((\alpha-\beta)j + \beta b - \alpha i) m(b).
\end{aligned}$$

It is straightforward to see that $\Delta U_{ij}(\mu-1)$ is independent of μ, which is the capacity of graph $T_{i,\mu-1}$. Therefore, we use ΔU_{ij} instead of $\Delta U_{ij}(\mu-1)$ hearafter.

To sum up, we have proven $\Pi(\psi_{ij}(T_{i,\mu-1})) - \Pi(T_{i,\mu-1}) = \Delta U_{ij} - \Delta F_{ij}(\mu-1)$ where $\Delta U_{ij} = \sum_{b=j}^{N} \alpha(j-i)m(b) + \sum_{b=\bar{b}}^{j-1} ((\alpha-\beta)j + \beta b - \alpha i) m(b)$ and $\Delta F_{ij}(\mu-1) = F(j+1,\mu) - F(i+1,\mu-1)$ for $i,j,\mu \in [N]$, $i<j$ and $1<\mu \le j$. The subscripts of ΔU_{ij} and $\Delta F_{ij}(\mu-1)$ reflect the changes in the number of vertices. We remark that ΔU_{ij} only depends on i and j, but not on the number of complete paths μ, and $\Delta F_{ij}(\mu-1)$ depends on all of i, j, and μ. □

B.3 Proof of Lemma 6

(i) It is straightforward to see that the optimal (j,μ)-SQT $T^*_{j,\mu}$ where $j \in [N]$ and $\mu \in \{2,\dots,j\}$ must be constructed by a single-path transformation from a $(i,\mu-1)$-SQT $T_{i,\mu-1}$ where $i \in \{1,\dots,j-1\}$. Indeed, this $(i,\mu-1)$-SQT $T_{i,\mu-1}$ can be retrieved backwards by removing the j-length path. The remaining question is that whether $T_{i,\mu-1}$ is an optimal $(i,\mu-1)$-SQT (i.e., whether $T_{i,\mu-1} \in \mathcal{S}^*_{i,\mu-1}$).

We prove by contradiction. Suppose $T^*_{j,\mu} = \Pi(\psi_{ij}(T_{i,\mu-1}))$ and $T_{i,\mu-1}$ is not an optimal (i,μ)-SQT. We let $T^*_{i,\mu-1}$ be an optimal $(i,\mu-1)$-SQT. Then $\Pi(T^*_{i,\mu-1}) > \Pi(T_{i,\mu-1})$. Following Lemma 5, we have $\Pi(T^*_{j,\mu}) - \Pi(T_{i,\mu-1}) = \Pi(\psi_{ij}(T_{i,\mu-1})) - \Pi(T_{i,\mu-1}) = \Delta U_{ij} - \Delta F_{ij}(\mu-1)$ and $\Pi(\psi_{ij}(T^*_{i,\mu-1})) - \Pi(T^*_{i,\mu-1}) = \Delta U_{ij} - \Delta F_{ij}(\mu-1)$. Thus, the two differences $\Pi(T^*_{j,\mu}) - \Pi(T_{i,\mu-1})$ and $\Pi(\psi_{ij}(T^*_{i,\mu-1})) - \Pi(T^*_{i,\mu-1})$ should be the same, i.e., $\Pi(T^*_{j,\mu}) - \Pi(T_{i,\mu-1}) = \Pi(\psi_{ij}(T^*_{i,\mu-1})) - \Pi(T^*_{i,\mu-1})$. Because $\Pi(T^*_{i,\mu-1}) > \Pi(T_{i,\mu-1})$, we must end up with $\Pi(\psi_{ij}(T^*_{i,\mu-1})) > \Pi(T^*_{j,\mu})$. But this contradicts the fact that $T^*_{j,\mu}$ is an optimal (j,μ)-SQT. We reach a contradiction. Thus, for every optimal (j,μ)-SQT $T^*_{j,\mu}$, there must be an optimal $(i,\mu-1)$-SQT $T^*_{i,\mu-1} \in \mathcal{S}^*_{i,\mu-1}$ such that $T^*_{j,\mu} = \psi_{ij}(T^*_{i,\mu-1})$. In other words, the optimal (j,μ)-SQT is created by a single-path transformation from an optimal $(i,\mu-1)$-SQT. $\square$

(ii) Suppose $\psi_{ij}(T^*_{i,\mu-1})$ is an optimal (j,μ)-SQT that arises from an optimal $(i,\mu-1)$-SQT $T^*_{i,\mu-1} \in \mathcal{S}^*_{i,\mu-1}$. Then $\psi_{ij}(T^*_{i,\mu-1})$ has the maximum expected utility among all (j,μ)-SQT, i.e., $\Pi(\psi_{ij}(T^*_{i,\mu-1})) \geq \Pi(\psi_{ij}(T_{i,\mu-1}))$ for any $T_{i,\mu-1} \in \mathcal{S}_{i,\mu-1}$. Consider another optimal $(i,\mu-1)$-SQT $\hat{T}^*_{i,\mu-1} \in \mathcal{S}^*_{i,\mu-1}$ $(\hat{T}^*_{i,\mu-1} \neq T^*_{i,\mu-1})$. Since both are optimal $(i,\mu-1)$-SQT, we have $\Pi(T^*_{i,\mu-1}) = \Pi(\hat{T}^*_{i,\mu-1})$. By Lemma 5, we have $\Pi(\psi_{ij}(T^*_{i,\mu-1})) = \Pi(T^*_{i,\mu-1}) + \Delta U_{ij} - \Delta F_{ij}(\mu-1)$ and $\Pi_i(\psi_{ij}(\hat{T}^*_{i,\mu-1})) = \Pi(\hat{T}^*_{i,\mu-1}) + \Delta U_{ij} - \Delta F_{ij}(\mu-1)$. Therefore, $\Pi(\psi_{ij}(T^*_{i,\mu-1})) = \Pi(T^*_{i,\mu-1}) + \Delta U_{ij} - \Delta F_{ij}(\mu-1) = \Pi(\hat{T}^*_{i,\mu-1}) + \Delta U_{ij} - \Delta F_{ij}(\mu-1) = \Pi_i(\psi_{ij}(\hat{T}^*_{i,\mu-1}))$. Additionally, it implies $\Pi(\psi_{ij}(\hat{T}^*_{i,\mu-1})) = \Pi(\psi_{ij}(T^*_{i,\mu-1})) \geq \Pi(\psi_{ij}(T_{i,\mu-1}))$ for any $T_{i,\mu-1} \in \mathcal{S}_{i,\mu-1}$. That is, $\psi_{ij}(\hat{T}^*_{i,\mu-1})$ is also an optimal (j,μ)-SQT with $\Pi(\psi_{ij}(T^*_{i,\mu-1})) = \Pi_i(\psi_{ij}(\hat{T}^*_{i,\mu-1}))$. $\square$

(iii) We prove by contradiction. Clearly any $\psi_{ij}(T^*_{i,\mu-1})$ where $i \in [j-1]$ is an element of $\mathcal{S}_{j,\mu}$. Let the graph $\psi_{ij}(T^*_{i,\mu-1})$ with largest expected utility among $i \in [j-1]$ be denoted as $\hat{T}_{j,\mu}$. Suppose $\hat{T}_{j,\mu}$ is not an element of $\mathcal{S}^*_{j,\mu}$. It implies that $\psi_{ij}(T^*_{i,\mu-1})$ is not an element of $\mathcal{S}^*_{j,\mu}$ for all $i \in [j-1]$.

Consider an optimal (j,μ)-LQST $T^*_{j,\mu}$. Following (i), $T^*_{j,\mu}$ must arise from an optimal $(i,\mu-1)$-SQT for some $i \in [j-1]$ which we denote as $\tilde{T}^*_{i,\mu-1}$. In other words, $T^*_{j,\mu} = \psi_{ij}(\tilde{T}^*_{i,\mu-1})$ and $\Pi(\psi_{ij}(\tilde{T}^*_{i,\mu-1})) \geq \Pi(T_{j,\mu})$ for all $T_{j,\mu} \in \mathcal{S}_{j,\mu}$. Following (ii), since both $\tilde{T}^*_{i,\mu-1}$ and $T^*_{i,\mu-1}$ are optimal $(i,\mu-1)$-SQT, we have $\Pi(\psi_{ij}(\tilde{T}^*_{i,\mu-1})) = \Pi(\psi_{ij}(T^*_{i,\mu-1}))$, implying that $\psi_{ij}(T^*_{i,\mu-1})$ must be an optimal (j,μ)-SQT as well. We obtain a contradiction. Thus, the graph in $\{\psi_{ij}(T^*_{i,\mu-1}) | i \in [j-1]\}$ with largest Π value must be an element of $\mathcal{S}^*_{j,\mu}$. $\square$

B.4 Proof of Theorem 4

The proof consists of two parts. We first prove Algorithm 1 outputs the optimal side-quest tree. Then we show Algorithm 1 runs in polynomial time.

(i) The optimality of Algorithm 1

Our algorithm builds on the optimal structure discussed in Theorem 3 as well as Lemmas 4-6.

By Lemma 5 and (i) of Lemma 6, Algorithm 1 calculates all possible objective values of the optimal (j,μ)-SQT recursively in line 9 using $\Pi_{i,j,\mu} = \Pi_{i,j} + \Delta U_{ij} - \Delta F_{ij}(\mu-1)$, where $\Pi_{i,j}$, ΔU_{ij}

and $F_{ij}(\mu-1)$ are given constants. By (ii) of Lemma 6, when there are multiple optimal $(i,\mu-1)$-SQTs, the (j,μ)-SQTs constructed by any optimal $(i,\mu-1)$-SQTs are optimal sharing the same objective value. Therefore, we can select any optimal $(i,\mu-1)$-SQT to construct the optimal (j,μ)-SQT.

Next, Algorithm 1 computes the objective value of the optimal (j,μ)-SQT in line 11 by choosing the maximum $\Pi_{i,j,\mu}$ for $i\in[j-1]$. It is ensured by (iii) of Lemma 6 that the (j,μ)-SQT in the set $\{\psi_{ij}(T^*_{i,\mu-1})|i\in[j-1]\}$ with the maximum objective value is an optimal (j,μ)-SQT. The algorithm then constructs the optimal (j,μ)-SQT from the $(i^*,\mu-1)$-SQT in lines 12–14 for any optimal (j,μ)-SQT where $j\in[N]$ and $\mu\in[\min\{|\mathcal{B}|,j\}]$.

Lemma 4 indicates that the optimal side quest tree is an optimal (j^*,μ^*)-SQT whose objective value is the maximum among the optimal (j,μ)-SQTs where $j\in[N]$ and $\mu\in[\min\{|\mathcal{B}|,j\}]$. Algorithm 1 finds the optimal side quest tree T^* in line 18 by comparing the objective values of the optimal j-LSQT for all $j\in[N]$ and $\mu\in[\min\{|\mathcal{B}|,j\}]$.

(ii) Computational complexity of the algorithm

Algorithm 1 has $O(N|\mathcal{B}|)$ stages. At stage $j\in[N]$ and $\mu\in[\min\{|\mathcal{B}|,j\}]$, the optimal (j,μ)-SQT is computed. It takes $O(N)$ iterations to compute the possible objective value of the optimal (j,μ)-SQT where $\mu>1$. Hence, the computational complexity of Algorithm 1 is $O(N^2|\mathcal{B}|)$, which is in polynomial time. $\square$