Generalized Parametric Path Problems

Kshitij Gajjar[1] **Girish Varma**[2] **Prerona Chatterjee**[3] **Jaikumar Radhakrishnan**[3]

[1]School of Computing, National University of Singapore, Singapore
[2]CSTAR and ML Lab, International Institute of Information Technology, Hyderabad, Telengana, India
[3] STCS, Tata Institute of Fundamental Research, Mumbai, Maharashtra, India

Abstract

Parametric path problems arise independently in diverse domains, ranging from transportation to finance, where they are studied under various assumptions. We formulate a general path problem with relaxed assumptions, and describe how this formulation is applicable in these domains.
We study the complexity of the general problem, and a variant of it where preprocessing is allowed. We show that when the parametric weights are linear functions, algorithms remain tractable even under our relaxed assumptions. Furthermore, we show that if the weights are allowed to be non-linear, the problem becomes NP-hard. We also study the multi-dimensional version of the problem where the weight functions are parameterized by multiple parameters. We show that even with 2 parameters, this problem is NP-hard.

1 INTRODUCTION

Parametric shortest path problems arise in graphs where the cost of an edge depends on a parameter. Many real-world problems lend themselves to such a formulation, e.g., routing in transportation networks parameterized by time/cost (Carstensen [1983], Mulmuley and Shah [2001], Dean [2004]), and financial investment and arbitrage networks (Hajela and Pandey [2014], Hau [2014], Moosa [2003]). Path problems have been studied independently in these domains, under specific assumptions that are relevant to the domain. For example, the time-dependent shortest path problem used to model transportation problems assumes a certain FIFO condition (Equation 2.1). Arbitrage problems only model the rate of conversion and are defined with respect to a single currency parameter. These assumptions reduce the applicability of such algorithms to other domains.

We propose a generalized model for parametric path problems with relaxed assumptions, giving rise to an expressive formulation with wider applicability. We also present specific instances of real-world problems where such generalized models are required (see Section 2).

Definition 1.1. *The input to a Generalized Path Problem (GPP) is a* 4*-tuple* $(G,W,L,\mathbf{x}_0)$*, where* $G=(V\cup\{s,t\},E)$ *is a directed acyclic graph with two special vertices s and t,* $W=\{w_e:\mathbb{R}^k\to\mathbb{R}^k\ :\ e\in E\}$ *is a set of weight functions on the edges of G,* $L\in\mathbb{R}^k$ *is a vector used for computing the cost of a path from the k parameters, and* $\mathbf{x}_0\in\mathbb{R}^k$ *is the initial parameter.* ◊

The aim in a GPP is to find an *s-t* path P in the graph G that maximizes the dot product of L with the *composition* of the weight functions on P, evaluated at the initial parameter $\mathbf{x}_0$.

Problem 1.2 (Generalized Path Problem (GPP))**.**
Input: *An instance* $(G,W,L,\mathbf{x}_0)$ *of GPP.*
Output: *An s-t path* $P=(e_1,\ldots,e_r)$ *which maximizes*

$$L\cdot w_{e_r}(w_{e_{r-1}}(\cdots w_{e_2}(w_{e_1}(\mathbf{x}_0))\cdots)).$$

When $k=1$*, we call the GPP a* scalar *GPP. Sometimes we ignore the* $\mathbf{x}_0$ *and just write* (G,W,L).

Scalar GPP (see Figure 1) models shortest paths by choosing weights $w_e(x)=a_e\cdot x+b_e$ and fixing $L=-1$, to convert it to a minimization problem. Scalar GPP also models currency arbitrage problems (Hajela and Pandey [2014], Cormen et al. [2009]), where the cost of a path is the product of its edge weights, by choosing weight functions to be lines passing through the origin with slopes equal to the conversion rate.

Further, GPP can model more general path problems which involve multiple parameters to be optimized. For example, in transport networks, one needs to find a path that optimizes parameters like time traveled, cost of transportation, convenience, polluting emissions, etc. (Kamishetty et al. [2020]). In finance problems, an entity can have investments in different asset classes like cash, gold, stocks, bonds, etc.,

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

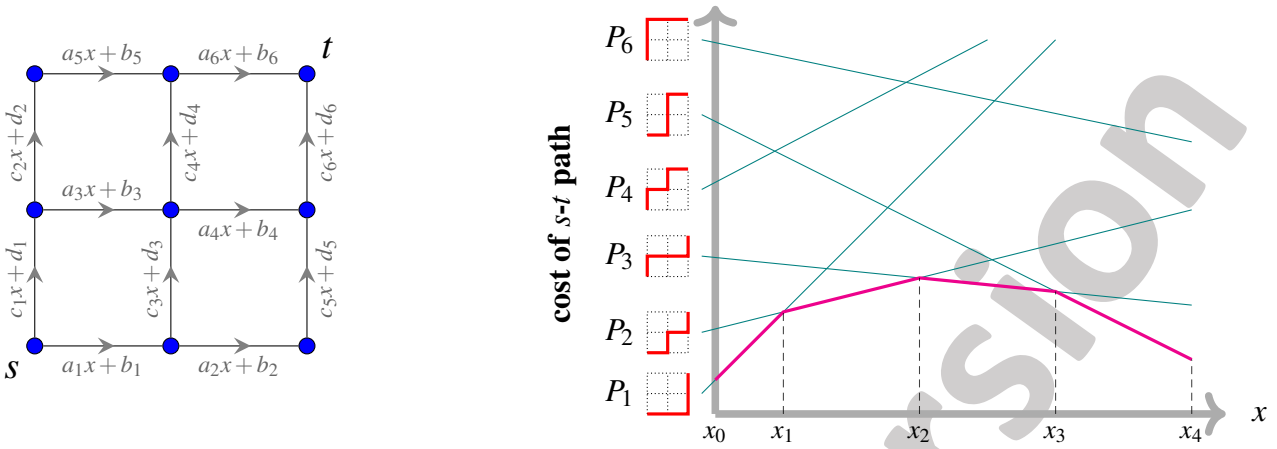


Figure 1: (Left) Graph of a scalar GPP instance whose edge weights are linear functions of a parameter x. (Right) A plot of x versus the costs of all possible s-t paths $P_1,\ldots,P_6$. All cost functions are linear, as the composition of linear functions is linear. For example, the cost of P_6 is $a_6(a_5(c_2(c_1x+d_1)+d_2)+b_5)+b_6$. Table 1 denotes the table for GPP with preprocessing (PGPP) with $L=-1$.

and a transaction, modeled by an edge, can affect these in complex ways (see more examples in Section 2). The edge parameters could contribute additively or multiplicatively to the cost of the path. We study weight functions that are affine linear transformations, which allows for both of these.

Note that the optimal path can vary based on the value of the initial parameter $\mathbf{x}_0$. For this, we also consider a version of GPP with preprocessing (called PGPP), where we can preprocess the inputs (G,W,L) and store them in a table which maps the initial values $\mathbf{x}_0$ to their optimal paths. Such a mapping is very useful in situations where the underlying network does not change too often and a large amount of computing power is available for preprocessing (e.g., the road map of a city typically does not change on a day-to-day basis). If the size of the table is managable, then it can be saved in memory and a query for an optimal path for a given $\mathbf{x}_0$ can be answered quickly using a simple table lookup.

Problem 1.3 (GPP with Preprocessing (PGPP))**.**
Input: *An instance* (G,W,L) *of GPP.*
Output: *A table which maps* $\mathbf{x}_0$ *to optimal paths.*

Table 1: PGPP Table for Figure 1

Interval	Path
(x_0,x_1)	P_1
(x_1,x_2)	P_2
(x_2,x_3)	P_3
(x_3,x_4)	P_5

We present an efficient algorithm for scalar GPP with linear weight functions. On the other hand, we show that if the GPP instance is non-scalar or the weight functions are non-linear, algorithms with worst-case guarantees cannot be obtained, assuming $\mathsf{P}\neq\mathsf{NP}$.

Our algorithm is based on the Bellman-Ford-Moore algorithm (Bellman [1958], Ford Jr [1956], Moore [1959]), whereas our NP-hardness reductions are from two well-known NP-hard problems, Set Partition and Product Partition.

The results for GPP with preprocessing (PGPP) are much more technical since they involve proving upper and lower bounds on the number of discontinuities of the cost of the optimal path as a function of the initial parameter x_0. They generalize previously known results in Time-Dependent Shortest Paths by Foschini et al. [2014]. Their work crucially uses the FIFO property Equation 2.1, whereas our analysis does not make this assumption, giving a more general result.

Our Contributions. Here is a summary of our results. In Section 2, we give specific instances from transportation and finance where these results can be applied.

1. There is an efficient algorithm for scalar GPP with linear weight functions (see Section 3).
2. Scalar PGPP with linear weight functions has a quasi-polynomial sized table, and thus table retrieval can be performed in poly-logarithmic time (see Section 4).
3. Scalar GPP with piecewise linear or quadratic weight functions is NP-hard to approximate (see Section 5).
4. For scalar PGPP with piecewise linear or quadratic weight functions, the size of the table could be exponential (see Section 6).
5. Non-scalar GPP (GPP with $k>1$) is NP-hard (see Section 7).

2 APPLICATIONS & RELATED WORKS

Applications to Transportation. We relate scalar GPP to an extensively well-studied problem, known as Time-Dependent Shortest Paths (TDSPs) in graphs, which comes up in routing/planning problems in transportation networks (Dean [2004], Dehne et al. [2012], Foschini et al. [2014]).

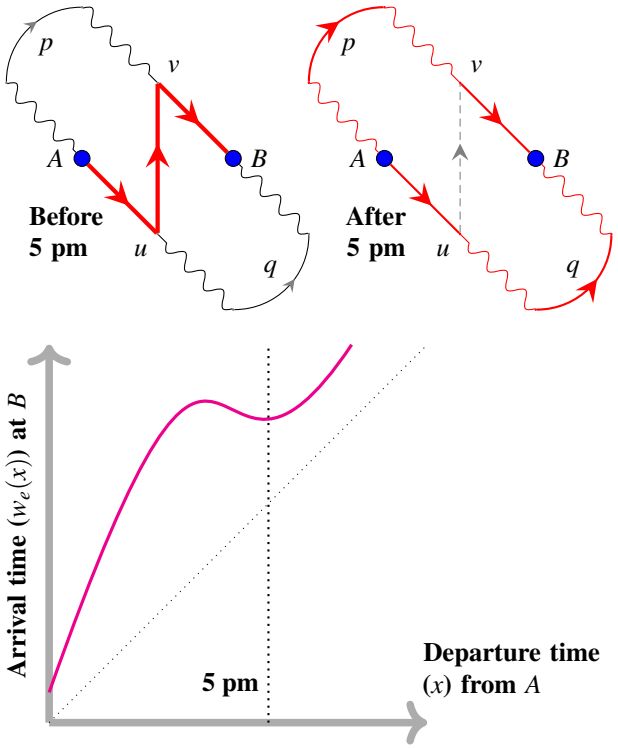


Figure 2: An illustration of how Braess' paradox can lead to a non-FIFO edge weight function. The plot denotes $w_e(x)$ for a single edge $e = (A,B)$ of a graph. Roads A-p-v and u-q-B are quite lengthy, and thus the road linking u to v is preferable for a journey from A to B. Before 5 pm, the u-v link is available, which leads to traffic congestion on the route A-u-v-B. Once the u-v link closes at 5 pm, the traffic splits equally on the routes A-p-v-B and A-u-q-B. As the plot indicates, there is a drop in the travel time just around 5 pm. During that brief interval, those departing from A after 5 pm can reach B earlier than those departing from A before 5 pm, by Braess' paradox. Hence, e is not FIFO, as per Equation 2.1.

In the TDSP setting, the parameter x denotes time, and the weight $w_e(x)$ of an edge $e = (A,B)$ denotes the arrival time at B if the departure time from A is x. If there is another edge $e' = (B,C)$ connected to B, then the arrival time at C along the path (A,B,C) is $w_{e'}(w_e(x))$, and so on. Thus, the cost of an s-t path is the arrival time at t as a function of the departure time from s. We say that an edge e is FIFO if its weight is a monotonically increasing function, i.e.,

$$x_1 \leq x_2 \iff w_e(x_1) \leq w_e(x_2) \qquad \forall x_1, x_2. \tag{2.1}$$

The study of TDSPs can be traced back to the work of Cooke and Halsey [1966]. Dreyfus [1969] gave a polynomial time algorithm when the edges were FIFO and the queries were made in discrete time steps. These results were extended to non-FIFO networks by Orda and Rom [1990], and generalized further by Ziliaskopoulos and Mahmassani [1993].

Dean [2004] summarised all known research on FIFO networks with linear edge weights. Dehne et al. [2012] presented an algorithm for TDSPs in this setting whose running time was at most the table size of the PGPP instance. Soon thereafter, Foschini et al. [2014] showed that the table size is at most $n^{O(\log n)}$, and that this is optimal, conclusively solving the problem for FIFO networks. We show that their bounds also hold for non-FIFO networks.

Some other closely related lines of work which might be of interest to the reader are Marcucci et al. [2021], Ben-Nun et al. [2020], Brunelli et al. [2021], Ruß et al. [2021], Wang et al. [2019] and Delling [2018].

Example: Braess' Paradox. The FIFO assumption makes sense because it seems that leaving from a source at a later time might not help one reach their destination quicker. However, somewhat counter-intuitively, Braess [1968] observed that this need not always the case (Figure 2 shows an example of Braess' paradox). Steinberg and Zangwill [1983] showed that Braess' paradox can occur with a high probability. Rapoport et al. [2009] backed their claim with empirical evidence. In fact, there are real-world instances where shutting down a road led to a decrease in the overall traffic congestion. Two examples are Stuttgart (Murchland [1970]) and Seoul ([Easley and Kleinberg, 2010, Page 71]).

Applications to Finance. Financial domain problems have been modelled as graph problems before (Dopfer and Potts [2014], Koenig and Battiston [2009], Eboli [2013], Bates et al. [2014], Attia [2019]). We model the currency arbitrage problem (Ross [1977], Shleifer and Vishny [1997], Delbaen and Schachermayer [2006]) as a GPP. In the currency arbitrage problem, we need to find an optimal conversion strategy from one currency to another via other currencies, assuming that all the conversion rates are known.

Example: Multi-currency Arbitrage. GPP can model generalized multi-currency arbitrage problems. In currency arbitrage, an entity can have money available in different currencies and engage in transactions (modelled by edges) which can change the entity's wealth composition in complex ways (Moosa [2003]). The transaction fees could have fixed as well as variable components, depending on the amount used. This can be modelled by affine linear transformations. Eventually the entity might liquidate all the money to a single currency, which can be modelled by the vector L in the GPP instance. The goal is to pick a sequence of transactions which maximizes the cash after liquidation. Hence, this problem naturally lends itself to a GPP formulation.

Example: Investment Planning. GPP can model investment planning by considering the nodes of the graph to be the state of the individual (which could be qualifications, contacts, experience, influence, etc). At each given state, the individual has a set of investment opportunities which are represented by directed edges. Every edge represents an investment opportunity, and the weight of the edge models the return as a function of the capital invested. Suppose an individual initially has y amount of money and makes

two investments in succession with returns $r_1(x), r_2(x)$, then the individual will end up with $r_2(r_1(y))$ amount of money. Though a generic investment plan could allow multiple partial investments, there are cases where this is not possible. For example, the full fees needs to be paid up front for attending a professional course or buying a house, which motivates restricting to investment plans given by paths. The vertices s, t denote the start and end of an investment period, and the optimal investment strategy is an s-t path which maximizes the composition of functions along the path.

3 ALGORITHM FOR SCALAR GPP WITH LINEAR WEIGHTS

In this section, we present our algorithm for scalar GPP with linear weight functions. Formally, we show the following.

Theorem 3.1. *There exists an algorithm that takes as input a scalar GPP instance (G,W,L,x_0) (where G has n vertices and $w_e(x) = a_e \cdot x + b_e$ for every edge e of G), and outputs an optimal s-t path in G in $O(n^3)$ running time.*

We use Algorithm 1 for solving GPP. Our algorithm is similar to the Bellman-Ford-Moore shortest path algorithm (Bellman [1958], Ford Jr [1956], Moore [1959]), where they keep track of *minimum* cost paths. The only subtlety in our case is that we need to keep track of both *minimum and maximum* cost paths with at most k edges from the start vertex s to every vertex v, as k varies from 1 to n. The variables $p_{\max}, p_{\min}$ act as parent pointers for the maximum cost path and the minimum cost path tree rooted at s. $r_{\max}, r_{\min}$ stores the cost of the maximum and minimum cost path. The running time of Algorithm 1 is clearly $O(n^3)$, the same as the running time of the Bellman-Ford-Moore algorithm. Its correctness follows from the following observation.

Observation 3.2. *Let $e = (u,v)$ be an edge of G, and a_e be the coefficient of x in w_e. That is, $w_e(x) = a_e x + b_e$.*

- *If e is the last edge on a* shortest *s-v path, then its s-u subpath is a shortest s-u path if a_e is positive, and a longest s-u path if a_e is negative.*
- *If e is the last edge on a* longest *s-v path, then its s-u subpath is a shortest s-u path if a_e is negative, and a longest s-u path if a_e is positive.*

Then, the argument is similar to the proof of the Bellman-Ford-Moore algorithm, using the optimal substructure property. Our algorithm can also handle time constraints on the edges which can come up in transport and finance problems. For example, each investment (modelled by an edge) could have a scalar value, which denotes the time taken for it to realize. The goal is to find an optimal sequence of investments (edges) from s to t, such that the sum of times along the path is at most some constant T. We can reduce such a problem to a GPP problem with a time constraint as follows.

Replace each edge e by a path of length t_e, where t_e is the time value associated with e. The weight function for the first edge is simply $w_e(x)$ and for the other $t_e - 1$ edges, it is the identity function. Then, Algorithm 1 can be modified so that the first for-loop stops at T instead of at $n-1$.

Algorithm 1: GPP with linear weight functions

For $v \in V \setminus \{s\}$, $r_{\max}(v) = -\infty, r_{\min}(v) = \infty$;
$r_{\max}(s) = r_{\min}(s) = x$;
for $k \in [1, n-1]$ **do**
  for $e = (u,v) \in E$ **do**
    if $a_e \geq 0$ **then**
      if $r_{max}(v) < w_e(r_{max}(u))$ **then**
        $r_{\max}(v) \leftarrow w_e(r_{\max}(u))$, $p_{\max}(v) \leftarrow u$;
      end
      if $r_{min}(v) > w_e(r_{min}(u))$ **then**
        $r_{\min}(v) \leftarrow w_e(r_{\min}(u))$, $p_{\min}(v) \leftarrow u$;
      end
    else
      if $r_{max}(v) < w_e(r_{min}(u))$ **then**
        $r_{\max}(v) \leftarrow w_e(r_{\min}(u))$, $p_{\max}(v) \leftarrow u$;
      end
      if $r_{min}(v) > w_e(r_{max}(u))$ **then**
        $r_{\min}(v) \leftarrow w_e(r_{\max}(u))$, $p_{\min}(v) \leftarrow u$;
      end
    end
  end
end
Output: The sequence $(t, p_{\max}(t), p_{\max}(p_{\max}(t)), \ldots, s)$ in reverse order is the optimal path at x with value $r_{\max}(t)$.

4 UPPER BOUND FOR SCALAR PGPP WITH LINEAR WEIGHTS

In this section, we study scalar PGPP (linear edge weights with $L = -1$), and show that the total number of different shortest s-t paths (for different values of $x_0 \in (-\infty, \infty)$) is at most quasi-polynomial in n. In PGPP (Problem 1.3), we are allowed to preprocess the graph. We compute all possible shortest s-t paths in the graph and store them in a table of quasi-polynomial size. More precisely, if (G,W,L) is a scalar GPP instance (where G has n vertices and $w_e(x) = a_e \cdot x + b_e$ for every edge e of G), then we show that the number of shortest s-t paths in G is at most $n^{O(\log n)}$. (For the example in Figure 1, this number is 4.) Since the entries of this table can be sorted by their corresponding x_0 values, a table lookup can be performed using a simple binary search in $\log(n^{O(\log n)}) = O((\log n)^2)$ time. Thus, a shortest s-t path for a queried x_0 can be retrieved in poly-logarithmic time.

In our proof, we will crucially use the fact that the edge weights of G are of the form $w_e(x) = a_e x + b_e$. Although our result holds in more generality, it is helpful and convenient to think of the edge weights from a TDSP perspective. That is, when travelling along an edge $e = (u,v)$ of G, if the start time at vertex u is x, then the arrival time at vertex v is $w_e(x)$.

As the edge weights are linear and the composition of linear functions is linear, the arrival time at t after starting from s at time x and travelling along a path P is a linear function of x, called the cost of the path and denoted by $\mathsf{cost}(P)(x)$. We show that the *piecewise linear lower envelope* (denoted by $\mathsf{cost}_G(x)$, indicated in pink in Figure 1) of the cost functions of the s-t paths of G has $n^{\log n+O(1)}$ pieces. Let $p(f)$ denote the number of pieces in a piecewise linear function f.

Theorem 4.1. *Let $\mathscr{P}$ be the set of s-t paths in G. Then, the cost function of the shortest s-t path, given by $\mathsf{cost}_G(x) = \min_{P:P\in\mathscr{P}} \mathsf{cost}(P)(x)$, is a piecewise linear function such that*

$$p(\mathsf{cost}_G(x)) \leq n^{\log n+O(1)}.$$

Before we can prove Theorem 4.1, we need some elementary facts about piecewise linear functions. Given a set of linear functions F, let $F_\downarrow$ and $F_\uparrow$ be defined as follows.

$$F_\downarrow(x) = \min_{f:f\in F} f(x) \qquad\qquad F_\uparrow(x) = \max_{f:f\in F} f(x)$$

In other words, $F_\downarrow$ and $F_\uparrow$ are the piecewise linear lower and upper envelopes of F, respectively.

Fact 4.2 (Some properties of piecewise linear functions)**.**

(i) If F is a set of linear functions, then $F_\downarrow$ is a piecewise linear concave function and $F_\uparrow$ is a piecewise linear convex function.

(ii) If $f(x)$ and $g(x)$ are piecewise linear concave *functions, then $h(x) = \min\{f(x), g(x)\}$ is a piecewise linear concave function such that $p(h) \leq p(f) + p(g)$.*

(iii) If $f(x)$ and $g(x)$ are piecewise linear functions and $g(x)$ is monotone*, then $h(x) = f(g(x))$ is a piecewise linear function such that $p(h) \leq p(f) + p(g)$.*

Proof. These facts and their proofs are inspired by (and similar to) some of the observations made by [Foschini et al., 2014, Lemma 2.1, Lemma 2.2].

(i) Linear functions are concave (convex), and the pointwise minimum (maximum) of concave (convex) functions is concave (convex).

(ii) Each piece of h corresponds to a unique piece of f or g. Since h is concave, different pieces of h have different slopes, corresponding to different pieces of f or g.

(iii) A break point is a point where two adjoining pieces of a piecewise linear function meet. Note that each break point of h can be mapped back to a break point of f or a break point of g. As g is monotone, different break points of h map to different break points of g. □

We now prove the following key lemma.

Lemma 4.3. *Let F and G be two sets of linear functions, and let $H = \{f \circ g \mid f \in F, g \in G\}$. Then*

$$H_\downarrow(x) = \min\{F_\downarrow(G_\downarrow(x)), F_\downarrow(G_\uparrow(x))\}; \tag{4.4}$$
$$H_\uparrow(x) = \max\{F_\uparrow(G_\downarrow(x)), F_\uparrow(G_\uparrow(x))\}; \tag{4.5}$$
$$p(H_\downarrow) \leq 4p(F_\downarrow) + 2p(G_\downarrow) + 2p(G_\uparrow); \tag{4.6}$$
$$p(H_\uparrow) \leq 4p(F_\uparrow) + 2p(G_\downarrow) + 2p(G_\uparrow). \tag{4.7}$$

Proof. We will first show Equation 4.4. Since F is the set of outer functions, it is easy to see that

$$H_\downarrow(x) = \min_{g:g\in G} F_\downarrow(g(x)). \tag{4.8}$$

To get Equation 4.4 from Equation 4.8, we need to show that the inner function g that minimizes $H_\downarrow$ is always either $G_\downarrow$ or $G_\uparrow$. Fix an $x_0 \in \mathbb{R}$. We will see which $g \in G$ minimizes $F_\downarrow(g(x_0))$. Note that for every $g \in G$, we have $G_\downarrow(x_0) \leq g(x_0) \leq G_\uparrow(x_0)$. Thus, the input to $F_\downarrow$ is restricted to the interval $[G_\downarrow(x_0), G_\uparrow(x_0)]$. Since $F_\downarrow$ is a *concave* function (Fact 4.2 (i)), it achieves its minimum at either $G_\downarrow(x_0)$ or at $G_\uparrow(x_0)$ within this interval. This shows Equation 4.4.

We will now show Equation 4.6 using Equation 4.4. Since $G_\downarrow$ is a concave function, it has two parts: a first part where it monotonically increases and a second part where it monotonically decreases. In each part, the number of pieces in $F_\downarrow(G_\downarrow(x))$ is at most $p(F_\downarrow) + p(G_\downarrow)$ (Fact 4.2 (iii)), which gives a total of $2(p(F_\downarrow) + p(G_\downarrow))$. Similarly, since $G_\uparrow$ is a convex function, it has two parts: a first part where it monotonically decreases and a second part where it monotonically increases. In each part, the number of pieces in $F_\downarrow(G_\uparrow(x))$ is at most $p(F_\downarrow) + p(G_\uparrow)$ (Fact 4.2 (iii)), which gives a total of $2(p(F_\downarrow) + p(G_\uparrow))$. Combining these using Equation 4.4 and Fact 4.2 (ii), we obtain

$$\begin{aligned} p(H_\downarrow) &\leq 2(p(F_\downarrow) + p(G_\downarrow)) + 2(p(F_\downarrow) + p(G_\uparrow)) \\ &= 4p(F_\downarrow) + 2p(G_\downarrow) + 2p(G_\uparrow). \end{aligned}$$

We skip the proof of Equation 4.5 and its usage to prove Equation 4.7 because it is along similar lines. □

Using this lemma, we complete the proof of Theorem 4.1.

Proof of Theorem 4.1. It suffices to prove the theorem for all positive integers n that are powers of 2. Let a, b, v be three vertices of G and let k be a power of 2. Let $\mathscr{P}_v(a,b,k)$ be the set of a-b paths P that pass through v such that the a-v subpath and the v-b subpath of P have at most $k/2$ edges each (k is even number since it is a power of 2). Let $\mathscr{P}(a,b,k)$ be the set of a-b paths that have at most k edges. Note that $\mathscr{P}(a,b,k) = \bigcup_{v\in V} \mathscr{P}_v(a,b,k)$. Let $f_v(a,b,k)$ be the number of pieces in the piecewise linear lower envelope or the piecewise linear upper envelope of $\mathscr{P}_v(a,b,k)$, whichever is larger. Similarly, $f(a,b,k)$ is the number of pieces in the piecewise linear lower envelope or the piecewise linear upper envelope of $\mathscr{P}(a,b,k)$, whichever is larger. Note that

every path that features in the lower envelope of $\mathscr{P}(a,b,k)$ also features in the lower envelope of $\mathscr{P}_v(a,b,k)$, for some v. Thus,

$$f(a,b,k) \leq \sum_{v \in V} f_v(a,b,k). \tag{4.9}$$

Since G has n vertices, $\mathscr{P}(a,b,n)$ is simply the set of all a-b paths. And since $p(\mathsf{cost}_G(x))$ is the number of pieces in the piecewise linear lower envelope of these paths, $p(\mathsf{cost}_G(x)) \leq f(a,b,n)$. Thus it suffices to show that $f(a,b,n) \leq n^{\log n + O(1)}$. We will show, by induction on k, that $f(a,b,k) \leq (8n)^{\log k}$. The base case, $f(a,b,1) \leq 1$, is trivial. Now, let $k > 1$ be a power of 2. We will now show the following recurrence.

$$f_v(a,b,k) \leq 4\left(f(a,v,k/2) + f(v,b,k/2)\right) \tag{4.10}$$

Fix a vertex $v \in V$. By induction, $f(a,v,k/2) \leq (8n)^{\log(k/2)}$ and $f(v,b,k/2) \leq (8n)^{\log(k/2)}$. Note that for every path $P \in \mathscr{P}_v(a,b,k)$, we have $\mathsf{cost}(P)(x) = \mathsf{cost}(P_2)(\mathsf{cost}(P_1)(x))$, where $P_1 \in \mathscr{P}(a,v,k/2)$ and $P_2 \in \mathscr{P}(v,b,k/2)$. Thus we can invoke Lemma 4.3 with F, G and H as the set of linear (path cost) functions corresponding to the paths $\mathscr{P}(v,b,k/2)$, $\mathscr{P}(a,v,k/2)$ and $\mathscr{P}_v(a,b,k)$, respectively. Applying Equation 4.6 and Equation 4.7, we get

$$f_v(a,b,k) \leq 4f(v,b,k/2) + 2f(a,v,k/2) + 2f(a,v,k/2),$$

which simplifies to Equation 4.10. Substituting Equation 4.10 in Equation 4.9, and using the fact that $|V| = n$, we get the following.

$$\begin{aligned} f(a,b,k) &\leq 4\sum_{v \in V} \left(f(a,v,k/2) + f(v,b,k/2)\right) \\ &\leq 4n\left((8n)^{\log(k/2)} + (8n)^{\log(k/2)}\right) \\ &= (4n)\cdot 2 \cdot (8n)^{\log(k/2)} = (8n)^{\log k}. \end{aligned}$$

Thus, $f(a,b,n) \leq (8n)^{\log n} = n^{\log n + 3}$. □

5 HARDNESS OF SCALAR GPP WITH NON-LINEAR WEIGHTS

In this section, we show that it is NP-hard to approximate scalar GPP, even if one of the edge weights is made piecewise linear while keeping all other edge weights linear.

Theorem 5.1. *Let (G,W,L,x_0) be a GPP instance with a special edge e^*, where G has n vertices and $w_e(x) = a_e x + b_e$ for every edge $e \in E(G) \setminus \{e^*\}$, and $w_{e^*}(x)$ is piecewise linear with 2 pieces. Then it is NP-hard to find an s-t path whose cost approximates the cost of the optimal s-t path in G to within a constant, both additively and multiplicatively.*

Note that Theorem 5.1 implies that Problem 1.3 with piecewise linear edge weights is NP-hard.

Proof of Theorem 5.1. We reduce from SET PARTITION, a well-known NP-hard problem [Garey and Johnson, 1979, Page 226][1]. The SET PARTITION problem asks if a given set of n integers $A = \{a_0, \ldots, a_{n-1}\}$ can be partitioned into two subsets A_0 and A_1 such that they have the same sum.

We now explain our reduction. Let ε be the multiplicative approximation factor and δ be the additive approximation term. Given a SET PARTITION instance $A = \{a_1, \ldots, a_n\}$, we multiply all its elements by the integer $\lceil \delta + 1 \rceil$. Note that this new instance can be partitioned into two subsets having the same sum if and only if the original instance can. Furthermore, after this modification, no subset of A has sum in the range $[-\delta, \delta]$, unless that sum is zero. Next, we define a graph instantiated by the SET PARTITION instance.

Definition 5.2. *G_n is a directed, acyclic graph on $n+1$ vertices. The vertex set of G_n is $\{v_0, v_1, \ldots, v_n\}$. For every $i \in \{0,1,\ldots,n-1\}$, there are two edges from v_i to v_{i+1} labelled by f_0 and f_1. The start vertex s is v_0 and the last vertex t is v_n (see figure below).* ◊

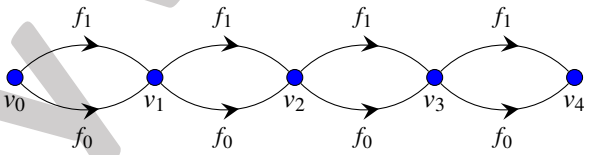


Figure 3: The graph G_n for $n = 4$.

Definition 5.3. *Each path of G_n can be denoted by a string in $\{0,1\}^n$, from left to right. For instance, if $\sigma = (0101)$, then the cost function $f_\sigma(x)$ of the path P_σ is given by*

$$f_\sigma(x) = f_{(0101)}(x) = f_1(f_0(f_1(f_0(x)))).$$

Note that the innermost function corresponds to the first edge of the path P_σ, and the outermost to the last (Figure 3). ◊

Consider the graph G_{n+1}. For each $i \in \{0,1,\ldots,n-1\}$ and each edge (v_i, v_{i+1}), the edge labelled by f_0 has weight $x + a_i$ and the edge labelled by f_1 has weight $x - a_i$. Both edges from v_n to v_{n+1} have weight $|x|$ (and can be replaced by a single edge e^*). Let $\mathscr{A}$ be an algorithm which solves Problem 1.3. We will provide G_{n+1} and $x_0 = 0$ as inputs to $\mathscr{A}$, and show that A can be partitioned into two subsets having the same sum if and only if $\mathscr{A}$ returns a path of cost 0.

Let $\sigma = (\sigma_0 \sigma_1 \cdots \sigma_{n-1}) \in \{0,1\}^n$. Let A_1 be the subset of A with characteristic vector σ, and let $A_0 = A \setminus A_1$. The cost of the path P_σ (Definition 5.3) from v_0 to v_n is

$$\mathsf{cost}(P_\sigma)(x) = x + \sum_{i=0}^{n-1} (-1)^{\sigma_i} a_i = x + \sum_{a_i \in A_0} a_i - \sum_{a_i \in A_1} a_i.$$

[1] A similar reduction can be found in [Nikolova et al., 2006, Theorem 3].

Now if we set the start time from vertex v_0 as $x = x_0 = 0$, then we obtain the following.

$$\mathsf{cost}(P_\sigma)(0) = 0 \implies \sum_{a_i \in A_0} a_i = \sum_{a_i \in A_1} a_i.$$

Let OPT be a shortest path in G_{n+1} and Q be the path returned by $\mathscr{A}$ at start time $x = x_0 = 0$. The last edge from v_n to v_{n+1} (whose weight is $|x|$) ensures that $\mathsf{OPT} \geq 0$. So, if $\mathsf{OPT} = 0$, then $\mathsf{cost}(Q)(0) \leq \varepsilon \cdot 0 + \delta = \delta$.

Since every path of non-zero cost in G_{n+1} has cost more than δ, $\mathsf{cost}(Q)(0) = 0$ if $\mathsf{OPT} = 0$. Further, if $\mathsf{OPT} > 0$, then $\mathsf{OPT} \geq \lceil \delta + 1 \rceil$, and so $\mathscr{A}$ returns a path of cost more than δ. Thus, A can be partitioned into two subsets having the same sum if and only if $\mathscr{A}$ returns a path of cost 0. □

Remark. Our reduction also works if we change the weight of the last edge from $|x|$ to x^2, implying that scalar GPP with polynomial functions is NP-hard, even if one of the edge weights is quadratic and all other edge weights are linear.

6 LOWER BOUND FOR SCALAR PGPP WITH NON-LINEAR WEIGHTS

In this section, we show that for the graph G_n defined in the previous section (Definition 5.2) with a suitable choice of the weight functions f_0 and f_1, the table size for PGPP (Problem 1.3) can be exponential in n. Note that G_n has exactly 2^n paths from s to t. We will show that each of these paths is a shortest s-t path, for some value of x. Thus, there is a scalar GPP instance (G, W, L) for which the table size is $2^{\Omega(n)}$, needing $\log(2^{\Omega(n)}) = \Omega(n)$ time for a table lookup.

Our proof is by induction on n. We define the functions f_0 and f_1 in such a way that their behaviour within the interval $[0,1]$ has some very special properties, stated in Lemma 6.1. This enables us to show that the number of times the compositions of these functions achieve their minimum within the interval $[0,1]$ doubles every time n increases by one.

We need some notation before we can proceed. For a function $f : \mathbb{R} \to \mathbb{R}$, and a subset $A \subseteq \mathbb{R}$, if $B \supseteq f(A)$, then we denote by $f|_A : A \to B$, the function defined by $f|_A(x) = f(x)$ for every $x \in A$, also known as the restriction of f to A.

Lemma 6.1. *Suppose $f_0, f_1 : \mathbb{R} \to \mathbb{R}$ are functions such that $f_0|_{[0,1/3]} : [0,1/3] \to [0,1]$ and $f_1|_{[2/3,1]} : [2/3,1] \to [0,1]$ are bijective. Further, suppose $|f_0(x)| \geq 1$ for every $x \in (-\infty, 0] \cup [2/3, \infty)$; and $|f_1(x)| \geq 1$ for every $x \in (-\infty, 1/3] \cup [1, \infty)$. Then, for every $n \geq 1$, there is a function $\alpha_n : \{0,1\}^n \to (0,1)$ such that*

(i) $\alpha_n(\sigma) \in [0,1/3]$ if $\sigma_1 = 0$;
(ii) $\alpha_n(\sigma) \in [2/3,1]$ if $\sigma_1 = 1$;
(iii) For every $\sigma, \tau \in \{0,1\}^n$, $\alpha_n(\sigma) = \alpha_n(\tau) \iff \sigma = \tau$;
(iv) For every $\sigma, \tau \in \{0,1\}^n$, $f_\sigma(\alpha_n(\tau)) = 0 \iff \sigma = \tau$.

Proof. As stated earlier, we prove this lemma by induction on n. For the **base case** ($n = 1$), we define

$$\alpha_1(0) = (f_0|_{[0,1/3]})^{-1}(0) \quad \& \quad \alpha_1(1) = (f_1|_{[2/3,1]})^{-1}(0).$$

First we check if α_1 is well-defined and its range lies in $(0,1)$. To see that $\alpha_1(0)$ and $\alpha_1(1)$ are well-defined, note that the inverses of the functions $f_0|_{[0,1/3]}$ and $f_0|_{[0,1/3]}$ are well-defined because they are bijective. To see that the range of α_1 lies in $(0,1)$, note that for $x \in \{0,1\}$, we have $|f_0(x)| \geq 1$ and $|f_1(x)| \geq 1$, implying that they are both non-zero. Thus, $0 < \alpha_1(x) < 1$.

We now show that α_n satisfies (i), (ii), (iii), (iv). Since $f_0|_{[0,1/3]}$ and $f_1|_{[2/3,1]}$ are bijective, $\alpha_1(0) \in [0,1/3]$ and $\alpha_1(1) \in [2/3,1]$. Thus, α_1 satisfies (i), (ii). Since these intervals are disjoint, α_1 satisfies (iii). Finally, note that $f_0(\alpha_1(0)) = 0 = f_1(\alpha_1(1))$. Also, since $|f_0(x)| \geq 1$ for every $x \in [2/3,1]$ and $|f_1(x)| \geq 1$ for every $x \in [0,1/3]$, both $f_0(\alpha_1(1))$ and $f_1(\alpha_1(0))$ are non-zero. Thus, α_1 satisfies (iv). This proves the base case.

Induction step ($n > 1$)**:** Assume that $\alpha_{n-1} : \{0,1\}^{n-1} \to (0,1)$ has been defined, and that it satisfies (i), (ii), (iii), (iv). We now define $\alpha_n : \{0,1\}^n \to (0,1)$. Let $\sigma \in \{0,1\}^n$ be such that $\sigma = \sigma_1 \sigma'$, where $\sigma_1 \in \{0,1\}$ and $\sigma' = \sigma_2 \cdots \sigma_n \in \{0,1\}^{n-1}$. We define $\alpha_n(\sigma)$ as follows.

$$\alpha_n(\sigma) = \begin{cases} (f_0|_{[0,1/3]})^{-1}(\alpha_{n-1}(\sigma')) & \text{if } \sigma_1 = 0 \\ (f_1|_{[2/3,1]})^{-1}(\alpha_{n-1}(\sigma')) & \text{if } \sigma_1 = 1 \end{cases}$$

More concisely,

$$\alpha_n(\sigma) = (f_{\sigma_1}|_A)^{-1}(\alpha_{n-1}(\sigma')), \tag{6.2}$$

where $A = [0,1/3]$ when $\sigma_1 = 0$ and $A = [2/3,1]$ when $\sigma_1 = 1$, Note that α_n is well-defined and its range lies in $(0,1)$ for the same reasons as explained in the base case. We will now show that α_n satisfies (i), (ii), (iii), (iv).

(i), (ii): By definition, $(f_0|_{[0,1/3]})^{-1} : [0,1] \to [0,1/3]$ and $(f_1|_{[2/3,1]})^{-1} : [0,1] \to [2/3,1]$. Thus, α_n satisfies (i), (ii).

(iii): Suppose $\sigma = \sigma_1 \sigma' \in \{0,1\}^n$ and $\tau = \tau_1 \tau' \in \{0,1\}^n$. Clearly if $\sigma = \tau$, then $\alpha_n(\sigma) = \alpha_n(\tau)$. This shows the $\Leftarrow$ direction. For the $\Rightarrow$ direction, suppose $\alpha_n(\sigma) = \alpha_n(\tau)$. Then the only option is $\sigma_1 = \tau_1$, since otherwise one of $\alpha_n(\sigma), \alpha_n(\tau)$ would lie in the interval $[0,1/3]$ and the other would lie in the interval $[2/3,1]$. Thus,

$$(f_{\sigma_1}|_A)^{-1}(\alpha_{n-1}(\sigma')) = (f_{\sigma_1}|_A)^{-1}(\alpha_{n-1}(\tau')),$$

where $A = [0,1/3]$ when $\sigma_1 = 0$ and $A = [2/3,1]$ when $\sigma_1 = 1$, Since $(f_{\sigma_1}|_A)$ is bijective, this means that $\alpha_{n-1}(\sigma') = \alpha_{n-1}(\tau')$. Using part (iii) of the induction hypothesis, this implies that $\sigma' = \tau'$. Thus, α_n satisfies (iii).

(iv): Suppose $\sigma = \sigma_1 \sigma' \in \{0,1\}^n$ and $\tau = \tau_1 \tau' \in \{0,1\}^n$.

Let us show the $\Leftarrow$ direction first. If $\sigma = \tau$, then

$$\begin{aligned} f_\sigma(\alpha_n(\tau)) &= f_\sigma(\alpha_n(\sigma)) \\ &= f_{\sigma'}(f_{\sigma_1}(\alpha_n(\sigma))) \qquad (\text{since } \sigma = \sigma_1\sigma') \\ &= f_{\sigma'}(f_{\sigma_1}((f_{\sigma_1}|_A)^{-1}(\alpha_{n-1}(\sigma')))) \quad (\text{Equation 6.2}) \\ &= f_{\sigma'}((f_{\sigma_1} \circ (f_{\sigma_1}|_A)^{-1})(\alpha_{n-1}(\sigma'))) \\ &\qquad (\text{function composition is associative}) \\ &= f_{\sigma'}(\alpha_{n-1}(\sigma')). \end{aligned}$$

Using part (iv) of the induction hypothesis, we get $f_{\sigma'}(\alpha_{n-1}(\sigma')) = 0$, which implies that $f_\sigma(\alpha_n(\tau)) = 0$. This shows the $\Leftarrow$ direction.

For the $\Rightarrow$ direction, suppose $f_\sigma(\alpha_n(\tau)) = 0$. We have two cases: $\sigma_1 = \tau_1$ and $\sigma_1 \neq \tau_1$. We will show that $\sigma = \tau$ in the first case, and that the second case is impossible. If $\sigma_1 = \tau_1$,

$$\begin{aligned} 0 = f_\sigma(\alpha_n(\tau)) &= f_{\sigma'}(f_{\sigma_1}((f_{\tau_1}|_A)^{-1}(\alpha_{n-1}(\tau')))) \\ &= f_{\sigma'}(f_{\sigma_1}((f_{\sigma_1}|_A)^{-1}(\alpha_{n-1}(\tau')))) = f_{\sigma'}(\alpha_{n-1}(\tau')), \end{aligned}$$

Using part (iv) of the induction hypothesis, $f_{\sigma'}(\alpha_{n-1}(\tau')) = 0 \Rightarrow \sigma' = \tau'$, and thus $\sigma = \tau$. This handles the case $\sigma_1 = \tau_1$. We will now show by contradiction that the case $\sigma_1 \neq \tau_1$ is impossible.

Suppose $\sigma_1 \neq \tau_1$. Let $\sigma_1 = 0$ and $\tau_1 = 1$ (the proof for $\sigma_1 = 1$ and $\tau_1 = 0$ is similar). Using the induction hypothesis, $(f_{\tau_1}|_{[2/3,1]})^{-1}(\alpha_{n-1}(\tau')) \in [2/3, 1]$. Since $|f_0(x)| \geq 1$ for every $x \in (-\infty, 0] \cup [2/3, \infty)$, this means that $\left|f_{\sigma_1}((f_{\tau_1}|_{[2/3,1]})^{-1}(\alpha_{n-1}(\tau')))\right| \geq 1$. Also note that if $|x| \geq 1$, then both $|f_0(x)| \geq 1$ and $|f_1(x)| \geq 1$. By repeatedly applying this fact, it is easy to see that

$$\begin{aligned} &|f_\sigma(\alpha_n(\tau))| \\ &= \left|f_{\sigma_n}(\cdots(f_{\sigma_1}((f_{\tau_1}|_{[2/3,1]})^{-1}(\alpha_{n-1}(\tau')))\cdots)\right| \geq 1. \end{aligned}$$

We started with $f_\sigma(\alpha_n(\tau)) = 0$ and obtained $|f_\sigma(\alpha_n(\tau))| \geq 1$, which is clearly a contradiction. This completes the proof of the $\Rightarrow$ direction, and thus α_n satisfies (iv). □

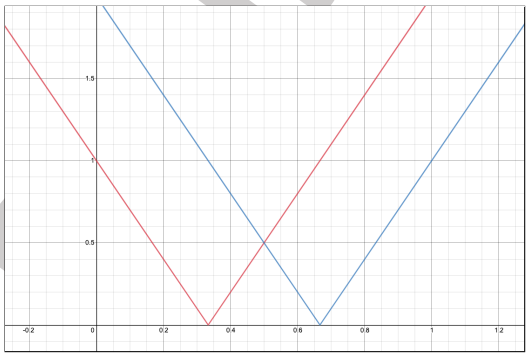


Figure 4: The functions f_0 and f_1.

Theorem 6.3. *Consider the graph G_n. Define piecewise linear functions $f_0, f_1 : \mathbb{R} \to \mathbb{R}$ as follows (see figure above).*

$$f_0(x) = \begin{cases} 1 - 3x, & \text{if } x \leq 1/3 \\ 3x - 1, & \text{if } x \geq 1/3 \end{cases} \qquad f_1(x) = \begin{cases} 2 - 3x, & \text{if } x \leq 2/3 \\ 3x - 2, & \text{if } x \geq 2/3 \end{cases}$$

For every $n \geq 1$ and $\sigma \in \{0,1\}^n$, the cost function f_σ of the path P_σ is a unique piece in the lower envelope formed by the cost functions $\{f_\sigma\}_{\sigma \in \{0,1\}^n}$. Thus, the piecewise linear shortest path cost function has 2^n pieces.

Proof of Theorem 6.3. It is easy to check that f_0 and f_1 possess the conditions needed to invoke Lemma 6.1. Thus for every $n \geq 1$, there exists a function α_n which satisfies properties (i), (ii) and (iii) of Lemma 6.1.

Let n be a positive integer. Consider the graph G_n (Definition 5.2). Each path of G_n is indexed by a binary string $\sigma \in \{0,1\}^n$ and has cost function f_σ (Definition 5.3). Note that $f_0(x) \geq 0$, $f_1(x) \geq 0$ for all $x \in \mathbb{R}$. Thus $f_\sigma(x) \geq 0$ for all $\sigma \in \{0,1\}^n$, $x \in \mathbb{R}$.

Let $\sigma \in \{0,1\}^n$. Using property (iii), $f_\sigma(\alpha(\sigma)) = 0$, and $f_\tau(\alpha(\sigma)) > 0$ for every $\sigma \neq \tau \in \{0,1\}^n$. Thus, the cost function f_σ of the path P_σ is a unique piece (which includes the point $\alpha_n(\sigma)$) in the lower envelope formed by the cost functions $\{f_\sigma\}_{\sigma \in \{0,1\}^n}$. □

Remark. The proof of Theorem 6.3 works for a quadratic choice of the functions f_0 and f_1 as well. However, then the degree of the composed functions blows up exponentially, thereby making their bit complexity prohibitively large.

7 HARDNESS OF NON-SCALAR GPP WITH LINEAR WEIGHTS

In this section, we show that non-scalar GPP is NP-hard.

Theorem 7.1. *Let $(G, W, L, \mathbf{x}_0)$ be a GPP instance, where G has n vertices and each edge e of G is labelled by a two dimensional vector $\mathbf{w}_e(x)$. The vertices s,t are labelled by two dimensional vectors $\mathbf{x}_0, \mathbf{t}_0$, respectively. Then it is NP-hard to compute an optimal s-t path in G.*

Note that Theorem 7.1 implies that Problem 1.2 with parameter $k = 2$ is NP-hard.

Proof of Theorem 7.1. We reduce from PRODUCT PARTITION problem, a well-known NP-hard problem (Ng et al. [2010]). The problem is similar to the set partition problem, except that *products* of the elements are taken instead of their *sums*. Formally, the problem asks if a given set of n positive integers $A = \{a_1, \ldots, a_n\}$ can it be partitioned into two subsets A_0 and A_1 such that their product is the same.

We now explain our reduction. Given a PRODUCT PARTITION instance $A = \{a_1, \ldots, a_n\}$, consider the graph G_{n+1}

(Definition 5.2). For every $i \in \{0,1,\ldots,n-1\}$, there are two edges from v_i to v_{i+1} labelled by matrices

$$\begin{bmatrix} a_i & 0 \\ 0 & a_i^{-1} \end{bmatrix} \text{ and } \begin{bmatrix} a_i^{-1} & 0 \\ 0 & a_i \end{bmatrix}.$$

We label s by the vector $\mathbf{x}_0 = [1,1]^{\mathrm{T}}$ and t by $\mathbf{t}_0 = [-1,-1]^{\mathrm{T}}$. Let $\mathscr{A}$ be an algorithm which solves Problem 1.2 with parameter $k = 2$. We will provide G_{n+1} as input to $\mathscr{A}$, and show that A can be partitioned into two subsets having the same sum if and only if $\mathscr{A}$ returns a path of cost -2.

Let $\sigma = (\sigma_1 \cdots \sigma_n) \in \{1,-1\}^n$. Let A_1 be the subset of A with characteristic vector σ, and let $A_0 = A \setminus A_1$. The cost of the path P_σ (Definition 5.3) from v_0 to v_n is

$$\mathsf{cost}(P_\sigma) = \begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \prod_{i=1}^n a_i^{\sigma_i} & 0 \\ 0 & \prod_{i=1}^n a_i^{-1\cdot\sigma_i} \end{bmatrix} \cdot \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

Evaluating this, we obtain $\mathsf{cost}(P_\sigma) = -(a + a^{-1})$, where $a = \prod_{i=1}^n a_i^{\sigma_i} = \prod_{a_i \in A_0} a_i \cdot \prod_{a_i \in A_1} a_i^{-1}$. Further, $a = 1 \iff \prod_{a_i \in A_0} a_i = \prod_{a_i \in A_1} a_i$.

Since $a > 0$, the AM-GM inequality implies that $a + a^{-1} > 2$ for every $a \neq 1$. Therefore, $-(a+a^{-1}) < -2$ for every $a \neq 1$, and so A can be partitioned into two subsets whose product is the same if and only if $\mathscr{A}$ returns a path of cost -2. □

8 CONCLUSION & DISCUSSION

We study Generalized Path Problems on graphs with parametric weights. We show that the problem is efficiently solvable when the weight functions are linear, but becomes intractable in general when they are piecewise linear.

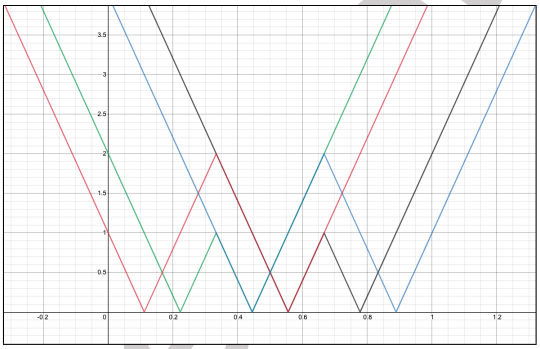
We assume that the weight functions are deterministic and fully known in advance. Modelling probabilistic and partially known weight functions and proposing algorithms for them is a direction for future work. Furthermore, we have assumed that only one edge can be taken at a time, resulting in an optimization over *paths*. This requirement could be relaxed to study *flows* on graphs with parametric weights. Though there is some literature on such models in route planning algorithms (Liebig et al. [2017]), results with rigorous guarantees such as the ones we have presented are challenging to obtain. In such cases, heuristic algorithms with empirical evaluation measures might be worth exploring.

Acknowledgements

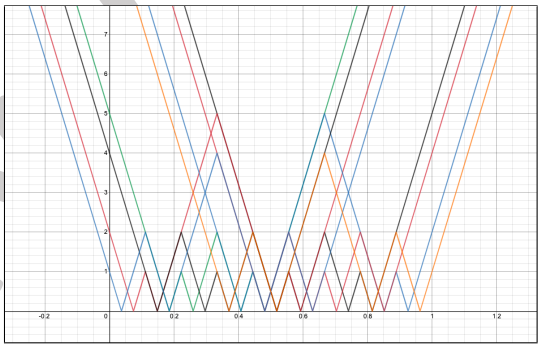
This work was done when the first author was a postdoctoral researcher at Technion, Israel. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 682203-ERC-[Inf-Speed-Tradeoff]. The authors from TIFR acknowledge support of the Department of Atomic Energy, Government of India, under project number RTI4001.

PLOTS

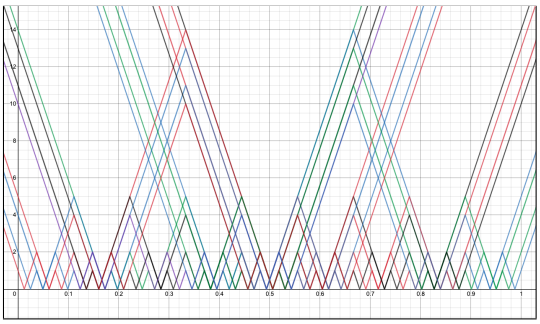
In this section, we exhibit the plots of all the s-t paths (Definition 5.3) in the graphs G_n (Definition 5.2) for the weight functions f_0, f_1 (Theorem 6.3), for some small values of n.



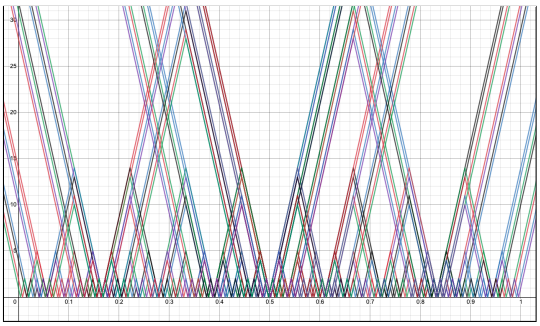
$n = 2$



$n = 3$



$n = 4$



$n = 5$

References

Joseph Attia. The applications of graph theory to investing. *arXiv preprint arXiv:1902.00786*, 2019.

Samuel Bates, Valérie Angeon, and Ahmed Ainouche. The pentagon of vulnerability and resilience: A methodological proposal in development economics by using graph theory. *Economic Modelling*, 42:445–453, 2014.

Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

Tal Ben-Nun, Lukas Gianinazzi, Torsten Hoefler, and Yishai Oltchik. Parametric graph templates: Properties and algorithms. *CoRR*, abs/2011.07001, 2020. URL https://arxiv.org/abs/2011.07001.

Dietrich Braess. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, 12(1):258–268, 1968.

Filippo Brunelli, Pierluigi Crescenzi, and Laurent Viennot. On computing pareto optimal paths in weighted time-dependent networks. *Inf. Process. Lett.*, 168:106086, 2021. doi: 10.1016/j.ipl.2020.106086. URL https://doi.org/10.1016/j.ipl.2020.106086.

Patricia J. Carstensen. *The complexity of some problems in parametric linear and combinatorial programming*. PhD thesis, University of Michigan, 1983.

Kenneth L Cooke and Eric Halsey. The shortest route through a network with time-dependent internodal transit times. *Journal of mathematical analysis and applications*, 14(3):493–498, 1966.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN 0262033844.

Brian C Dean. Shortest paths in fifo time-dependent networks: Theory and algorithms. *Rapport technique, Massachusetts Institute of Technology*, page 13, 2004.

Frank Dehne, Masoud T Omran, and Jörg-Rüdiger Sack. Shortest paths in time-dependent FIFO networks. *Algorithmica*, 62(1-2):416–435, 2012.

Freddy Delbaen and Walter Schachermayer. *The mathematics of arbitrage*. Springer Science & Business Media, 2006.

Daniel Delling. Route planning in transportation networks: from research to practice. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 2–2, 2018.

Kurt Dopfer and Jason Potts. *The new evolutionary economics*. Edward Elgar Publishing, 2014.

Stuart E Dreyfus. An appraisal of some shortest-path algorithms. *Operations research*, 17(3):395–412, 1969.

David Easley and Jon Kleinberg. *Networks, crowds, and markets*, volume 8. Cambridge university press Cambridge, 2010.

Mario Eboli. *Financial Applications of Flow Network Theory*, pages 21–29. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-32903-6. doi: 10.1007/978-3-642-32903-6_3. URL https://doi.org/10.1007/978-3-642-32903-6_3.

Lester R Ford Jr. Network flow theory. Technical report, Rand Corp Santa Monica, CA, 1956.

Luca Foschini, John Hershberger, and Subhash Suri. On the complexity of time-dependent shortest paths. *Algorithmica*, 68(4):1075–1097, 2014. doi: 10.1007/s00453-012-9714-7. URL https://doi.org/10.1007/s00453-012-9714-7.

Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

Gaurav Hajela and Manish Pandey. Parallel implementations for solving shortest path problem using bellman-ford. *International Journal of Computer Applications*, 95 (15):1–6, June 2014. Full text available.

Harald Hau. The exchange rate effect of multi-currency risk arbitrage. *Journal of International Money and Finance*, 47(C):304–331, 2014. doi: 10.1016/j.jimonfin.2014.0. URL https://ideas.repec.org/a/eee/jimfin/v47y2014icp304-331.html.

Sreeja Kamishetty, Soumya Vadlamannati, and Praveen Paruchuri. Towards a better management of urban traffic pollution using a pareto max flow approach. *Transportation Research Part D: Transport and Environment*, 79:102194, 2020. ISSN 1361-9209. doi: https://doi.org/10.1016/j.trd.2019.11.023. URL https://www.sciencedirect.com/science/article/pii/S1361920919303530.

Michael D Koenig and Stefano Battiston. From graph theory to models of economic networks. a tutorial. *Networks, topology and dynamics*, 613:23–63, 2009.

Thomas Liebig, Nico Piatkowski, Christian Bockermann, and Katharina Morik. Dynamic route planning with real-time traffic predictions. *Information Systems*, 64:258–265, 2017.

Tobia Marcucci, Jack Umenberger, Pablo A. Parrilo, and Russ Tedrake. Shortest paths in graphs of convex sets. *CoRR*, abs/2101.11565, 2021. URL https://arxiv.org/abs/2101.11565.

E. F. Moore. The shortest path through a maze. *Proc. Int. Symp. Switching Theory, 1959*, part II:285–292, 1959. URL https://ci.nii.ac.jp/naid/10015375086/en/.

Imad A. Moosa. *Two-Currency, Three-Currency and Multi-Currency Arbitrage*, pages 1–18. Palgrave Macmillan UK, London, 2003. ISBN 978-1-4039-4603-4. doi: 10.1057/9781403946034_1. URL https://doi.org/10.1057/9781403946034_1.

Ketan Mulmuley and Pradyut Shah. A lower bound for the shortest path problem. *J. Comput. Syst. Sci.*, 63(2):253–267, 2001. doi: 10.1006/jcss.2001.1766. URL https://doi.org/10.1006/jcss.2001.1766.

John D Murchland. Braess's paradox of traffic flow. *Transportation Research*, 4(4):391–394, 1970.

Chi To Ng, MS Barketau, TC Edwin Cheng, and Mikhail Y Kovalyov. "product partition" and related problems of scheduling and systems reliability: computational complexity and approximation. *European Journal of Operational Research*, 207(2):601–604, 2010.

Evdokia Nikolova, Matthew Brand, and David R Karger. Optimal route planning under uncertainty. In *ICAPS*, volume 6, pages 131–141, 2006.

Ariel Orda and Raphael Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM (JACM)*, 37(3):607–625, 1990.

Amnon Rapoport, Tamar Kugler, Subhasish Dugar, and Eyran J Gisches. Choice of routes in congested traffic networks: Experimental tests of the braess paradox. *Games and Economic Behavior*, 65(2):538–571, 2009.

Stephen Ross. Return, risk and arbitrage. *Risk and Return in Finance*, Vol. I, 01 1977.

Matthias Ruß, Gunther Gust, and Dirk Neumann. The constrained reliable shortest path problem in stochastic time-dependent networks. *Operations Research*, 2021.

Andrei Shleifer and Robert W Vishny. The limits of arbitrage. *The Journal of finance*, 52(1):35–55, 1997.

Richard Steinberg and Willard I Zangwill. The prevalence of braess' paradox. *Transportation Science*, 17(3):301–318, 1983.

Yong Wang, Guoliang Li, and Nan Tang. Querying shortest paths on time dependent road networks. *Proceedings of the VLDB Endowment*, 12(11):1249–1261, 2019.

Athanasios K. Ziliaskopoulos and Hani S. Mahmassani. Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications. *Transportation Research Record*, 1993.