Generalized Parametric Path Problems

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

1. kshitij@comp.nus.edu.sg | School of Computing, National University of Singapore.

2. girish.varma@iiit.ac.in | CSTAR and ML Lab, IIIT Hyderabad, India.

3. School of Technology and Computer Science, Tata Institute of Fundamental Research, Mumbai, India.

37th Conference on Uncertainty in Artificial Intelligence
July 27-30, 2021

uai2021

1. Parametric Path Problems

Given i.) city road map as a directed acyclic graph (Fig 1), ii.) time taken to cross each link as a function of start time on the link and iii.) a starting time x_0, find the shortest path from s to t.

The time taken to reach t start from s at time x and following the path P_1, is given by

$$c_6(c_5(a_2(a_1x+b_1)+b_2)+d_5)+d_6.$$

Fig 2 shows time taken in each of the 6 paths as a function of time and Tab 1 lists shortest paths for every interval of time. Parametrized path problems can model transporation planning, investment planning, multicurrency arbitrage etc.

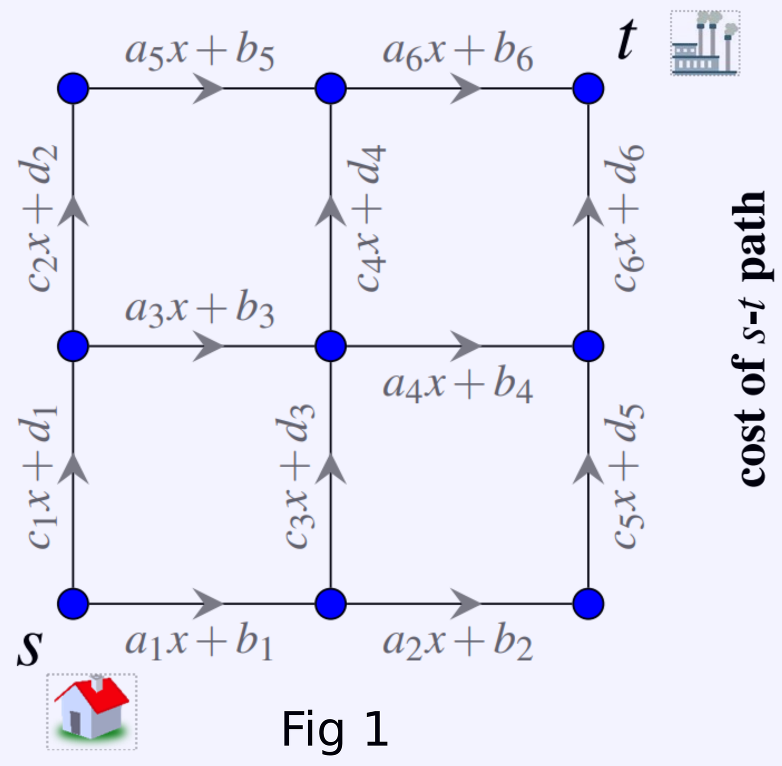


Fig 1

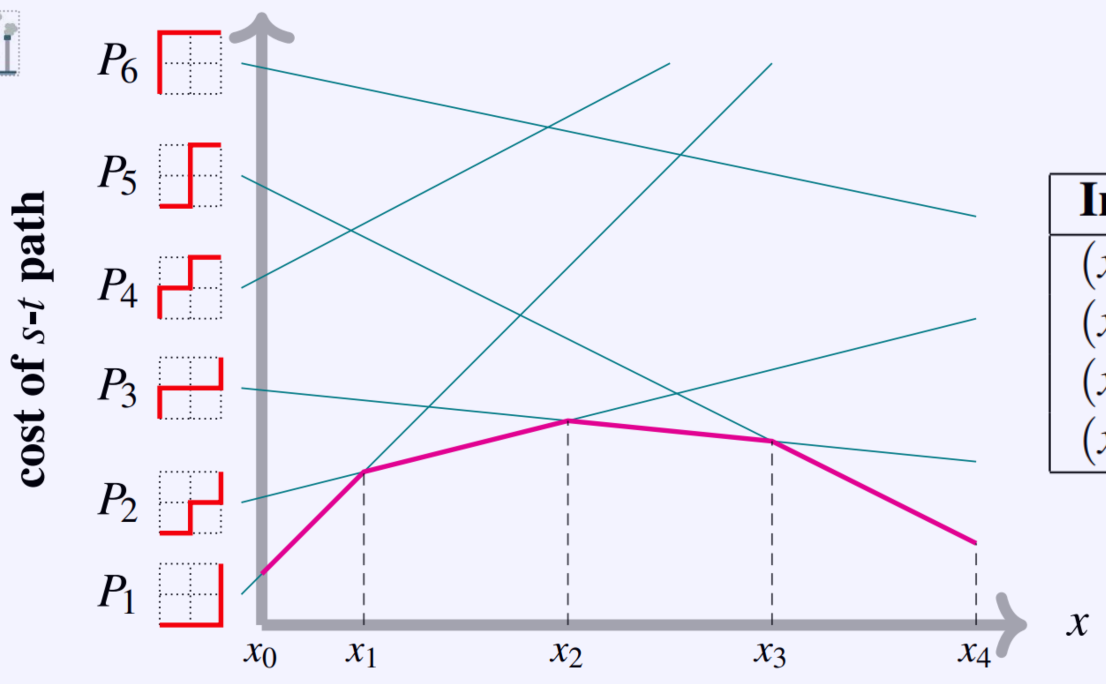


Fig 2

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

Tab 1

2. A Generalization

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.

Problem (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).

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).

Problem (GPP with Preprocessing (PGPP)).
Input: *An instance* (G,W,L) *of GPP.*
Output: *A mapping of* $\mathbf{x}_0$ *to optimal paths (see Tab 1).*

3. Applications

Parametrized path problems have been studied in Transportation and Finance domains under specific assumptions. For eg. the Time Dependent Shortest Path problem studied in the Transportation domain (Dean [2004], Dehne et al. [2012], Foschini et al. [2014]) assumes monotone weight functions (called FIFO). Our results does not assume these, making them more widely applicable.

Braess Paradox in Transport Networks. Intuitively adding an extra road to a road network reduces the traffic. However Braess (1968) showed that adding a road can increase traffic congestion. There have been documented real-world occurrences: Stuttgart (1969), Seoul (2005), New York City (2009). Since our results do not assume monotone FIFO conditions, transport planning in some situations where the paradox applies can be done using our algorithms.

4. Algorithm for Scalar GPP with Linear Weight Functions

Theorem. *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.*

Our algorithm is similar to the Bellman-Ford-Moore shortest path algorithm, 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.

5. Upper bound for Scalar PGPP with Linear Weights

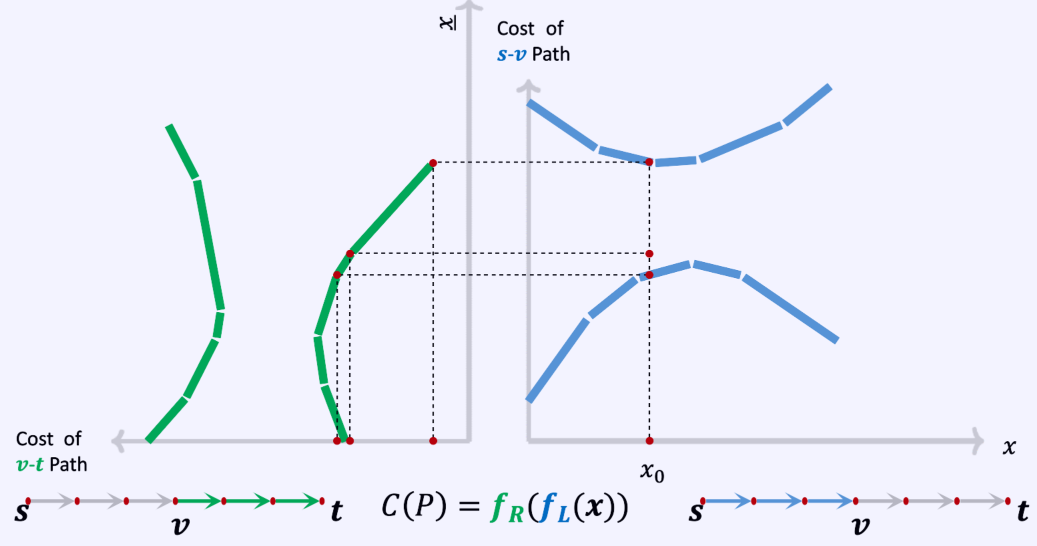
We study scalar PGPP, and show that the total number of different shortest s-t paths (for different values of $x_0\in(-\infty,\infty)$) is at most quasi-polynomial in n.

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 cost$(P)(x)$. We show that the *piecewise linear lower envelope* (denoted by cost$_G(x)$, indicated in pink in Fig 2) 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. *Let* $\mathcal{P}$ *be the set of* s-t *paths in* G*. Then, the cost function of the shortest* s-t *path, given by* $cost_G(x)=\min_{P:P\in\mathcal{P}} cost(P)(x)$*, is a piecewise linear function such that*

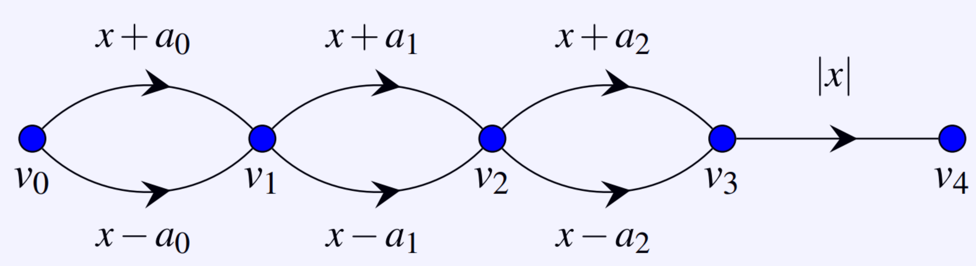
$$p(cost_G(x))\le n^{\log n+O(1)}.$$

The key technical ingredient in the proof is that, when cost functions are composed, the number of discontinuities in the upper and lower envelope only grows additively.



6. Hardness of Scalar GPP with Non-linear Weights

We reduce an instance of the SetPartition (which is NP-Hard) to an instance of Scalar GPP with the following DAG having only 1 discontinous edge weight function. The SetPartition problem asks if a given set of n integers $A=\{a_0,\ldots,a_{n-1}\}$ can be partitioned into two subsets S and $A\setminus S$ such that they have the same sum. It can be observed that the cost of the shortest path is 0 iff SetPartition instance has a solution.



Theorem. *Let* $(G,W,L,\mathbf{x}_0)$ *be a GPP instance with a special edge* e^**, where* G *has* n *vertices and* $w_e(x)=a_ex+b_e$ *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.*

7. Lower bound for Scalar PGPP with Non-linear Weights

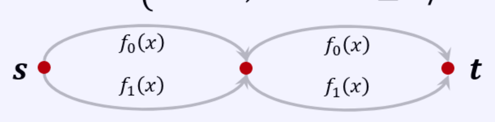
Theorem. *Let* (G,W,L,x_0) *be a GPP instance with a special edge* e^**, where* G *has* n *vertices and* $w_e(x)=a_ex+b_e$ *for every edge* $e\in E(G)\setminus\{e^*\}$*, and* $w_{e^*}(x)$ *is piecewise linear with 2 pieces. Then it is NP-hard to find an* s-t *path whose cost approximates the cost of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

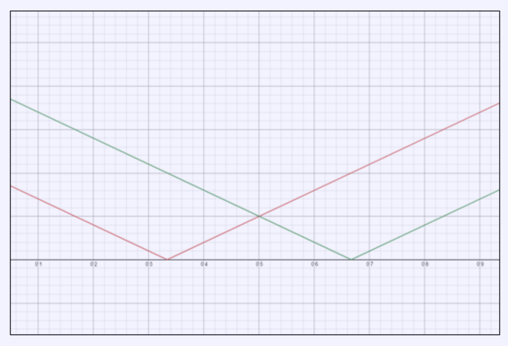
Proof Idea: There exists a graph on n nodes with 2^n paths such that each paths shows up in the lower envelope. Each time a new vertex is added to the graph, the number of pieces in the shortest path cost function is doubled.

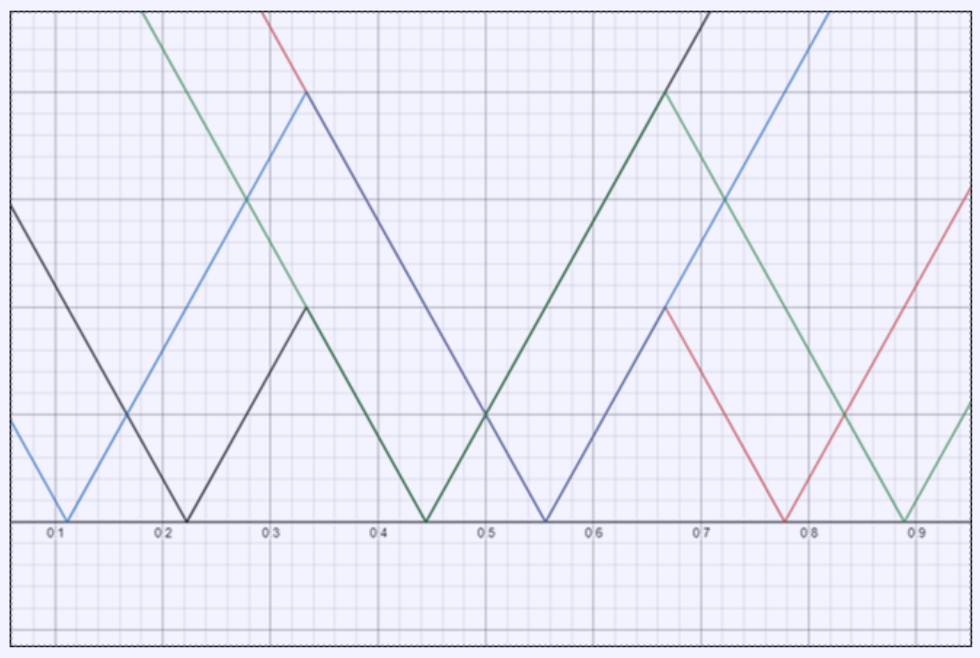
$n=2$

$$f_0(x)=\begin{cases}1-3x, & \text{if } x\le 1/3\\ 3x-1, & \text{if } x\ge 1/3\end{cases}$$

$$f_1(x)=\begin{cases}2-3x, & \text{if } x\le 2/3\\ 3x-2, & \text{if } x\ge 2/3\end{cases}$$

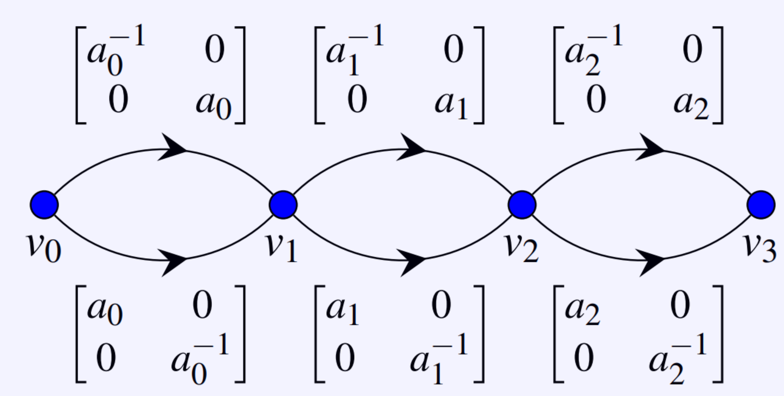






8. Hardness of Non-scalar GPP with Linear Weights

We reduce an instance of the ProductPartition (which is NP-Hard) to an instance of GPP with the following DAG having 2 dimensional linear edge weight function. The ProductPartition problem asks if a given set of n integers $A=\{a_0,\ldots,a_{n-1}\}$ can be partitioned into two subsets S and $A\setminus S$ such that they have the same product. It can be observed that the cost of the shortest path is 1 iff ProductPartition instance has a solution.



Theorem. *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.