Path of Solutions for Fused Lasso Problems

Torpong Nitayanont[a], Cheng Lu[b] and Dorit S. Hochbaum[c]
Department of Industrial Engineering and Operations Research, University of California, Berkeley, Berkeley, CA, U.S.A.

Keywords: Fused Lasso, Path of Solutions, Minimum Cut, Hyperparameter Selection, Signal Processing.

Abstract: In a fused lasso problem on sequential data, the objective consists of two competing terms: the fidelity term and the regularization term. The two terms are often balanced with a tradeoff parameter, the value of which affects the solution, yet the extent of the effect is not a priori known. To address this, there is an interest in generating the *path of solutions* which maps values of this parameter to a solution. Even though there are infinite values of the parameter, we show that for the fused lasso problem with convex piecewise linear fidelity functions, the number of different solutions is bounded by n^2q where n is the number of variables and q is the number of breakpoints in the fidelity functions. Our path of solutions algorithm, PoS, is based on an efficient minimum cut technique. We compare our PoS algorithm with a state-of-the-art solver, Gurobi, on synthetic data. The results show that PoS generates all solutions whereas Gurobi identifies less than 22% of the number of solutions, on comparable running time. Even allowing for hundreds of times factor increase in time limit, compared with PoS, Gurobi still cannot generate all the solutions.

1 INTRODUCTION

We consider here the class of convex piecewise linear fused lasso problems:

$$\text{(PFL)} \min_{x_1,\ldots,x_n} \sum_{i=1}^{n} f_i^{pl}(x_i) + \lambda \sum_{i=1}^{n-1} |x_i - x_{i+1}| \quad (1)$$

where each fidelity function $f_i^{pl}(x_i)$ is a convex piecewise linear function and λ is a parameter that controls the tradeoff between the fidelity term and the regularization term, which penalizes the differences between consecutive variables.

Applications of problems in the class of PFL include the study of DNA copy number gains and losses on chromosomes using Comparative Genomic Hybridization (CGH) (Eilers and De Menezes, 2005). In CGH, each piecewise linear function takes the form of $\tau(x_i - a_i)_+ + (1-\tau)(a_i - x_i)_+$ for a given $\tau \in [0,1]$ and data point a_i, and the problem is called quantile regression. In signal processing (Storath et al., 2016), each piecewise linear function is $f_i^{pl}(x_i) = w_i|x_i - a_i|$ for a given weight w_i and data point a_i. Each piecewise linear function in these examples contains exactly one breakpoint.

Clearly, the optimal solution of PFL (1) varies with regard to the tradeoff parameter λ. In the works mentioned above, the choice of λ is either handpicked (Storath et al., 2016) or selected via cross validation (Eilers and De Menezes, 2005). However, there could still be λ values outside of the search range that lead to desirable results, which go unnoticed, or the resolution of the cross validation grid search could still be made finer. Our goal is to find *the path of solutions* to PFL for all $\lambda \in [0,\infty)$, that is we want to identify a set of ranges or intervals of λ (we call them λ-ranges) that span $[0,\infty)$, i.e. $\Lambda = \{[0,\lambda_1),[\lambda_1,\lambda_2),\ldots,[\lambda_m,\infty)\}$, such that the solutions to PFL for two values of λ from the same range are similar.

In addition to the identification of the set of λ-ranges, we also find the optimal solution for each λ-range, $\{\boldsymbol{x}^*([0,\lambda_1)),\boldsymbol{x}^*([\lambda_1,\lambda_2)),\ldots,\boldsymbol{x}^*([\lambda_m,\infty))\}$. Note that we use the notation of $\boldsymbol{x}^*(\lambda)$ to refer to the solution of PFL for a specific value of λ. The notation of $\boldsymbol{x}^*([\lambda_1,\lambda_2))$, for example, refers to the optimal solution to PFL for any value of λ in the range $[\lambda_1,\lambda_2)$. It also implies that the solution for all values of λ in this range are identical.

The relevant literature on finding the path of solutions include the work on fused-lasso signal approximator (FLSA) by Hoefling (2010). The problem is

[a] https://orcid.org/0009-0002-6976-1951
[b] https://orcid.org/0000-0001-5137-7199
[c] https://orcid.org/0000-0002-2498-0512

Nitayanont, T., Lu, C. and Hochbaum, D.
Path of Solutions for Fused Lasso Problems.
DOI: 10.5220/0012433200003654

In *Proceedings of the 13th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2024)*, pages 107-118
ISBN: 978-989-758-684-2; ISSN: 2184-4313

defined as follows:

$$\min_{x_1,\ldots,x_n} \frac{1}{2}\sum_{i=1}^{n}(x_i-a_i)^2+\lambda_1\sum_{i=1}^{n}|x_i|+\lambda_2\sum_{i=1}^{n-1}|x_i-x_{i+1}|$$

Hoefling (2010) solves for the optimal solution $\boldsymbol{x}^*(\lambda_1,\lambda_2)$ for all λ_1 and λ_2 by first finding the solution for $\lambda_1 = 0$ in terms of λ_2, $\boldsymbol{x}^*(0,\lambda_2)$, in $O(n\log n)$. Then, $\boldsymbol{x}^*(\lambda_1,\lambda_2)$ is computed as a function of $\boldsymbol{x}^*(0,\lambda_2)$. Tibshirani (2011) presents path of solution algorithms to a generalized lasso problem, which is formulated as

$$\min_{\boldsymbol{x}\in\mathbb{R}^n}\frac{1}{2}\|\boldsymbol{b}-A\boldsymbol{x}\|_2^2+\lambda\|D\boldsymbol{x}\|_1$$

for different configurations of A and D such as when D is an arbitrary matrix and $A = I$. Note that the regularization term of this problem can be made equivalent to our regularization term, $\sum_{i=1}^{n-1}|x_i-x_{i+1}|$, when D is set to an appropriate matrix.

Another work on the path of solutions is the work by Wang et al. (2006), which solves the regularized least absolute deviation regression problem (RLAD):

$$\min_{x\in\mathbb{R}^p}||y-Ax||_1+\lambda||x||_1$$

given $A\in\mathbb{R}^{n\times p}, y\in\mathbb{R}^n$ for every possible value of λ in $O(\log(np)\,min(p,n)^2)$. They applied their method on the image reconstruction problem.

Generalized isotonic median regression or GIMR problem, which has the following formulation:

$$\text{(GIMR)}\quad \min_{x_1,\ldots,x_n}\sum_{i=1}^{n}f_i^{pl}(x_i)+\sum_{i=1}^{n-1}d_{i,i+1}\cdot(x_i-x_{i+1})_+ + \sum_{i=1}^{n-1}d_{i+1,i}\cdot(x_{i+1}-x_i)_+ \qquad (2)$$

for a given set of $\{d_{i,i+1},d_{i+1,i}\}_{i=1}^{n-1}$, is solved by an algorithm, called HL-algorithm hereafter, in the work of Hochbaum and Lu (2017). HL-algorithm solves GIMR by formulating the problem into a sequence of minimum cut problems, which can be solved efficiently in $O(q\log n)$ where q is the total number of breakpoints in fidelity functions. This is the best complexity for solving GIMR to date. PFL is a special case of GIMR and therefore can be solved, for a fixed λ, using the HL-algorithm. The path of solutions algorithm proposed here that solves PFL for all $\lambda\geq 0$ is an extension of the HL algorithm. We introduce here the notations and concepts that will be used throughout the paper.

Notation and Preliminaries. GIMR (2) can be viewed as defined on a bi-directional path graph $G=(V,A)$ with node set $V=\{1,2,\ldots,n\}$ and arc set $A=\{(i,i+1),(i+1,i)\}_{i=1,\ldots,n-1}$. Each node $i\in V$ corresponds to the variable x_i. Let the node interval $[i,j]$ in the graph G for $i\leq j$ be the subset of consecutive nodes in $V,\{i,i+1,\ldots,j-1,j\}$.

We define an associated graph G_{st} with the set of vertices $V_{st}=V\cup\{s,t\}$ and the set of arcs $A_{st}=A\cup A_s\cup A_t$. The two appended nodes s and t are called the source and the sink node, respectively. $A_s=\{(s,i): i\in V\}$ and $A_t=\{(i,t): i\in V\}$ are the sets of source adjacent arcs and sink adjacent arcs. Each arc $(i,j)\in A_{st}$ has an associated nonnegative capacity $c_{i,j}$.

An s,t-cut is a partition of $V_{st},(\{s\}\cup S,T\cup\{t\})$, where $T=\bar{S}=V\backslash S$. For simplicity, we refer to an s,t-cut partition as (S,T). We refer to S as the source set of the cut, excluding s, and T the sink set, excluding t. For each node $i\in V$, we define its *status* in graph G^{st} as status$(i)=s$ if $i\in S$ (referred as an s-node), otherwise status$(i)=t)(i\in T)$ (referred as a t-node).

The capacity of a cut (S,T) is defined as $C(\{s\}\cup S,T\cup\{t\})$ where $C(V_1,V_2)=\sum_{i\in V_1,j\in V_2}c_{i,j}$. A minimum s,t-cut in G^{st} is an s,t-cut (S,T) that minimizes $C(\{s\}\cup S,T\cup\{t\})$. Hereafter, any reference to a minimum cut is to the unique minimum s,t-cut with the maximal source set, that is, the source set that is not contained in any other source set of a minimum cut in case that there are multiple minimum cuts.

A convex piecewise linear function $f_i^{pl}(x_i)$ is specified by its ascending list of q_i breakpoints, $a_{i,1}<a_{i,2}<\ldots<a_{i,q_i}$, and the slopes of the q_i+1 linear pieces between every two adjacent breakpoints, denoted by $w_{i,0}<w_{i,1}<\ldots<w_{i,q_i}$. To define a breakpoint more formally, each convex piecewise linear function $f_i^{pl}(x_i)$ can be viewed as a maximum of multiple affine functions, which are sorted according to their slopes. We call the x-coordinate of an intersection of two consecutive affine functions a *breakpoint*. Let the sorted list of the union of q breakpoints of all the n convex piecewise linear functions be $a_{i_1,j_1}<a_{i_2,j_2}<\ldots<a_{i_q,j_q}$ (w.l.o.g. we may assume that the n sets of breakpoints are disjoint (Hochbaum and Lu, 2017)), where a_{i_k,j_k}, the k-th breakpoint in the sorted list, is the breakpoint between the (j_k-1)-th and the j_k-th linear pieces of $f_{i_k}^{pl}(x_{i_k})$.

Overview of Paper. Section 2 describes the algorithm that solves PFL (1) for a fixed value of λ, the HL-algorithm. In Section 3, we give a description of the path of solutions algorithm, or PoS, for ℓ_1-fidelity fused lasso problem. In Section 4, we generalize it to PFL with convex piecewise linear fidelity function. We provide the bound of the number of different solutions of PFL across all nonnegative λ values as well as the time complexity of the algorithm in Section 5. Lastly, we conclude with experimental results in Section 6.

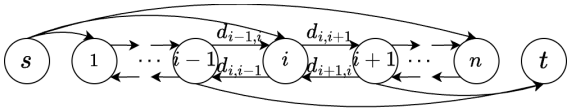

Figure 1: $G^{st}(\alpha)$: For node i, if its right sub-gradient at α, $(f_i^{pl})'(\alpha)$, is negative, s is connected to i with an arc of capacity $c_{s,i} = -(f_i^{pl})'(\alpha)$. Otherwise, i is connected to t with an arc of capacity $c_{i,t} = (f_i^{pl})'(\alpha)$.

2 ALGORITHM THAT SOLVES FUSED LASSO FOR A FIXED VALUE OF λ

The PFL problem (1) that we consider in this work can be solved for a fixed λ using the HL-algorithm, introduced by Hochbaum and Lu (2017). The HL-algorithm solves problems in the class of generalized isotonic median regression or GIMR (2). PFL is a special case of GIMR (2) when $d_{i,i+1} = d_{i+1,i} = \lambda$ for all $i \in \{1,2,...,n-1\}$. In this section, we give a description of how the HL-algorithm solves GIMR, which also explains for the case of PFL.

We construct a parametric graph $G^{st}(\alpha) = (V_{st}, A_{st})$, shown in Figure 1, that is associated with the bi-directional path graph $G = (V, A)$, for any scalar value α. The capacities of arcs $(i, i+1), (i+1, i) \in A$ are $c_{i,i+1} = d_{i,i+1}$ and $c_{i+1,i} = d_{i+1,i}$ respectively. Arc $(s,i) \in A_s$ has capacity $c_{s,i} = \max\{0, -(f_i^{pl})'(\alpha)\}$ and arc $(i,t) \in A_t$ has capacity $c_{i,t} = \max\{0, (f_i^{pl})'(\alpha)\}$, where $(f_i^{pl})'(\alpha)$ is the right sub-gradient of function $f_i^{pl}(\cdot)$ at argument α. (One can select instead the left sub-gradient.) For any given value of α, we have either $c_{s,i} = 0$ or $c_{i,t} = 0$, that is, i is never connected to both s and t for the same α.

The link between the minimum cut for any given value of α and the optimal solution to GIMR (2) is characterized in the following threshold theorem:

Theorem 2.1 (Threshold theorem (Hochbaum, 2001))**.** *For any given α, let S^* be the maximal source set of the minimum cut in graph $G^{st}(\alpha)$. Then, there is an optimal solution $\boldsymbol{x}^*$ to GIMR (2) satisfying $x_i^* \geq \alpha$ if $i \in S^*$ and $x_i^* < \alpha$ if $i \in T^*$.*

An important property of $G^{st}(\alpha)$ is that the capacities of source adjacent arcs and sink adjacent arcs are nonincreasing and nondecreasing functions of α, respectively. The capacities of all the other arcs are constants. This implies the following nested cut property:

Lemma 2.2 (Nested cut property (Gallo et al., 1989; Hochbaum, 2001, 2008))**.** *For any two parameter values $\alpha_1 \leq \alpha_2$, let S_{α_1} and S_{α_2} be the respective maximal source set of the minimum cuts of $G^{st}(\alpha_1)$ and $G^{st}(\alpha_2)$, then $S_{\alpha_1} \supseteq S_{\alpha_2}$.*

We remark that the above threshold theorem and nested cut property both work not only for GIMR (2) defined on a bi-path graph, but also for a generalization of GIMR that is defined on arbitrary (directed) graphs, where the regularization term may penalize the difference between arbitrary pairs of variables rather than just consecutive variables.

Based on the threshold theorem, it is sufficient to solve the minimum cuts in the parametric graph $G^{st}(\alpha)$ for all values of α, in order to solve GIMR (2). In piecewise linear functions, the right sub-gradients for α values between any two adjacent breakpoints are constant. Thus, the source and sink adjacent arc capacities remain constant for α between any two adjacent breakpoint values in the sorted list of breakpoints over all the n convex piecewise linear functions. This result leads to the following lemma given by Hochbaum and Lu (2017).

Lemma 2.3. *The minimum cuts in $G^{st}(\alpha)$ remain unchanged for α assuming any value between any two adjacent breakpoints in the sorted list of breakpoints of all the n convex piecewise linear functions, $\{f_i^{pl}(x_i)\}_{i=1,...,n}$.*

Thus, the values of α to be considered can be restricted to the set of breakpoints of the n convex piecewise linear functions, $\{f_i^{pl}(x_i)\}_{i=1,...,n}$. The HL algorithm solves GIMR (2) by efficiently computing the minimum cuts of $G^{st}(\alpha)$ for subsequent values of α in the ascending list of breakpoints, $a_{i_1,j_1} < a_{i_2,j_2} < \ldots < a_{i_q,j_q}$.

Let G_k, for $k \geq 1$, denote the parametric graph $G^{st}(\alpha)$ for α equal to a_{i_k,j_k}, i.e., $G_k = G^{st}(a_{i_k,j_k})$. For $k = 0$, we let $G_0 = G^{st}(a_{i_1,j_1} - \varepsilon)$ for a small value of $\varepsilon > 0$. Let (S_k, T_k) be the minimum cut in G_k, for $k \geq 0$. Recall that S_k is the maximal source set. The nested cut property (Lemma 2.2) implies that $S_k \supseteq S_{k+1}$ for $k \geq 0$. Based on the threshold theorem and the nested cut property, we know that for each node $j \in \{1,\ldots,n\}$, $x_j^* = a_{i_k,j_k}$ for the index k such that $j \in S_{k-1}$ and $j \in T_k$.

The HL-algorithm generates the respective minimum cuts of G_k in increasing order of k. It is shown by Hochbaum and Lu (2017) that (S_k, T_k) can be computed from (S_{k-1}, T_{k-1}) in time $O(\log n)$. Hence the total complexity of the algorithm is $O(q \log n)$. The efficiency of updating (S_k, T_k) from (S_{k-1}, T_{k-1}) is based on the following key results.

The update of the graph from G_{k-1} to G_k is simple as it only involves a change in the capacities of the source and sink adjacent arcs of $i_k, (s, i_k)$ and (i_k, t). Recall that from G_{k-1} to G_k, the right sub-gradient

$(f_{i_k}^{pl})'$ changes from w_{i_k,j_k-1}, the slope of the k-th linear piece of $f_{i_k}^{pl}$, to w_{i_k,j_k}, the slope of the $(k+1)$-th linear piece of $f_{i_k}^{pl}$. Thus, the changes of c_{s,i_k} and $c_{i_k,t}$ from G_{k-1} to G_k depend on the signs of w_{i_k,j_k-1} and w_{i_k,j_k}, which have three possible cases.

Case 1. $w_{i_k,j_k-1} \leq 0, w_{i_k,j_k} \leq 0 : c_{s,i_k}$ changes from $-w_{i_k,j_k-1}$ to $-w_{i_k,j_k}$ while $c_{i_k,t}$ remains zero.

Case 2. $w_{i_k,j_k-1} \leq 0, w_{i_k,j_k} \geq 0 : c_{s,i_k}$ changes from $-w_{i_k,j_k-1}$ to 0 and $c_{i_k,t}$ changes from 0 to w_{i_k,j_k}.

Case 3. $w_{i_k,j_k-1} \geq 0, w_{i_k,j_k} \geq 0 : c_{i_k,t}$ changes from w_{i_k,j_k-1} to w_{i_k,j_k} while c_{s,i_k} remains zero.

Note that the update from G_{k-1} to G_k does not involve the values of $d_{i,i+1}$ and $d_{i+1,i}$ in GIMR (2), of which both take the values of λ in PFL (1).

Based on the nested cut property, for any node i, if $i \in T_{k-1}$, then i remains in T_k and the sink set for all subsequent cuts. Hence, an update of the minimum cut in G_k from the minimum cut in G_{k-1} can only involve shifting some nodes from source set S_{k-1} to sink set T_k. Formally, the relation between (S_{k-1}, T_{k-1}) and (S_k, T_k) is characterized in Lemma 2.4 and 2.5 given by Hochbaum and Lu (2017):

Lemma 2.4. *If $i_k \in T_{k-1}$, then $(S_k, T_k) = (S_{k-1}, T_{k-1})$.*

We define *s-interval* in graph G to be the interval of consecutive s-nodes, that is the set of consecutive nodes that are in the source set S of the minimum cut of G. An *s-interval containing i* refers to an interval of s-nodes in G that contains node i. If i is a t-node, then such s-interval is an empty interval. The *maximal s-interval containing i* refers to the s-interval containing i that is not a subset of any other such s-intervals.

Lemma 2.5. *If $i_k \in S_{k-1}$, then all the nodes that change their status from s in G_{k-1} to t in G_k must form a (possibly empty) s-interval of i_k in G_{k-1}.*

Both lemmas imply that the derivation of the minimum cut in G_k from the minimum cut in G_{k-1} can be done by finding an optimal s-interval of nodes containing i_k that change their status from s to t. If this interval of nodes, referred to as *node status change interval*, is $[i_{kl}, i_{kr}]$ then it follows that $(S_k, T_k) = (S_{k-1} \backslash [i_{kl}, i_{kr}], T_{k-1} \cup [i_{kl}, i_{kr}])$. Combining this with the threshold theorem (Theorem 2.1), all x_i for $i \in [i_{kl}, i_{kr}]$ have their optimal solutions in GIMR (2) equal to a_{i_k,j_k}.

To find the node status change interval from G_{k-1} to G_k that gives the minimum cut in G_k, we rely on the following lemma from Hochbaum and Lu (2017).

Lemma 2.6. *Given the maximal s-interval containing i_k in G_{k-1}, $[i_{kl}, i_{kr}]$, the node status change interval $[i_{k1}, i_{k2}]$ is the optimal solution to the following problem defined on G_k*

$$\min_{[i_{k1}, i_{k2}]} C(\{s\}, [i_{k1}, i_{k2}]) + C([i_{kl}, i_{kr}] \backslash [i_{k1}, i_{k2}], \{t\}) \quad (3)$$
$$+ C([i_{kl}, i_{kr}] \backslash [i_{k1}, i_{k2}], [i_{k1}, i_{k2}] \cup \{i_{kl} - 1, i_{kr} + 1\})$$
$$s.t. \quad i_k \in [i_{k1}, i_{k2}] \subseteq [i_{kl}, i_{kr}] \text{ or } [i_{k1}, i_{k2}] = \emptyset$$

Solving problem (3) in Lemma 2.6 can be easier than directly solving the minimum cut problem on G_k. Our path of solutions algorithm rely on all of the stated theorems and lemmas, but particularly on Lemma 2.5 and 2.6.

3 PATH OF SOLUTIONS ALGORITHM FOR ℓ_1 FIDELITY FUNCTIONS

The proposed path of solutions algorithm, or PoS, finds the optimal solution $\boldsymbol{x}^*$ for all nonnegative λ. We first demonstrate in this section how PoS solves a specific PFL problem, the ℓ_1-fidelity fused lasso:

$$(\ell_1\text{-}PFL) \min_{x_1,\ldots,x_n} \sum_{i=1}^{n} w_i |x_i - a_i| + \lambda \sum_{i=1}^{n-1} |x_i - x_{i+1}| \quad (4)$$

for all nonnegative λ, given a set of breakpoints $\{a_i\}_{i=1}^n$ and a set of positive penalty weights $\{w_i\}_{i=1}^n$.

As we consider ℓ_1-PFL, which is a special case of both GIMR and PFL, we substitute some notations introduced in Section 2 by simpler terms.

For ℓ_1-PFL, $f_i^{pl} = w_i|x_i - a_i|$ consists of two linear pieces and a single breakpoint a_i. $w_{i,0} = -w_i, w_{i,1} = w_i$ and $a_{i,0} = a_i$. Moreover, both $d_{i,i+1}$ and $d_{i+1,i}$ are equal to λ. Similar to the HL-algorithm, we let the indices $i_1, i_2, ..., i_n$ be the sorted indices of n breakpoints such that $a_{i_1} < a_{i,2} < \ldots < a_{i_n}$. In the graph G_k or $G^{st}(a_{i_k})$, the weights of the source adjacent arc $(s,i) \in A_s$ and the sink adjacent arc $(i,t) \in A_t$ are

$$c_{s,i} = \begin{cases} -w_i & \text{if } a_i < a_{i_k} \\ 0 & \text{otherwise} \end{cases}, \quad c_{i,t} = \begin{cases} 0 & \text{if } a_i < a_{i_k} \\ w_i & \text{otherwise} \end{cases}$$

The arcs $(i, i+1)$ and $(i+1, i)$ have weights equal to λ, for $i \in \{1, 2, \ldots, n-1\}$. The HL-algorithm computes the optimal solution of problem (4) by solving for the minimum cuts of graphs $G_0, G_1, \ldots, G_n$.

At iteration k of the HL-algorithm, we update the graph from G_{k-1} to G_k and compute the minimum cut solution (S_k, T_k), which allows us to discover the optimal solutions of variables whose nodes are in T_k but not in T_{k-1}, as explained last section. In PoS where we consider all $\lambda \in [0, \infty)$, the minimum cut solutions of G_k for different values of λ may be different. To reflect the dependence on λ, we use the notations of

$G_k(\lambda)$ and $(S_k(\lambda), T_k(\lambda))$ for the graph and its minimum cut solutions at step k of the algorithm. We may revert to the notations of G_k and (S_k, T_k) when we discuss the graph for any λ in general.

The overall idea of PoS is as follow: at the first iteration of the algorithm, we start with the graph G_0 and the range of all possible values of λ, $[0,\infty)$, denoted by $\mathbf{\Lambda_0}$, on which all values of λ result in the same minimum cut $(S_0(\lambda), T_0(\lambda))$ where $S_0(\lambda) = \{1,\ldots,n\}$ and $T_0(\lambda) = \emptyset$. We then update the graph to G_1 and partition $\mathbf{\Lambda_0}$ into different ranges of λ such that λ in the same range has the same minimum cut solution $(S_1(\lambda), T_1(\lambda))$. The resulting set of ranges of λ for G_1 is denoted $\mathbf{\Lambda_1}$. For conciseness, we call each range of values of λ a λ-*range*.

At iteration k, we start with a set of ranges of λ-values, denoted by $\mathbf{\Lambda}_{k-1}$, spanning $[0,\infty)$, such that each λ-range $\Lambda \in \mathbf{\Lambda}_{k-1}$ corresponds to a particular minimum cut solution $(S_{k-1}(\Lambda), T_{k-1}(\Lambda))$ of $G_{k-1}(\Lambda)$. We update G_{k-1} to G_k, and partition each λ-range in $\mathbf{\Lambda}_{k-1}$ into smaller λ-ranges according to the solutions of (S_k, T_k). These λ-ranges that are offspring of ranges in $\mathbf{\Lambda}_{k-1}$ together form the set $\mathbf{\Lambda}_k$. The set $\mathbf{\Lambda}_k$ is then passed onto the next iteration.

We explain the first iteration of PoS in Subsection 3.1. Subsection 3.2 describes the iteration k. In Subsection 3.2.1, we show the computation of the minimum cuts of G_k given the minimum cut solutions of G_{k-1}. Subsection 3.2.2 completes the iteration k by using the minimum cut solutions to partition λ-ranges from the previous iteration. We conclude with the outputs upon the completion of PoS in Subsection 3.3.

3.1 First Iteration of the Path of Solutions Algorithm

At the beginning of the first iteration, it is clear that the minimum cut $(S_0(\lambda), T_0(\lambda))$ of $G_0(\lambda)$, shown in Figure 2, is always $(\{1,2,...,n\},\{\})$ for any $\lambda \in [0,\infty)$ since all nodes are connected to s and none is connected to t. We let $\mathbf{\Lambda_0}$ denote the set of λ-ranges such that for λ_1 and λ_2 that are in the same λ-range, we have $(S_0(\lambda_1), T_0(\lambda_1)) = (S_0(\lambda_2), T_0(\lambda_2))$. Since there is only one solution of $(S_0(\lambda), T_0(\lambda))$ across all $\lambda \in [0,\infty)$, we have $\mathbf{\Lambda_0} = \{[0,\infty)\}$.

The first step of the algorithm is to determine the minimum cuts $(S_1(\lambda), T_1(\lambda))$ of $G_1(\lambda)$, in Figure 3. We consider the outcomes of the cut capacity in terms of λ, for each possible case of $T_1(\lambda)$ according to Lemma 2.5, which states that $T_1(\lambda) \backslash T_0(\lambda)$ must form an s-interval containing i_1 in $G_0(\lambda)$. Since all nodes in $G_0(\lambda)$ are s-nodes for all λ, possible $T_1(\lambda)$ are node intervals with the form of $[i_{1l}, i_{1r}]$ where $i_1 \in [i_{1l}, i_{1r}] \subseteq [1,n]$. We group them into cases such that the coefficient of λ in the cut capacity of each case is different from other cases.

Case 1. $T_1(\lambda) = \emptyset$. None of the nodes changes its status from s to t. $C(S_1(\lambda), T_1(\lambda)) = w_{i_1}$.

Case 2. $T_1(\lambda) = [1,n]$.
$C(S_1(\lambda), T_1(\lambda)) = (\sum_{i=1}^{n} w_i) - w_{i_1}$

Case 3. $T_1(\lambda)$ consists of some, but not all, nodes. There are three subcases.

Case 3.1. $T_1(\lambda) = [1, i_{1r}]$ where $i_1 \leq i_{1r} < n$.
$C(S_1(\lambda), T_1(\lambda)) = \lambda + \min_{i_1 \leq i_{1r} < n} (\sum_{i=1}^{i_{1r}} w_i) - w_{i_1}$
The optimal $T_1(\lambda)$ under Case 3.1 is $[1, i_1]$, when $i_{1r} = i_1$, with $C(S_1(\lambda), T_1(\lambda)) = \lambda + \sum_{i=1}^{i_1 - 1} w_i$.

Case 3.2. $T_1(\lambda) = [i_{1l}, n]$ where $1 < i_{1l} \leq i_1$. $C(S_1(\lambda), T_1(\lambda)) = \lambda + \min_{1 < i_{1l} \leq i_1} (\sum_{i=i_{1l}}^{n} w_i) - w_{i_1}$
The optimal $T_1(\lambda)$ under Case 3.2 is $[i_1, n]$, when $i_{1l} = i_1$, with $C(S_1(\lambda), T_1(\lambda)) = \lambda + \sum_{i=i_1+1}^{n} w_i$.

Case 3.3. $T_1(\lambda) = [i_{1l}, i_{1r}]$ where $1 < i_{1l} \leq i_1$ and $i_1 \leq i_{1r} < n$. $C(S_1(\lambda), T_1(\lambda)) = 2\lambda + \min_{\substack{1 < i_{1l} \leq i_1 \\ i_1 \leq i_{1r} < n}} (\sum_{i=i_{1l}}^{i_{1r}} w_i) - w_{i_1}$. The optimal $T_1(\lambda)$ under Case 3.3 is $\{i_1\}$ or $[i_1, i_1]$, when $i_{1l} = 1$ and $i_{1r} = 1$. $C(S_1(\lambda), T_1(\lambda)) = 2\lambda$.

The computations of the optimal cuts in Case 3.1, 3.2 and 3.3. rely on the fact that $\{w_i\}_{i=1}^{n}$ are positive.

Notice that the comparison between solutions with similar coefficients of λ in the cut capacity is independent of λ. For instance, Case 3.1, with the cut capacity $\lambda + \sum_{i=1}^{i_1 - 1} w_i$, and Case 3.2, with the cut capacity of $\lambda + \sum_{i=i_1+1}^{n} w_i$, have the same coefficient of λ, which 1, in their cut capacities. Determining which one is more optimal is independent of λ. We only need to compare the constant terms of the two cases.

Let T^{ρ} denote the cut with the smallest term constant among the cuts whose coefficients of λ are equal to ρ, and c_ρ denote the corresponding minimum constant terms. The cut capacity due to the sink set T^{ρ} is then equal to $\rho\lambda + c_\rho$. If multiple cut solutions have the smallest constant, we select the cut solution that results in the largest source set possible, due to the maximal source set requirement.

For $\rho = 0$, we compare Case 1 and Case 2. $c_0 = \min(w_{i_1}, (\sum_{i=1}^{n} w_i) - w_{i_1})$. If $w_{i_1} \leq (\sum_{i=1}^{n} w_i) - w_{i_1})$ then $T^0 = \emptyset$ (Case 1), otherwise, $T^0 = [1,n]$ (Case 2). Notice that when $w_{i_1} = (\sum_{i=1}^{n} w_i) - w_{i_1})$, we select $T^0 = \emptyset$ (Case 1) rather than $[1,n]$ (Case 2) since $T^0 = \emptyset$ always results in a larger source set.

For $\rho = 1$, we compare Case 3.1 and Case 3.2. $c_1 = \min(\sum_{i=1}^{i_1 - 1} w_i, \sum_{i=i_1+1}^{n} w_i)$. If $\sum_{i=1}^{i_1 - 1} w_i < \sum_{i=i_1+1}^{n} w_i$ then $T^1 = [1, i_1]$ (Case 3.1). If $\sum_{i=1}^{i_1 - 1} w_i > \sum_{i=i_1+1}^{n} w_i$ then $T^1 = [i_1, n]$ (Case 3.2). However, if $\sum_{i=1}^{i_1 - 1} w_i = \sum_{i=i_1+1}^{n} w_i$, we select the one, between Case 3.1 and 3.2, that results in a larger source set.

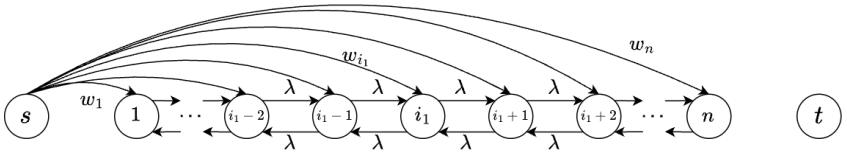

Figure 2: $G_0(\lambda)$ for ℓ_1-PFL (1).

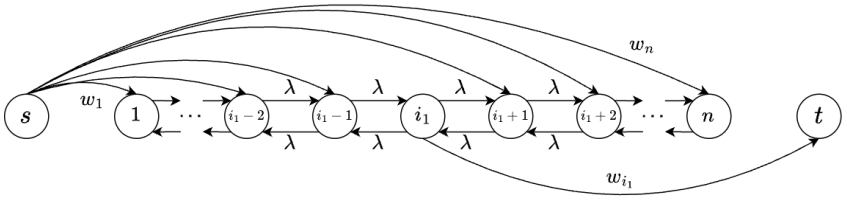

Figure 3: $G_1(\lambda)$ for ℓ_1-PFL (1).

For $\rho = 2$, there is only one case, Case 3.3, whose coefficients of λ in cut capacity is 2. The cut capacity of Case 3.3, $T_1(\lambda) = \{i_1\}$, is 2λ. Hence, $c_2 = 0$ and $T^2 = \{i_1\}$.

With the notations of T^ρ and c_ρ, we compare the capacities between cases of $T_1(\lambda)$ listed above and give the optimal solution of $T_1(\lambda)$ for different λ intervals as follow:

If $T^0 = \emptyset$ then

$$T_1(\lambda) = \begin{cases} T^0, & \text{if } \lambda \geq \frac{c_0}{2} \text{ and } \lambda \geq c_0 - c_1 \\ T^1, & \text{if } \lambda < c_0 - c_1 \text{ and } \lambda > c_1 \\ T^2 & \text{if } \lambda \leq c_1 \text{ and } \lambda < \frac{c_0}{2} \end{cases} \quad (5)$$

However, if $T^0 = [1,n]$ then

$$T_1(\lambda) = \begin{cases} T^0, & \text{if } \lambda > \frac{c_0}{2} \text{ and } \lambda > c_0 - c_1 \\ T^1, & \text{if } \lambda \leq c_0 - c_1 \text{ and } \lambda > c_1 \\ T^2 & \text{if } \lambda \leq c_1 \text{ and } \lambda \leq \frac{c_0}{2} \end{cases} \quad (6)$$

The derivation of both (5) and (6) comes from the comparison of three cut capacities: c_0, $\lambda + c_1$ and 2λ, of which we select the smallest one for different ranges of λ. The difference between (5) and (6) is a result of the fact that we always select the solution with the maximal source set. Hence, for λ such that T^0 and T^1 are equally optimal, we select the one with a larger source set. When $T^0 = \emptyset$, T^0 is always preferred. When $T^0 = [1,n]$, T^1 is always preferred. The comparison for such situation between T^0 and T^2, and between T^1 and T^2 can be done similarly.

Note that when $2c_1 > c_0$, the second case in (5) does not exist since $c_1 > c_0 - c_1$. Hence, when $T^0 = \emptyset$ and $2c_1 > c_0$, the solution of $T_1(\lambda)$ is

$$T_1(\lambda) = \begin{cases} T^0, & \text{if } \lambda \geq \frac{c_0}{2} \\ T^2 & \text{otherwise} \end{cases} \quad (5.1)$$

Otherwise, when $T^0 = \emptyset$ and $2c_1 \leq c_0$,

$$T_1(\lambda) = \begin{cases} T^0, & \text{if } \lambda \geq c_0 - c_1 \\ T^1, & \text{if } c_1 < \lambda < c_0 - c_1 \\ T^2 & \text{if } \lambda \leq c_1 \end{cases} \quad (5.2)$$

For the case where $T^0 = [1,n]$, we can write the solutions of $T_1(\lambda)$ for when $2c_1 > c_0$ and $2c_1 \leq c_0$ in a similar way as (5.1) and (5.2), which we omit here due to the space limit.

Suppose the given weights $\{w_i\}_{i=1}^n$ and breakpoints $\{a_i\}_{i=1}^n$ result in the case where $T^0 = \emptyset$ and $2c_1 \leq c_0$, in which the solution of $T_1(\lambda)$ is (5.2). We divide the λ-range $[0,\infty)$ into three λ-ranges: $[0,c_1], (c_1, c_0 - c_1)$ and $[c_0 - c_1, \infty)$. We denote this set of λ-ranges by $\mathbf{\Lambda}_1$, i.e. $\mathbf{\Lambda}_1 = \{[0,c_1], (c_1, c_0 - c_1), [c_0 - c_1, \infty)\}$. Each λ-range in $\mathbf{\Lambda}_1$ has a different solution of $T_1(\lambda)$, according to (5.2).

When appropriate, we may use the notation of $T_1(\Lambda)$ in place of $T_1(\lambda)$ for $\lambda \in \Lambda \in \mathbf{\Lambda}_1$ to emphasize the fact that $T_1(\lambda)$ for all $\lambda \in \Lambda$ are identical.

The first iteration of PoS ends here as we obtain the set $\mathbf{\Lambda}_1$ and the minimum cut solutions of G_1 for λ-ranges in $\mathbf{\Lambda}_1$. In the next iteration of PoS, we solve for the minimum cuts of the graph G_2 for each λ-range in $\mathbf{\Lambda}_1$, which is then partitioned further according to their minimum cut solutions. The subsequent iterations of PoS will be elaborated in the next section.

Figure 4 illustrates $\mathbf{\Lambda}_0, \mathbf{\Lambda}_1$ and $\mathbf{\Lambda}_2$ which are the sets of λ-ranges that lead to different minimum cut solutions. Each λ-range in $\mathbf{\Lambda}_2$ corresponds to a particular sequence of minimum cut solution $\{T_0(\lambda), T_1(\lambda), T_2(\lambda)\}$.

The process continues until we complete the iteration n of the algorithm and obtain the set of λ-ranges $\mathbf{\Lambda}_n$. Each λ-range in $\mathbf{\Lambda}_n$ corresponds to a particular sequence of minimum cut solution $\{T_0(\lambda), T_1(\lambda), \ldots, T_n(\lambda)\}$, which corresponds to a particular solution of $\boldsymbol{x}^*(\lambda)$.

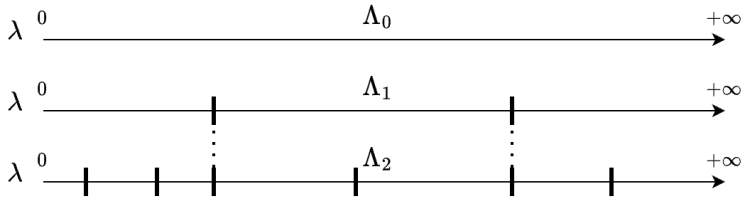

Figure 4: λ-Ranges according to the minimum cut solutions of $G_0(\lambda)$, $G_1(\lambda)$ and $G_2(\lambda)$.

3.2 Iteration k of the Path of Solutions Algorithm

Iteration k of the algorithm involves similar steps and has the same goal as in the first iteration, that is to partition each λ-range from the previous iteration according to their respective minimum cut solutions.

3.2.1 Finding the Minimum Cuts of $G_k(\lambda)$

By the end of iteration $k-1$, we have found the set of λ-ranges, $\mathbf{\Lambda}_{k-1}$, and $(S_{k-1}(\Lambda), T_{k-1}(\Lambda)$ for each $\Lambda \in \mathbf{\Lambda}_{k-1}$. Our goal in the iteration k is to find the minimum cut solution $(S_k(\lambda), T_k(\lambda))$ of the graph $G_k(\lambda)$ for λ in each $\Lambda \in \mathbf{\Lambda}_{k-1}$. This minimum cut solution might contain several cases depending on the value of λ, like the result in (5). These different cases will then divide the λ-range Λ into smaller λ-ranges.

If $i_k \in T_{k-1}(\lambda)$, then according to Lemma 2.4, the minimum cut solution $(S_k(\lambda), T_k(\lambda))$ is then equal to $(S_{k-1}(\lambda), T_{k-1}(\lambda))$. A more interesting and less trivial case is when $i_k \in S_{k-1}(\lambda)$.

We discussed toward the end of Section 2 that, based on Lemma 2.5, finding the minimum cut $(S_k(\lambda), T_k(\lambda))$ of $G_k(\lambda)$ is equivalent to finding the node status change interval in $S_{k-1}(\lambda)$ that move to $T_k(\lambda)$. According to Lemma 2.6, given the maximal s-interval containing i_k in $G_{k-1}(\lambda)$, $[i_{kl}, i_{kr}]$, we can find the node status change interval containing i_k, $[i_{k1}, i_{k2}]$, by solving problem (3) defined on $G_k(\lambda)$, restated here:

$$\min_{[i_{k1}, i_{k2}]} C(\{s\}, [i_{k1}, i_{k2}]) + C([i_{kl}, i_{kr}] \backslash [i_{k1}, i_{k2}], \{t\}) \quad (3)$$
$$+ C([i_{kl}, i_{kr}] \backslash [i_{k1}, i_{k2}], [i_{k1}, i_{k2}] \cup \{i_{kl} - 1, i_{kr} + 1\})$$
$$\text{s.t.} \quad i_k \in [i_{k1}, i_{k2}] \subseteq [i_{kl}, i_{kr}] \text{ or } [i_{k1}, i_{k2}] = \emptyset$$

For each $\Lambda \in \mathbf{\Lambda}_{k-1}$, we know its maximal s-interval containing i_k, $[i_{kl}, i_{kr}]$, from $(S_{k-1}(\Lambda), T_{k-1}(\Lambda))$. To solve for the node status change interval $[i_{k1}, i_{k2}]$, we write out the cost (3) for different feasible solutions of $[i_{k1}, i_{k2}] \subseteq [i_{kl}, i_{kr}]$, for general $\lambda \in [0, \infty)$. Similar to the computation of $T_1(\lambda)$ in Subsection 3.1, the cost functions for different possible cases of $[i_{k1}, i_{k2}]$ may have different coefficients of λ. These different coefficients of λ in (3) are contingent upon the positions of i_k and $[i_{k1}, i_{k2}]$, as well as $[i_{kl}, i_{kr}]$.

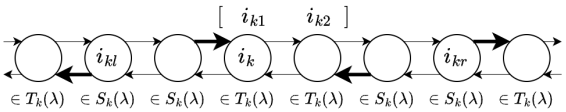

Figure 5: Example when $1 < i_{kl} < i_{k1} \leq i_k \leq i_{k2} < i_{kr} < n$. Here, we only show arcs of weight λ, which connect consecutive nodes.

Consider, for example, the most typical case of the maximal s-interval containing i_k, $[i_{kl}, i_{kr}]$, such that $i_{kl} \in (1, i_k)$ and $i_{kr} \in (i_k, n)$. Different cases of $[i_{k1}, i_{k2}]$ that are feasible solutions to problem (3) consist of:

Case 1. $i_{k1} \in (i_{kl}, i_k]$ and $i_{k2} \in [i_k, i_{kr})$, illustrated in Figure 5. The cost function (3) due to this case of $[i_{k1}, i_{k2}]$ can be written as $4\lambda + \sum_{i=i_{k1}}^{i_{k2}} w_i \cdot \mathbb{1}(a_i > a_{i_k}) + \sum_{i=i_{kl}}^{i_{k1}-1} w_i \cdot \mathbb{1}(a_i \leq a_{i_k}) + \sum_{i=i_{k2}+1}^{i_{kr}} w_i \cdot \mathbb{1}(a_i \leq a_{i_k})$. The first term is the cost due to arcs of weight λ. There are 4 such arcs of weights λ that connect nodes in $[i_{kl}, i_{kr}] \backslash [i_{k1}, i_{k2}]$, which remain in the source set $S_k(\lambda)$, to other nodes in the sink set $T_k(\lambda)$. The coefficients of λ is then equal to 4. These 4 arcs are displayed in bold in Figure 5.

The second term is the sum of the weights w for nodes in $[i_{k1}, i_{k2}]$ that are still connected to the node s. The sum of the third term and the fourth term is the sum of the weights w for nodes in $[i_{kl}, i_{k1} - 1] \cup [i_{k2} + 1, i_{kr}]$ that are connected to the node t.

The optimal node status change interval for Case 1 is $\arg\min_{[i_{k1}, i_{k2}]: i_{k1} \in (i_{k1}, i_k], i_{k2} \in [i_k, i_{kr})} \sum_{i=i_{k1}}^{i_{k2}} w_i \cdot \mathbb{1}(a_i > a_{i_k}) + \sum_{i=i_{kl}}^{i_{k1}-1} w_i \cdot \mathbb{1}(a_i \leq a_{i_k}) + \sum_{i=i_{k2}+1}^{i_{kr}} w_i \cdot \mathbb{1}(a_i \leq a_{i_k})$.

Case 2. There are two subcases.

Case 2.1. $i_{k1} = i_{kl}, i_{k2} \in [i_k, i_{kr})$. The cost is $2\lambda + \sum_{i=i_{kl}}^{i_{k2}} w_i \cdot \mathbb{1}(a_i > a_{i_k}) + \sum_{i=i_{k2}+1}^{i_{kr}} w_i \cdot \mathbb{1}(a_i \leq a_{i_k})$.

Case 2.2. $i_{k1} \in (i_{k1}, i_k], i_{k2} = i_{kr}$. The cost is equal to $2\lambda + \sum_{i=i_{k1}}^{i_{kr}} w_i \cdot \mathbb{1}(a_i > a_{i_k}) + \sum_{i=i_{kl}}^{i_{k1}-1} w_i \cdot \mathbb{1}(a_i \leq a_{i_k})$.

Case 3. $[i_{k1}, i_{k2}]$ is an empty interval, that is, $[i_{k1}, i_{k2}] \subseteq S_k(\lambda)$. The cost function can be written as $2\lambda + \sum_{i=i_{kl}}^{i_{kr}} w_i \cdot \mathbb{1}(a_i \leq a_{i_k})$.

Case 4. $i_{k1} = i_{kl}, i_{k2} = i_{kr}$. The cost is $\sum_{i=i_{kl}}^{i_{kr}} w_i \cdot \mathbb{1}(a_i > a_{i_k})$.

Let $[i_{k1}, i_{k2}]^\rho$ denote the node status change interval with the smallest cost among all s-intervals in $[i_{kl}, i_{kr}]$ whose coeffcients of λ in the cost are ρ, for $\rho = 0, 2, 4$. This is a similar notation as in Subsection 3.1. For $\rho = 0$, there is only one case, that is Case 4. For $\rho = 2$, we compare Case 2 and 3. For $\rho = 4$, there is only one case, Case 1. We do not provide explicit forms of $[i_{k1}, i_{k2}]^\rho$ here due to the space limit.

With the same method as presented in Subsection 3.1, we can write out the optimal solution of $[i_{kl}, i_{kr}]$ in the following form:

$$[i_{k1}, i_{k2}](\lambda) = \begin{cases} [i_{k1}, i_{k2}]^0, & \text{if } \lambda > \tau_2 \\ [i_{k1}, i_{k2}]^2, & \text{if } \tau_1 \leq \lambda \leq \tau_2 \\ [i_{k1}, i_{k2}]^4 & \text{if } \lambda \leq \tau_1 \end{cases} \quad (7)$$

(see the next paragraph for the solution when $\lambda = \tau_1$*)* where τ_1 and τ_2 are the threshold values that split $\lambda \in [0, \infty)$ into ranges. τ_1 and τ_2 can be obtained by comparing the objective functions (3) due to $[i_{k1}, i_{k2}]^0, [i_{k1}, i_{k2}]^2$and $[i_{k1}, i_{k2}]^4$.

Remark here that when $\lambda = \tau_1$, we select between $[i_{k1}, i_{k2}]^2$ and $[i_{k1}, i_{k2}]^4$ the solution that results in a larger source set, as it depends on the problem instance. When $\lambda = \tau_2$, we always prefer $[i_{k1}, i_{k2}]^2$ over $[i_{k1}, i_{k2}]^0 = [i_{kl}, i_{kr}]$.

We see for this example of $[i_{kl}, i_{kr}]$, where $i_{kl} \in (1, i_k)$ and $i_{kr} \in (i_k, n)$, that there can be up to 3 different solutions, as shown in (7), due to 3 different coefficients of λ in the cost function across all possible cases of $[i_{k1}, i_{k2}]$. These 3 different solutions correspond to 3 λ-ranges, $[0, \tau_1], [\tau_1, \tau_2]$ and (τ_2, ∞).

For other types of the maximal s-interval containing i_k, $[i_{kl}, i_{kr}]$, we list all possible coefficients of λ and the number of solutions of the node status change interval, $[i_{k1}, i_{k2}]$, in Table 1.

We have shown here the procedure to find the node status change interval $[i_{k1}, i_{k2}]$ from $G_{k-1}(\lambda)$ to $G_k(\lambda)$, for any $\lambda \geq 0$, given a maximal s-interval containing i_k, $[i_{kl}, i_{kr}]$, which can be obtained from the minimum cut $(S_{k-1}(\lambda), T_{k-1}(\lambda))$. Next, we show how we apply this procedure to partition each λ-ranges in $\mathbf{\Lambda}_{k-1}$ and get the set $\mathbf{\Lambda}_k$ to complete the iteration k of PoS.

3.2.2 Partitioning of λ-Ranges

The implication of the procedure in Subsection 3.2.1 is that λ-ranges in $\mathbf{\Lambda}_{k-1}$ that have the same maximal s-interval containing i_k in G_{k-1} also have the same node status change interval solution. Therefore, the final step of the iteration k of PoS is to group λ-ranges in $\mathbf{\Lambda}_{k-1}$ based on their maximal s-intervals containing i_k. For each group, with a particular maximal s-intervals containing i_k, we compute for the node status change interval, like in Subsection 3.2.1, and apply the corresponding solution to λ-ranges in that group.

Table 1: Different types of the maximal s-intervals $[i_{kl}, i_{kr}]$ containing i_k in the minimum cut of G_{k-1}, considered in the k-th iteration of PoS. For each type of $[i_{kl}, i_{kr}]$, we list all possible coefficients of λ in the objective function (3).

Types of $[i_{kl}, i_{kr}]$: $i_k \neq 1, n$ **and ...**	**Coef. of** λ	**# cases**
$i_{kl} \in (1, i_k), i_{kr} \in (i_k, n)$	0, 2, 4	3
$i_{kl} \in (1, i_k), i_{kr} = n$	0, 1, 2, 3	4
$i_{kl} = 1, i_{kr} \in (i_k, n)$	0, 1, 2, 3	4
$i_{kl} = 1, i_{kr} = n$	0, 1, 2	3
$i_{kl} = i_k, i_{kr} = n$	0, 2	2
$i_{kl} = 1, i_{kr} = i_k$	0, 2	2
$i_{kl} = i_k, i_{kr} \in (i_k, n)$	0, 1	2
$i_{kl} \in (1, i_k), i_{kr} = i_k$	0, 1	2
$i_{kl} = i_k, i_{kr} = i_k$	0, 2	2
Types of $[i_{kl}, i_{kr}]$: $i_k \in \{1, n\}$ **and ...**	**Coef. of** λ	**# cases**
$i_{kl}, i_{kr} \in \{1, n\}$	0, 1	2
$i_{kl} = i_k = 1, i_{kr} \in (1, n)$	0, 1, 2	3
$i_{kl} \in (1, n), i_{kr} = i_k = n$	0, 1, 2	3

For example, suppose the λ-ranges depicted by rounded rectangular boxes in Figure 6(b) are λ-ranges in $\mathbf{\Lambda}_{k-1}$ that have the same maximal s-interval containing i_k, and their node status change interval solution, in the form similar to (7), is shown in Figure 6(a). τ_1, τ_2 and τ_3 are threshold values that divide $[0, \infty)$ into 4 segments, each corresponds to a particular node status change interval solution. Here, we show the case with 4 solutions, separated by 3 thresholds, which is the maximum number possible (see Table 1). This is a different case from (7) where we have 3 solutions.

Then, we intersect these λ-ranges, which have the same maximal s-interval containing i_k (Figure 6(b)), with the ranges of λ from the node status change interval solution (Figure 6(a)). The result is shown in Figure 6(c) where some λ-ranges are split into multiple λ-ranges by threshold values of λ that belong to the ranges. Each of these resulting λ-ranges takes the node status change interval solution from the segments of λ-values (Figure 6(a)) that it belongs to, as written in the rectangular boxes in Figure 6(c). For each of these λ-ranges, say a λ-range Λ, with the node status change interval solution $[i_{kl}, i_{kr}](\Lambda)$, its minimum cut in G_k is $(S_k(\Lambda), T_k(\Lambda))$ where $T_k(\Lambda)$ is $T_{k-1}(\Lambda) \cup [i_{kl}, i_{kr}](\Lambda)$.

After we perform this step on all groups of λ-ranges that share the same maximal s-intervals containing i_k in G_{k-1}, we obtain a set of λ-ranges spanning $[0, \infty)$, which together form the set $\mathbf{\Lambda}_k$.

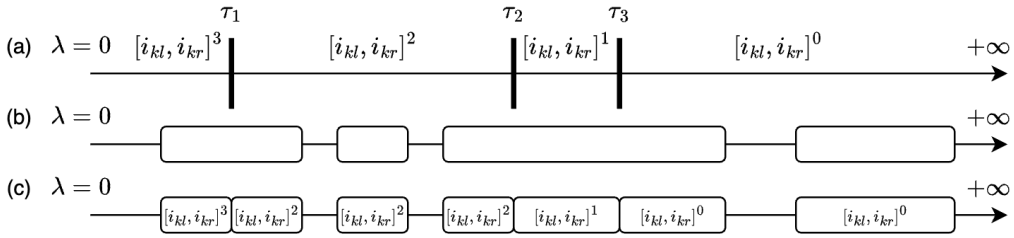

Figure 6: At iteration k, we group λ-ranges that have the same maximal s-interval containing i_k in the minimum cut of G_{k-1} together. Let λ-ranges in (b) be an example of such group of λ-ranges for a specific maximal s-interval containing i_k. The node status change interval solution can be computed accordingly for this s-interval. An example of such solution is illustrated in (a). This solution (a) is then applied onto λ-ranges in (b), resulting in a set of λ-ranges in (c), with their node status change intervals solutions written in their respective rectangular boxes.

3.3 Outputs of the PoS Algorithm

The Path of Solutions algorithm divides the range of nonnegative λ values, $[0,\infty)$, into a set of smaller ranges, which are then subsequently divided into smaller ranges in each iteration.

At the end of PoS, we obtain the set of λ-ranges, $\mathbf{\Lambda}_n$. We show in Section 5 that the number of λ-ranges in $\mathbf{\Lambda}_n$ is at most n^3. For an interval Λ in $\mathbf{\Lambda}_n$, we have an associated sequence of graph cuts $\{(S_k(\Lambda), T_k(\Lambda))\}_{k=0}^n$. We then obtain the solution $\boldsymbol{x}^*(\lambda)$ for $\lambda \in \Lambda$ based on the HL-algorithm by setting $x_i^*(\lambda)$ to a_{i_k}, the k-th smallest breakpoint, for the index k such that $i \in S_{k-1}(\Lambda)$ and $i \in T_k(\Lambda)$.

PoS solves ℓ_1-PFL in $O(n^2 q \log n)$. We show this in Section 5 as well.

4 PATH OF SOLUTIONS ALGORITHM FOR CONVEX PIECEWISE LINEAR FIDELITY FUNCTIONS

Here, we address the path of solutions for a general fused lasso problem with convex piecewise linear fidelity function, or PFL (1):

$$\text{(PFL)} \quad \min_{x_1,\ldots,x_n} \sum_{i=1}^{n} f_i^{pl}(x_i) + \lambda \sum_{i=1}^{n-1} |x_i - x_{i+1}| \qquad (1)$$

The extension from ℓ_1-PFL to PFL can be done in a similar way as the extension of the HL-algorithm on GIMR from the case of ℓ_1-fidelity function to convex piecewise linear fidelity functions, given by Hochbaum and Lu (2017). We give a brief description here:

Let the number of breakpoints of $f_i^{pl}(x_i)$ be q_i, and they are denoted by $a_{i,1}, a_{i,2}, \ldots, a_{i,q_i}$. The total number of breakpoints is $\sum_{i=1}^n q_i$, denoted by q. We sort the q breakpoints in the ascending order with the indices $(i_1, j_1), (i_2, j_2), \ldots (i_q, j_q)$ such that $a_{i_1,j_1} < a_{i_2,j_2} < \ldots < a_{i_q,j_q}$.

The PoS algorithm applied on PFL involves the computation of the minimum cuts of graphs $G_0, \ldots, G_q$, for different λ-ranges, which is done in a similar manner as when PoS is applied on ℓ_1-PFL in Section 3. The difference here comes from the update of G_{k-1} to G_k, which is described in Section 2. The differences between the two graphs are the weights of the source and sink adjacent arcs of node i_k. In G_k, $w_{s,i_k} = max(-(f_i^{pl})'(a_{i_k}), 0)$ and $w_{i_k,t} = max((f_i^{pl})'(a_{i_k}), 0)$. The maximal s-interval of interest when finding the node status change interval in iteration k is the maximal s-interval containing i_k, similar to when we solve ℓ_1-PFL.

5 BOUND OF THE NUMBER OF SOLUTIONS AND TIME COMPLEXITY

We first provide the following lemma and theorem that lead to a conclusion that the number of different solutions to PFL across all nonnegative λ is equal to the number of λ-ranges in $\mathbf{\Lambda}_n$.

Lemma 5.1. *The set of λ-ranges obtained from the iteration k of PoS, $\mathbf{\Lambda}_k$, has the following property: for two different nonnegative values of λ, λ_a and λ_b, they belong to the same λ-range in $\mathbf{\Lambda}_k$ if and only if $T_j(\lambda_a) = T_j(\lambda_b)$ for all $j = 0, 1, ..., k$.*

Proof. Suppose $T_j(\lambda_a) \neq T_j(\lambda_b)$ for some $j \in \{0, 1, ..., k\}$. Let h be the smallest index such that $T_h(\lambda_a) \neq T_h(\lambda_b)$. We know that $h \geq 1$ since $T_0(\lambda_a) = T_0(\lambda_b) = \{\}$. It follows that $T_j(\lambda_a) = T_j(\lambda_b)$ for all $j \leq h-1$. Hence, λ_a and λ_b are in the same λ-range in $\mathbf{\Lambda}_{h-1}$. Since $T_h(\lambda_a) \neq T_h(\lambda_b)$ and $T_{h-1}(\lambda_a) = T_{h-1}(\lambda_b)$, the node status change interval going from G_{h-1} to G_h for λ_a, which is $T_h(\lambda_a) \backslash T_{h-1}(\lambda_a)$, and

that for λ_b, which is $T_h(\lambda_b) \backslash T_{h-1}(\lambda_b)$, must be different. After the partition of their common λ-range in $\mathbf{\Lambda}_{h-1}$, they must be in separate λ-ranges in $\mathbf{\Lambda}_h$. Since any two λ-ranges never merge, they must also be in different λ-intervals in $\mathbf{\Lambda}_k$. Hence, we have proven that if λ_a and λ_b are in the same λ-range in $\mathbf{\Lambda}_k$ then $T_j(\lambda_a) = T_j(\lambda_b)$ for all $j = 0, 1, ..., k$.

For the opposite direction of the proof, suppose λ_a and λ_b are not in the same λ-range in $\mathbf{\Lambda}_k$. Let h be the largest index such that λ_a and λ_b are in the same λ-range in $\mathbf{\Lambda}_h$ (there exists such h because λ_a and λ_b are in the same range of $[0,\infty) \in \mathbf{\Lambda}_0$). From the proof of the first direction, it follows that $T_h(\lambda_a) = T_h(\lambda_b)$. Since λ_a and λ_b are not in the same λ-range in $\mathbf{\Lambda}_{h+1}$, it implies that their node status change intervals, denoted by $[i_{k1}, i_{k2}](\lambda_a)$ and $[i_{k1}, i_{k2}](\lambda_b)$ are different. Hence, $T_{h+1}(\lambda_a)$, which is equal to $T_h(\lambda_a) \cup [i_{k1}, i_{k2}](\lambda_a)$, and $T_{h+1}(\lambda_b)$, which is equal to $T_h(\lambda_b) \cup [i_{k1}, i_{k2}](\lambda_b)$, must be different. □

Theorem 5.2. *The solutions of PFL (1) for two different values of λ, $\boldsymbol{x}^*(\lambda_a)$ and $\boldsymbol{x}^*(\lambda_b)$, are equal if and only if λ_a and λ_b are in the same λ-range in Λ_n.*

Proof. It follows from Lemma 5.1 that λ_a and λ_b are in the same λ-range in Λ_n if and only if $T_j(\lambda_a) = T_j(\lambda_b)$ for all $j = 0, 1, ..., n$.

The threshold theorem implies that, if $T_j(\lambda_a) = T_j(\lambda_b)$ for $j = 0, \ldots, n$ then $\boldsymbol{x}^*(\lambda_a) = \boldsymbol{x}^*(\lambda_b)$.

If $T_j(\lambda_a) \neq T_j(\lambda_b)$ for some $j \in \{0, \ldots, n\}$, then there is a node i such that $i \in T_j(\lambda_a)$ but $i \notin T_j(\lambda_b)$ (or $i \notin T_j(\lambda_a)$ but $i \in T_j(\lambda_b)$). Suppose $i \in T_j(\lambda_a)$ but $i \notin T_j(\lambda_b)$. It follows that $x_i^*(\lambda_a) \leq a_j < x_i^*(\lambda_b)$. Hence, $\boldsymbol{x}^*(\lambda_a) \neq \boldsymbol{x}^*(\lambda_b)$. □

The bounds of the number of solutions to ℓ_1-PFL and PFL across all $\lambda \geq 0$ are given in Theorem 5.3 and 5.4.

Theorem 5.3. *The number of different optimal solutions of ℓ_1-PFL (4) across all λ values is at most $\frac{n^3+3n^2+2n+2}{2}$, which is smaller than n^3 when $n \geq 4$.*

Proof. Theorem 5.2 implies that the number of solutions is equal to the number of λ-ranges in Λ_n.

The iteration k of the algorithm involves dividing λ-ranges in $\mathbf{\Lambda}_{k-1}$ into groups based on the maximal s-interval containing i_k in G_{k-1} and then solving for the node status change interval solution for each maximal s-interval containing i_k.

The node status change interval solution for each maximal s-interval containing i_k may consist of multiple solutions, similar to how (7) consists of multiple cases. If there are p cases of the node status change interval solution, there will be $p-1$ λ-threshold values that cut into the set of λ-ranges that have the same maximal s-interval containing i_k. In Figure 6, for example, $p = 4$. In (7), $p = 3$. Each threshold contributes to an increase of at most one additional λ-ranges. In Figure 6(c), there is an increase of 3 λ-ranges from Figure 6(b) after we use the λ-threshold values to split the λ-ranges in Figure 6(b).

Our summary in Table 1 shows that the number of cases of the node status change interval solution for any type of the maximal s-interval is at most 4. This implies that for a set of λ-ranges in $\mathbf{\Lambda}_{k-1}$ that have the same maximal s-interval containing i_k, there will be at most 3 threshold values of λ that cut through them and the increase in the number of λ-ranges in $\mathbf{\Lambda}_k$ due to this maximal s-interval can be at most 3.

Since the number of s-intervals that contain i_k, or the number of $[i_{kl}, i_{kr}]$ such that $1 \leq i_{kl} \leq i_k \leq i_{kr} \leq n$ is $i_k(n - i_k + 1)$, the increase in the number of λ-ranges from $\mathbf{\Lambda}_{k-1}$ to $\mathbf{\Lambda}_k$ is at most $3i_k(n - i_k + 1)$.

$\mathbf{\Lambda}_0 = \{[0,\infty)\}$ has only one λ-range. After n iterations, in $\mathbf{\Lambda}_n$, the number of λ-ranges is at most $1 + \sum_{i_k=1}^{n} 3i_k(n - i_k + 1) = 1 + \frac{n(n+1)(n+2)}{2} = \frac{n^3+3n^2+2n+2}{2}$. For $n \geq 4$, this bound is smaller than n^3. □

Theorem 5.4. *The number of different optimal solutions of PFL (1), with q breakpoints, across all non-negative λ is at most $1 + \frac{3q(n+1)^2}{4}$, which is smaller than $n^2 q$ for $n \geq 7$.*

Proof. The proof is similar to that of Theorem 5.3. In each graph update, or each iteration, say iteration k of the algorithm, the increase in the number of λ-ranges in $\mathbf{\Lambda}_k$ from $\mathbf{\Lambda}_{k-1}$ can be at most 3. When solving PFL with q breakpoints, the number of iterations is q, as we update the graph from G_0 to G_1, and so forth, until G_q.

Suppose the k-th smallest breakpoint comes from the i-th fidelity function, f_i^{pl}. At iteration k, we do the computation on the maximal s-interval that contains node i. The increase in the number of λ-ranges in this iteration is then at most $3i(n - i + 1)$.

There are q_i iterations where we consider the maximal s-interval that contains the node i. Hence, this increase of at most $3i(n - i + 1)$ happens q_i times. This implies that the bound of the number of λ-ranges in $\mathbf{\Lambda}_q$ is $1 + \sum_{i=1}^{n} 3q_i i(n - i + 1)$.

When n is an odd number, $\max_{i=1,\ldots,n} i(n - i + 1) = \frac{(n+1)^2}{4}$. For an even number n, $\max_{i=1,\ldots,n} i(n - i + 1) = \frac{n^2+2n}{4} < \frac{(n+1)^2}{4}$. Hence, the bound of the number of λ-ranges in $\mathbf{\Lambda}_q$ is at most $1 + \sum_{i=1}^{n} 3q_i \frac{(n+1)^2}{4} = 1 + \frac{3q(n+1)^2}{4}$, which is at most $n^2 q$ for $n \geq 7$. □

Theorem 5.5. *PFL (1) is solved by PoS in*

Table 2: The number of solutions found by our path of solutions algorithm and Gurobi as well as the computation times, taken from one of the 10 experiments. For Gurobi, we vary the resolution of the list of values of λ that we provide to the solver.

	Gurobi								PoS
Resolution (Δ)	0.1	0.05	0.025	0.01	0.005	0.0025	0.001	0.0005	
# Solutions	24	36	53	77	93	101	104	108	114
Time (s)	0.45	0.82	1.63	4.10	8.19	16.43	40.72	83.55	0.42

Table 3: Average percentage of solutions found by Gurobi compared to those found by our path of solutions algorithm as well as the average relative computation time of Gurobi compared to the path of solutions algorithm, across 10 runs.

	Gurobi								PoS
Resolution (Δ)	0.1	0.05	0.025	0.01	0.005	0.0025	0.001	0.0005	
Avg % solutions found	23%	34%	48%	69%	81%	88%	95%	97%	100%
Avg runtime compared to PoS	1.22	2.40	4.78	11.92	23.72	47.47	119.21	237.05	1

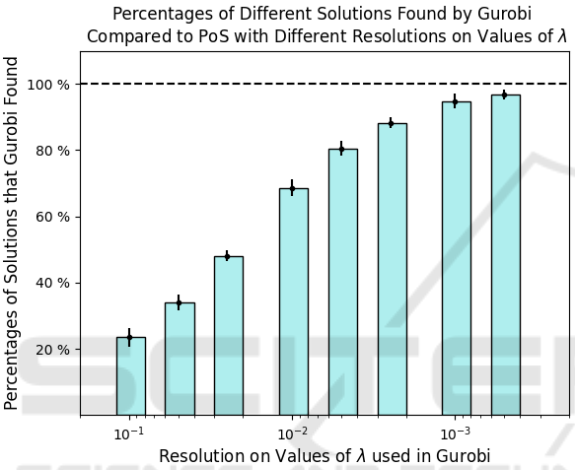

Figure 7: Percentages of different solutions of ℓ_1-PFL found by Gurobi with varying resolutions, averaged across 10 experiments. Error bars of 1 standard deviation are displayed atop the bar plots. PoS can find all solutions across all λ, indicated by the dashed line at the 100% level.

$O(n^2 q \log n)$ time where q is the number of breakpoints of n convex piecewise linear fidelity functions.

Proof. Suppose the k-th smallest breakpoint comes from the i-th fidelity function, f_i^{pl}. At iteration k of the algorithm, we compute for the node status change interval for each group of λ-ranges in $\mathbf{\Lambda}_{k-1}$ that have the same maximal s-interval containing i. There are $i(n-i+1)$ such maximal s-intervals. This is also the number of the node status change interval computation at iteration k since we do one computation for one maximal s-interval.

With the same reasoning as in the proof of Theorem 5.4, the number of the computation for node status change interval across q iterations is $\sum_{i=1}^{n} q_i\, i(n-i+1)$, which is bounded by $\frac{n^2 q}{3}$ as shown earlier.

Hochbaum and Lu (2017) provided a method that performs a computation of the node status change interval in $O(\log n)$. By relying on the same method and the data structures used in that work, the total time complexity due to the node status change interval computation throughout the algorithm is $O(n^2 q \log n)$.

Other steps such as the sorting of the breakpoints as well as the updates in the data structures are dominated by the node status change interval computation time. Therefore, the time complexity of PoS in solving PFL for all $\lambda \geq 0$ is $O(n^2 q \log n)$. □

6 EXPERIMENTS

To demonstrate the performance of PoS in finding different solutions for all $\lambda \geq 0$, we compare it to Gurobi, a state-of-the-art linear programming solver (Gurobi Optimization, LLC, 2023). We implemented PoS in Python and run both PoS and Gurobi on the same laptop with Apple M2 CPU, 16GB RAM.

We evaluate both PoS and Gurobi on ℓ_1-PFL (4) where the number of samples, n, is 100, the weights $w_1, \ldots, w_n$ and the breakpoints $a_1, \ldots, a_n$ are sampled from a uniform distribution between 0 and 1.

PoS generates the solutions for all λ. However, Gurobi solves the problem only for a specific value of λ, which needs to be specified, one at a time. We provide a list of values of λ to Gurobi: $[0, \Delta, 2\Delta, \ldots, U]$. Δ is the resolution for the grid search on the values of λ, and U is the upper bound of λ. We take the value of U from the result of PoS. We vary the resolution Δ and report the result for all resolutions that we tested.

We run the experiment 10 times. In each run, the weights and breakpoints are resampled. We measure the computation time of PoS and Gurobi as well as the number of different solutions that they found. The results for one particular run, as an example, are reported in Table 2. In this table, we report the number of different solutions found by both algorithms and their runtimes. The reported number of solutions for

PoS is the true number of different solutions since it finds all solutions for all nonnegative λ.

As shown in Table 2, Gurobi, with a low resolution (large Δ), fails to find some solutions that correspond to λ not included in the list. With $\Delta = 0.1$, both PoS and Gurobi have about the same computation time. However, Gurobi found only 24 different solutions while PoS was able to produce all 114 different solutions. As Δ decreases, the list of λ for Gurobi becomes finer. Gurobi is able to find more different solutions, but it also takes longer time. At the resolution of 0.0005, Gurobi found 108 solutions, which is close to the total number of different solutions. However, it takes Gurobi more than 80 seconds while PoS can achieve a better result in 0.42 seconds.

In Table 3 and Figure 7, we report the percentages of solutions found by Gurobi compared to PoS, averaged across 10 runs. A reported number of 100% implies that we find all possible solutions. We also report the average relative runtime of Gurobi compared to PoS, where the reported number of *a* for Gurobi implies the runtime that is *a* times of that of PoS.

When we set the resolution for Gurobi at 0.1, Gurobi and PoS take approximately the same amount of time. However, with this resolution, Gurobi only found about 23% of all possible solutions while PoS can find all of them. At the finest resolution included in the experiment, which is 0.0005, Gurobi can find 97% of all solutions. However, it takes more than 200 times longer than PoS.

7 CONCLUSIONS

We provide in this work an efficient minimum cut-based algorithm called Path of Solutions, or PoS, that generates the solutions of the convex piecewise linear fused lasso problem (1) for all values of the tradeoff parameter, λ. PoS is a more efficient alternative to the traditional method of parameter tuning, in which a single parameter value is evaluated one at a time. In traditional parameter tuning, we might unknowingly overlook some important values or test two redundant values that lead to the same result. These challenges are overcome by PoS.

In addition to the algorithm design, we provide both the time complexity of the algorithm and the bound of the number of different solutions across all nonnegative λ in terms of the number of variables and the number of breakpoints in the fidelity functions.

We demonstrate the efficiency of PoS via a set of experiments in comparison with Gurobi, a state-of-the-art solver for mathematical programming. PoS is capable of generating all solutions for all λ while Gurobi found only a small fraction in the same amount of time. To find all solutions, Gurobi requires more than a factor of 200 of the time that PoS needs.

Regarding future directions of this work, we are interested in extending our algorithm to solve other variants of the fused lasso problems as well as conducting more experiments on synthetic and real data.

ACKNOWLEDGEMENTS

This research was supported in part by AI Institute NSF Award 2112533.

REFERENCES

Eilers, P. H. and De Menezes, R. X. (2005). Quantile smoothing of array cgh data. *Bioinformatics*, 21(7):1146–1153.

Gallo, G., Grigoriadis, M. D., and Tarjan, R. E. (1989). A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18(1):30–55.

Gurobi Optimization, LLC (2023). Gurobi Optimizer Reference Manual.

Hochbaum, D. S. (2001). An efficient algorithm for image segmentation, markov random fields and related problems. *Journal of the ACM (JACM)*, 48(4):686–701.

Hochbaum, D. S. (2008). The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Operations research*, 56(4):992–1009.

Hochbaum, D. S. and Lu, C. (2017). A faster algorithm solving a generalization of isotonic median regression and a class of fused lasso problems. *SIAM Journal on Optimization*, 27(4):2563–2596.

Hoefling, H. (2010). A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics*, 19(4):984–1006.

Storath, M., Weinmann, A., and Unser, M. (2016). Exact algorithms for l^1-tv regularization of real-valued or circle-valued signals. *SIAM Journal on Scientific Computing*, 38(1):A614–A630.

Tibshirani, R. J. (2011). *The solution path of the generalized lasso*. Stanford University.

Wang, L., Gordon, M. D., and Zhu, J. (2006). Regularized least absolute deviations regression and an efficient algorithm for parameter tuning. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 690–700. IEEE.