technische universität dortmund

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

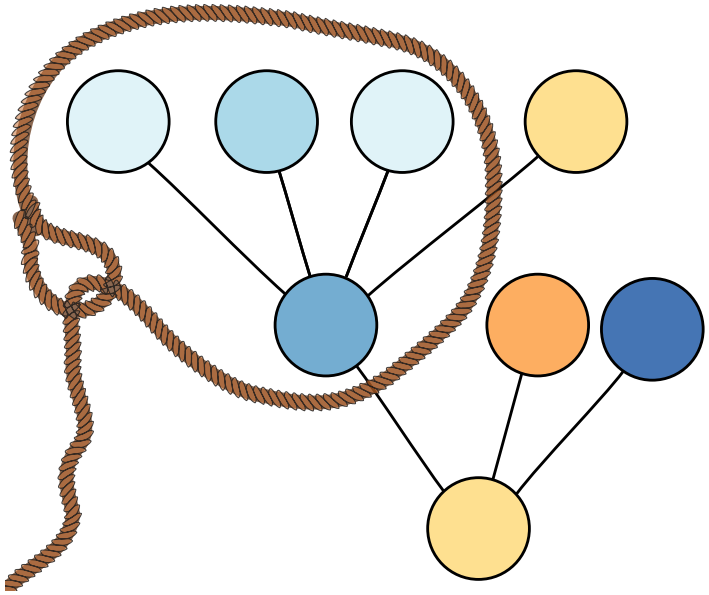
Engineering Fused Lasso Solvers on Trees

Elias Kuthe[1] Sven Rahmann[2]

SEA 2020

[1] Bioinformatics
Computer Science XI
TU Dortmund University Germany

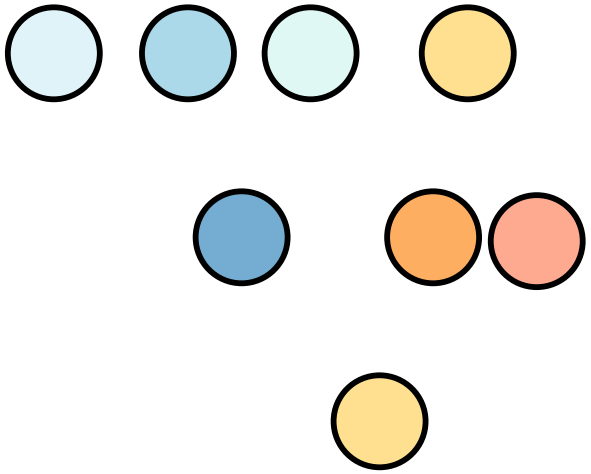
[2] Genome Informatics
Institute of Human Genetics
University of Duisburg-Essen
Germany



github.com/**eqt/treelas**

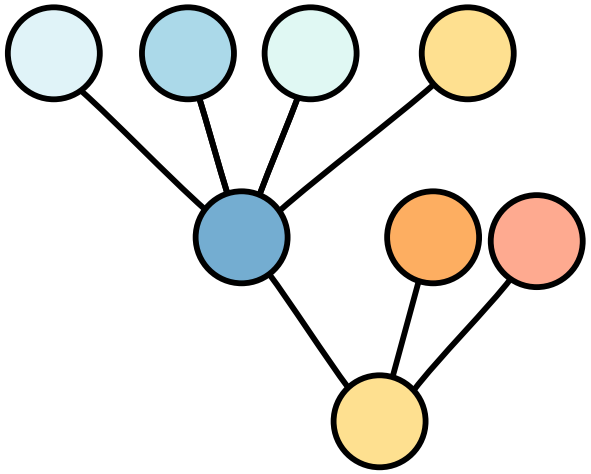
Fused Lasso Signal Approximator [Tibshirani et al (2005)]

Input Signal $y \in \mathbb{R}^{|V|}$,



Fused Lasso Signal Approximator [Tibshirani et al (2005)]

Input Signal $y \in \mathbb{R}^{|V|}$,
Graph $G = (V, E)$,
Parameter $\lambda > 0$

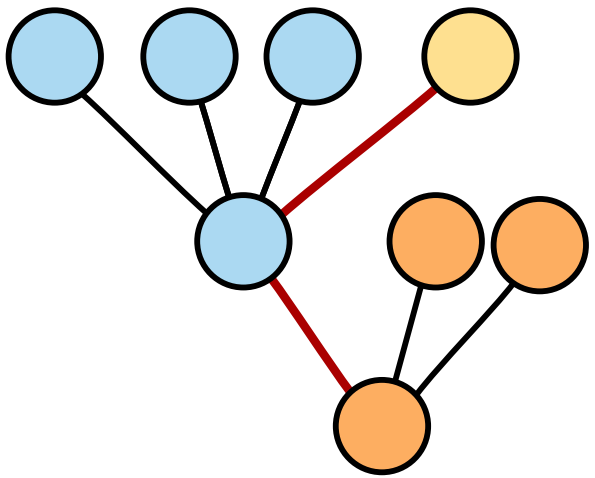


Find **Minimizer** $x^* \in \mathbb{R}^{|V|}$ of

$$f(x) := \sum_{i \in V} (y_i - x_i)^2 + \sum_{\{i,j\} \in E} \lambda \left| x_i - x_j \right|$$

Fused Lasso Signal Approximator [Tibshirani et al (2005)]

Input Signal $y \in \mathbb{R}^{|V|}$,
Graph $G = (V, E)$,
Parameter $\lambda > 0$



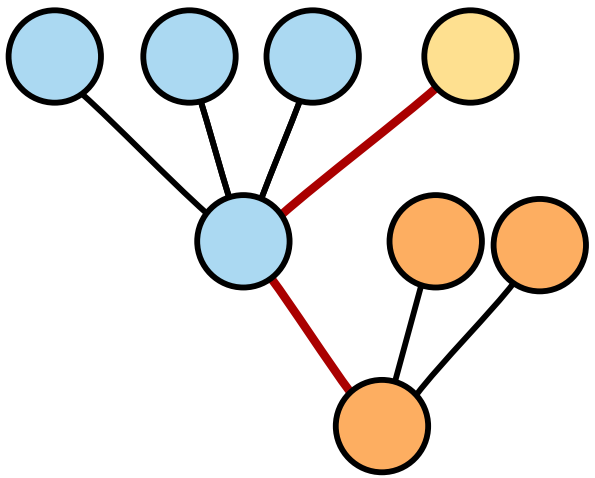
Find **Minimizer** $x^* \in \mathbb{R}^{|V|}$ of

$$f(x) := \sum_{i \in V} (y_i - x_i)^2 + \sum_{\{i,j\} \in E} \lambda \left| x_i - x_j \right|$$

Fused Lasso Signal Approximator [Tibshirani et al (2005)]

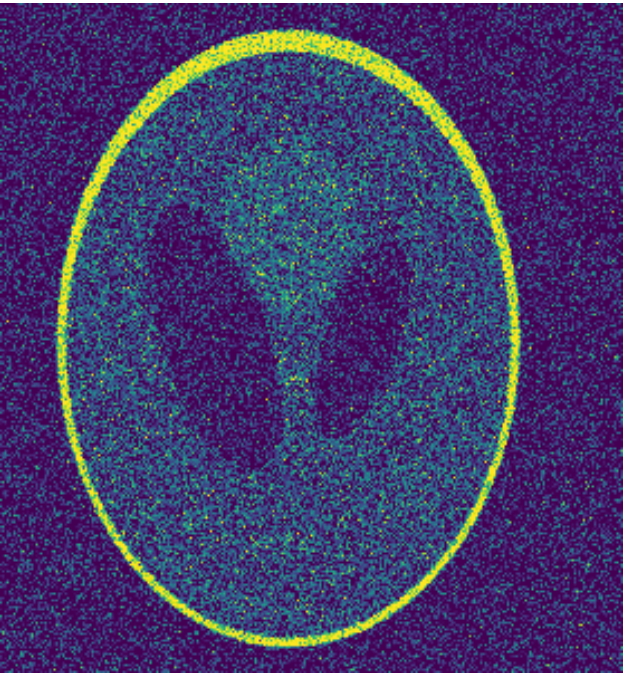
Input Signal $y \in \mathbb{R}^{|V|}$,
Graph $G = (V, E)$,
Parameter(s) $\lambda > 0$
Node weights $\mu \geqslant 0$.

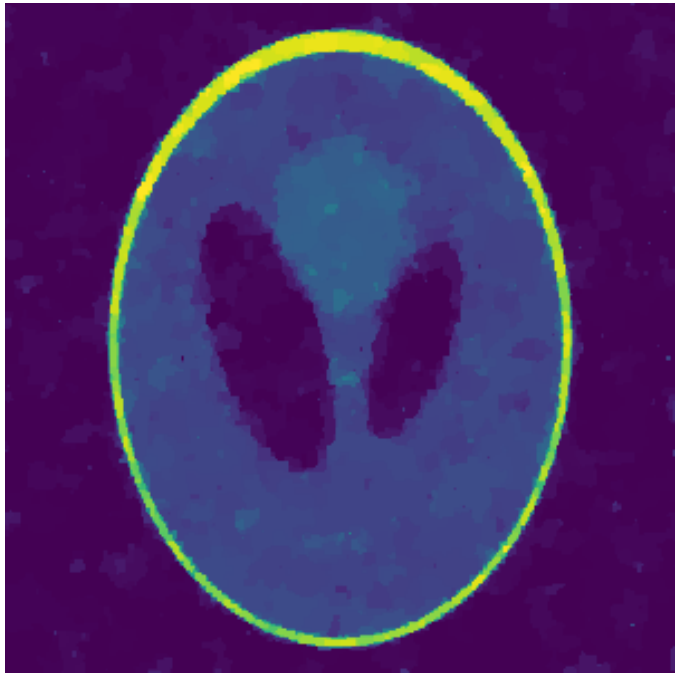
Find **Minimizer** $x^* \in \mathbb{R}^{|V|}$ of

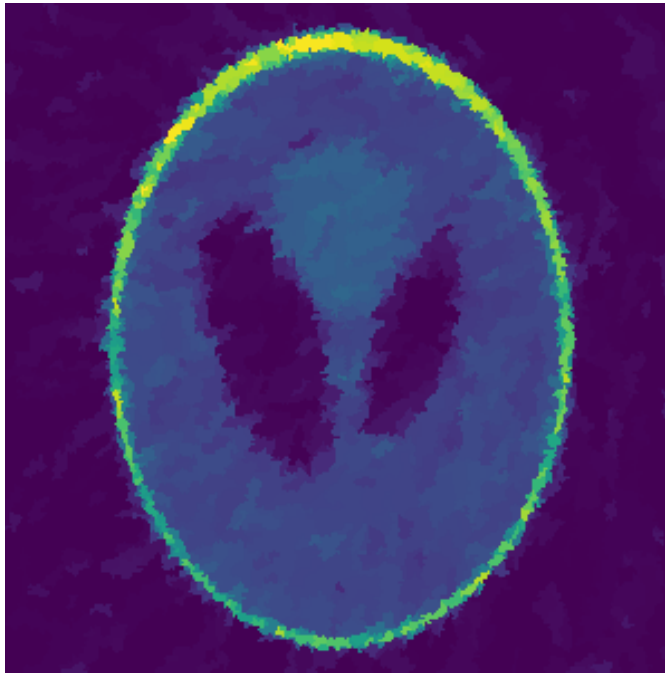


$$f(x) := \sum_{i \in V} \mu_i (y_i - x_i)^2 + \sum_{\{i,j\} \in E} \lambda_{ij} \left| x_i - x_j \right|$$

Applications



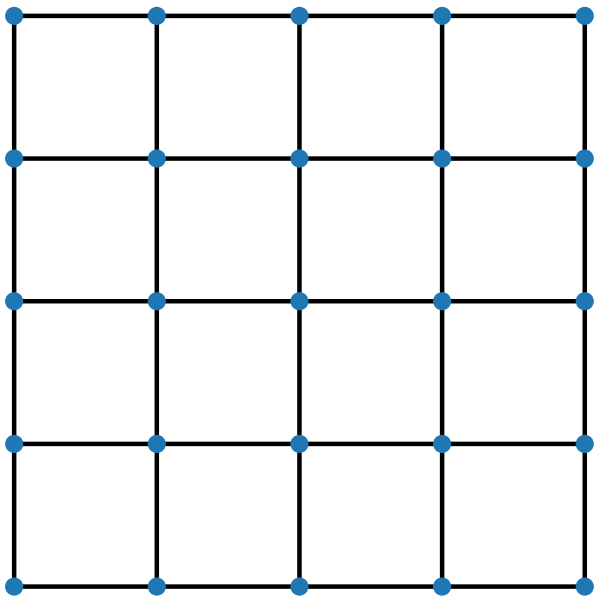




- Denoising
- Segmentation
- Clustering/Grouping

Related Work

Beck & Teboulle (2009) Fast Subgradient Descent (FISTA) on the Dual

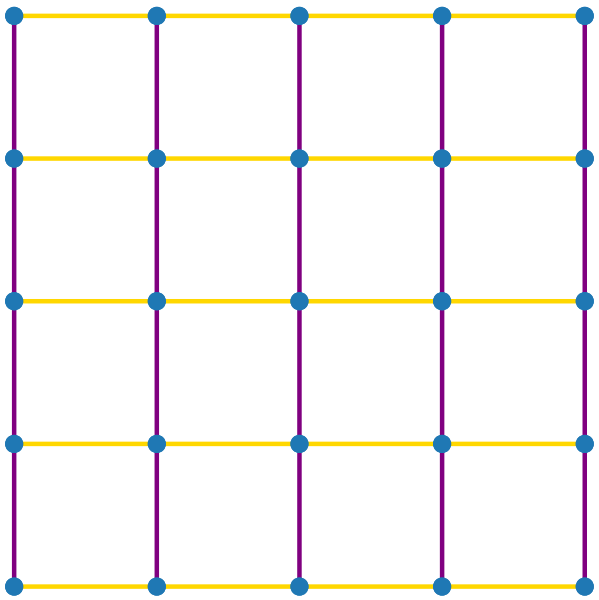


Related Work

Beck & Teboulle (2009) Fast Subgradient Descent (FISTA) on the Dual

Johnson (2013) Line graphs in $\mathcal{O}(n)$

Tansey & Scott (2015) Leverage Johnson's algorithm by Consensus ADMM



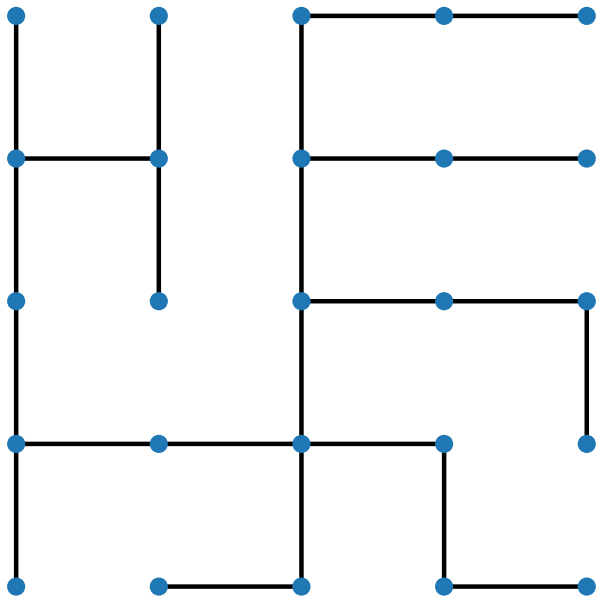
Related Work

Beck & Teboulle (2009) Fast Subgradient Descent (FISTA) on the Dual

Johnson (2013) Line graphs in $\mathcal{O}(n)$

Tansey & Scott (2015) Leverage Johnson's algorithm by Consensus ADMM

Kolmogorov, Pock & Rolinek (2016) Tree graphs in $\mathcal{O}(n \log n)$



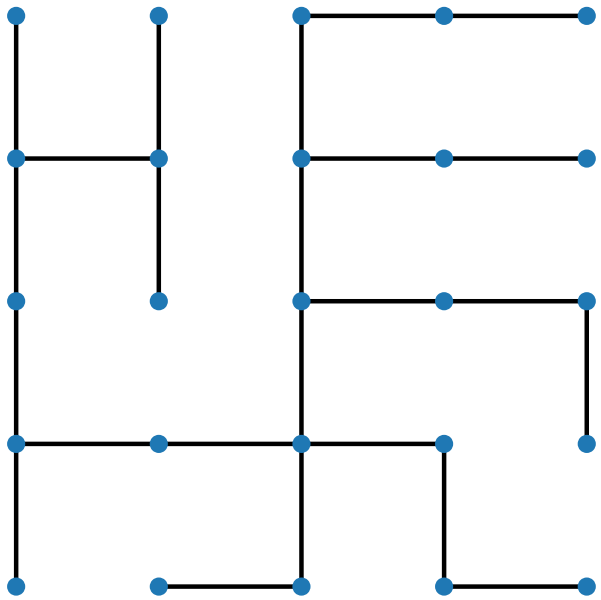
Related Work

Beck & Teboulle (2009) Fast Subgradient Descent (FISTA) on the Dual

Johnson (2013) Line graphs in $\mathcal{O}(n)$

Tansey & Scott (2015) Leverage Johnson's algorithm by Consensus ADMM

Kolmogorov, Pock & Rolinek (2016) Tree graphs in $\mathcal{O}(n \log n)$



Fused Lasso On Trees

"However, dynamic programming on a tree is

(a) ***much more complex*** *than dynamic programming on a chain (since it relies on sophisticated data structures), and*

(b) ***noticeably slower in practice*** *than dynamic programming over a chain, especially for large problem sizes."*

[Padilla, Sharpnack & Scott (2017)]

Overview

1. Exact Algorithm
2. Approximation Scheme
3. Experiments

Min-Convolution

$\lambda \geqslant 0$

$$\boxed{h(x) := \min_z g(z) + \lambda|z - x|}$$

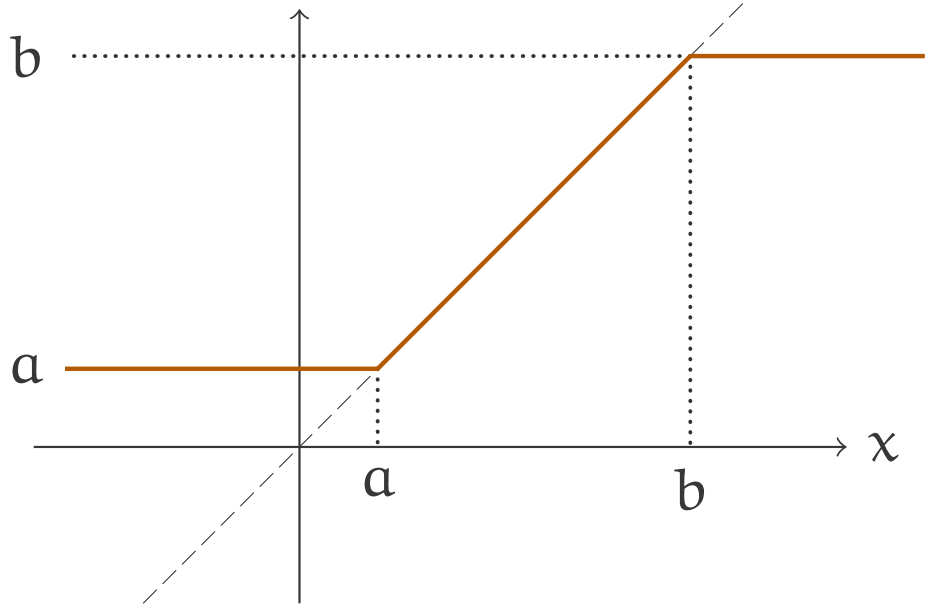
Lemma (Johnson 2013**)**

g *convex diff'able* $\Rightarrow$

h *convex, diff'able:*

$$\boxed{h'(x) = \mathrm{clip}_{-\lambda}^{+\lambda}\left[g'(x)\right]}$$

$\mathrm{clip}_a^b(x) := \min\{b, \max\{a, x\}\}$



Min-Convolution

$\lambda \geqslant 0$

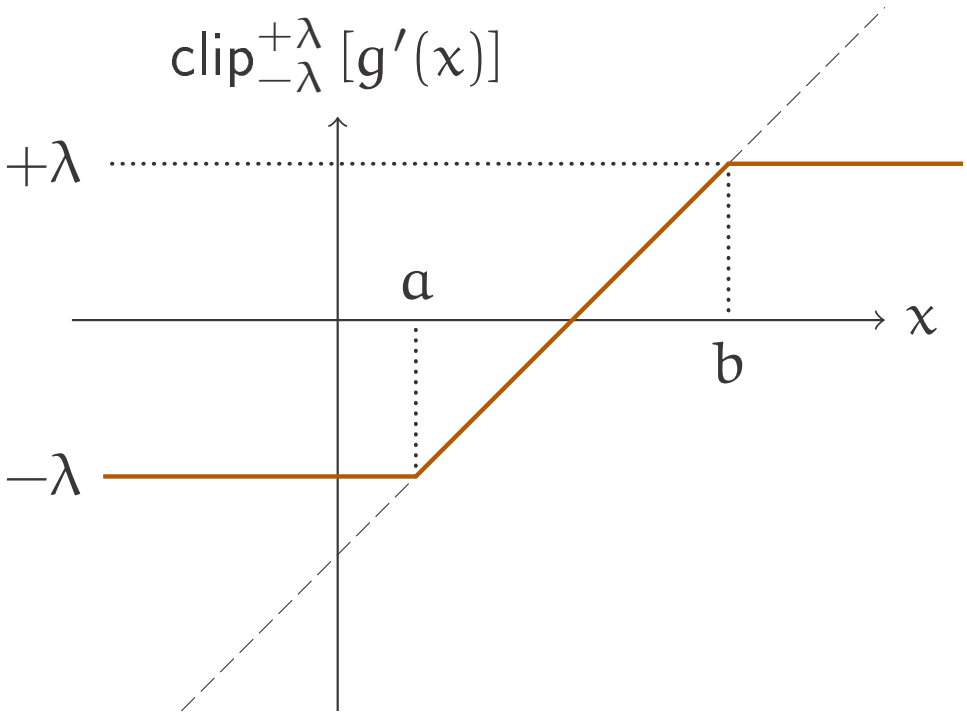
$$\boxed{h(x) := \min_z g(z) + \lambda|z - x|}$$

Lemma (Johnson 2013)

g *convex diff'able* $\Rightarrow$

h *convex, diff'able:*

$$\boxed{h'(x) = \mathrm{clip}_{-\lambda}^{+\lambda}\left[g'(x)\right]}$$

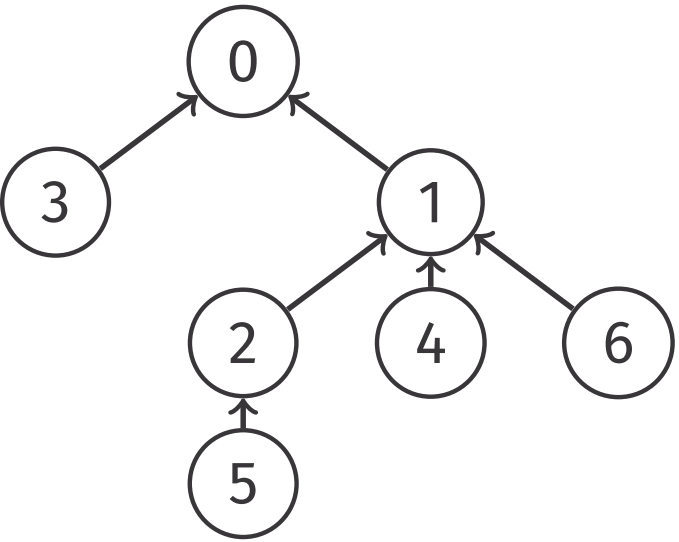


Furthermore

$z^* = \mathrm{clip}_a^b(x)$ with

$[a, b] := (g')^{-1}([-\lambda, \lambda])$

Dynamic Programming

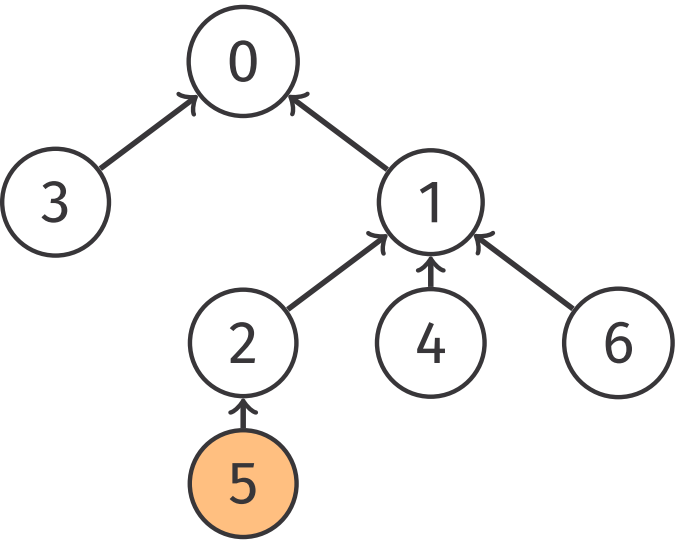


Idea: Decompose into subproblems

$f_i : \mathbb{R} \to \mathbb{R}$: optimize over subtree T_i

$f_i(x)$ minimal objective with $x_i = x$ fixed.

Dynamic Programming



Idea: Decompose into subproblems

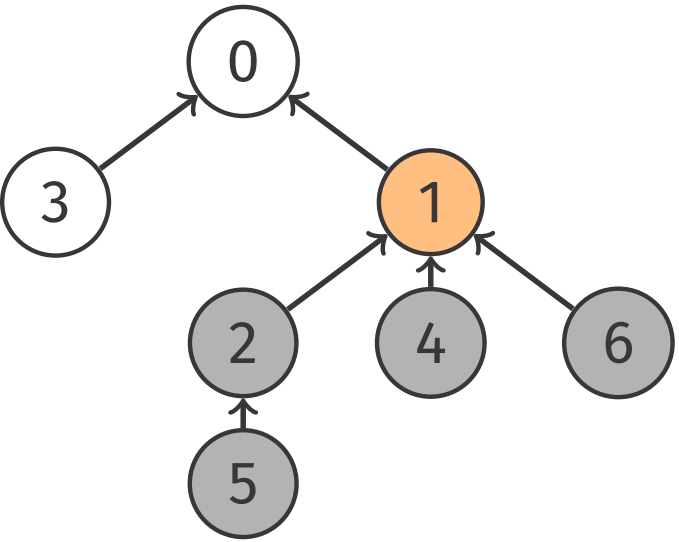
$f_i : \mathbb{R} \to \mathbb{R}$: optimize over subtree T_i

$f_i(x)$ minimal objective with $x_i = x$ fixed.

$$f_5(x) = \frac{1}{2}(x - y_5)^2$$

$$f_5'(x) = x - y_5$$

Dynamic Programming



Idea: Decompose into subproblems

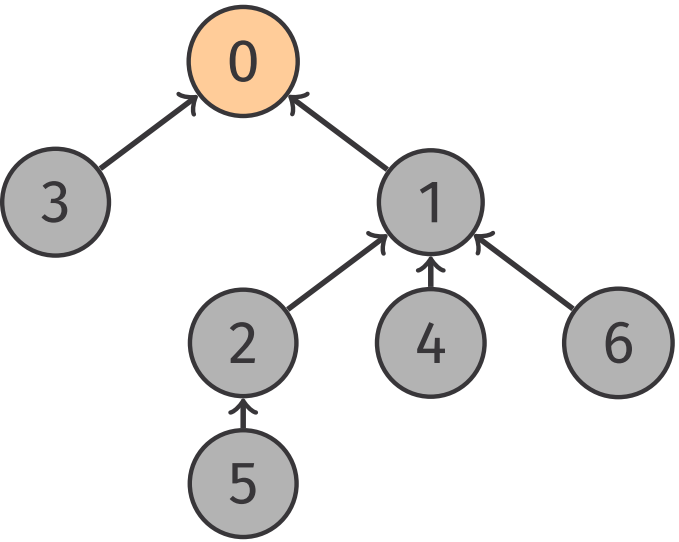
$f_i : \mathbb{R} \to \mathbb{R}$: optimize over subtree T_i

$f_i(x)$ minimal objective with $x_i = x$ fixed.

$$f_1(x) = \tfrac{1}{2}(x - y_1)^2 + \min \sum_{j \in \{2,4,6\}} \left(\lambda_j |x - x_j| + f_j(x_j)\right)$$

$$f_1'(x) = x - y_i + \sum_{j \in \{2,4,6\}} \operatorname{clip}_{-\lambda_j}^{+\lambda_j} \left[f_j'(x)\right]$$

Dynamic Programming



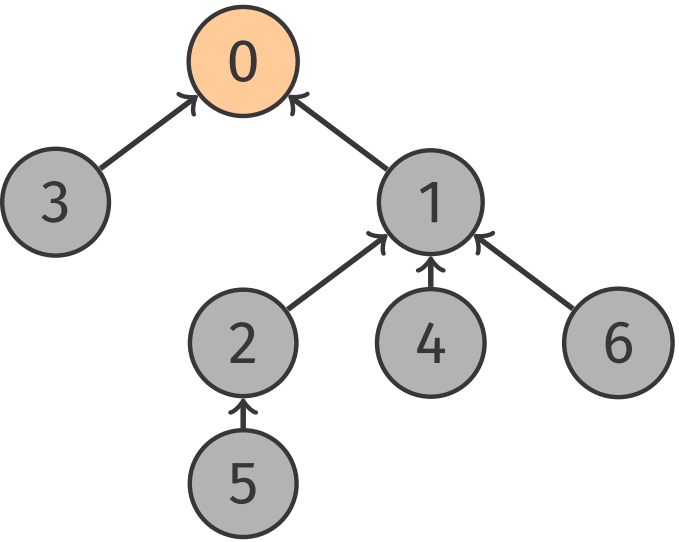
Idea: Decompose into subproblems

$f_i : \mathbb{R} \to \mathbb{R}$: optimize over subtree T_i

$f_i(x)$ minimal objective with $x_i = x$ fixed.

$$f_i'(x) = x - y_i + \sum_{j \in \mathsf{Kids}(i)} \mathrm{clip}_{-\lambda_j}^{+\lambda_j}\left(f_j'(x)\right)$$

Dynamic Programming



Idea: Decompose into subproblems

$f_i : \mathbb{R} \to \mathbb{R}$: optimize over subtree T_i

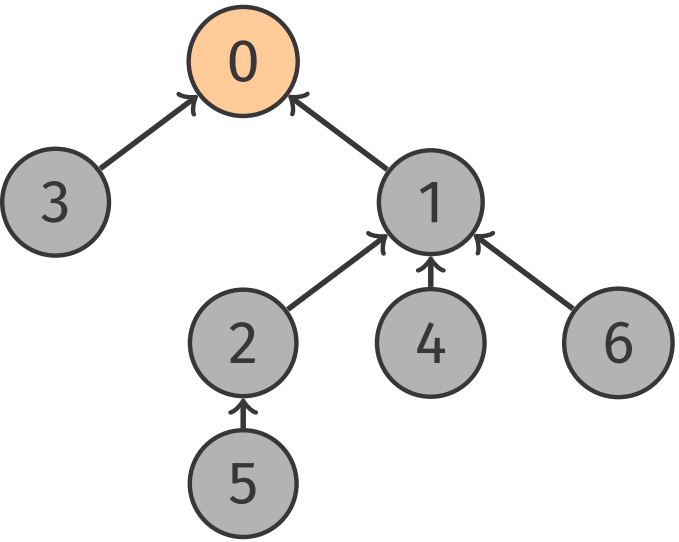
$f_i(x)$ minimal objective with $x_i = x$ fixed.

$$\boxed{f_i'(x) = x - y_i + \sum_{j \in \mathsf{Kids}(i)} \mathrm{clip}_{-\lambda_j}^{+\lambda_j}\left(f_j'(x)\right)}$$

$$f(x^*) = f_r(x_r^*)$$

Goal: Find x_r^* with $f_r'(x_r^*) \stackrel{!}{=} 0$

Dynamic Programming



Idea: Decompose into subproblems

$f_i : \mathbb{R} \to \mathbb{R}$: optimize over subtree T_i

$f_i(x)$ minimal objective with $x_i = x$ fixed.

$$\boxed{f_i'(x) = x - y_i + \sum_{j \in \mathsf{Kids}(i)} \mathrm{clip}_{-\lambda_j}^{+\lambda_j}\left(f_j'(x)\right)}$$

$$f(x^*) = f_r(x_r^*)$$

Goal: Find x_r^* with $f_r'(x_r^*) \stackrel{!}{=} 0$

Backtrace $x_i^* \leftarrow \mathrm{clip}_{a_i}^{b_i}(x_{\pi(i)}^*)$

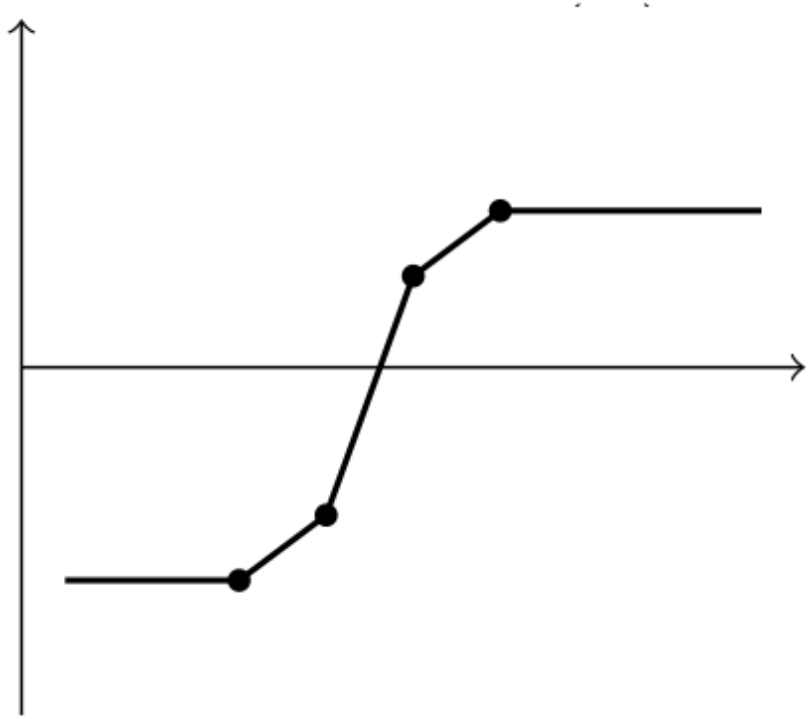
Exact Algorithm

Piecewise Linear Function (PWL)

$$\boxed{f_i'(x) = x - y_i + \sum_{j \in \mathsf{Kids}(i)} \mathsf{clip}_{-\lambda_j}^{+\lambda_j}\left(f_j'(x)\right)}$$

Piecewise Linear Function (PWL)

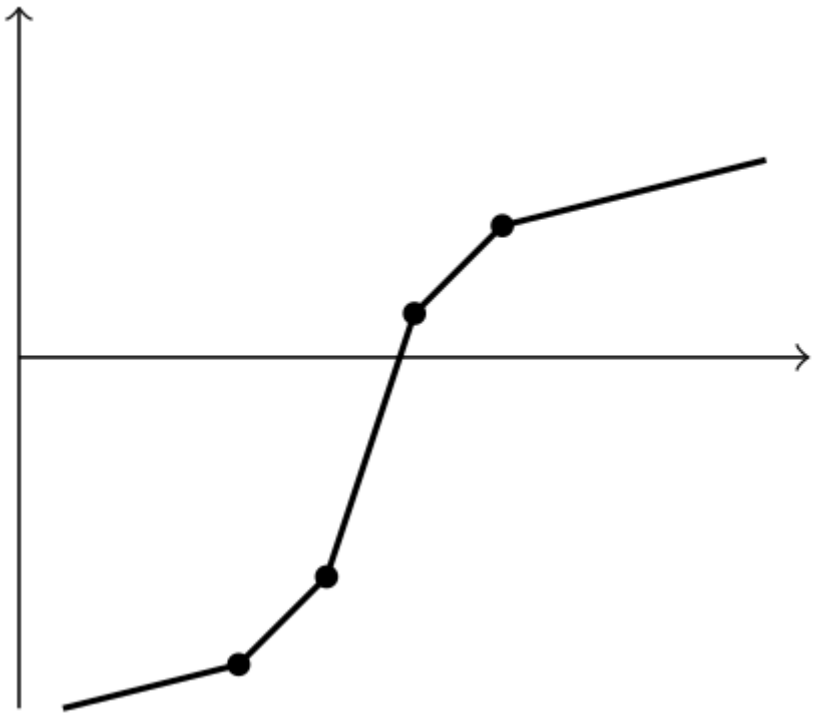
$$f_i'(x) = x - y_i + \sum_{j \in \mathsf{Kids}(i)} \mathsf{clip}_{-\lambda_j}^{+\lambda_j}\left(f_j'(x)\right)$$



- f_j' is piece-wise linear (PWL)

Piecewise Linear Function (PWL)

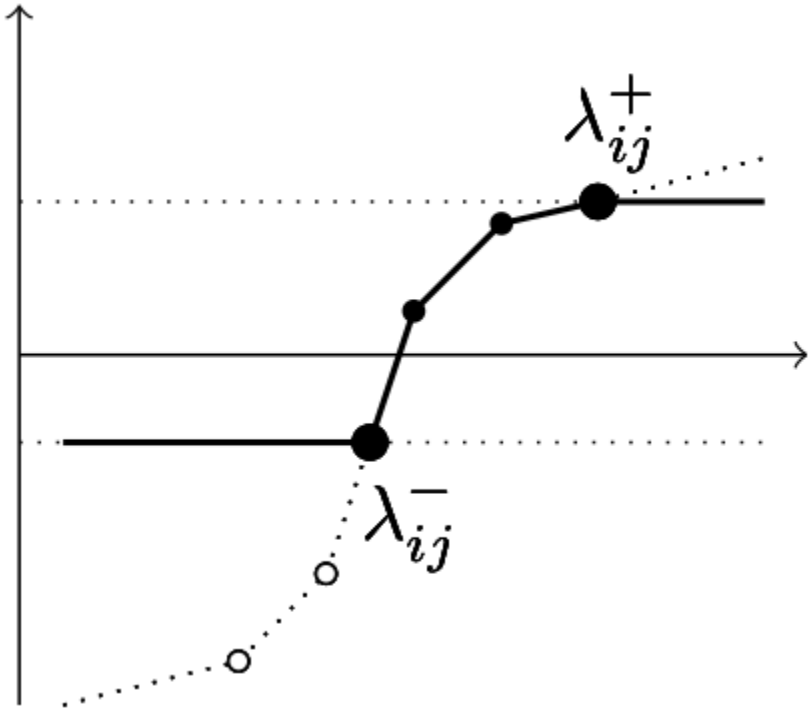
$$f_i'(x) = x - y_i + \sum_{j \in \mathsf{Kids}(i)} \mathsf{clip}_{-\lambda_j}^{+\lambda_j}\left(f_j'(x)\right)$$



- f_j' is piece-wise linear (PWL)
- $x \mapsto x - y_i$ is PWL
- sum of PWL is PWL

Piecewise Linear Function (PWL)

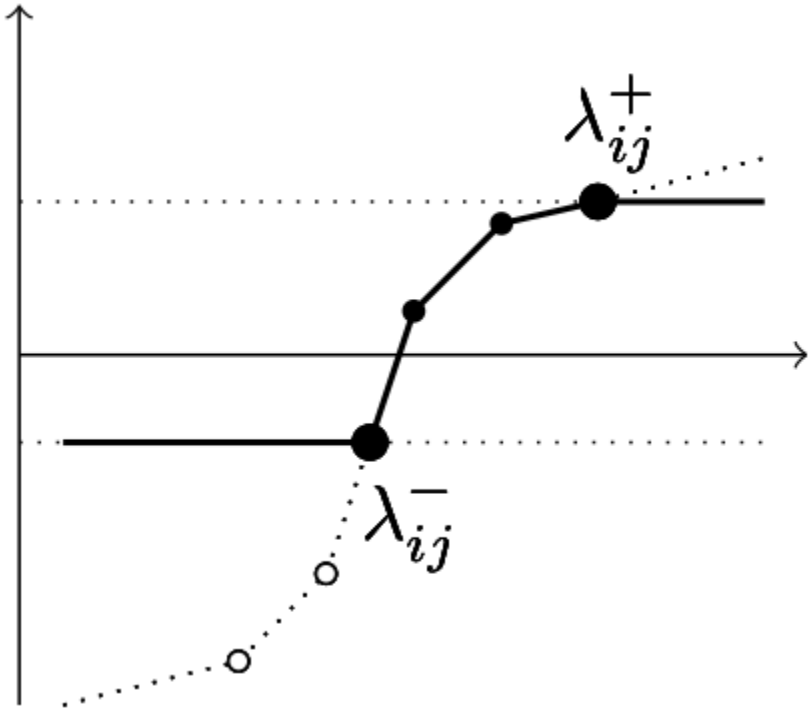
$$f_i'(x) = x - y_i + \sum_{j \in \mathsf{Kids}(i)} \mathsf{clip}_{-\lambda_j}^{+\lambda_j}\left(f_j'(x)\right)$$



- f_j' is piece-wise linear (PWL)
- $x \mapsto x - y_i$ is PWL
- sum of PWL is PWL
- clip of PWL is PWL

Piecewise Linear Function (PWL)

$$f_i'(x) = x - y_i + \sum_{j \in \mathsf{Kids}(i)} \mathsf{clip}_{-\lambda_j}^{+\lambda_j}\left(f_j'(x)\right)$$



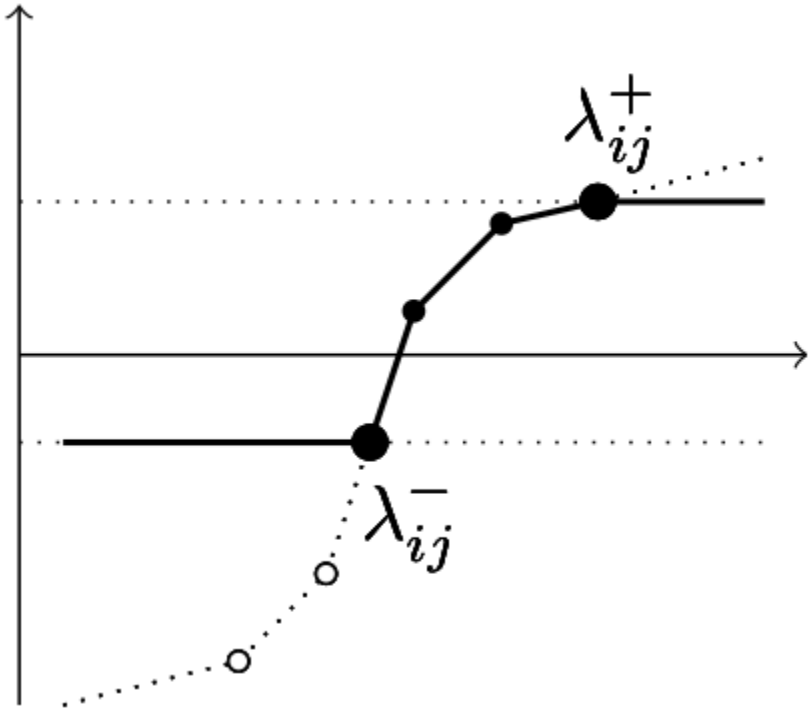
- f_j' is piece-wise linear (PWL)
- $x \mapsto x - y_i$ is PWL
- sum of PWL is PWL
- clip of PWL is PWL

Store change-points (knots):

- position $\tilde{x}_k$
- $\Delta\mathsf{slope}_k$
- $\Delta\mathsf{intercept}_k$

Piecewise Linear Function (PWL)

$$f'_i(x) = x - y_i + \sum_{j \in \text{Kids}(i)} \text{clip}_{-\lambda_j}^{+\lambda_j}\left(f'_j(x)\right)$$



- f'_j is piece-wise linear (PWL)
- $x \mapsto x - y_i$ is PWL
- sum of PWL is PWL
- clip of PWL is PWL

Store change-points (knots):

- position $\tilde{x}_k$
- Δslope_k
- $\Delta\text{intercept}_k$

Operations: EXTRAC-MIN(q) • EXTRACT-MAX(q) • MERGE(q_1, q_2)

Algorithm **CLIP↑**

Input : Queue q, Slope s, Intercept t, Target t_0
Output: Position $\hat{x}$ (modified q)

---

while *not* EMPTY(q) **and** $(s \cdot q.min_x + t < t_0)$ **do**
  knot $\leftarrow$ EXTRACT-MIN(q)
  $s \leftarrow s +$ knot$.\Delta s$
  $t \leftarrow t +$ knot$.\Delta t$
$\hat{x} \leftarrow (t_0 - t)/s$
INSERT$(q,$ knot: Δslope s, Δintercept t, position $\hat{x})$

Algorithm CLIP↑

Input : Queue q, Slope s, Intercept t, Target t_0
Output: Position $\hat{x}$ (modified q)

---

while *not* EMPTY(q) **and** $(s \cdot q.\mathit{min_x} + t < t_0)$ **do**
  knot $\leftarrow$ EXTRACT-MIN(q)
  $s \leftarrow s +$ knot$.\Delta s$
  $t \leftarrow t +$ knot$.\Delta t$
$\hat{x} \leftarrow (t_0 - t)/s$
INSERT$(q,$ knot: Δslope s, Δintercept t, position $\hat{x})$

CLIP↓ analogue

Algorithm **TreeOpt** [Kolmogorov et al. (2016)]

Input : Signal $y \in \mathbb{R}^n$, Weights $\mu \geqslant 0$, $\lambda > 0$, Tree T: π, Root r
Output: Minimizer $x^* \in \mathbb{R}^n$.

Initialize empty queues f'_i for each $i \in V$

Compute $\hat{\lambda}_i = \sum_{j \in \mathsf{Kids}(i)} \lambda_i$ for each $i \in V$

for *each node* i *in* *Post-Order*(T) **do**

  $a_i \leftarrow \textsc{Clip} \uparrow \left(f'_i, \mu_i, -\mu_i y_i - \hat{\lambda}_i, -\lambda_i\right)$

  $b_i \leftarrow \textsc{Clip} \downarrow \left(f'_i, \mu_i, -\mu_i y_i + \hat{\lambda}_i, +\lambda_i\right)$

  $f'_{\pi(i)} \leftarrow f'_{\pi(i)} \cup f'_i$

$x^*_r \leftarrow \textsc{Clip} \uparrow \left(f'_r, \mu_r, -\mu_r y_r + \hat{\lambda}_r, 0\right)$

for *each node* i *in* *Pre-Order*(T) **do**

  $x^*_i \leftarrow \mathrm{clip}_{a_i}^{b_i}(x^*_{\pi(i)})$

Runtime TreeOpt

[Kolmogorov et al. (2016)]

Theorem (Tree Fused Lasso)

x^* *is computable in* $\mathcal{O}(n \log n)$

Proof sketch.

- $2n$ knots are created
- Fibonacci Heaps for queues f_i'

$\square$

Runtime TreeOpt

Theorem (Tree Fused Lasso)

x^* is computable in $\mathcal{O}(n \log n)$

Proof sketch.

- $2n$ knots are created
- Fibonacci Heaps for queues f_i'

$\square$

Observation (Experiments)

Queues contain few elements

Runtime TreeOpt

Theorem (Tree Fused Lasso)

x^* is computable in $\mathcal{O}(n \log n)$

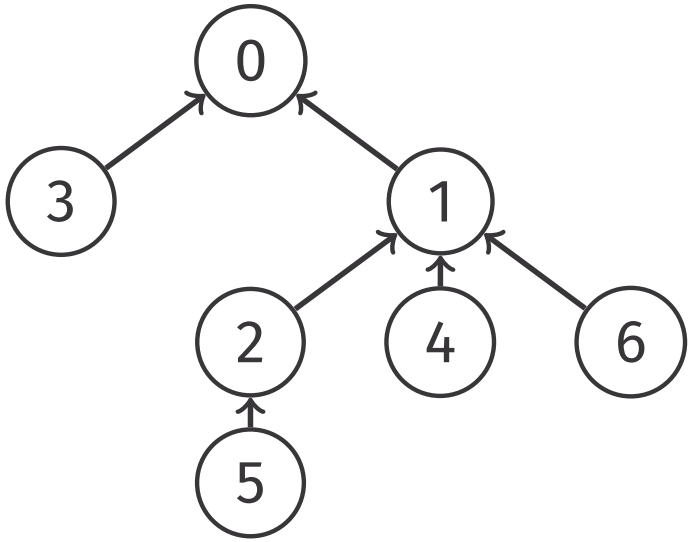
Proof sketch.

- $2n$ knots are created
- Fibonacci Heaps for queues f'_i

$\square$

Observation (Experiments)

Queues contain few elements



`0:(1:(3)((2:(5):2)(4)(6)):1):0`

Runtime TreeOpt

Theorem (Tree Fused Lasso)

x^* is computable in $\mathcal{O}(n \log n)$

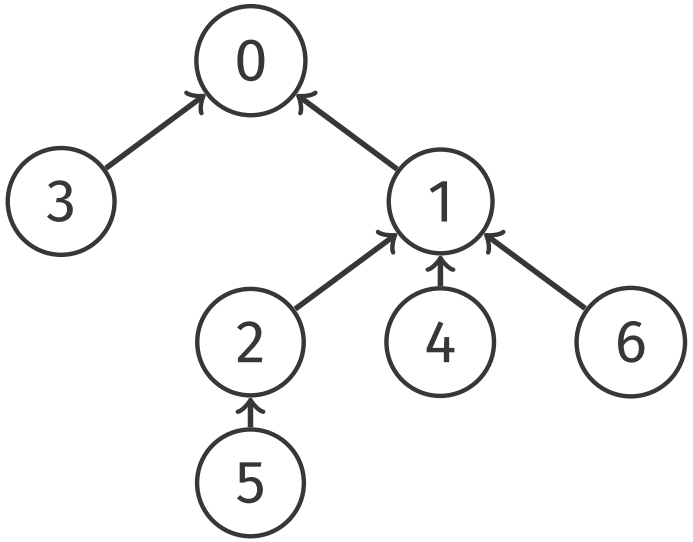
Proof sketch.

- $2n$ knots are created
- Fibonacci Heaps for queues f'_i

$\square$

Observation (Experiments)

Queues contain few elements



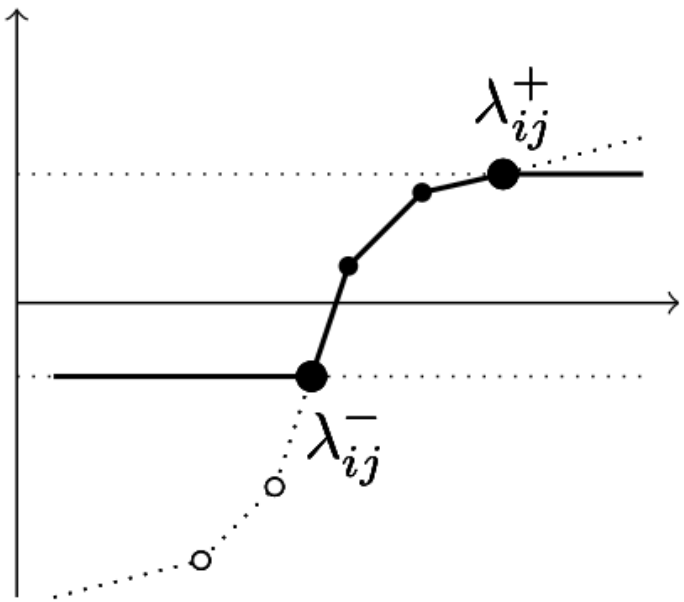
`0:(1:(3)((2:(5):2)(4)(6)):1):0`

$\rightsquigarrow$ sorted array(s)

Approximation Scheme

Idea: **Sample** f_i'

$$\phi_i := f_i'(x) = x - y_i + \sum_{j \in \mathsf{Kids}(i)} \mathrm{clip}_{-\lambda_j}^{+\lambda_j}\left(f_j'(x)\right)$$

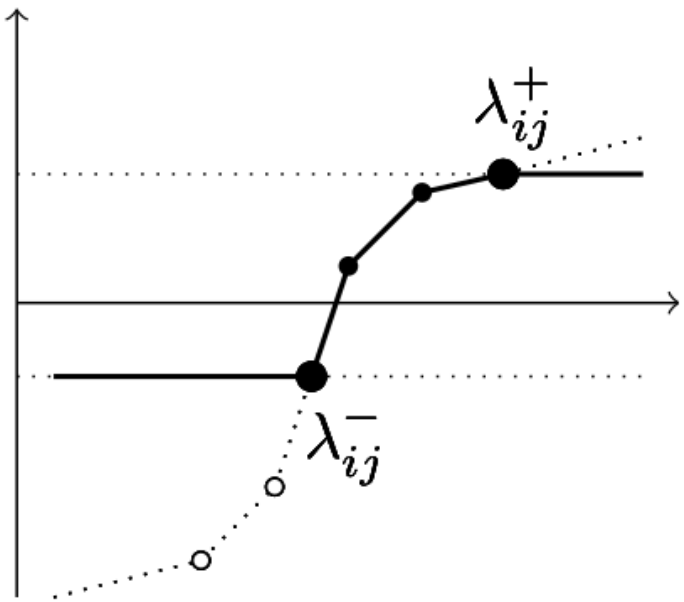


f_i convex $\Rightarrow$

f_i' mon. increasing

Idea: **Sample** f_i'

$$\phi_i := f_i'(x) = x - y_i + \sum_{j \in \mathsf{Kids}(i)} \mathrm{clip}_{-\lambda_j}^{+\lambda_j}\left(f_j'(x)\right)$$



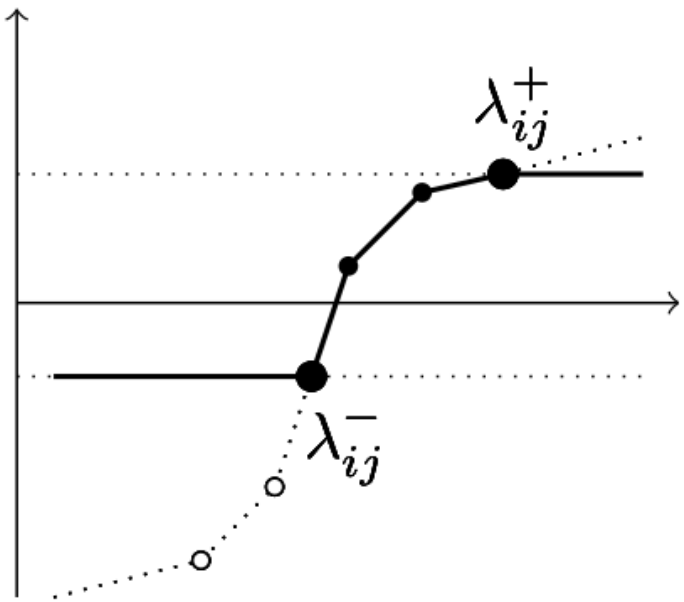
Assume edge i is clipped

$$\phi_i \leqslant -\lambda_i$$
$$\Rightarrow x_i^* \geqslant x$$

f_i convex $\Rightarrow$
f_i' mon. increasing

Idea: **Sample** f_i'

$$\phi_i := f_i'(x) = x - y_i + \sum_{j \in \mathsf{Kids}(i)} \mathrm{clip}_{-\lambda_j}^{+\lambda_j}\left(f_j'(x)\right)$$



Assume edge i is clipped

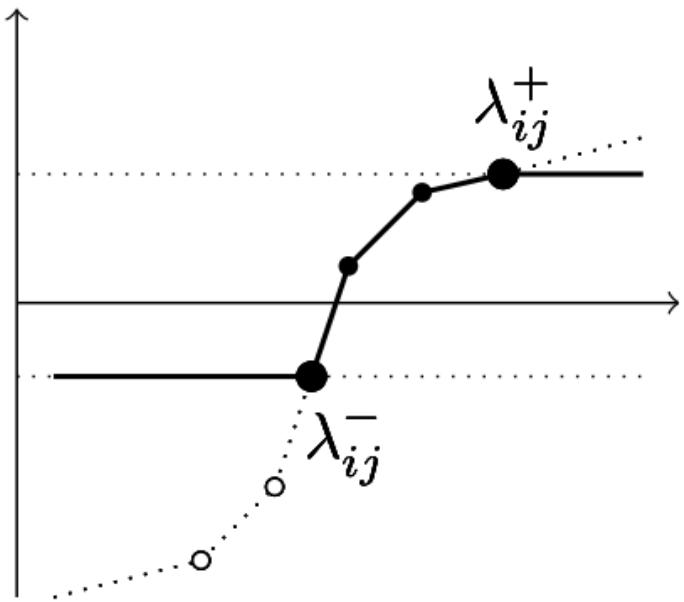
$\phi_i \leqslant -\lambda_i$	$\phi_i \geqslant +\lambda_i$
$\Rightarrow x_i^* \geqslant x$	$\Rightarrow x_i^* \leqslant x$

f_i convex $\Rightarrow$

f_i' mon. increasing

Idea: **Sample** f_i'

$$\phi_i := f_i'(x) = x - y_i + \sum_{j \in \mathsf{Kids}(i)} \mathrm{clip}_{-\lambda_j}^{+\lambda_j}\left(f_j'(x)\right)$$



f_i convex $\Rightarrow$
f_i' mon. increasing

Assume edge i is clipped

$\phi_i \leqslant -\lambda_i$	$\phi_i \geqslant +\lambda_i$
$\Rightarrow x_i^* \geqslant x$	$\Rightarrow x_i^* \leqslant x$

If edge i not clipped

Overtake parent's value $\pi(i)$

$\rightsquigarrow$ Binary Search

Algorithm TreeApx

Input : Probes $x_i \in I_i$, Bounds $I_i = [a_i, b_i]$ for $i \in V$; Tree T

Output: Improved bounds $[\hat{a}_i, \hat{b}_i] \subset [a_i, b_i]$

---

for *each node* i *in* Post-Order(T) **do**

  $\phi_i \leftarrow \mu_i(x_i - y_i) + \sum_{j \in \mathsf{Kids}(i)} \mathsf{clip}_{-\lambda_j}^{+\lambda_j}[\phi_j]$

for *each node* i *in* Pre-Order(T) **do**

  if $\phi_i \leqslant -\lambda_i$ **then**

    $[\hat{a}_i, \hat{b}_i] \leftarrow [x_i, b_i]$

  else if $\phi_i \geqslant +\lambda_i$ **then**

    $[\hat{a}_i, \hat{b}_i] \leftarrow [a_i, x_i]$

  else

    $[\hat{a}_i, \hat{b}_i] \leftarrow [a_{\pi(i)}, b_{\pi(i)}]$

Runtime: **TreeApx**

Theorem

In time $\mathcal{O}(n \log_2 1/\delta)$ *TreeApx computes* x *with* $\|x^* - x\|_\infty \leqslant \delta$

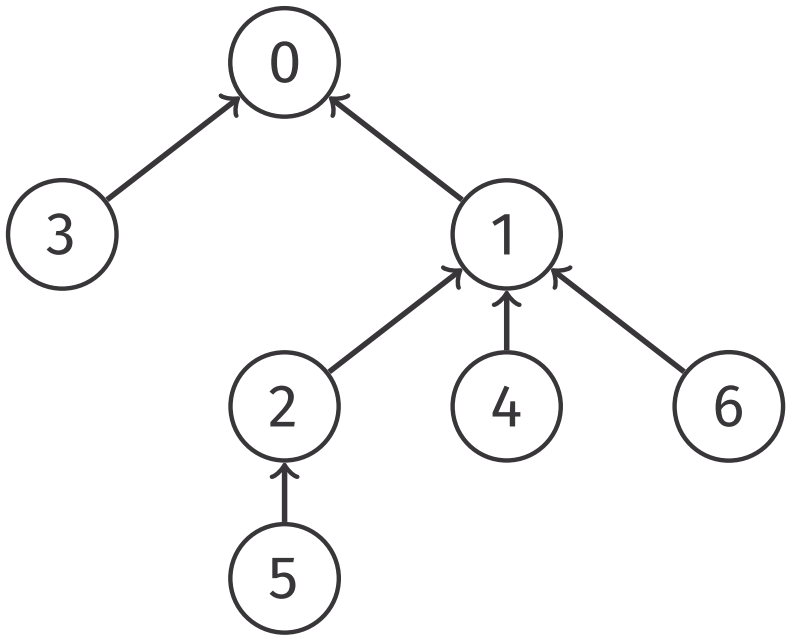
Fixed precision δ (e.g. `float32`)
$\rightsquigarrow$ linear runtime.

Runtime: **TreeApx**

Theorem

In time $\mathcal{O}(n \log_2 1/\delta)$ *TreeApx computes* x *with* $\|x^* - x\|_\infty \leqslant \delta$

Fixed precision δ (e.g. `float32`)
$\rightsquigarrow$ linear runtime.



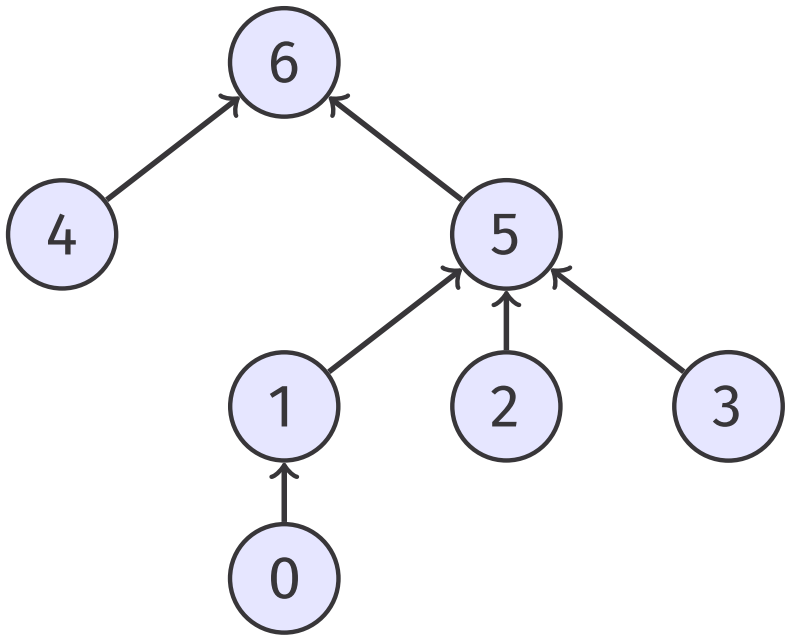
	0	1	2	3	4	5	6
BFS^R	5	2	4	6	3	1	0

Runtime: **TreeApx** $\rightsquigarrow$ **Relabel Tree**

Theorem

In time $\mathcal{O}(n \log_2 1/\delta)$ *TreeApx computes* x *with* $\|x^* - x\|_\infty \leqslant \delta$

Fixed precision δ (e.g. `float32`)
$\rightsquigarrow$ linear runtime.



	0	1	2	3	4	5	6
BFS^R	0	1	2	3	4	5	6

Experiments

Fused Lasso Tree Instances

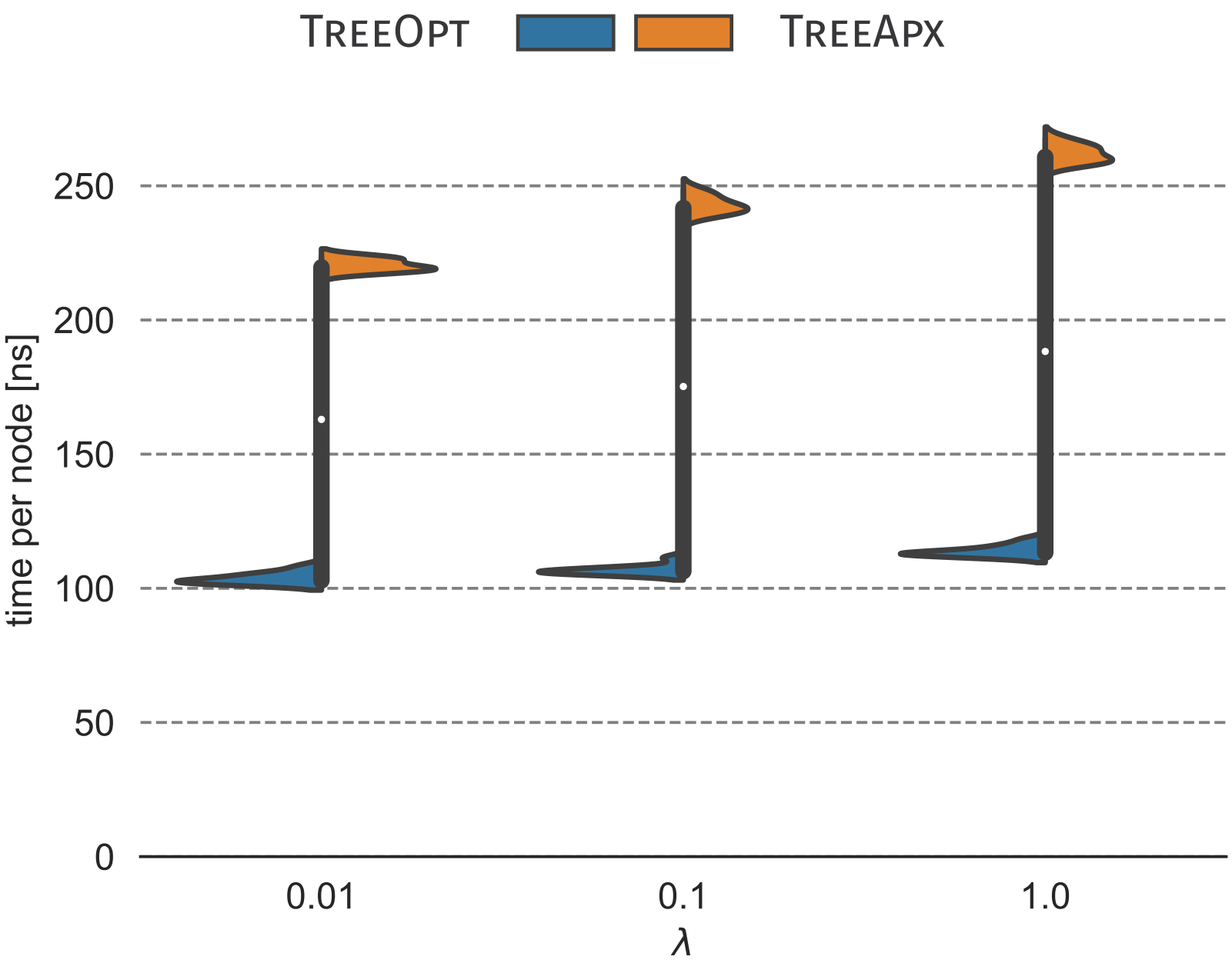
Name	$n = \lvert V \rvert$	leaf	mean $\deg_0$
Binary	100 000 000	50.0 %	3.00 ± 0
Europe	50 912 018	11.3 %	2.13 ± 0.37
HighDeg/LowDeg	50 000 000	37.2 %	2.59 ± 71.32
com-Orkut	3 072 441	48.6 %	2.94 ± 3.31
Phantom	1 000 000	33.6 %	2.50 ± 0.64

Fused Lasso Tree Instances

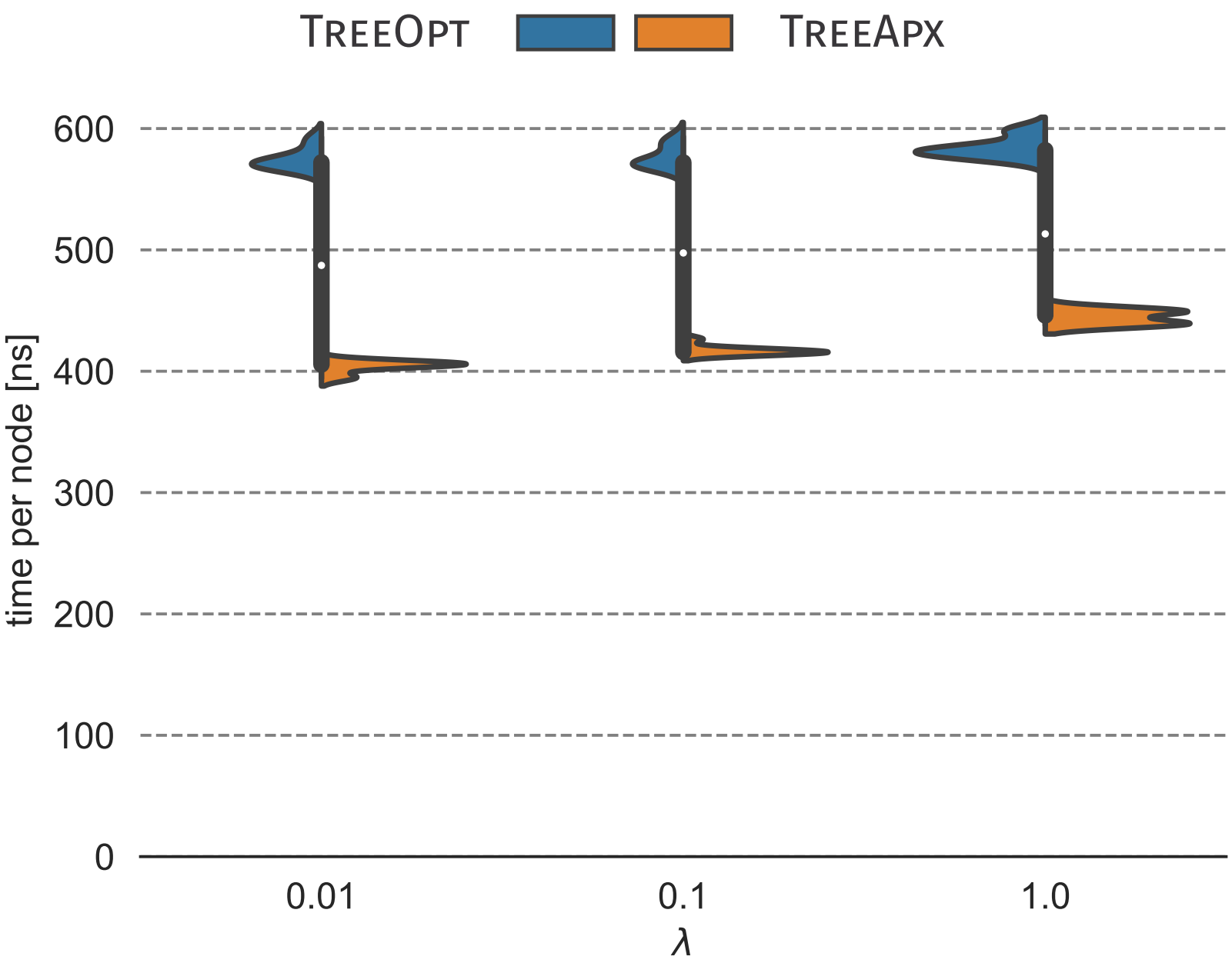
Name	$n = \lvert V \rvert$	leaf	mean $\deg_0$
BINARY	100 000 000	50.0 %	3.00 ± 0
EUROPE	50 912 018	11.3 %	2.13 ± 0.37
HIGHDEG/LOWDEG	50 000 000	37.2 %	2.59 ± 71.32
COM-ORKUT	3 072 441	48.6 %	2.94 ± 3.31
PHANTOM	1 000 000	33.6 %	2.50 ± 0.64

- Sample 10 random spanning trees
- $y_i \sim \mathcal{N}(0, 1)$ for all $i \in V$
- $\lambda \in \{0.01, 0.1, 1\}$
- $\delta = 2^{-20} \approx 10^{-6}$

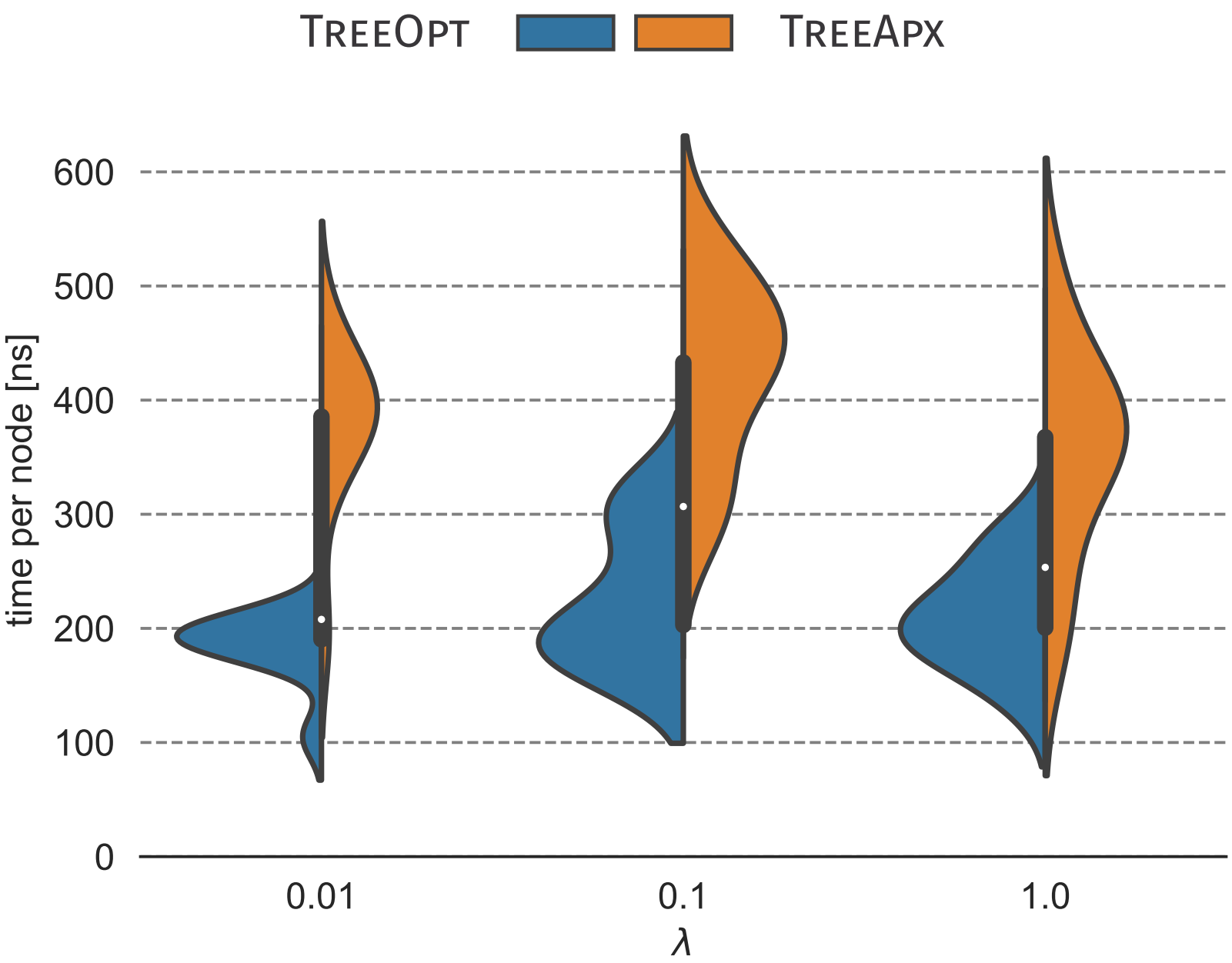
Results: **Europe** $\deg_0 \approx 2.13 \pm 0.37$ $n = 50\,912\,018$



Results: **HighDeg/LowDeg** $\deg_0 \approx 2.59 \pm 71.32$ $n = 50\,000\,000$



Results: **PHANTOM** $\deg_0 \approx 2.50 \pm 0.64$ $n = 1\,000\,000$



Conclusions

TreeOpt Exact Dynamic Programming

- $\mathcal{O}(nq \log q)$
- Sophisticated data structures not needed
- Memory layout is important

TreeApx Approximation Scheme

- $\mathcal{O}(n \log 1/\delta)$
- Simple Gradient Descent
- Better for high degrees

Conclusions

TreeOpt Exact Dynamic Programming

- $\mathcal{O}(nq \log q)$
- Sophisticated data structures not needed
- Memory layout is important

TreeApx Approximation Scheme

- $\mathcal{O}(n \log 1/\delta)$
- Simple Gradient Descent
- Better for high degrees

Asymptotically fast does not mean fast ***in practice***:

- 5 to 20* times slower than on Lines
- Depends on degree distribution

References

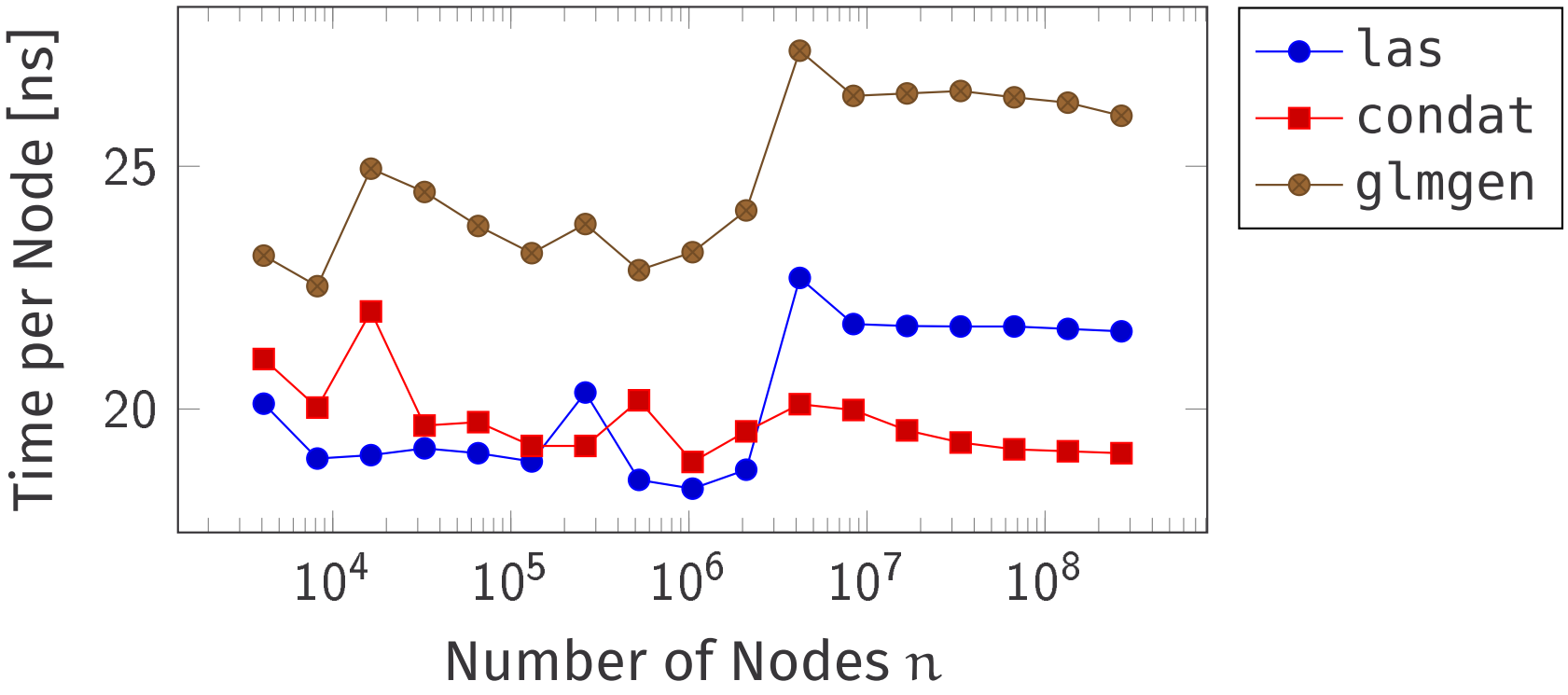
Nicholas Johnson.
A dynamic programming algorithm for the fused lasso and L_0-segmentation.
Journal of Computational and Graphical Statistics, 22(2): 246–260, 2013.
doi: 10.1080/10618600.2012.681238.

Vladimir Kolmogorov, Thomas Pock, and Michal Rolinek.
Total variation on a tree.
SIAM J. Imaging Sciences, 9(2):605–636, 2016.

Oscar H. M. Padilla, James Sharpnack, and James G Scott.
The DFS fused lasso: Linear-time denoising over general graphs.
The Journal of Machine Learning Research, 18(1):6410–6445, 2017.

Appendix

Experiments: **Line Graphs**



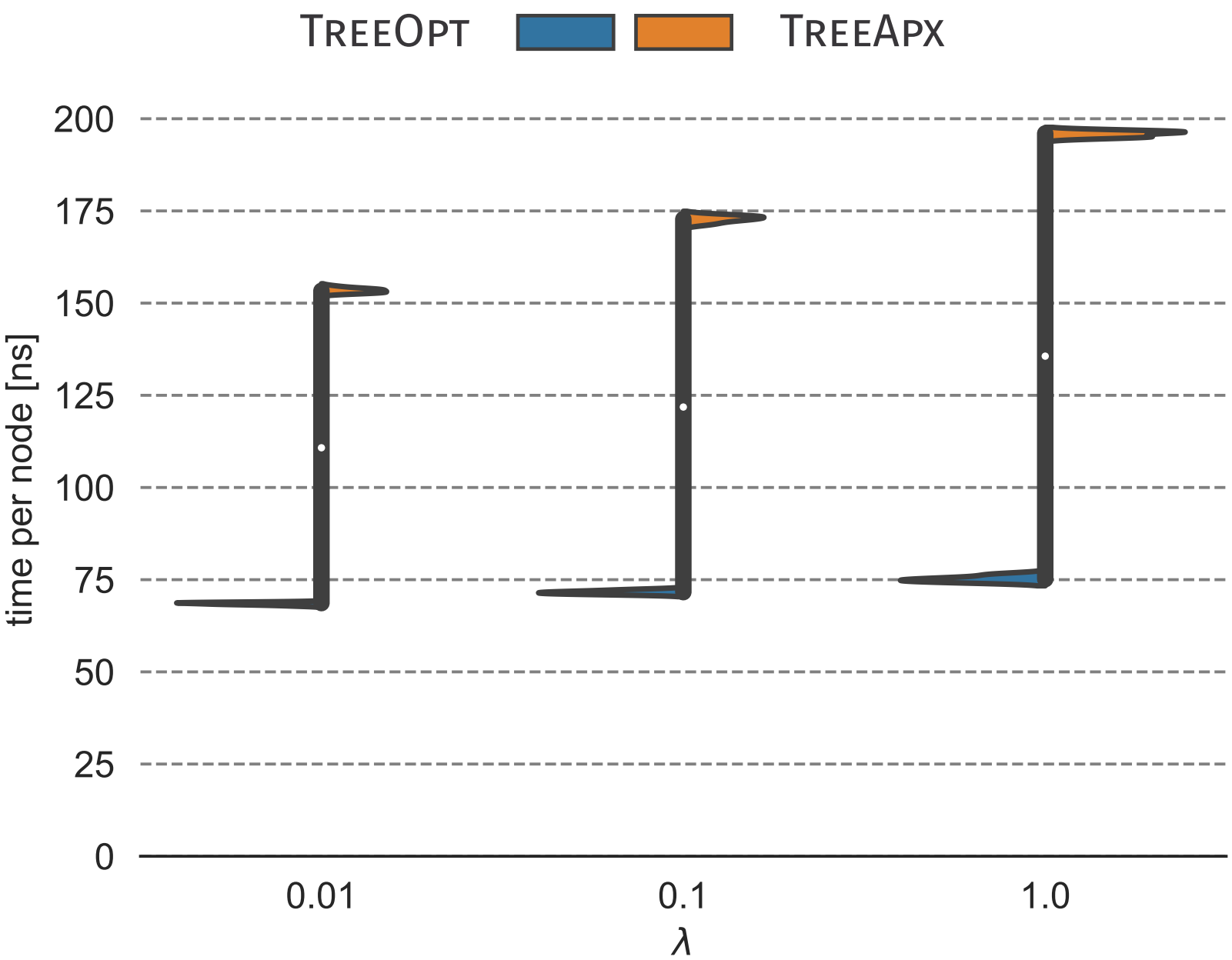
Dual Problem

$$\min_{\alpha} \phi(\alpha) := \sum_{i \in V} \left(y_i - \sum_{j \,:\, ij \in E} \alpha_{ij} \right)^2$$

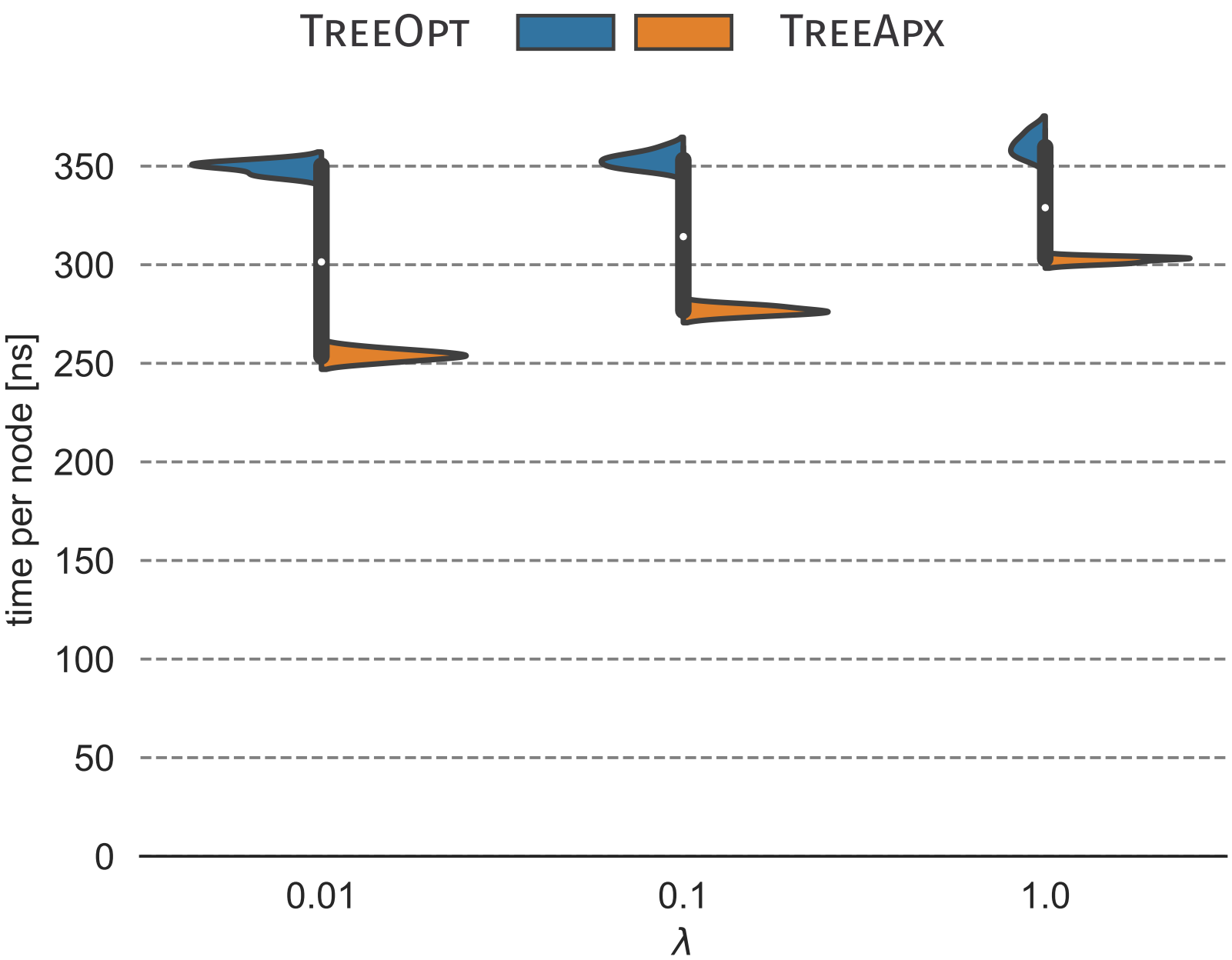
$$\text{s.t.} \quad |\alpha_e| \leqslant \lambda_e \quad \forall\, e \in E$$

- Paramatrized st-MinCut problem
- ADMM: Decompose into subgraphs

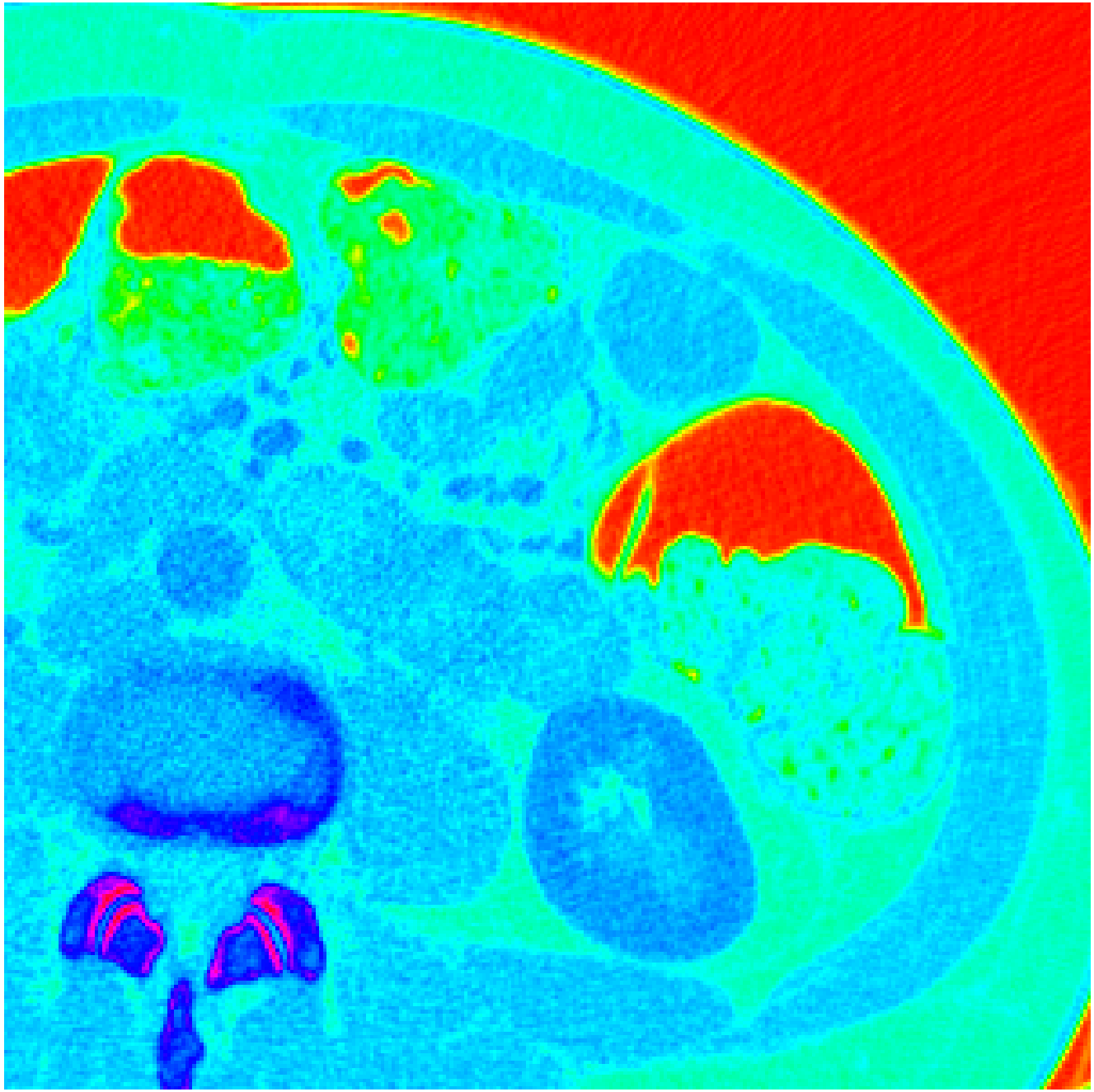
Results: **Binary** $\deg_0 \approx 3.00 \pm 0$ $n = 10^8$



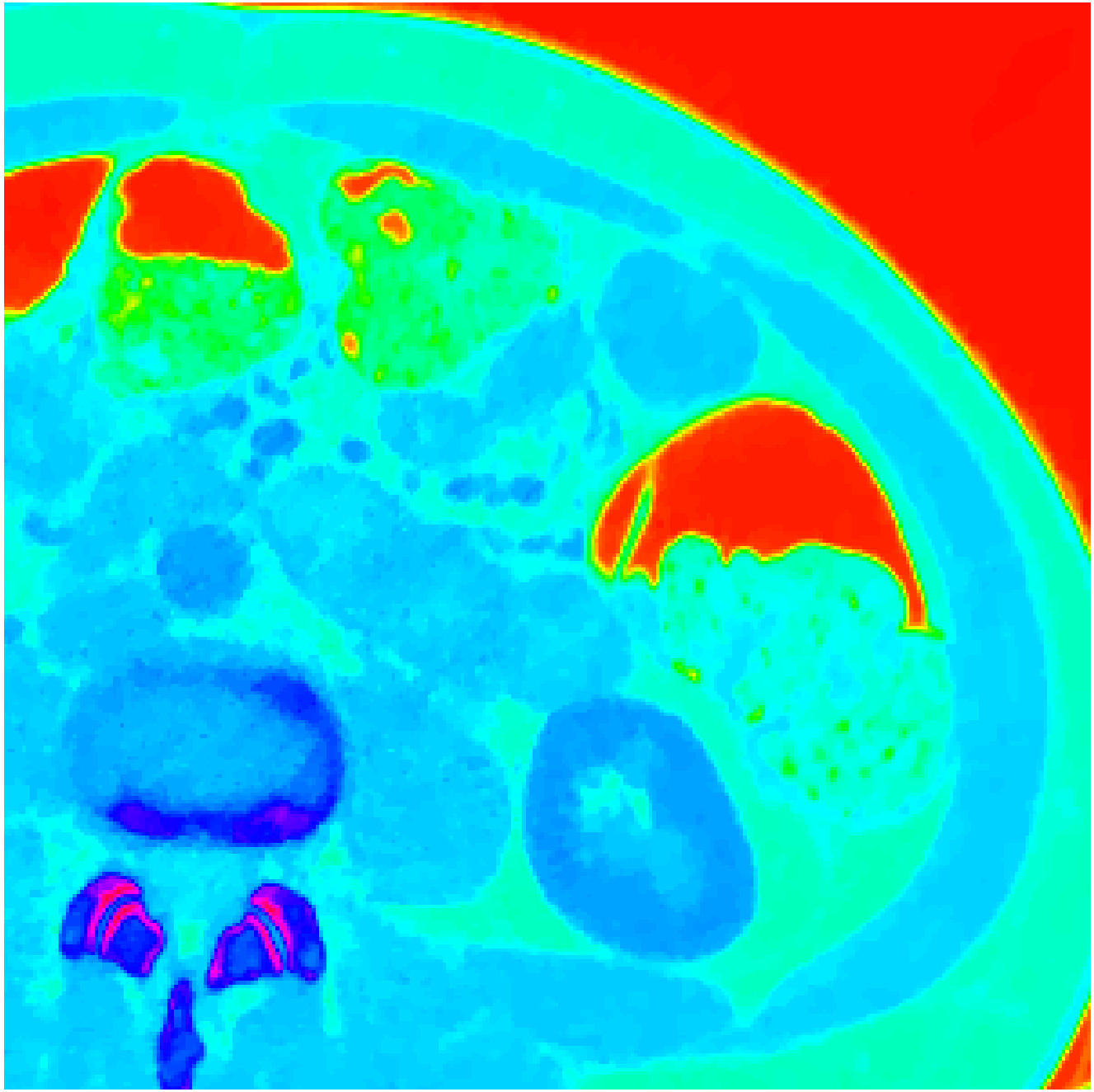
Results: COM-ORKUT $\deg_0 \approx 2.50 \pm 0.64$ $n = 3\,072\,441$

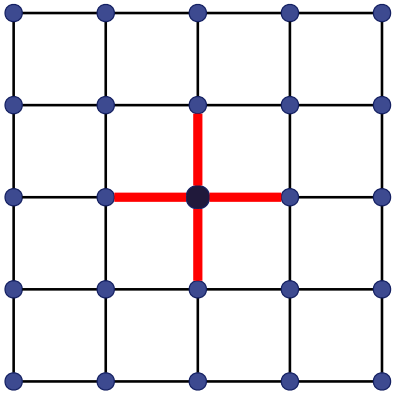


Application: **Magnetic Resonance Imaging (MRI)**

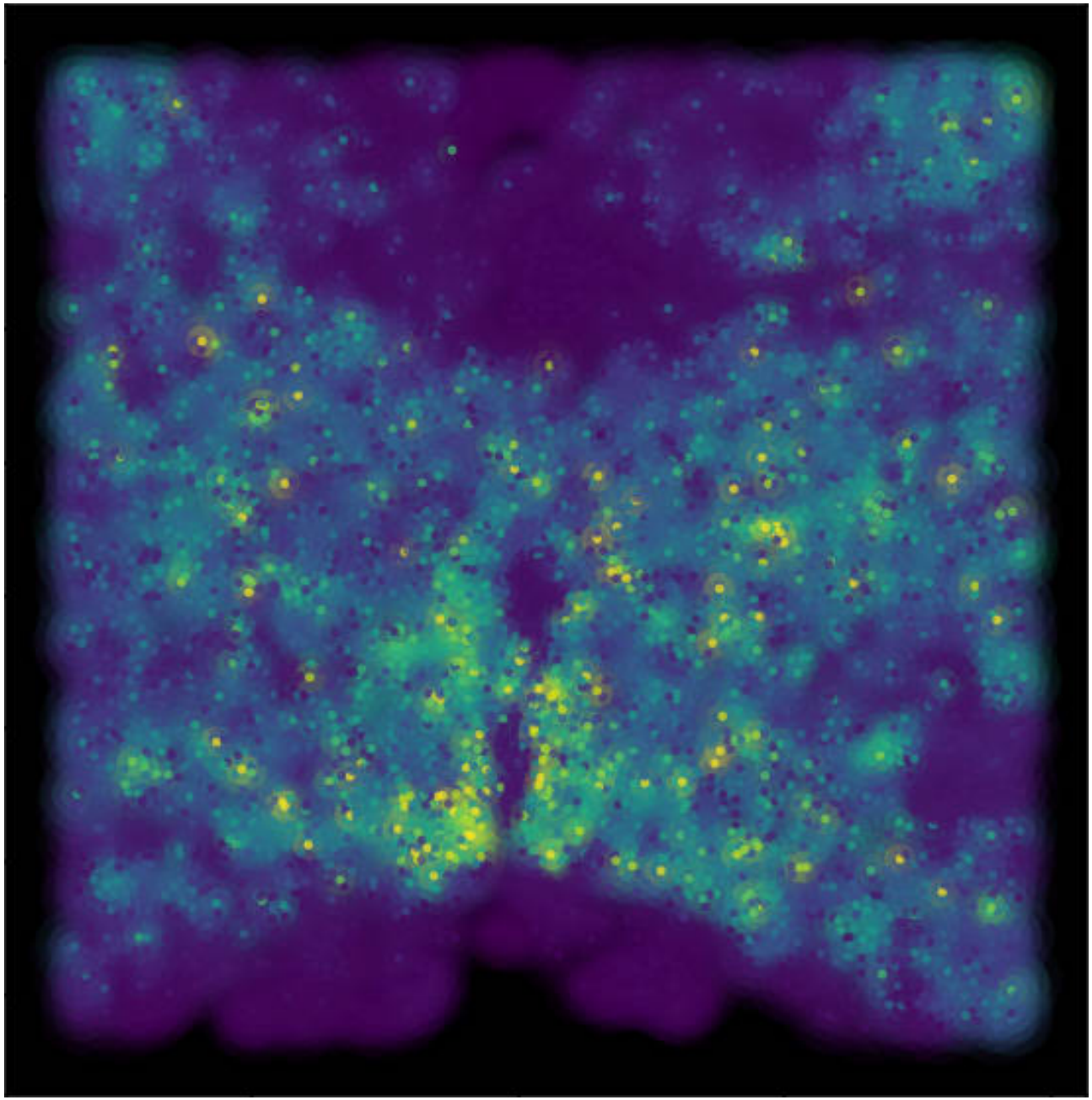


Application: **Magnetic Resonance Imaging (MRI)**

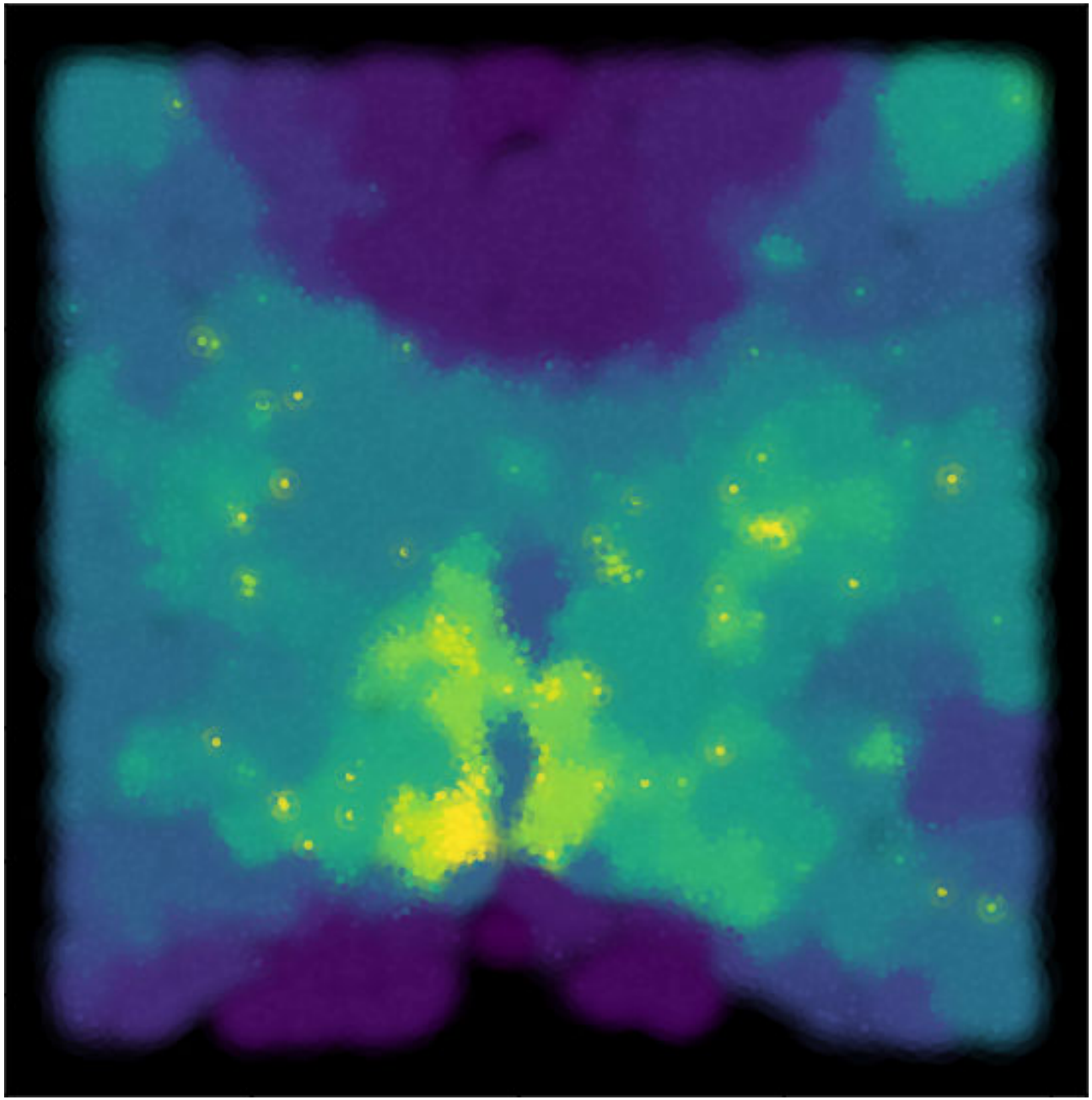
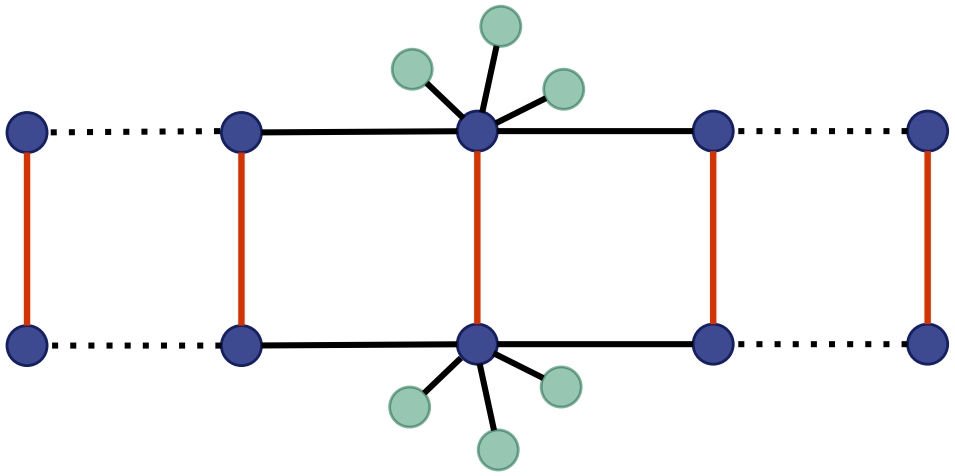




MERFISH



MERFISH



Application: **Differentially Methylated Regions (DMR)**