Polar Operators for Structured Sparse Estimation

Xinhua Zhang
Machine Learning Research Group
National ICT Australia and ANU
xinhua.zhang@anu.edu.au

Yaoliang Yu and Dale Schuurmans
Department of Computing Science, University of Alberta
Edmonton, Alberta T6G 2E8, Canada
{yaoliang,dale}@cs.ualberta.ca

Abstract

Structured sparse estimation has become an important technique in many areas of data analysis. Unfortunately, these estimators normally create computational difficulties that entail sophisticated algorithms. Our first contribution is to uncover a rich class of structured sparse regularizers whose *polar operator* can be evaluated efficiently. With such an operator, a simple conditional gradient method can then be developed that, when combined with smoothing and local optimization, significantly reduces training time vs. the state of the art. We also demonstrate a new reduction of polar to proximal maps that enables more efficient latent fused lasso.

1 Introduction

Sparsity is an important concept in high-dimensional statistics [1] and signal processing [2] that has led to important application successes by reducing model complexity and improving interpretability of the results. Standard computational strategies such as greedy feature selection [3] and generic convex optimization [4–7] can be used to implement simple sparse estimators. However, sophisticated notions of *structured* sparsity have been recently developed that can encode *combinatorial* patterns over variable subsets [8]. Although combinatorial structure greatly enhances modeling capability, it also creates computational challenges that require sophisticated optimization approaches. For example, current structured sparse estimators often adopt an *accelerated proximal gradient* (APG) strategy [9, 10], which has a low per-step complexity and enjoys an optimal convergence rate among black-box first-order procedures [10]. Unfortunately, APG must also compute a *proximal update* (PU) of the nonsmooth regularizer during each iteration. Not only does the PU require a highly nontrivial computation for structured regularizers [4]—*e.g.*, requiring tailored network flow algorithms in existing cases [5, 11, 12]—it yields dense intermediate iterates. Recently, [6] has demonstrated a class of regularizers where the corresponding PUs can be computed by a sequence of submodular function minimizations, but such an approach remains expensive.

Instead, in this paper, we demonstrate that an alternative approach can be more effective for many structured regularizers. We base our development on the *generalized conditional gradient* (GCG) algorithm [13, 14], which also demonstrates promise for sparse model optimization. Although GCG possesses a slower convergence rate than APG, it demonstrates competitive performance if its updates are interleaved with local optimization [14–16]. Moreover, GCG produces sparse intermediate iterates, which allows additional sparsity control. Importantly, unlike APG, GCG requires computing the *polar* of the regularizer, instead of the PU, in each step. This difference allows important new approaches for characterizing and evaluating structured sparse regularizers.

Our first main contribution is to characterize a rich class of structured sparse regularizers that allow efficient computation of their polar operator. In particular, motivated by [6], we consider a family of structured sparse regularizers induced by a cost function on variable subsets. By introducing a "lifting" construction, we show how these regularizers can be expressed as linear functions, which after some reformulation, allows efficient evaluation by a simple linear program (LP). Important examples covered include overlapping group lasso [5] and path regularization in directed acyclic graphs [12]. By exploiting additional structure in these cases, the LP can be reduced to a piecewise

linear objective over a simple domain, allowing further reduction in computation time via smoothing [17]. For example, for the overlapping group lasso with n groups where each variable belongs to at most r groups, the cost of evaluating the polar operator can be reduced from $O(rn^3)$ to $O(rn\sqrt{n}/\epsilon)$ for a desired accuracy of ϵ. Encouraged by the superior performance of GCG in these cases, we then provide a simple reduction of the polar operator to the PU. This reduction makes it possible to extend GCG to cases where the PU is easy to compute. To illustrate the usefulness of this reduction we provide an efficient new algorithm for solving the fused latent lasso [18].

2 Structured Sparse Models

Consider the standard regularized risk minimization framework

$$\min_{\mathbf{w}\in\mathbb{R}^n} f(\mathbf{w}) + \lambda\,\Omega(\mathbf{w}), \tag{1}$$

where f is the empirical risk, assumed to be convex with a Lipschitz continuous gradient, and Ω is a convex, positively homogeneous regularizer, *i.e.* a *gauge* [19, §4]. Let $2^{[n]}$ denote the power set of $[n] := \{1, \ldots, n\}$, and let $\overline{\mathbb{R}}_+ := \mathbb{R}_+ \cup \{\infty\}$. Recently, [6] has established a principled method for deriving regularizers from a subset cost function $F : 2^{[n]} \to \overline{\mathbb{R}}_+$ based on defining the gauge:

$$\Omega_F(\mathbf{w}) = \inf\{\gamma \geq 0 \colon \mathbf{w} \in \gamma\,\mathrm{conv}(S_F)\}, \text{ where } S_F = \big\{\mathbf{w}_A \colon \|\mathbf{w}_A\|_{\tilde{p}}^p = 1/F(A), \emptyset \neq A \subseteq [n]\big\}. \tag{2}$$

Here γ is a scalar, $\mathrm{conv}(S_F)$ denotes the convex hull of the set S_F, $\tilde{p}, p \geq 1$ with $\frac{1}{\tilde{p}} + \frac{1}{p} = 1$, $\|\cdot\|_p$ throughout is the usual ℓ_p-norm, and $\mathbf{w}_A$ denotes a duplicate of $\mathbf{w}$ with all coordinates not in A set to 0. Note that we have tacitly assumed $F(A) = 0$ iff $A = \emptyset$ in (2). The gauge Ω_F defined in (2) is also known as the *atomic norm* with the set of atoms S_F [20]. It will be useful to recall that the *polar* of a gauge Ω is defined by [19, §15]:

$$\Omega^\circ(\mathbf{g}) := \sup_{\mathbf{w}}\{\langle \mathbf{g}, \mathbf{w}\rangle : \Omega(\mathbf{w}) \leq 1\}. \tag{3}$$

In particular, the polar of a norm is its dual norm. (Recall that any norm is also a gauge.) For the specific gauge Ω_F defined in (2), its polar is simply the support function of S_F [19, Theorem 13.2]:

$$\Omega_F^\circ(\mathbf{g}) = \max_{\mathbf{w}\in S_F} \langle \mathbf{g}, \mathbf{w}\rangle = \max_{\emptyset\neq A\subseteq[n]} \|\mathbf{g}_A\|_p \,/[F(A)]^{1/p}. \tag{4}$$

(The first equality uses the definition of support function, and the second follows from (2).) By varying $\tilde{p}$ and F, one can generate a class of sparsity inducing regularizers that includes most current proposals [6]. For instance, if $F(A) = 1$ whenever $|A|$ (the cardinality of A) is 1, and $F(A) = \infty$ for $|A| > 1$, then Ω_F° is the ℓ_∞ norm and Ω_F is the usual ℓ_1 norm. More importantly, one can encode structural information through the cost function F, which selects and establishes preferences over the set of atoms S_F. As pointed out in [6], when F is submodular, (4) can be evaluated by a secant method with submodular minimizations ([21, §8.4], see also Appendix B). However, as we will show, it is possible to do significantly better by completely avoiding submodular optimization. Before presenting our main results, we first review the state of the art for solving (1), and demonstrate how the performance of current methods can hinge on efficient computation of (4).

2.1 Optimization Algorithms

A standard approach for minimizing (1) is the *accelerated proximal gradient* (APG) algorithm [9, 10], where each iteration involves solving the *proximal update* (PU): $\mathbf{w}_{k+1} = \arg\min_{\mathbf{w}} \langle \mathbf{d}_k, \mathbf{w}\rangle + \frac{1}{2s_k}\|\mathbf{w} - \mathbf{w}_k\|_2^2 + \lambda\Omega_F(\mathbf{w})$, for some step size s_k and descent direction $\mathbf{d}_k$. Although it can be shown that APG finds an ϵ accurate solution in $O(1/\sqrt{\epsilon})$ iterations [9, 10], each update can be quite difficult to compute when Ω_F encodes combinatorial structure, as noted in the introduction.

An alternative approach to solving (1) is the *generalized conditional gradient* (GCG) method [13, 14], which has recently received renewed attention. Unlike APG, GCG only requires the *polar operator* of the regularizer Ω_F to be computed in each iteration, given by the argument of (4):

$$\mathbb{P}_F^\circ(\mathbf{g}) \;=\; \arg\max_{\mathbf{w}\in S_F} \langle \mathbf{g}, \mathbf{w}\rangle = F(C)^{\frac{-1}{p}} \arg\max_{\mathbf{w}:\|\mathbf{w}\|_{\tilde{p}}=1} \langle \mathbf{g}_C, \mathbf{w}\rangle \;\text{ for }\; C = \arg\max_{\emptyset\neq A\subseteq[n]} \|\mathbf{g}_A\|_p^p \,/F(A). \tag{5}$$

Algorithm 1 outlines a GCG procedure for solving (1) that only requires the evaluation of $\mathbb{P}_F^\circ$ in each iteration without needing the full PU to be computed. The algorithm is quite simple: Line 3

Algorithm 1 Generalized conditional gradient (GCG) for optimizing (1).

1: Initialize $\mathbf{w}_0 \leftarrow \mathbf{0}$, $s_0 \leftarrow 0$, $\boldsymbol{\ell}_0 \leftarrow \mathbf{0}$.
2: **for** $k = 0, 1, \ldots$ **do**
3: Polar operator: $\mathbf{v}_k \leftarrow \mathbb{P}_F^\circ(\mathbf{g}_k)$, $A_k \leftarrow C(\mathbf{g}_k)$, where $\mathbf{g}_k = -\nabla f(\mathbf{w}_k)$ and C is defined in (5).
4: 2-D Conic search: $(\alpha, \beta) := \arg\min_{\alpha \geq 0, \beta \geq 0} f(\alpha \mathbf{w}_k + \beta \mathbf{v}_k) + \lambda(\alpha s_k + \beta)$.
5: Local re-optimization: $\{\mathbf{u}^i\}_1^k := \arg\min_{\{\mathbf{u}^i = \mathbf{u}^i_{A_i}\}} f(\sum_i \mathbf{u}^i) + \lambda \sum_i F(A_i)^{\frac{1}{p}} \|\mathbf{u}^i\|_{\tilde{p}}$
where the $\{\mathbf{u}^i\}$ are initialized by $\mathbf{u}^i = \alpha \boldsymbol{\ell}_i$ for $i < k$ and $\mathbf{u}^i = \beta \mathbf{v}^i$ for $i = k$.
6: $\mathbf{w}_{k+1} \leftarrow \sum_i \mathbf{u}^i$, $\boldsymbol{\ell}_i \leftarrow \mathbf{u}^i$ for $i \leq k$, $s_{k+1} \leftarrow \sum_i F(A_i)^{\frac{1}{p}} \|\mathbf{u}^i\|_{\tilde{p}}$.
7: **end for**

evaluates the polar operator, which provides a descent direction $\mathbf{v}_k$; Line 4 finds the optimal step sizes for combining the current iterate $\mathbf{w}_k$ with the direction $\mathbf{v}_k$; and Line 5 locally improves the objective (1) by maintaining the same support patterns but re-optimizing the parameters. It has been shown that GCG can find an ϵ accurate solution to (1) in $O(1/\epsilon)$ steps, provided only that the polar (5) is computed to ϵ accuracy [14]. Although GCG has a slower theoretical convergence rate than APG, the introduction of local optimization (Line 5) often yields faster convergence in practice [14–16]. Importantly, Line 5 does not increase the sparsity of the intermediate iterates. Our main goal in this paper therefore is to extend this GCG approach to structured sparse models by developing efficient algorithms for computing the polar operator for the structured regularizers defined in (2).

3 Polar Operators for Atomic Norms

Let $\mathbf{1}$ denote the vector of all 1s with length determined by context. Our first main contribution is to develop a general class of atomic norm regularizers whose polar operator (5) can be computed efficiently. To begin, consider the case of a (partially) *linear* function F where there exists a $\mathbf{c} \in \mathbb{R}^n$ such that $F(A) = \langle \mathbf{c}, \mathbf{1}_A \rangle$ for all $A \in \operatorname{dom} F$ (note that the domain need not be a lattice). A few useful regularizers can be generated by linear functions: for example, the ℓ_1 norm can be derived from $F(A) = \langle \mathbf{1}, \mathbf{1}_A \rangle$ for $|A| = 1$, which is linear. Unfortunately, linearity is too restrictive to capture most structured regularizers of interest, therefore we will need to expand the space of functions F we consider. To do so, we introduce the more general class of *marginalized linear functions*: we say that F is marginalized linear if there exists a nonnegative linear function M on an extended domain $2^{[n+l]}$ such that its marginalization to $2^{[n]}$ is exactly F:

$$F(A) = \min_{B : A \subseteq B \subseteq [n+l]} M(B), \quad \forall\, A \subseteq [n]. \tag{6}$$

Essentially, such a function F is "lifted" to a larger domain where it becomes linear. The key question is whether the polar Ω_F° can be efficiently evaluated for such functions.

To develop an efficient procedure for computing the polar Ω_F°, first consider the simpler case of computing the polar Ω_M° for a nonnegative linear function M. Note that by linearity the function M can be expressed as $M(B) = \langle \mathbf{b}, \mathbf{1}_B \rangle$ for $B \in \operatorname{dom} M \subseteq 2^{[n+l]}$ ($\mathbf{b} \in \mathbb{R}_+^{n+l}$). Since the effective domain of M need not be the whole space in general, we make use of the specialized polytope:

$$P := \operatorname{conv}\{\mathbf{1}_B : B \in \operatorname{dom} M\} \subseteq [0,1]^{n+l}. \tag{7}$$

Note P may have exponentially many faces. From the definition (4) one can then re-express the polar Ω_M° as:

$$\Omega_M^\circ(\mathbf{g}) = \max_{\emptyset \neq B \in \operatorname{dom} M} \|\mathbf{g}_B\|_p / M(B)^{1/p} = \left(\max_{\mathbf{0} \neq \mathbf{w} \in P} \frac{\langle \tilde{\mathbf{g}}, \mathbf{w} \rangle}{\langle \mathbf{b}, \mathbf{w} \rangle} \right)^{1/p} \quad \text{where } \tilde{g}_i = |g_i|^p \ \ \forall i, \tag{8}$$

where we have used the fact that the linear-fractional objective must attain its maximum at vertices of P; that is, at $\mathbf{1}_B$ for some $B \in \operatorname{dom} M$. Although the linear-fractional program (8) can be reduced to a sequence of LPs using the classical method of [22], a single LP suffices for our purposes. Indeed, let us first remove the constraint $\mathbf{w} \neq \mathbf{0}$ by considering the alternative polytope:

$$Q := P \cap \{\mathbf{w} \in \mathbb{R}^{n+l} : \langle \mathbf{1}, \mathbf{w} \rangle \geq 1\}. \tag{9}$$

As shown in Appendix A, all vertices of Q are scalar multiples of the nonzero vertices of P. Since the objective in (8) is scale invariant, we can restrict the constraints to $\mathbf{w} \in Q$. Then, by applying transformations $\tilde{\mathbf{w}} = \mathbf{w} / \langle \mathbf{b}, \mathbf{w} \rangle$, $\sigma = 1 / \langle \mathbf{b}, \mathbf{w} \rangle$, problem (8) can be equivalently re-expressed by:

$$\max_{\tilde{\mathbf{w}}, \sigma > 0} \ \langle \tilde{\mathbf{g}}, \tilde{\mathbf{w}} \rangle, \ \text{subject to } \ \tilde{\mathbf{w}} \in \sigma Q, \ \langle \mathbf{b}, \tilde{\mathbf{w}} \rangle = 1. \tag{10}$$

Of course, whether this LP can be solved efficiently depends on the structure of Q (and of P indeed).

Finally, we note that the same formulation allows the polar to be efficiently computed for a *marginalized* linear function F via a simple reduction: Consider any $\mathbf{g} \in \mathbb{R}^n$ and let $[\mathbf{g}; \mathbf{0}] \in \mathbb{R}^{n+l}$ denote $\mathbf{g}$ padded by l zeros. Then $\Omega_F^\circ(\mathbf{g}) = \Omega_M^\circ([\mathbf{g}; \mathbf{0}])$ for all $\mathbf{g} \in \mathbb{R}^n$ because

$$\max_{\emptyset \neq A \subseteq [n]} \frac{\|\mathbf{g}_A\|_p^p}{F(A)} = \max_{\emptyset \neq A \subseteq [n]} \frac{\|\mathbf{g}_A\|_p^p}{\min_{B: A \subseteq B \subseteq [n+l]} M(B)} = \max_{\emptyset \neq A \subseteq B} \frac{\|\mathbf{g}_A\|_p^p}{M(B)} = \max_{B: \emptyset \neq B \subseteq [n+l]} \frac{\|[\mathbf{g}; \mathbf{0}]_B\|_p^p}{M(B)}. \quad (11)$$

To see the last equality, fixing B the optimal A is attained at $A = B \cap [n]$. If $B \cap [n]$ is empty, then $\|[\mathbf{g}; \mathbf{0}]_B\| = 0$ and the corresponding B cannot be the maximizer of the last term, unless $\Omega_F^\circ(\mathbf{g}) = 0$ in which case it is easy to see $\Omega_M^\circ([\mathbf{g}; \mathbf{0}]) = 0$.

Although we have kept our development general so far, the idea is clear: once an appropriate "lifting" has been found so that the polytope Q in (9) can be compactly represented, the polar (5) can be reformulated as the LP (10), for which efficient implementations can be sought. We now demonstrate this new methodology for the two important structured regularizers: group sparsity and path coding.

3.1 Group Sparsity

For a general formulation of group sparsity, let $\mathcal{G} \subseteq 2^{[n]}$ be a set of variable groups (subsets) that possibly overlap [3, 6, 7]. Here we use $i \in [n]$ to index variables and $G \in \mathcal{G}$ to index groups. Consider the cost function over variable groups $F_{\mathrm{g}} : 2^{[n]} \to \mathbb{R}_+$ defined by:

$$F_{\mathrm{g}}(A) = \sum_{G \in \mathcal{G}} c_G \, \mathbb{I}(A \cap G \neq \emptyset), \quad (12)$$

where c_G is a nonnegative cost and $\mathbb{I}$ is an indicator such that $\mathbb{I}(\cdot) = 1$ if its argument is true, and 0 otherwise. The value $F_{\mathrm{g}}(A)$ provides a weighted count of how many groups overlap with A. Unfortunately, F_{g} is not linear, so we need to re-express it to recover an efficient polar operator. To do so, augment the domain by adding $l = |\mathcal{G}|$ variables such that each new variable G corresponds to a group G. Then define a weight vector $\mathbf{b} \in \mathbb{R}_+^{n+l}$ such that $b_i = 0$ for $i \leq n$ and $b_G = c_G$ for $n < G \leq n + l$. Finally, consider the linear cost function $M_{\mathrm{g}} : 2^{[n+l]} \to \overline{\mathbb{R}}_+$ defined by:

$$M_{\mathrm{g}}(B) = \langle \mathbf{b}, \mathbf{1}_B \rangle \text{ if } i \in B \Rightarrow G \in B, \; \forall \, i \in G \in \mathcal{G}; \;\; M_{\mathrm{g}}(B) = \infty \text{ otherwise.} \quad (13)$$

The constraint ensures that if a variable $i \leq n$ appears in the set B, then every variable G corresponding to a group G that contains i must also appear in B. By construction, M_{g} is a nonnegative linear function. It is also easy to verify that F_{g} satisfies (6) with respect to M_{g}.

To compute the corresponding polar, observe that the effective domain of M_{g} is a lattice, hence (4) can be solved by combinatorial methods. However, we can do better by exploiting problem structure in the LP. For example, observe that the polytope (7) can now be compactly represented as:

$$P_{\mathrm{g}} = \{\mathbf{w} \in \mathbb{R}^{n+l} : \mathbf{0} \leq \mathbf{w} \leq \mathbf{1}, w_i \leq w_G, \forall \, i \in G \in \mathcal{G}\}. \quad (14)$$

Indeed, it is easy to verify that the integral vectors in P_{g} are precisely $\{\mathbf{1}_B : B \in \operatorname{dom} M_{\mathrm{g}}\}$. Moreover, the linear constraint in (14) is totally unimodular (TUM) since it is the incidence matrix of a bipartite graph (variables and groups), hence P_{g} is the convex hull of its integral vectors [23]. Using the fact that the scalar σ in (10) admits a closed form solution $\sigma = \langle \mathbf{1}, \tilde{\mathbf{w}} \rangle$ in this case, the LP (10) can be reduced to:

$$\max_{\tilde{\mathbf{w}}} \; \sum_{i \in [n]} \tilde{g}_i \min_{G: i \in G \in \mathcal{G}} \tilde{w}_G, \;\; \text{subject to } \; \tilde{\mathbf{w}} \geq 0, \; \sum_{G \in \mathcal{G}} b_G \tilde{w}_G = 1. \quad (15)$$

Note only $\{\tilde{w}_G\}$ appear in the problem as implicitly $\tilde{w}_i = \min_{G: i \in G} \tilde{w}_G$, $\forall \, i \in [n]$. This is now just a piecewise linear objective over a (reweighted) simplex. Since projecting to a simplex can be performed in linear time, the smoothing method of [17] can be used to obtain a very efficient implementation. We illustrate a particular case where each variable $i \in [n]$ belongs to at most $r > 1$ groups. (Appendix D considers when the groups form a directed acyclic graph.)

Proposition 1 *Let $h(\tilde{\mathbf{w}})$ denote the* negated *objective of* (15). *Then for any $\epsilon > 0$, $h_\epsilon(\tilde{\mathbf{w}}) := \frac{\epsilon}{n \log r} \sum_{i \in [n]} \log \sum_{G: i \in G} r^{-n \tilde{g}_i \tilde{w}_G / \epsilon}$ satisfies: (i) the gradient of h_ϵ is $\left(\frac{n}{\epsilon} \|\tilde{\mathbf{g}}\|_\infty^2 \log r\right)$-Lipschitz, (ii) $h(\tilde{\mathbf{w}}) - h_\epsilon(\tilde{\mathbf{w}}) \in (-\epsilon, 0]$ for all $\tilde{\mathbf{w}}$, and (iii) the gradient of h_ϵ can be computed in $O(nr)$ time.*

(The proof is given in Appendix C.) With this construction, APG can be run on h_ϵ to achieve a 2ϵ accurate solution to (15) within $O(\frac{1}{\epsilon}\sqrt{n \log r})$ steps [17], using a total time cost of $O(\frac{nr}{\epsilon}\sqrt{n \log r})$. Note that this is significantly cheaper than the $O(n^2(l+n)r)$ worst case complexity of [11, Algorithm 2]. More importantly, we gain explicit control of the trade-off between accuracy ϵ and computational cost. A detailed comparison to related approaches is given in Appendix B.1 and E.

3.2 Path Coding

Another interesting regularizer, recently investigated by [12], is determined by path costs in a directed acyclic graph (DAG) defined over the set of variables $i \in [n]$. For convenience, we add two nodes, a source s and a sink t, with dummy edges (s,i) and (i,t) for all $i \in [n]$. An (s,t)-path (or simply path) is then given by a sequence $(s,i_1),(i_1,i_2),\ldots,(i_{k-1},i_k),(i_k,t)$ with $k \geq 1$. A non-negative cost is associated with each edge including (s,i) and (i,t), so the cost of a path is the sum of its edge costs. A regularizer can then be defined by (2) applied to the cost function $F_{\mathsf{p}} : 2^{[n]} \to \overline{\mathbb{R}}_+$

$$F_{\mathsf{p}}(A) = \begin{cases} \text{cost of the path} & \text{if the nodes in } A \text{ form an } (s,t)\text{-path (unique for DAG)} \\ \infty & \text{if such a path does not exist} \end{cases}. \tag{16}$$

Note F_{p} is *not* submodular. Although F_{p} is not linear, a similar "lifting" construction can be used to show that it is marginalized linear, hence it supports efficient computation of the polar. To explain the construction, let $V := [n] \cup \{s,t\}$ be the node set including s and t, E be the edge set including (s,i) and (i,t), $T = V \cup E$, and let $\mathbf{b} \in \mathbb{R}_+^{|T|}$ be the concatenation of zeros for node costs and the given edge costs. Let $m := |E|$ be the number of edges. It is then easy to verify that F_{p} satisfies (6) with respect to the linear cost function $M_{\mathsf{p}} : 2^T \to \overline{\mathbb{R}}_+$ defined by:

$$M_{\mathsf{p}}(B) = \langle \mathbf{b}, \mathbf{1}_B \rangle \text{ if } B \text{ represents a path; } \infty \text{ otherwise.} \tag{17}$$

To efficiently compute the resulting polar, we consider the form (8) using $\tilde{g}_i = |g_i|^p \ \forall i$ as before:

$$\Omega^\circ_{M_{\mathsf{p}}}(\mathbf{g}) = \max_{\mathbf{0} \neq \mathbf{w} \in [0,1]^{|T|}} \frac{\langle \tilde{\mathbf{g}}, \mathbf{w} \rangle}{\langle \mathbf{b}, \mathbf{w} \rangle}, \quad \text{s.t.} \quad w_i = \sum\nolimits_{j:(i,j)\in E} w_{ij} = \sum\nolimits_{k:(k,i)\in E} w_{ki}, \ \forall i \in [n]. \tag{18}$$

Here the constraints form the well-known flow polytope whose vertices are exactly all the paths in a DAG. Similar to (15), the normalized LP (10) can be simplified by solving for the scalar σ to obtain:

$$\max_{\tilde{\mathbf{w}} \geq \mathbf{0}} \sum_{i \in [n]} \tilde{g}_i \left(\sum_{j:(i,j)\in E} \tilde{w}_{ij} + \sum_{k:(k,i)\in E} \tilde{w}_{ki} \right), \text{ s.t. } \langle \mathbf{b}, \tilde{\mathbf{w}} \rangle = 1, \ \sum_{j:(i,j)\in E} \tilde{w}_{ij} = \sum_{k:(k,i)\in E} \tilde{w}_{ki}, \ \forall i \in [n]. \tag{19}$$

Due to the extra constraints, the LP (19) is more complicated than (15) obtained for group sparsity. Nevertheless, after some reformulation (essentially dualization), (19) can still be converted to a simple piecewise linear objective, hence it is amenable to smoothing; see Appendix F for details. To find a 2ϵ accurate solution, the cutting plane method takes $O(\frac{mn}{\epsilon^2})$ computations to optimize the nonsmooth piecewise linear objective, while APG needs $O(\frac{1}{\epsilon}\sqrt{n})$ steps to optimize the smoothed objective, using a total time cost of $O(\frac{m}{\epsilon}\sqrt{n})$. This too is faster than the $O(nm)$ worst case complexity of [12, Appendix D.5] in the regime where n is large and the desired accuracy ϵ is moderate.

4 Generalizing Beyond Atomic Norms

Although we find the above approach to be effective, many useful regularizers are not expressed in form of an atomic norm (2), which makes evaluation of the polar a challenge and thus creates difficulty in applying Algorithm 1. For example, another important class of structured sparse regularizers is given by an alternative, composite gauge construction:

$$\Omega_s(\mathbf{w}) = \sum\nolimits_i \kappa_i(\mathbf{w}), \text{ where } \kappa_i \text{ is a } \textit{closed} \text{ gauge that can be different for different } i. \tag{20}$$

The polar for such a regularizer is given by $\Omega_s^\circ(\mathbf{g}) = \inf\{\max_i \kappa_i^\circ(\mathbf{w}^i) : \sum_i \mathbf{w}^i = \mathbf{g}\}$, where each $\mathbf{w}^i$ is an independent vector and κ_i° corresponds to the polar of κ_i (proof given in Appendix H). Unfortunately, a polar in this form does not appear to be easy to compute. However, for some regularizers in the form (20) the following proximal objective can indeed be computed efficiently:

$$\mathsf{Prox}_\Omega(\mathbf{g}) = \min_{\boldsymbol{\theta}} \tfrac{1}{2}\|\mathbf{g}-\boldsymbol{\theta}\|_2^2 + \Omega(\boldsymbol{\theta}), \qquad \mathsf{ArgProx}_\Omega(\mathbf{g}) = \arg\min_{\boldsymbol{\theta}} \tfrac{1}{2}\|\mathbf{g}-\boldsymbol{\theta}\|_2^2 + \Omega(\boldsymbol{\theta}). \tag{21}$$

The key observation is that computing Ω° can be efficiently reduced to just computing Prox_Ω.

Proposition 2 *For any closed gauge Ω, its polar Ω° can be equivalently expressed by:*

$$\Omega^\circ(\mathbf{g}) = \inf\{\, \zeta \geq 0 : \mathsf{Prox}_{\zeta\Omega}(\mathbf{g}) = \tfrac{1}{2}\|\mathbf{g}\|_2^2 \,\}. \tag{22}$$

(The proof is included in Appendix I.) Since the left hand side of the inner constraint is decreasing in ζ, one can efficiently compute the polar Ω° by a simple root finding search in ζ. Thus, regularizers in the form of (20) can still be accommodated in an efficient GCG method in the form of Algorithm 1.

4.1 Latent Fused Lasso

To demonstrate the usefulness of this reduction we consider the recently proposed latent fused lasso model [18], where for given data $X \in \mathbb{R}^{m\times n}$ one seeks a dictionary matrix $W \in \mathbb{R}^{m\times t}$ and coefficient matrix $U \in \mathbb{R}^{t\times n}$ that allow X to be accurately reconstructed from a dictionary that has desired structure. In particular, for a reconstruction loss f, the problem is specified by:

$$\min_{W,U\in\mathcal{U}} f(WU, X) + \Omega_p(W), \quad \text{where} \quad \Omega_p(W) = \sum_i \left(\lambda_1 \|W_{:i}\|_p + \lambda_2 \|W_{:i}\|_{\mathsf{TV}}\right), \tag{23}$$

such that $\|\cdot\|_{\mathsf{TV}}$ is given by $\|\mathbf{w}\|_{\mathsf{TV}} = \sum_{j=1}^{m-1} |w_{j+1} - w_j|$ and $\|\cdot\|_p$ is the usual ℓ_p-norm. The fused lasso [24] corresponds to $p = 1$. Note that U is constrained to be in a compact set $\mathcal{U}$ to avoid degeneracy. To ease notation, we assume w.l.o.g. $\lambda_1 = \lambda_2 = 1$.

The main motivation for this regularizer arises from biostatistics, where one wishes to identify DNA copy number variations *simultaneously* for a group of related samples [18]. In this case the total variation norm $\|\cdot\|_{\mathsf{TV}}$ encourages the dictionary to vary smoothly from entry to entry while the ℓ_p norm shrinks the dictionary so that few latent features are selected. Conveniently, Ω_p decomposes along the columns of W, so one can apply the reduction in Proposition 2 to compute its polar assuming Prox_{Ω_p} can be efficiently computed. Solving Prox_{Ω_p} appears non-trivial due to the composition of two *overlapping* norms, however [25] showed that for $p = 1$ the polar can be solved efficiently by computing Prox for each of the two norms successively. Here we extend this results by proving in Appendix J that the same fact holds for any ℓ_p norm.

Proposition 3 *For any* $1 \le p \le \infty$, $\mathsf{ArgProx}_{\|\cdot\|_{\mathsf{TV}}+\|\cdot\|_p}(\mathbf{w}) = \mathsf{ArgProx}_{\|\cdot\|_p}\big(\mathsf{ArgProx}_{\|\cdot\|_{\mathsf{TV}}}(\mathbf{w})\big)$.

Since $\mathsf{Prox}_{\|\cdot\|_p}$ is easy to compute, the only remaining problem is to develop an efficient algorithm for computing $\mathsf{Prox}_{\|\cdot\|_{\mathsf{TV}}}$. Although [26] has recently proposed an approximate iterative method, we provide an algorithm in Appendix K that is able to efficiently compute the exact solution. Therefore, by combining this result with Propositions 2 and 3 we are able to efficiently compute the polar Ω_p° and hence apply Algorithm 1 to solving (23) with respect to W.

5 Experiments

To investigate the effectiveness of these computational schemes we considered three applications: group lasso, path coding, and latent fused lasso. All algorithms were implemented in Matlab unless otherwise noted.

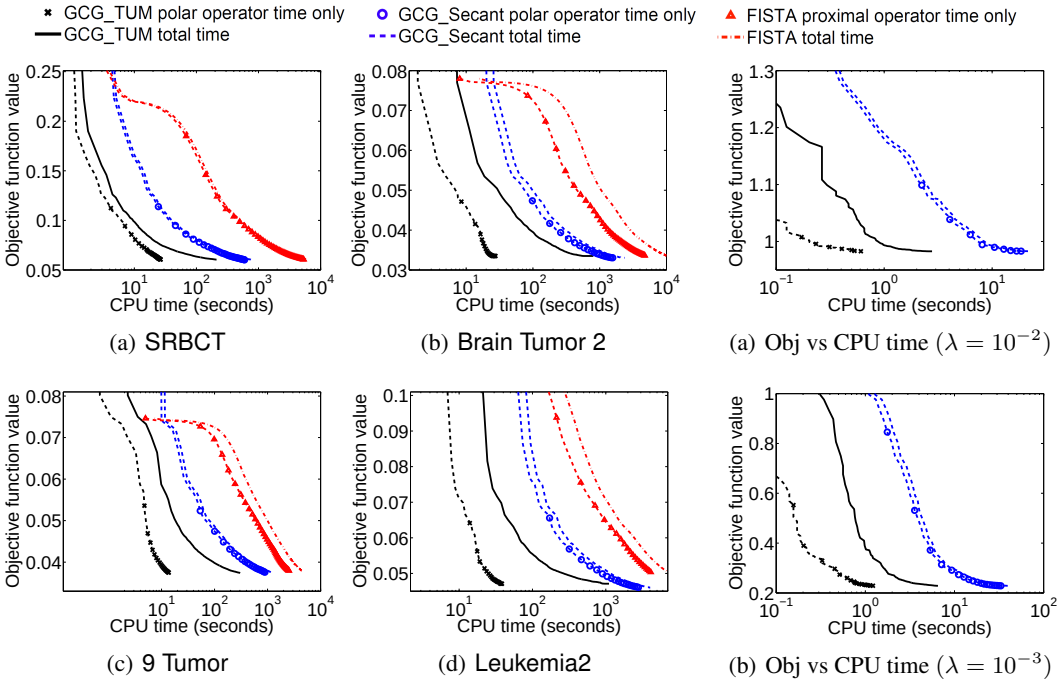
5.1 Group Lasso: CUR-like Matrix Factorization

Our first experiment considered an example of group lasso that is inspired by CUR matrix factorization [27]. Given a data matrix $X \in \mathbb{R}^{n\times d}$, the goal is to compute an approximate factorization $X \approx CUR$, such that C contains a subset of c columns from X and R contains a subset of r rows from X. Mairal et al. [11, §5.3] proposed a convex relaxation of this problem:

$$\min_W \tfrac{1}{2}\|X - XWX\|^2 + \lambda\big(\textstyle\sum_i \|W_{i:}\|_\infty + \sum_j \|W_{:j}\|_\infty\big). \tag{24}$$

Conveniently, the regularizer fits the development of Section 3.1, with $p = 1$ and the groups defined to be the rows and columns of W. To evaluate different methods, we used four gene-expression data sets [28]: SRBCT, Brain_Tumor_2, 9_Tumor, and Leukemia2, of sizes 83×2308, 50×10367, 60×5762, and 72×11225, respectively. The data matrices were first centered columnwise and then rescaled to have unit Frobenius norm.

Algorithms. We compared three algorithms: GCG (Algorithm 1) with our polar operator which we call GCG_TUM, GCG with the polar operator of [11, Algorithm 2] (GCG_Secant), and APG (see Section 2.1). The PU in APG uses the routine mexProximalGraph from the SPAMS package [29]. The polar operator of GCG_Secant was implemented with a mex wrapper of a max-flow package [30], while GCG_TUM used L-BFGS to find an optimal solution $\{w_G^*\}$ for the smoothed version of



(a) SRBCT (b) Brain Tumor 2 (c) 9 Tumor (d) Leukemia2

Figure 1: Convex CUR matrix factorization results.

(a) Obj vs CPU time ($\lambda = 10^{-2}$) (b) Obj vs CPU time ($\lambda = 10^{-3}$)

Figure 2: Path coding results.

(15) given in Proposition 1, with smoothing parameter ϵ set to 10^{-3}. To recover an integral solution it suffices to find an optimal solution to (15) that has the form $w_G = c$ for some groups and $w_G = 0$ for the remainder (such a solution must exist). So we sorted $\{w_G^*\}$ and set the w_G of the smallest k groups to 0, and w_G for the remaining groups set to a common value that satisfies the constraint. The best k can be recovered from $\{0, 1, \ldots, |\mathcal{G}| - 1\}$ in $O(nr)$ time. See more details in Appendix G. Both GCG methods relinquish local optimization (step 5) in Algorithm 1, but use a totally corrective variant of step 4, which allows efficient optimization by L-BFGS-B via pre-computing $X\mathbb{P}^{\circ}_{F_{\mathcal{G}}}(\mathbf{g}_k)X$.

Results. For simplicity, we tested three values for λ: 10^{-3}, 10^{-4}, and 10^{-5}, which led to increasingly dense solutions. Due to space limitations we only show in Figure 1 the results for $\lambda = 10^{-4}$ which gives moderately sparse solutions. On these data sets, GCG_TUM proves to be an order of magnitude faster than GCG_Secant in computing the polar. As [11] observes, network flow based algorithms often find solutions in practice far more quickly than their theoretical bounds. Thanks to the efficiency of totally corrective update, almost all computations taken by GCG_Secant were devoted to the polar operator. Therefore the acceleration proffered by GCG_TUM in computing the polar leads to a reduction of *overall* optimization time by at least 50%. Finally, APG is always even slower than GCG_Secant by an order of magnitude, with PU taking up the most computation.

5.2 Path Coding

Following [12, §4.3], we consider a logistic regression problem where one is given training examples $\mathbf{x}_i \in \mathbb{R}^n$ with corresponding labels $y_i \in \{-1, 1\}$. For this problem, we formulate (1) with a path coding regularizer Ω_{F_p} and the empirical risk:

$$f(\mathbf{w}) = \textstyle\sum_i \frac{1}{n_i} \log(1 + \exp(-y_i \langle \mathbf{w}, \mathbf{x}_i \rangle)), \tag{25}$$

where n_i is the number of examples that share the same label as y_i. We used the breast cancer data set for this experiment, which consists of 8141 genes and 295 tumors [31]. The gene network is adopted from [32]. Similar to [12, §4.3], we removed all isolated genes (nodes) to which no edge is incident, randomly oriented the raw edges, and removed cycles to form a DAG using the function mexRemoveCyclesGraph in SPAMS. This resulted in 34864 edges and $n = 7910$ nodes.

Algorithms. We again considered three methods: APG, GCG with our polar operator (GCG_TUM), and GCG with the polar operator from [12, Algorithm 1], which we label as GCG_Secant. The PU in APG uses the routine mexProximalPathCoding from SPAMS, which solves a quadratic network flow problem. It turns out the time cost for a single call of the PU was enough for GCG_TUM and

GCG_Secant to converge to a final solution, and so the APG result is not included in our plots. We implemented the polar operator for GCG_Secant based on Matlab's built-in shortest path routine graphshortestpath (C++ wrapped by mex). For GCG_TUM, we used cutting plane to solve a variant of the dual of (19) (see Appendix F), which is much simipler than smoothing in implementation, but exhibits similar efficiency in practice. An integral solution can also be naturally recovered in the course of computing the objective. Again, both GCG methods only used totally corrective updates.

Results. Figure 2 shows the result for path coding, with the regularization coefficient λ set to 10^{-2} and 10^{-3} so that the solution is moderately sparse. Again it is clear that GCG_TUM is an order of magnitude faster than GCG_Secant.

5.3 Latent Fused Lasso

Finally, we compared GCG and APG on the latent fused lasso problem (23). Two algorithms were tested as the PU in APG: our proposed method and the algorithm in [26], which we label as APG-Liu. The synthetic data is generated by following [18]. For each basis (column) of the dictionary, we use the model $\tilde{W}_{ij} = \sum_{s=1}^{S_j} c_s \mathbb{I}(i_s \le i \le i_s + l_s)$, where $S_j \in \{3, 5, 8, 10\}$ specifies the number of consecutive blocks in the j-th basis, $c_s \in \{\pm 1, \pm 2, \pm 3, \pm 4, \pm 5\}$, $i_s \in \{1, \dots, m - 10\}$ and $l_s \in \{5, 10, 15, 20\}$, which are the magnitude, starting position, and length of the s-th block, respectively. Note that we choose c_s, i_s, l_s randomly (and independently for each block s) from their respective sets. The coefficient matrix $\tilde{U}$ are sampled from the Gaussian distribution $N(0, 1)$ (independently for each entry) and normalized to have unit ℓ_2 norm for each row. Finally, we generate the observation matrix $X = \tilde{W}\tilde{U} + \varepsilon$, with added (zero mean and unit variance) Gaussian noise ε. We set the dimension $m = 300$, the number of samples $n = 200$, and the number of bases (latent dimension) $\tilde{t} = 10$.

Since the noise is Gaussian, we choose the squared loss $f(WU, X) = \frac{1}{2}\|X - WU\|_F^2$, but the algorithm is applicable to any other smooth loss as well. To avoid degeneracy, we constrained each row of U to have unit ℓ_2 norm. Finally, to pick an appropriate dictionary size, we tried $t \in \{5, 10, 20\}$, which corresponds to under-, perfect- and over-estimation, respectively. The regularization constants λ_1, λ_2 in Ω_p were chosen from $\{0.01, 0.1, 1, 10, 100\}$.

Note that problem (23) is not jointly convex in W and U, so we followed the same strategy as [18]; that is, we alternatively optimized W and U keeping the other fixed. For each subproblem, we ran both APG and GCG to compare their performance. For space limitations, we only report the running time for the setting $\lambda_1 = \lambda_2 = 0.1$, $t = 20$ and $p \in \{1, 2\}$. In these experiments we observed that the polar typically only requires 5 to 6 calls to Prox. As can be seen from Figure 3, GCG is significantly faster than APG and APG-Liu in reducing the objective. This is due to the greedy nature of GCG, which yields very sparse iterates, and when interleaved with local search achieves fast convergence.

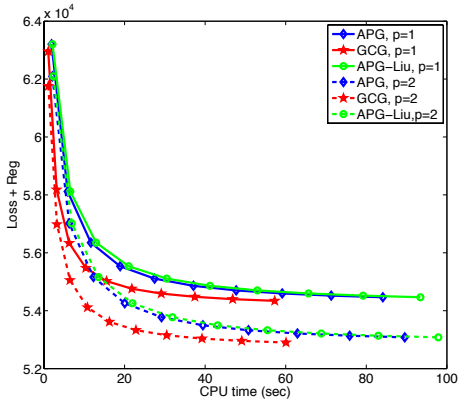


Figure 3: Latent fused lasso.

6 Conclusion

We have identified and investigated a new class of structured sparse regularizers whose polar can be reformulated as a linear program with totally unimodular constraints. By leveraging smoothing techniques, we are able to compute the corresponding polars with significantly better efficiency than previous approaches. When plugged into the GCG algorithm, one can observe significant reductions in run time for both group lasso and path coding regularization. We have further developed a generic scheme for converting an efficient proximal solver to an efficient method for computing the polar operator. This reduction allowed us to develop a fast new method for latent fused lasso. For future work, we plan to study more general subset cost functions and investigate new structured regularizers amenable to our approach. It will also be interesting to extend GCG to handle nonsmooth losses.

References

[1] P. Bühlmann and S. van de Geer. *Statistics for High-Dimensional Data*. Springer, 2011.

[2] Y. Eldar and G. Kutyniok, editors. *Compressed Sensing: Theory and Applications*. Cambridge, 2012.

[3] J. Huang, T. Zhang, and D. Metaxas. Learning with structured sparsity. *JMLR*, 12:3371–3412, 2011.

[4] S. Kim and E. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*, 2010.

[5] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for hierarchical sparse coding. *JMLR*, 12:2297–2334, 2011.

[6] G. Obozinski and F. Bach. Convex relaxation for combinatorial penalties. Technical Report HAL 00694765, 2012.

[7] P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *Annals of Statistics*, 37(6A):3468–3497, 2009.

[8] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[9] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[10] Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140: 125–161, 2013.

[11] J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Convex and network flow optimization for structured sparsity. *JMLR*, 12:2681–2720, 2011.

[12] J. Mairal and B. Yu. Supervised feature selection in graphs with path coding penalties and network flows. *JMLR*, 14:2449–2485, 2013.

[13] M. Dudik, Z. Harchaoui, and J. Malick. Lifted coordinate descent for learning with trace-norm regularizations. In *AISTATS*, 2012.

[14] X. Zhang, Y. Yu, and D. Schuurmans. Accelerated training for matrix-norm regularization: A boosting approach. In *NIPS*, 2012.

[15] S. Laue. A hybrid algorithm for convex semidefinite optimization. In *ICML*, 2012.

[16] B. Mishra, G. Meyer, F. Bach, and R. Sepulchre. Low-rank optimization with trace norm penalty. Technical report, 2011. http://arxiv.org/abs/1112.2318.

[17] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[18] G. Nowak, T. Hastie, J. R. Pollack, and R. Tibshirani. A fused lasso latent feature model for analyzing multi-sample aCGH data. *Biostatistics*, 12(4):776–791, 2011.

[19] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[20] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky. The convex geometry of linear inverse problems. *Foundations of Computational Mathematics*, 12(6):805–849, 2012.

[21] F. Bach. Convex analysis and optimization with submodular functions: a tutorial. Technical Report HAL 00527714, 2010.

[22] W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7), 1967.

[23] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1st edition, 1986.

[24] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B*, 67:91–108, 2005.

[25] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

[26] J. Liu, L. Yuan, and J. Ye. An efficient algorithm for a class of fused lasso problems. In *Conference on Knowledge Discovery and Data Mining*, 2010.

[27] M. Mahoney and P. Drineas. CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

[28] URL `http://www.gems-system.or`.

[29] URL `http://spams-devel.gforge.inria.fr`.

[30] URL `http://drwn.anu.edu.au/index.html`.

[31] M. Van De Vijver *et al.* A gene-expression signature as a predictor of survival in breast cancer. *The New England Journal of Medicine*, 347(25):1999–2009, 2002.

[32] H. Chuang, E. Lee, Y. Liu, D. Lee, and T. Ideker. Network-based classification of breast cancer metastasis. *Molecular Systems Biology*, 3(140), 2007.

Appendix of *Polar Operators for Structured Sparse Estimation*

A Vertices of Q must be scalar multiples of those of P

First note that if $\mathbf{0} \notin P$, we have nothing to prove since $P = Q$. Thus we assume $\mathbf{0} \in P$ below.

Consider an arbitrary vertex $\mathbf{q} \in Q$. Clearly $\mathbf{q} \neq \mathbf{0}$ and $\mathbf{q} \in P$, hence $\mathbf{q} = \sum_{i=1}^n \alpha_i \cdot \mathbf{p}^{(i)}$, where $n \geq 1$, $\alpha_i > 0$, $\langle \mathbf{1}, \boldsymbol{\alpha} \rangle \leq 1$, and $\mathbf{p}^{(i)}$ are nonzero vertices of P. Clearly $\mathbf{p}^{(i)} \in Q$ as $\mathbf{p}^{(i)} \in P$ and $l_i := \langle \mathbf{1}, \mathbf{p}^{(i)} \rangle \geq 1$. It suffices to show $n = 1$. To prove by contradiction, suppose $n \geq 2$.

(i) If $\langle \mathbf{1}, \boldsymbol{\alpha} \rangle = 1$, then $\mathbf{q}$ is a convex combination of at least two points in Q, hence it cannot be a vertex.

(ii) If $\langle \mathbf{1}, \mathbf{q} \rangle = \sum_i \alpha_i l_i = 1$, then $\mathbf{q} = \sum_{i=1}^n (\alpha_i l_i) \frac{\mathbf{p}^{(i)}}{l_i}$. But $\frac{\mathbf{p}^{(i)}}{l_i} \in Q$ as $\frac{\mathbf{p}^{(i)}}{l_i} = \frac{1}{l_i}\mathbf{p}^{(i)} + (1 - \frac{1}{l_i})\mathbf{0} \in P$ and $\langle \mathbf{1}, \mathbf{p}^{(i)} \rangle = l_i \geq 1$. Again contradiction.

(iii) If $\langle \mathbf{1}, \mathbf{q} \rangle > 1$ and $\langle \mathbf{1}, \boldsymbol{\alpha} \rangle < 1$, then $\beta := \frac{1}{\langle \mathbf{1}, \mathbf{q} \rangle} < 1 < \frac{1}{\langle \mathbf{1}, \boldsymbol{\alpha} \rangle} =: \gamma$. Clearly $\beta\mathbf{q} \in Q$ because $\beta\mathbf{q} = \beta\mathbf{q} + (1 - \beta)\mathbf{0} \in P$ and $\langle \mathbf{1}, \beta\mathbf{q} \rangle = 1$. Also $\gamma\mathbf{q} \in Q$ as $\gamma\mathbf{q} = \frac{\sum_{i=1}^n \alpha_i \mathbf{p}^{(i)}}{\sum_{i=1}^n \alpha_i} \in P$ and $\langle \mathbf{1}, \gamma\mathbf{q} \rangle = \frac{\sum_{i=1}^n \alpha_i \langle \mathbf{1}, \mathbf{p}^{(i)} \rangle}{\langle \mathbf{1}, \boldsymbol{\alpha} \rangle} \geq \frac{\sum_{i=1}^n \alpha_i}{\langle \mathbf{1}, \boldsymbol{\alpha} \rangle} = 1$. So $\mathbf{q}$ lies between two points in Q: $\beta\mathbf{q}$ and $\gamma\mathbf{q}$. Contradiction.

Therefore $n = 1$, which completes the proof.

To summarize, we have proved that if $\mathbf{q}$, a vertex of Q, is not a vertex of P, then it must sum to 1 and be a scalar multiple of some vertex of P.

B Polar Oracle via Secant Method and Submodular Minimization

By (5), the key optimization problem in computing the polar operator is

$$\lambda^* = \max_{\emptyset \neq A \subseteq [n]} \frac{\langle \tilde{\mathbf{g}}, \mathbf{1}_A \rangle}{F(A)}, \quad \text{where} \quad \tilde{g}_i = |g_i|^p \,. \tag{26}$$

Let $A^* \in 2^{[n]} \backslash \emptyset$ be a maximizer. The following solution is a slight simplification of [21, §8.4]. Let

$$h(\lambda) := \max_{A \subseteq [n]} \langle \tilde{\mathbf{g}}, \mathbf{1}_A \rangle - \lambda F(A). \tag{27}$$

Note $A = \emptyset$ is allowed here. Clearly $h(\lambda)$ is convex and non-increasing. $h(\lambda) \geq \langle \tilde{\mathbf{g}}, \mathbf{1}_\emptyset \rangle - \lambda F(\emptyset) = 0$. By the definition of λ^*, for all $A \in 2^{[n]} \backslash \emptyset$ we have $\lambda^* \geq \frac{\langle \tilde{\mathbf{g}}, \mathbf{1}_A \rangle}{F(A)}$, *i.e.* $\langle \tilde{\mathbf{g}}, \mathbf{1}_A \rangle - \lambda^* F(A) \leq 0$. So

$$h(\lambda^*) = \max \left\{ \langle \tilde{\mathbf{g}}, \mathbf{1}_\emptyset \rangle - \lambda^* F(\emptyset), \max_{\emptyset \neq A \subseteq [n]} \langle \tilde{\mathbf{g}}, \mathbf{1}_A \rangle - \lambda^* F(A) \right\} = 0. \tag{28}$$

As a result, $h(\lambda) = 0$ for all $\lambda > \lambda^*$. For any $\lambda < \lambda^*$, $\frac{\langle \tilde{\mathbf{g}}, \mathbf{1}_{A^*} \rangle}{F(A^*)} = \lambda^* > \lambda$, and therefore

$$h(\lambda) \geq \langle \tilde{\mathbf{g}}, \mathbf{1}_{A^*} \rangle - \lambda F(A^*) > 0. \tag{29}$$

In summary,

$$\lambda^* = \sup \{\lambda : h(\lambda) > 0\} = \min \{\lambda : h(\lambda) = 0\}, \tag{30}$$

i.e. λ^* is the smallest root of h, which can be easily found by a secant method thanks to the convexity of h. The details are given in Algorithm 2.

Note if $h(\lambda_t) > 0$ upon termination, then the A_t returned must be non-empty. But if $h(\lambda_t) = 0$, then $A_t = \emptyset$ is possible, depending on the solver for the maximization problem in (27). Fortunately, since $\lambda_t = \lambda^*$, it can be easily verified that $\langle \tilde{\mathbf{g}}, \mathbf{1}_{A_{t-1}} \rangle - \lambda^* F(A_{t-1}) = 0$. So we can simply return A_{t-1} without having to customize the solver.

In terms of computational cost, the bottleneck is clearly Step 2 which solves (27) given $\lambda = \lambda_t$. This is deemed as tractable if F is submodular.

Algorithm 2 Polar oracle via secant method

Pick arbitrary $A_0 \in [n] \backslash \emptyset$, and set $\lambda_1 = \frac{\langle \tilde{\mathbf{g}}, \mathbf{1}_{A_0} \rangle}{F(A_0)}$. Clearly $\lambda_1 \leq \lambda^*$ and so $h(\lambda_1) \geq 0$.
for $t = 1, 2, \ldots$ **do**
  Compute $h(\lambda_t)$ by finding an optimal A in the definition of $h(\lambda_t)$ in (27). Call this A as A_t.
  if $h(\lambda_t) \in (0, \epsilon)$ **then**
    return A_t. $\lambda_\epsilon^* = \frac{\langle \tilde{\mathbf{g}}, \mathbf{1}_{A_t} \rangle}{F(A_t)}$ can be at most ϵ smaller than the true λ^*.
  end if
  if $h(\lambda_t) = 0$ **then**
    return A_{t-1}. It follows that $\lambda^* = \frac{\langle \tilde{\mathbf{g}}, \mathbf{1}_{A_{t-1}} \rangle}{F(A_{t-1})}$.
  end if
  Linearize $h(\lambda)$ at λ_t as $\tilde{h}_t(\lambda) = h(\lambda_t) - (\lambda - \lambda_t) F(A_t)$.
  Set λ_{t+1} as the root of $\tilde{h}_t$: $\lambda_{t+1} = \lambda_t + \frac{h(\lambda_t)}{F(A_t)} = \lambda_t + \frac{\langle \tilde{\mathbf{g}}, \mathbf{1}_{A_t} \rangle - \lambda_t F(A_t)}{F(A_t)} = \frac{\langle \tilde{\mathbf{g}}, \mathbf{1}_{A_t} \rangle}{F(A_t)}$. Since h is convex and hence $\tilde{h}_t$ must be upper bounded by h, it follows $\lambda_{t+1} \leq \lambda^*$. Thus $h(\lambda_{t+1}) \geq 0$.
end for

B.1 Network Min-cut Algorithm for Submodular Minimization with Overlapping Group

Next we show the network max-flow/min-cut algorithm for solving (27) in overlapping group lasso. Using the notation and setup in Section 3.1, the problem (27) can be written as

$$\min_{\mathbf{w} \in \{0,1\}^{n+l}} \lambda \sum_{G \in \mathcal{G}} c_G w_G - \sum_{i \in [n]} \tilde{g}_i w_i, \quad s.t. \quad w_G \geq w_i, \ \forall i \in G \in \mathcal{G}. \tag{31}$$

This is obviously equivalent to

$$\min_{\mathbf{w} \in \{0,1\}^{n+l}} \sum_{G \in \mathcal{G}} (\lambda c_G) w_G + \sum_{i \in [n]} \tilde{g}_i (1 - w_i), \quad s.t. \quad w_G \geq w_i, \ \forall i \in G \in \mathcal{G}. \tag{32}$$

Now we show this is exactly a min-cut problem on a directed graph. Let us construct a directed graph with source node s, sink node t, and all nodes w_G and w_i. There is a directed edge from s to each node w_G ($G \in \mathcal{G}$), and the weight is $\eta_G := \lambda c_G$. In addition, there is a directed edge from each node w_i ($i \in [n]$) to the sink t, with weight $\eta_i := \tilde{g}_i$. Finally, for each $i \in G \in \mathcal{G}$, there is a edge from node w_G to w_i, and the weight is $\eta_{G,i} := \infty$.

The min-cut problem essentially divides all nodes in a graph into two groups S and T with $s \in S$ and $t \in T$, and minimizes the sum of the weight of all edges from u to v where $u \in S$ and $v \in T$. Note edges with $u \in T$ and $v \in S$ are not counted into the cut-edge by the definition of min-cut. Let us fix $p_s = 0$, $p_t = 1$, and use $p_i, p_G = 0$ (or 1) if the node belongs to S (or T). Then the min-cut objective for this directed graph can be written as

$$\min_{p_i, p_G \in \{0,1\}} \sum_{i \in G \in \mathcal{G}: p_G = 0, p_i = 1} \eta_{G,i} + \sum_{i \in [n]: p_i = 0} \eta_i + \sum_{G \in \mathcal{G}: p_G = 1} \eta_G. \tag{33}$$

Since $\eta_{G,i} = \infty$, we have to exclude the solutions where $p_G = 0$ and $p_i = 1$. This can be compactly enforced by adding constraints $p_G \geq p_i$. Moreover, it is obvious from $p_i, p_G \in \{0, 1\}$ that

$$\sum_{i \in [n]: p_i = 0} \eta_i = \sum_{i \in [n]} \eta_i (1 - p_i), \qquad \text{and} \qquad \sum_{G \in \mathcal{G}: p_G = 1} \eta_G = \sum_{G \in \mathcal{G}} \eta_G p_G. \tag{34}$$

Substituting them back into (33) and noting the definition of η_i and η_G, it is straightforward to observe the equivalence between (32) and (33), with p_G and p_i corresponding to w_G and w_i respectively.

Finally, by using the well-known equivalence between max-flow and min-cut (problem (33)), it is trivial to write out the max-flow formulation for the graph defined above, which exactly recovers the solution proposed by [11, Algorithm 2]. In comparison, our min-cut formulation is clearly more straightforward because it completely eliminates the dualization step and directly provides the solution to (27).

C Proof of Proposition 1

The proof is based on the well-known duality between strong convexity and smoothness (Lipschitz continuous gradient) [17]. Note that we assume that r, the upper bound on the number of groups each variable can belong to, is greater than 1 since otherwise the problem is trivial.

Proof: Note that there are n variables which we index by i and there are ℓ groups (subsets of variables) which we index by G. The input vector $\tilde{\mathbf{w}} \in \mathbb{R}^n \times \mathbb{R}^\ell$.

Let l_i be the number of groups that contain variable i, and $\mathcal{S}_i := \{\mathbf{s} \in \mathbb{R}_+^{l_i} : \langle \mathbf{1}, \mathbf{s} \rangle = 1\}$ be the $(l_i - 1)$-dimensional simplex. Using the well-known variational representation of max function, we rewrite the (negated) objective h in (15) as

$$h(\tilde{\mathbf{w}}) = \sum_{i \in [n]} \tilde{g}_i \max_{\boldsymbol{\alpha}^{(i)} \in \mathcal{S}_i} \left\{ - \sum_{G: i \in G} \alpha_G^{(i)} \tilde{w}_G \right\} = \max_{\boldsymbol{\alpha}^{(i)} \in \mathcal{S}_i} \sum_{i \in [n]} \sum_{G: i \in G} -\tilde{g}_i \tilde{w}_G \alpha_G^{(i)}, \tag{35}$$

which is to be minimized. Here the second equality follows from the separability of the variables $\boldsymbol{\alpha}^{(i)}$. Fix $\epsilon > 0$ and denote $c := \frac{\epsilon}{n \log r}$. Consider

$$h_\epsilon(\tilde{\mathbf{w}}) = \max_{\boldsymbol{\alpha}^{(i)} \in \mathcal{S}_i} \sum_{i \in [n]} \sum_{G: i \in G} \left(-\tilde{g}_i \tilde{w}_G \alpha_G^{(i)} - c \cdot \alpha_G^{(i)} \log \alpha_G^{(i)} \right),$$

i.e., we add to h the scaled entropy function $-c \sum_{i \in [n], G: i \in G} \alpha_G^{(i)} \log \alpha_G^{(i)}$ whose negation is known to be strongly convex on the simplex (*w.r.t.* the ℓ_1-norm) [17]. Since the entropy is nonnegative, we have for any $\tilde{\mathbf{w}}$, $h(\tilde{\mathbf{w}}) \leq h_\epsilon(\tilde{\mathbf{w}})$ and moreover

$$h_\epsilon(\tilde{\mathbf{w}}) - h(\tilde{\mathbf{w}}) \leq c \max_{\boldsymbol{\alpha}^{(i)} \in \mathcal{S}_i} \sum_{i \in [n]} \sum_{G: i \in G} -\alpha_G^{(i)} \log \alpha_G^{(i)} \leq c \cdot n \log r = \epsilon,$$

where the last inequality is due to the well-known upper bound of the entropy over the probability simplex, *i.e.* entropy attains its maximum when all odds are equally likely. Therefore $h(\tilde{\mathbf{w}}) - h_\epsilon(\tilde{\mathbf{w}}) \in (-\epsilon, 0]$, and we have proved part (ii) of Proposition 1.

By straightforward calculation

$$\begin{aligned} h_\epsilon(\tilde{\mathbf{w}}) &= \sum_{i \in [n]} \max_{\boldsymbol{\alpha}^{(i)} \in \mathcal{S}_i} \sum_{G: i \in G} \left(-\tilde{g}_i \tilde{w}_G \alpha_G^{(i)} - c \cdot \alpha_G^{(i)} \log \alpha_G^{(i)} \right) \\ &= c \sum_{i \in [n]} \log \sum_{G: i \in G} \exp\left(-\frac{\tilde{g}_i \tilde{w}_G}{c} \right), \end{aligned} \tag{36}$$

$$\frac{\partial}{\partial \tilde{w}_G} h_\epsilon(\tilde{\mathbf{w}}) = - \sum_{i: i \in G} \tilde{g}_i p_i(G), \quad \text{where} \quad p_i(G) := \frac{\exp\left(-\frac{\tilde{g}_i \tilde{w}_G}{c} \right)}{\sum_{\tilde{G}: i \in \tilde{G}} \exp\left(-\frac{\tilde{g}_i \tilde{w}_{\tilde{G}}}{c} \right)}. \tag{37}$$

Hence $h_\epsilon(\tilde{\mathbf{w}})$ can be computed in $O(nr)$ time (since the second summation in (36) contains at most r terms). Similarly all $\{p_i(G) : i \in [n], i \in G\}$ can be computed in $O(nr)$ time. Therefore part (iii) of Proposition 1 is established.

Finally, to bound the Lipschitz constant of the gradient of h_ϵ, we observe that $h_\epsilon(\tilde{\mathbf{w}}) = \eta^*(A\tilde{\mathbf{w}})$, where η^* is the Fenchel conjugate of the scaled negative entropy

$$\eta(\boldsymbol{\alpha}) = c \sum_{i \in [n]} \sum_{G: i \in G} \alpha_G^{(i)} \log \alpha_G^{(i)},$$

and A is defined as the matrix satisfying

$$\langle \boldsymbol{\alpha}, A\tilde{\mathbf{w}} \rangle = \sum_{i \in [n]} \sum_{G: i \in G} -\alpha_G^{(i)} \tilde{g}_i \tilde{w}_G.$$

It is known that the scaled negative entropy η is strongly convex with modulus c (*w.r.t.* the ℓ_1-norm). Furthermore, employing ℓ_1 norm on $\boldsymbol{\alpha}$ and ℓ_2 norm on $\tilde{\mathbf{w}}$, the operator norm of the matrix A can be

bounded as

$$\|A\|_{2,1} := \max_{\boldsymbol{\alpha}:\|\boldsymbol{\alpha}\|_1=1} \max_{\tilde{\mathbf{w}}:\|\tilde{\mathbf{w}}\|_2=1} \langle \boldsymbol{\alpha}, A\tilde{\mathbf{w}} \rangle = \max_{\tilde{\mathbf{w}}:\|\tilde{\mathbf{w}}\|_2=1} \max_{\boldsymbol{\alpha}:\|\boldsymbol{\alpha}\|_1=1} \sum_{i\in[n]} \sum_{G:i\in G} -\alpha_G^{(i)} \tilde{g}_i \tilde{w}_G \tag{38}$$

$$\leq \left(\max_{i\in[n]} \tilde{g}_i \right) \cdot \max_{\tilde{\mathbf{w}}\geq 0:\|\tilde{\mathbf{w}}\|_2=1} \max_{\boldsymbol{\alpha}\geq 0:\|\boldsymbol{\alpha}\|_1=1} \sum_{i\in[n]} \sum_{G:i\in G} \alpha_G^{(i)} \tilde{w}_G \tag{39}$$

$$\leq \left(\max_{i\in[n]} \tilde{g}_i \right) \cdot \max_{\boldsymbol{\alpha}\geq 0:\|\boldsymbol{\alpha}\|_1=1} \sum_{i\in[n]} \sum_{G:i\in G} \alpha_G^{(i)} = \max_{i\in[n]} \tilde{g}_i = \|\tilde{\mathbf{g}}\|_\infty . \tag{40}$$

The equality is obviously attainable. Therefore by Theorem 1 of [17], $h_\epsilon(\tilde{\mathbf{w}}) = \eta^*(A\tilde{\mathbf{w}})$ has Lipschitz continuous gradient *w.r.t.* ℓ_2 norm, and the Lipschitz constant is

$$\frac{1}{c} \|A\|_{2,1}^2 = \frac{1}{\epsilon} \|\tilde{\mathbf{g}}\|_\infty^2 \, n \log r.$$

This completes our proof of part (i) of Proposition 1. ■

D DAG Groups

We discuss here another interesting special case of the group sparse model formulated in Section 3.1.

Suppose the variables $\{1, 2, \ldots, n\}$ form the nodes of a directed acyclic graph (DAG), and each node i corresponds to a group consisting of all nodes j that are reachable from i by transversing the DAG. For simplicity we assign unit cost to each group. Since a node in this model may belong to n groups, *i.e.* $r = \Theta(n)$ (recall that r is the upper bound on the number of groups that any variable may belong to), hence a naive application of Proposition 1 results in the overall complexity for computing the polar as $O(\frac{1}{\epsilon}\sqrt{n^5 \log n})$. Fortunately this can be reduced to $O(\frac{1}{\epsilon} m\sqrt{n})$, where m is the number of edges (in the worst case on the order of n^2).

We recall from the main paper the polar of the general group sparse regularizer

$$\min_{\tilde{\mathbf{w}}\geq 0} \sum_{i\in[n]} \tilde{g}_i \cdot \max_{G:i\in G\in\mathcal{G}} (-\tilde{w}_G), \;\; \text{s.t.} \sum_{G\in\mathcal{G}} b_G \cdot \tilde{w}_G = 1.$$

In the DAG case, each variable i corresponds to a group that consists of all descendants of i. Let us denote the group as G_i. For simplicity, assume the costs $b_G = 1$ for all groups G. By symmetry, if there is an edge from i to j then at optimum $\tilde{w}_{G_i} \geq \tilde{w}_{G_j}$, because otherwise we can swap their values without increasing the objective or violating the constraint. To lighten notation, we just write $\tilde{w}_{G_i}$ as $\tilde{w}_i$. Thus we simplify the above problem into

$$\min_{\tilde{\mathbf{w}}\geq 0} - \sum_{i\in[n]} \tilde{g}_i \tilde{w}_i, \;\; \text{s.t.} \sum_{i\in[n]} \tilde{w}_i = 1, \;\; \text{and } \tilde{w}_i \geq \tilde{w}_j \; \forall \; (i,j) \in E. \tag{41}$$

Here we use the pair (i, j) to denote an edge from i to j, and E is the set of all edges. Next introduce the dual variables $\alpha_{ij} \geq 0$ for the constraint $\tilde{w}_i \geq \tilde{w}_j$ and ξ for the constraint $\sum_{i\in[n]} \tilde{w}_i = 1$. Consider the Lagrangian dual

$$\min_{\xi, \boldsymbol{\alpha}\geq \mathbf{0}} \xi + \sum_{i\in[n]} \max_{\tilde{w}_i \geq 0} \tilde{w}_i \left(\tilde{g}_i - \xi + \sum_{j:(i,j)\in E} \alpha_{ij} - \sum_{k:(k,i)\in E} \alpha_{ki} \right),$$

which, after taking into account $\tilde{w}_i \leq 1$, simplifies to

$$\min_{\xi, \boldsymbol{\alpha}\geq \mathbf{0}} \xi + \sum_{i\in[n]} \left(\tilde{g}_i - \xi + \sum_{j:(i,j)\in E} \alpha_{ij} - \sum_{k:(k,i)\in E} \alpha_{ki} \right)_+, \tag{42}$$

where $(x)_+ := \max\{x, 0\}$. As in Appendix C we can easily smooth the function $(\cdot)_+$ and therefore solve (42) using APG. To summarize, a 2ϵ accurate solution can be found in $O(\frac{1}{\epsilon}\sqrt{n})$ iterations

with $O(m)$ cost per iteration. Overall this is faster than the complexity $O(mn^2 \log \frac{1}{\epsilon})$ of [6] (which involves a binary search). See Appendix E for details.

Moreover, if the DAG is a rooted tree, *i.e.*, each node can only be pointed to by at most one edge, we can further reduce the overall cost to $O(n \log \frac{1}{\epsilon})$. Indeed, let the root be node 1, and denote as $\mathrm{pa}(i)$ and $\mathrm{ch}(i)$ the parent and children nodes of i, respectively. Note that by the definition of rooted tree, $|\mathrm{pa}(i)| = 1$ for any node i that is not the root. Again, for any non-root node $i > 1$, we introduce a dual variable α_i for the constraint $x_{\mathrm{pa}(i)} \geq x_i$. For convenience let $\alpha_1 = 0$. Then the Lagrangian dual of (41) in the rooted tree case is

$$\min_{\xi, \boldsymbol{\alpha} \geq 0} \xi + \sum_{i \in [n]} \left(\tilde{g}_i - \xi + \sum_{j \in \mathrm{ch}(i)} \alpha_j - \alpha_i \right)_+ . \tag{43}$$

At the optimum, there cannot be two summands that are positive, because then the subgradient of ξ would be negative. If only one summand is positive, we can increase ξ to make it 0 without changing the objective value. Thus we can assume all summands are 0, and solve

$$\min_{\xi, \boldsymbol{\alpha} \geq 0} \xi, \;\; \text{s.t. } \forall\, i,\; \alpha_i \geq \tilde{g}_i - \xi + \sum_{j \in \mathrm{ch}(i)} \alpha_j. \tag{44}$$

In effect, we search for the smallest ξ that makes the feasible region nonempty. For any $\xi > 0$, its feasibility can be checked by propagating towards the root via

$$\alpha_i = \max \left\{ 0,\; \tilde{g}_i - \xi + \sum_{j \in \mathrm{ch}(i)} \alpha_j \right\}. \tag{45}$$

Note that for all leaf nodes, that is $\{j : \mathrm{ch}(j) = \emptyset\}$, their dual variables $\alpha_j = 0$. At the root if $\alpha_1 = 0 \geq \tilde{g}_1 - \xi + \sum_{j \in \mathrm{ch}(1)} \alpha_j$ is met, then we claim that ξ is feasible. Clearly $\xi \in [\tilde{g}_1, \max_i \tilde{g}_i]$, hence using binary search an ϵ accurate solution can be found in $O(n \log \frac{1}{\epsilon})$. Finally, given ξ, the optimal primal variable $\tilde{\mathbf{w}}$ can be easily recovered using KKT conditions. Overall our approach is faster than the $O(n\mathsf{d})$ complexity in [5], where d is the depth of the tree and in the worst case can be $\Theta(n)$.

E Comparisons for Group Sparse Models

In this section we compare the complexity of our approach (under the group sparse model developed in Section 3.1) with two related methods in literature, namely, [11] and [6].

Consider first [11]. The Algorithm 2 there proceeds in loops, with each iteration involving a max-flow problem on the canonical graph. The loop can take at most n iterations, while each max-flow problem can be solved with $O(|V|\,|E|)$ cost where $|V|$ and $|E|$ are the number of nodes and edges in the canonical graph, respectively. By construction, $|V| = n + l$, and $|E| \leq nr$ since each pair of (G, i) with the node i belong to the group G contributes an edge. Therefore the total cost is upper bounded by $O(n^2(n + l)r)$. Note that in the worst case $\ell = \Theta(nr)$. In contrast, the approach we developed in Section 3.1 for bounded degree groups costs $O(\frac{nr}{\epsilon}\sqrt{n \log r})$, significantly cheaper in the regime where n is big and ϵ is moderate.

For the DAG groups considered in Appendix D, again Algorithm 2 in [11] can take $\Theta(n)$ iterations, while $|V| = 2n$ and $|E| \leq mn$ (since in the worst case each node can belong to $\Theta(n)$ groups). Thus overall [11, Algorithm 2] costs $O(n^3 m)$ for DAG groups, worse than the complexity $O(\frac{1}{\epsilon} m\sqrt{n})$ we obtained in Appendix D.

Next consider [6] which developed a line search scheme to compute the polar. The major computational step there is to solve

$$\tilde{\mathbf{w}}_\sigma = \operatorname*{arg\,max}_{\tilde{\mathbf{w}} \in Q} \langle \tilde{\mathbf{g}}, \tilde{\mathbf{w}} \rangle - \sigma \langle \mathbf{b}, \tilde{\mathbf{w}} \rangle , \tag{46}$$

recursively, each time with a updated $\sigma > 0$. In the case of bounded degree groups, this is again a max-flow problem which costs $O(n(n+l)r)$, and therefore the overall cost is $O(n(n+l)r \log \frac{1}{\epsilon})$. In the case of DAG groups (Appendix D), the max-flow problem costs $O(n^2 m)$, and hence the overall cost is $O(n^2 m \log \frac{1}{\epsilon})$. In both cases, [6] improves over [11] but is still worse than our approach.

F Path Coding: Efficient Linear Programming

We show in this section how to efficiently solve the LP for the path coding regularizer discussed in Section 3.2. First recall that we have arrived at the following LP in Section 3.2:

$$\max_{\tilde{\mathbf{w}}} \quad \sum_i \tilde{g}_i \left(\sum_{j:(i,j)\in E} \tilde{w}_{ij} + \sum_{k:(k,i)\in E} \tilde{w}_{ki} \right), \tag{47}$$

$$\text{s.t.} \quad \tilde{\mathbf{w}} \geq \mathbf{0}, \quad \sum_{(i,j)\in E} b_{ij}\tilde{w}_{ij} = 1, \quad \sum_{j:(i,j)\in E} \tilde{w}_{ij} = \sum_{k:(k,i)\in E} \tilde{w}_{ki}, \ \forall i. \tag{48}$$

This LP appears to be more complicated than the one in Section 3.1, due to the two extra constraints in the end. We start with removing these constraints by introducing dual variables.

Denote $z_i = \sum_{j:(i,j)\in E} \tilde{w}_{ij}$. Since $\tilde{w}_{ij} \geq 0$, we can parameterize $\tilde{w}_{ij}$ as $\tilde{w}_{ij} = z_i \tau_j^{(i)}$, where $z_i \geq 0$ and $\boldsymbol{\tau}^{(i)}$ belongs to the simplex $\mathcal{S}_i := \{\boldsymbol{\tau}^{(i)} \geq \mathbf{0} : \langle \mathbf{1}, \boldsymbol{\tau}^{(i)} \rangle = 1\}$. Introduce Lagrange multipliers $\vartheta = (\lambda, \alpha_i)$ for the three constraints in (47), respectively. For convenience also let $\alpha_s = \alpha_t = \tilde{g}_s = \tilde{g}_t = 0$. Denote

$$d_{ij}(\vartheta) = \tilde{g}_i + \tilde{g}_j - \alpha_i + \alpha_j - \lambda b_{ij}.$$

After some tedious algebra we obtain the Lagrangian

$$\min_{\boldsymbol{\alpha},\lambda} \left\{ \lambda + \sum_{(i,j)\in E} \max_{\tilde{w}_{ij}\geq 0} \tilde{w}_{ij} d_{ij}(\vartheta) \right\} = \min_{\boldsymbol{\alpha},\lambda} \left\{ \lambda + \sum_{i\in[n]\cup\{s\}} \max_{z_i\geq 0} z_i \max_{\boldsymbol{\tau}^{(i)}\in\mathcal{S}_i} \sum_{j:(i,j)\in E} \tau_j^{(i)} d_{ij}(\vartheta) \right\} \tag{49}$$

$$= \min_{\boldsymbol{\alpha},\lambda} \left\{ \lambda + \sum_{i\in[n]\cup\{s\}} \max_{z_i\geq 0} z_i \left(\max_{j:(i,j)\in E} d_{ij}(\vartheta) \right) \right\}. \tag{50}$$

Our key observation is that z_i can be upper bounded. Note the constraints $\sum_{(i,j)\in E} b_{ij}\tilde{w}_{ij} = 1$ and $\tilde{\mathbf{w}} \geq 0$ in (48). Let C be the lowest cost of all (s,t)-paths, and naturally $C > 0$ by assumption. Then trivially any path will satisfy $z_i \leq \rho := \frac{1}{C}$. A more conservative upper bound on z_i is

$$z_i \leq \rho := \left(\min_{(i,j)\in E} b_{ij} \right)^{-1}, \tag{51}$$

assuming all $b_{ij} > 0$. Taking into account these upper bounds, we arrive at our final objective

$$\min_{\lambda} \{\lambda + \rho f(\lambda)\}, \quad \text{where} \quad f(\lambda) := \min_{\boldsymbol{\alpha}} \sum_{i\in[n]\cup\{s\}} \left(\max_{j:(i,j)\in E} d_{ij}(\vartheta) \right)_+. \tag{52}$$

As before $(x)_+ = \max\{x, 0\}$. Note given λ, the inner optimization over $\boldsymbol{\alpha}$ has a closed form thanks to the absence of cycles. Specifically, let $\alpha_t(\lambda) = 0$ and define for any $i \in [n] \cup \{s\}$

$$\alpha_i(\lambda) = \max_{j:(i,j)\in E} \{\alpha_j(\lambda) + \tilde{g}_i + \tilde{g}_j - b_{ij}\lambda\}. \tag{53}$$

Since the graph is a DAG, we can always find a topological ordering of the indices i, such that before computing $\alpha_i(\lambda)$ for node i, all its descendants $\alpha_j(\lambda)$ have been computed. It is not hard to see

$$f(\lambda) = \max\{\alpha_s(\lambda), 0\}, \tag{54}$$

and the optimal $\boldsymbol{\alpha}$ in the definition of f in (52) is attained at $\{\alpha_i = \alpha_i(\lambda) : i \in [n]\}$, because, as can be easily verified, $\mathbf{0}$ is a subgradient. This relationship allows us to compute a subgradient of f at λ via recursion

$$\partial\alpha_i(\lambda) = \left\{ \sum_{j\in J} \gamma_j (v_j - b_{ij}) : J = \text{(set of) } \arg\max \text{ in (53)}, v_j \in \partial\alpha_j(\lambda), \gamma_j \geq 0, \langle \mathbf{1}, \boldsymbol{\gamma} \rangle = 1 \right\}. \tag{55}$$

Obviously, the recursion in both (53) and (55) can be accomplished in $O(m)$ time. Indeed a trivial subgradient of $\alpha_s(\lambda)$ is the negative cost of the path that is induced by the $\arg\max$ in (53) (breaking

tie arbitrarily). Finally we solve (52) over λ by cutting plane method, which can find an ϵ accurate solution in $O(\frac{n}{\epsilon^2})$ iterations, *i.e.* with $O(\frac{mn}{\epsilon^2})$ total computation.

Further reducing the computational cost to $O(\frac{m\sqrt{n}}{\epsilon})$ is possible by smoothing the $\max$ function in

$$\min_{\boldsymbol{\alpha},\lambda} \left\{ \lambda + \rho \sum_{i\in[n]\cup\{s\}} \left(\max_{j:(i,j)\in E} d_{ij}(\vartheta) \right)_+ \right\}. \tag{56}$$

This cost is potentially better than the $O(mn)$ worst case complexity in [12, Algorithm 1]. Algorithmically, this can be done in exactly the same way as in Appendix C. After that we run APG on the smoothed problem. To summarize, following exactly the same argument as in the proof of Proposition 1 we have

Proposition 4 *Denote the objective in* (56) *as $h(\vartheta)$, For any $\epsilon > 0$, there exists a convex function h_ϵ such that (i) $\forall\ \vartheta$, $h(\vartheta) - h_\epsilon(\vartheta) \in (-\epsilon, 0]$, (ii) h_ϵ has $L = O(\frac{n}{\epsilon})$ Lipschitz continuous gradient, and (iii) the gradient of h_ϵ can be computed in $O(m)$ time.*

G Recovery of Integral Solutions to Polar Oracle

Recall our ultimate goal in polar oracle is to find integral solutions to (8) which we copy here for convenience

$$\lambda^* := \max_{\mathbf{0}\neq\mathbf{w}\in P} \frac{\langle \tilde{\mathbf{g}}, \mathbf{w}\rangle}{\langle \mathbf{b}, \mathbf{w}\rangle}. \tag{57}$$

As we showed in Section 3, the optimal objective value is exactly equal to that of (10), which we also copy here

$$\max_{\tilde{\mathbf{w}},\sigma>0} \ \langle \tilde{\mathbf{g}}, \tilde{\mathbf{w}}\rangle, \text{ subject to } \ \tilde{\mathbf{w}} \in \sigma Q, \ \ \langle \mathbf{b}, \tilde{\mathbf{w}}\rangle = 1. \tag{58}$$

We have shown how to smooth this objective and find an ϵ accurate solution for it. That means we have obtained a λ_ϵ (smooth objective function value) with the guarantee that $\lambda_\epsilon \in [\lambda^* - \epsilon, \lambda^*]$. With this λ_ϵ in hand, we now show how to find an ϵ accurate solution for (8), *i.e.* a $\mathbf{w}_\epsilon \in P\backslash\{\mathbf{0}\}$ such that

$$\frac{\langle \tilde{\mathbf{g}}, \mathbf{w}_\epsilon\rangle}{\langle \mathbf{b}, \mathbf{w}_\epsilon\rangle} \geq \lambda^* - \epsilon. \tag{59}$$

Indeed, this is simple according to Proposition 5.

Proposition 5 *Given $\lambda_\epsilon \in [\lambda^* - \epsilon, \lambda^*]$, find*

$$\mathbf{w}_\epsilon := \arg\max_{\mathbf{w}\in P\backslash\{\mathbf{0}\}} \ \{\langle \tilde{\mathbf{g}}, \mathbf{w}\rangle - \lambda_\epsilon \langle \mathbf{b}, \mathbf{w}\rangle\}. \tag{60}$$

Then $\mathbf{w}_\epsilon$ must satisfy (59)*.*

Proof: By the definition of λ^*, $\max_{\mathbf{w}\in P\backslash\{\mathbf{0}\}}\{\langle \tilde{\mathbf{g}}, \mathbf{w}\rangle - \lambda^* \langle \mathbf{b}, \mathbf{w}\rangle\} = 0$. As $\lambda_\epsilon \leq \lambda^*$, so $\max_{\mathbf{w}\in P\backslash\{\mathbf{0}\}}\{\langle \tilde{\mathbf{g}}, \mathbf{w}\rangle - \lambda_\epsilon \langle \mathbf{b}, \mathbf{w}\rangle\} \geq 0$. This implies $\frac{\langle \tilde{\mathbf{g}}, \mathbf{w}_\epsilon\rangle}{\langle \mathbf{b}, \mathbf{w}_\epsilon\rangle} \geq \lambda_\epsilon \geq \lambda^* - \epsilon$. ■

Note (60) is exactly the submodular minimization problem that the secant method is based on (step 3 of Algorithm 2). This step is computationally expensive and has to be solved for multiple values of λ_t in that method. In contrast, our our strategy needs to solve this problem only once.

In group sparsity, it leads to a max-flow problem as in Appendix B.1 which is again expensive. Fortunately, by exploiting the structure of the problem it is possible to design a heuristic solution. For convenience let us copy (15) to here, the linear programming for group sparsity.

$$\max_{\tilde{\mathbf{w}}} \ \sum_{i\in[n]} \tilde{g}_i \min_{G: i\in G\in\mathcal{G}} \tilde{w}_G, \ \text{ subject to } \ \tilde{\mathbf{w}} \geq 0, \ \sum_{G\in\mathcal{G}} b_G \tilde{w}_G = 1. \tag{61}$$

A solution $\tilde{\mathbf{w}}$ corresponds to an integral solution to the polar oracle if and only if $\tilde{w}_G \in \{0, c\}$ where c ensures $\sum_{G \in \mathcal{G}} b_G \tilde{w}_G = 1$. By solving the smoothed objective, we obtain a solution $\tilde{\mathbf{w}}^*$ which does not necessarily satisfy this condition. However, a smaller value of the component $\tilde{w}^*_G$ does suggest a higher likelihood for $\tilde{w}_G$ to be 0. Therefore, we sorted $\{w^*_G\}$ and set the w_G of the smallest k groups to 0 (k ranging from 0 to $|\mathcal{G}| - 1$), and the w_G for the remaining groups were set to a common value that satisfies the constraint. Given k, this leads to an objective value, and the k that maximizes this value can be selected by enumerating $k \in \{0, 1, \ldots, |\mathcal{G}| - 1\}$. By exploiting the structure of the objective, it is easy to design an algorithm which accomplishes the enumeration in $O(nr)$ time.

The optimal objective value over all k also allows us to compute its distance to the optimal objective value of the smoothed objective. If the gap (used as a certificate) is below ϵ, this integral solution is exactly ϵ sub-optimal. Otherwise we fall back on (60), and this case rarely happens in practice.

In path coding, the path can be simply recovered by following the $\arg\max$ in (53), with λ set to an optimal solution to (52).

H Polar of $\Omega_s(\mathbf{w}) = \sum_i \|\mathbf{w}\|_{(i)}$

The polar of $\Omega_s(\mathbf{w}) = \sum_i \|\mathbf{w}\|_{(i)}$ follows from the following proposition by taking $\phi(\boldsymbol{\alpha}) = \sum_i \alpha_i$. We note that Proposition 6 itself is a slight generalization of [19, Theorem 15.3].

Proposition 6 *Let $\kappa_i : \mathbb{R}^d \to \bar{\mathbb{R}}_+, 1 \le i \le n$ be closed gauges, $\phi : \bar{\mathbb{R}}^n_+ \to \bar{\mathbb{R}}_+$ be closed, convex, non-constant in each coordinate[1] with $\phi(\mathbf{0}) = 0$, and $\exists \mathbf{x} \in \cap_i \operatorname{ri}\operatorname{dom}\kappa_i$ such that $(\kappa_1(\mathbf{x}), \ldots, \kappa_n(\mathbf{x})) \in \operatorname{ri}\operatorname{dom}\phi$, then the Fenchel conjugate of $h := \phi(\kappa_1, \ldots, \kappa_n)$ is*

$$h^*(\mathbf{x}) = \min_{\sum_i \mathbf{x}^i = \mathbf{x}} \phi^+(\kappa_1^\circ(\mathbf{x}^1), \ldots, \kappa_n^\circ(\mathbf{x}^n)), \tag{62}$$

where κ_i° is the polar of κ_i and $\phi^+(\mathbf{y}) := \max_{\mathbf{x} \ge 0} \langle \mathbf{x}, \mathbf{y} \rangle - \phi(\mathbf{x})$ is the monotone conjugate of ϕ. Moreover, if ϕ is a gauge so is h whose polar

$$h^\circ(\mathbf{x}) = \min_{\sum_i \mathbf{x}^i = \mathbf{x}} \phi^\circ(\kappa_1^\circ(\mathbf{x}^1), \ldots, \kappa_n^\circ(\mathbf{x}^n)), \tag{63}$$

where ϕ° is the polar of ϕ.

Proof: Let us define the diagonal operator $A : \bar{\mathbb{R}}^d \to (\bar{\mathbb{R}}^d)^n, \mathbf{x} \mapsto (\mathbf{x}, \ldots, \mathbf{x})$. Then $h(\mathbf{x}) = H(A(\mathbf{x}))$, where

$$H(\mathbf{x}^1, \ldots, \mathbf{x}^n) := \phi(\kappa_1(\mathbf{x}^1), \ldots, \kappa_n(\mathbf{x}^n)).$$

The Fenchel conjugate of G is

$$\begin{aligned}
H^*(\mathbf{y}^1, \ldots, \mathbf{y}^n) &= \sup_{\mathbf{x}^i} \sum_i \langle \mathbf{x}^i, \mathbf{y}^i \rangle - H(\mathbf{x}^1, \ldots, \mathbf{x}^n) \\
&= \sup_{\mathbf{x}^i} \sum_i \langle \mathbf{x}^i, \mathbf{y}^i \rangle - \phi(\kappa_1(\mathbf{x}^1), \ldots, \kappa_n(\mathbf{x}^n)) \\
&= \sup_{\kappa_i(\mathbf{x}^i) \le \lambda_i} \sum_i \langle \mathbf{x}^i, \mathbf{y}^i \rangle - \phi(\lambda_1, \ldots, \lambda_n) \\
&= \sup_{\lambda_i \ge 0} \sum_i \langle \kappa_i^\circ(\mathbf{y}^i), \lambda_i \rangle - \phi(\lambda_1, \ldots, \lambda_n) \\
&= \phi^+(\kappa_1^\circ(\mathbf{y}^1), \ldots, \kappa_n^\circ(\mathbf{y}^n)),
\end{aligned}$$

where the third equality is due to the monotonicity of ϕ (since $\phi \ge 0$ and $\phi(\mathbf{0}) = 0$). Since both ϕ and κ_i are closed, H is closed. Also by assumption $\exists \mathbf{x}$ such that $A\mathbf{x} \in \operatorname{ri}\operatorname{dom} H$. Therefore we can apply [19, Theorem 16.3] to conclude that $h^* = (HA)^* = A^* H^*$, where A^* is the adjoint of A. Expanding the last expression we get (62).

[1] This assumption allows us to interpret $\phi(\infty, \ldots)$ as ∞.

The second claim follows from the relations

$$\kappa^* = \delta(\kappa^\circ \leq 1) \tag{64}$$
$$\kappa^\circ = \delta^*(\kappa \leq 1), \tag{65}$$

where κ is any gauge and $\delta(\cdot) = 0$ if $\cdot$ is true otherwise $\delta(\cdot) = \infty$. Indeed, when ϕ is a gauge, so is h, and

$$\begin{aligned} h^*(\mathbf{x}) &= \min_{\sum_i \mathbf{x}^i = \mathbf{x}} \phi^+(\kappa_1^\circ(\mathbf{x}^1), \ldots, \kappa_n^\circ(\mathbf{x}^n)) \\ &= \min_{\sum_i \mathbf{x}^i = \mathbf{x}} \delta(\phi^\circ(\kappa_1^\circ(\mathbf{x}^1), \ldots, \kappa_n^\circ(\mathbf{x}^n) \leq 1) \\ &= \delta\left(\left[\min_{\sum_i \mathbf{x}^i = \mathbf{x}} \phi^\circ(\kappa_1^\circ(\mathbf{x}^1), \ldots, \kappa_n^\circ(\mathbf{x}^n)\right] \leq 1\right) \\ &= \delta(h^\circ(\mathbf{x}) \leq 1), \end{aligned}$$

due to (64). Since both functions (inside δ) are positively homogeneous, we must have (63). ■

I Proof of Proposition 2

Since the polar Ω° is closed, we have

$$0 = \min_{\boldsymbol{\theta}:\Omega^\circ(\boldsymbol{\theta}) \leq \zeta} \tfrac{1}{2}\|\boldsymbol{\theta} - \mathbf{g}\|_2^2$$

if and only if $\Omega^\circ(\mathbf{g}) \leq \zeta$, therefore

$$\Omega^\circ(\mathbf{g}) = \inf\left\{\zeta \geq 0 : 0 = \min_{\boldsymbol{\theta}:\Omega^\circ(\boldsymbol{\theta}) \leq \zeta} \tfrac{1}{2}\|\boldsymbol{\theta} - \mathbf{g}\|_2^2\right\}. \tag{66}$$

Recall Moreau's identity [19, Theorem 31.5], that is,

$$\mathsf{Prox}_f(\mathbf{g}) + \mathsf{Prox}_{f^*}(\mathbf{g}) = \tfrac{1}{2}\|\mathbf{g}\|_2^2, \tag{67}$$

where f^* denotes the Fenchel conjugate of f. Setting $f(\mathbf{g}) = \delta(\Omega^\circ(\mathbf{g}) \leq \zeta)$ we obtain $f^*(\mathbf{g}) = \zeta\Omega(\mathbf{g})$, hence

$$\min_{\boldsymbol{\theta}:\Omega^\circ(\boldsymbol{\theta}) \leq \zeta} \tfrac{1}{2}\|\boldsymbol{\theta} - \mathbf{g}\|_2^2 = \mathsf{Prox}_f(\mathbf{g}) = \tfrac{1}{2}\|\mathbf{g}\|_2^2 - \mathsf{Prox}_{f^*}(\mathbf{g}),$$

which plugged into (66) completes the proof of Proposition 2.

J Proof of Proposition 3

The proof is quite straightforward. Let

$$\mathbf{u} := \arg\min_{\boldsymbol{\theta}} \tfrac{1}{2}\|\mathbf{w} - \boldsymbol{\theta}\|_2^2 + \|\boldsymbol{\theta}\|_{\mathsf{TV}} \tag{68}$$
$$\mathbf{v} := \arg\min_{\boldsymbol{\theta}} \tfrac{1}{2}\|\mathbf{u} - \boldsymbol{\theta}\|_2^2 + \|\boldsymbol{\theta}\|_p \tag{69}$$
$$\mathbf{z} := \arg\min_{\boldsymbol{\theta}} \tfrac{1}{2}\|\mathbf{w} - \boldsymbol{\theta}\|_2^2 + \|\boldsymbol{\theta}\|_{\mathsf{TV}} + \|\boldsymbol{\theta}\|_p, \tag{70}$$

then Proposition 3 amounts to claiming that $\mathbf{z} = \mathbf{v}$.

Indeed, by the first order optimality conditions for convex programming [19], we must have

$$\mathbf{0} \in \mathbf{u} - \mathbf{w} + \partial\|\mathbf{u}\|_{\mathsf{TV}} \tag{71}$$
$$\mathbf{0} \in \mathbf{v} - \mathbf{u} + \partial\|\mathbf{v}\|_p, \tag{72}$$

where $\partial\|\mathbf{x}\|$ denotes the subdifferential of the norm $\|\cdot\|$ at point $\mathbf{x}$. It is easy to argue from (72) that $u_i \geq u_j \implies v_i \geq v_j$, therefore exploiting the special structure of $\|\cdot\|_{\mathsf{TV}}$ we can conclude that $\partial\|\mathbf{u}\|_{\mathsf{TV}} \subseteq \partial\|\mathbf{v}\|_{\mathsf{TV}}$. Adding (71) and (72) we obtain

$$\mathbf{0} \in \mathbf{v} - \mathbf{w} + \partial\|\mathbf{v}\|_p + \partial\|\mathbf{v}\|_{\mathsf{TV}}, \tag{73}$$

which implies that $\mathbf{v}$ minimizes (70). Thus $\mathbf{v} = \mathbf{z}$, since both are optimal while the minimizer is unique.

Algorithm 3 Exact algorithm for the proximal map (74).

$h_1(-1) = w_1 - 1, h_1(1) = w_1 + 1$. $\mathbb{K}_1 \leftarrow \{(-1, h_1(-1)); (1, h_1(1))\}$.
for $j = 2, \ldots, m-1$ **do**
  $h_j(z) = z + w_j - \text{Median}(-1, 1, (h_{j-1} + I)^{-1}(w_j + z))$ for $z \in \{-1, 1\}$.
  $\mathbb{K}_j \leftarrow \{(-1, h_j(-1)), (1, h_j(1))\}$.
  for all $(\alpha_i, \beta_i) \in \mathbb{K}_{j-1}$ **do**
    if $-1 < \alpha_i' := \alpha_i + \beta_i - w_j < 1$ **then**
      $\mathbb{K}_j \leftarrow \mathbb{K}_j \cup \{(\alpha_i', \beta_i)\}$
    end if
  end for
end for

K Fused Lasso: An Efficient Exact Algorithm for Computing $\mathsf{Prox}_{\|\cdot\|_{\mathsf{TV}}}$

Given a vector $\mathbf{w}$, the problem of computing $\mathsf{Prox}_{\|\cdot\|_{\mathsf{TV}}}(\mathbf{w})$ amounts to solving

$$\min_{\boldsymbol{\theta}} \tfrac{1}{2}\|\mathbf{w} - \boldsymbol{\theta}\|_2^2 + \|\boldsymbol{\theta}\|_{\mathsf{TV}}. \tag{74}$$

Applying Moreau's identity [19, Theorem 31.5] we see that $\boldsymbol{\theta}$ minimizes (74) iff for some $\mathbf{z} \in \mathbb{R}^{m-1}$ that solves

$$\min_{\mathbf{z}\in[-1,1]^{m-1}} (z_1+w_1)^2+(z_{m-1}-w_m)^2+\sum\nolimits_j (z_j-z_{j-1}+w_j)^2, \tag{75}$$

we have $\theta_1 = w_1 + z_1$, $\theta_m = w_m - z_{m-1}$, and $\theta_j = w_j + z_j - z_{j-1}$ for all $2 \le j \le m-1$.

For $z \in [-1, 1]$, define $H_1(z) = \frac{1}{2}(z + w_1)^2$ and recursively for $2 \le j \le m-1$ define

$$H_j(z) = \min_{|z_{j-1}|\le 1} H_{j-1}(z_{j-1}) + \tfrac{1}{2}(z - z_{j-1} + w_j)^2. \tag{76}$$

It is readily verified that solving (75) amounts to minimizing $H_{m-1}(z) + \frac{1}{2}(z - w_m)^2$. Inductively, we infer that H_j is a convex piecewise quadratic univariate function. Therefore its derivative, denoted as h_j, is increasing and piecewise linear. Denote subdifferential $\partial h_j(1) = [\lim_{z\uparrow 1} h_j(z), \infty)$ and $\partial h_j(-1) = (-\infty, \lim_{z\downarrow -1} h_j(z)]$. Moreover, for all $2 \le j \le m-1$

$$h_j(z_j) = z_j + w_j - z_{j-1}, \tag{77}$$

$$\text{where} \quad z_{j-1} = \arg\min_{-1\le z\le 1} H_{j-1}(z) + \tfrac{1}{2}(z_j - z + w_j)^2 \tag{78}$$

$$= \text{Median}(-1, 1, (h_{j-1} + I)^{-1}(z_j + w_j)). \tag{79}$$

Therefore if h_{j-1} has k (linear) pieces, h_j has at most $k+1$ (linear) pieces (taking into account the end points $z = \pm 1$). Using dynamic programming we can recursively identify all the "kink points" of h_j (denoted as $\mathbb{K}_j$) for $j = 1, \ldots, m-1$, and hence easily find the minimizer of $H_{m-1}(z) + \frac{1}{2}(z - w_m)^2$, that is, (74).

Thus we can summarize the procedure in Algorithm 3.

Note the space cost is $O(m)$ and upon completion of Algorithm 3, we only have $\mathbb{K}_{m-1}$, based on which the optimal z_{m-1}^* can be found. To recover the optimal $z_1^*, \ldots, z_{m-2}^*$, we backtrack the values of z_j^* and $h_j(z_j^*)$. By (77), it is obvious that for $2 \le j \le m-1$

$$z_{j-1}^* = z_j^* + w_j - h_j(z_j^*). \tag{80}$$

Then by (79), we have three cases:

- $z_{j-1}^* = -1 \Rightarrow h_{j-1}(z_{j-1}^*) = h_{j-1}(-1)$ which we have recorded in Algorithm 3.
- $z_{j-1}^* = 1 \Rightarrow h_{j-1}(z_{j-1}^*) = h_{j-1}(1)$ which we have also recorded in Algorithm 3.
- $z_{j-1}^* = (h_{j-1} + I)^{-1}(z_j^* + w_j) \Rightarrow h_{j-1}(z_{j-1}^*) = z_j^* + w_j - z_{j-1}^* = h_j(z_j^*)$.

K.1 More Experiments on Fused Lasso with Comparison to Liu et. al. [26]

We compared two algorithms that solve the proximal operator $\mathsf{Prox}_{\|\cdot\|_{\mathsf{TV}}}$ in fused lasso. One is our dynamic programming (DP) Algorithm 3, and one is from Liu et. al. [26] whose implementation was extracted from the SLEP package[2]. In particular, we randomly generated an m-dimensional vector $\mathbf{w}$ and used the two methods to solve

$$\min_{\boldsymbol{\theta}} \tfrac{1}{2}\|\mathbf{w}-\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_{\mathsf{TV}}. \tag{81}$$

The components of $\mathbf{w}$ were drawn independently from unit Gaussians, and the dimension m ranged from 10^4 to 10^6. We varied $\lambda \in \{0.01, 0.1, 1, 10, 100\}$ and the resulting run time is shown in Figure 4 to 8 respectively. For each combination of m and λ, 50 random samples of $\mathbf{w}$ were drawn which allowed us to plot the error bar.

It is clear that the run time of both algorithms is linear in m. However, our DP algorithm is 2 to 6 times faster than [26], and the margin grows wider as the values of λ increase.

Figure 9 shows the total number of kinks generated along the execution of our DP algorithm. It is also linear in m and the slope is 2 to 12 depending on λ.

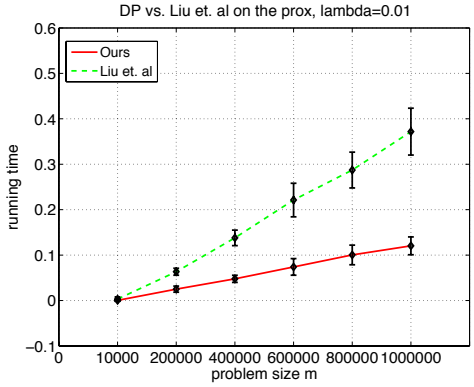


Figure 4: Running time (in seconds) of our DP algorithm vs [26] for $\lambda = 0.01$.

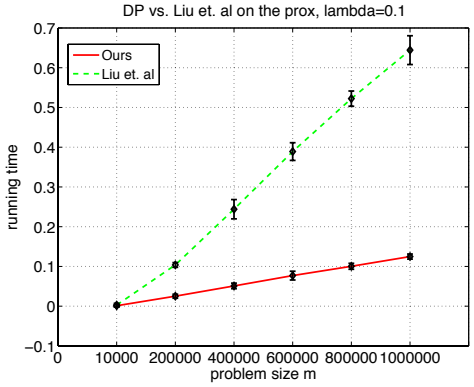


Figure 5: Running time (in seconds) of our DP algorithm vs [26] for $\lambda = 0.1$.

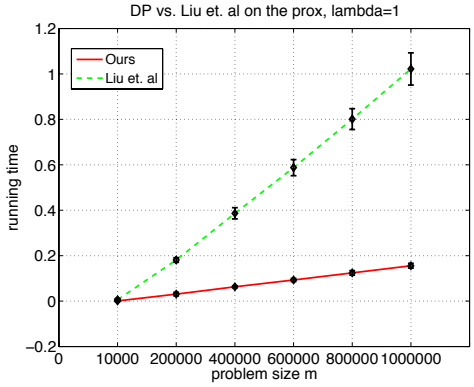


Figure 6: Running time (in seconds) of our DP algorithm vs [26] for $\lambda = 1$.

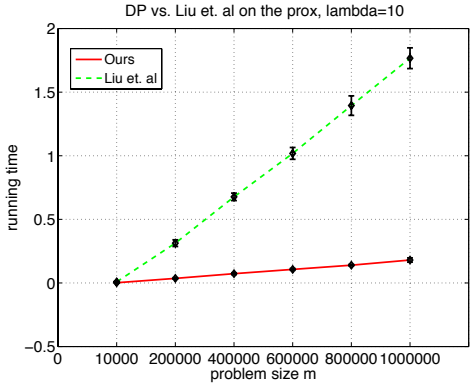


Figure 7: Running time (in seconds) of our DP algorithm vs [26] for $\lambda = 10$.

[2] http://www.public.asu.edu/~jye02/Software/SLEP/index.htm

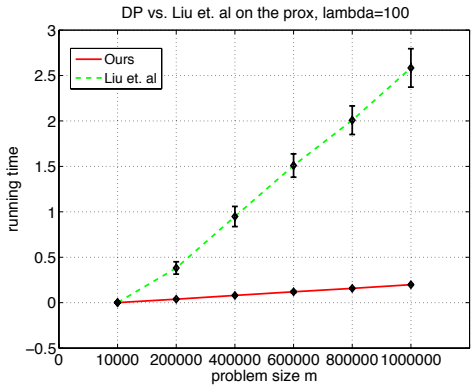


Figure 8: Running time (in seconds) of our DP algorithm vs [26] for $\lambda = 100$.

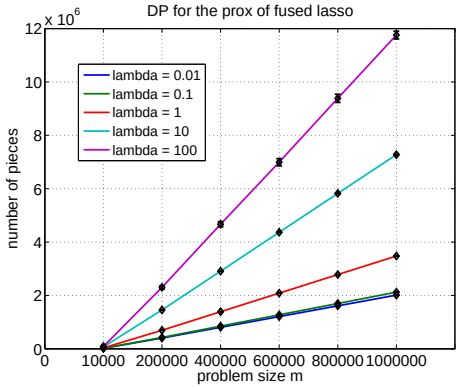


Figure 9: Number of pieces in our DP