Path Coding Penalties for Directed Acyclic Graphs

Julien Mairal*
Department of Statistics
University of California, Berkeley
julien@stat.berkeley.edu

Bin Yu*
Department of Statistics
University of California, Berkeley
binyu@stat.berkeley.edu

Abstract

We consider supervised learning problems where the features are embedded in a graph, such as gene expressions in a gene network. In this context, it is of much interest to automatically select a subgraph which has a small number of connected components, either to improve the prediction performance, or to obtain better interpretable results. Existing regularization or penalty functions for this purpose typically require solving among all connected subgraphs a selection problem which is combinatorially hard. In this paper, we address this issue for directed acyclic graphs (DAGs) and propose structured sparsity penalties over paths on a DAG (called "path coding" penalties). We design minimum cost flow formulations to compute the penalties and their proximal operator in polynomial time, allowing us in practice to efficiently select a subgraph with a small number of connected components. We present experiments on image and genomic data to illustrate the sparsity and connectivity benefits of path coding penalties over some existing ones as well as the scalability of our approach for prediction tasks.

1 Introduction

Supervised sparse estimation problems have been the topic of much research in statistical machine learning and signal processing. We consider in this paper such problems where more information is available than just sparsity of the solution. More precisely, when the features (or variables) can be identified to the vertices of a graph, such as gene expressions in a gene network, it can be desirable to automatically select a subgraph with a few connected components involved in a phenomenon, groups of genes involved in a disease for example [1]. There are two equally important reasons for this: either connectivity of the solution is good prior information which might improve the prediction performance, or connected components may be easier to interpret than isolated variables.

Formally, let us consider a supervised sparse estimation problem involving p features, and assume that a directed graph $G = (V, E)$ is given, where V is a vertex set identified to $\{1, \dots, p\}$, and $E \subseteq V \times V$ is an arc set. Classical empirical risk minimization problems can be written as

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

where $\mathbf{w}$ is a weight vector we wish to estimate, $g : \mathbb{R}^p \to \mathbb{R}$ is a convex smooth function, and $\Omega : \mathbb{R}^p \to \mathbb{R}$ is a regularization function. Typical choices for Ω to obtain a sparse solution are the ℓ_0- or ℓ_1-penalties, but we are interested in penalties that also encourage the sparsity pattern of $\mathbf{w}$ (the set of non-zero coefficients) to form a subgraph of G with a small number of connected components.

Of much interest to us are the sparsity-inducing penalties introduced by Jacob et al. [1] and Huang et al. [2]. Given a pre-defined set of (possibly overlapping) groups of features $\mathcal{G}$, their regularization functions encourage a sparsity pattern to be *in the union of a small number of groups*. By defining $\mathcal{G}$ as the set of all connected subgraphs of G, one could obtain the desired regularization effect, but unfortunately the number of connected subgraphs is exponential in the graph size and this approach

*This work was supported by NSF grants SES-0835531, CCF-0939370, DMS-1107000, DMS-0907632, and by ARO-W911NF-11-1-0114. J.M. would like to thank Laurent Jacob for interesting discussions and his former research lab, the INRIA Willow and Sierra project-teams, for letting him use computational ressources.

leads to hard combinatorial problems. Another strategy, which has been used in various contexts [1, 3], is to define $\mathcal{G}$ as the pairs of vertices linked by an arc, which, as a result, encourages neighbors in the graph to be simultaneously selected. This last formulation is computationally tractable, but does not model long-range interactions between features, and as experimentally shown in Section 4, does not promote large connected components. Another possibility suggested in [1, 2] consists of defining $\mathcal{G}$ as the set of connected subgraphs up to a size L, but the number of such subgraphs is exponential in L, making this approach difficult to use even for small subgraph sizes ($L = 3, 4$) as soon as the graph is large ($p \approx 10\,000$) and connected enough.[1] These observations naturally raise the question: *Can we replace connected subgraphs by another structure which (i) is rich enough to model long-range interactions in the graph, and (ii) leads to computationally feasible penalties?*

We propose a solution to the above question when the directed graph G is acyclic. We introduce for the penalties of [1] and [2] a novel group structure $\mathcal{G}_p$ which contains *all the paths*[2] *in* G. These "path coding" penalties go beyond pairwise interactions and allow controlling the connectivity of selected variables, whether one wishes to select large or small connected components. To deal with the exponential number of paths, we choose appropriate costs for each path (the "price" one has to pay to select a path), and introduce a new link between such *supervised path selection problems* in DAGs and network flow optimization [4]. The main idea is to design minimum cost flow problems such that sending a positive amount of flow along a path (for minimizing a cost) is equivalent to selecting the path in the context of the path coding penalties. This allows us to efficiently compute these penalties and their proximal operators (see [5]) and address the corresponding regularized problems.[3]

2 Proposed Path Coding Penalties

In this section, we formally present our path coding penalties, which leverage the work of [1] and [2]. For a vector $\mathbf{w}$ in $\mathbb{R}^p$ and group structure $\mathcal{G}$, the penalty of Huang et al. [2] can be written as

$$\varphi_{\mathcal{G}}(\mathbf{w}) \triangleq \min_{\mathcal{J} \subseteq \mathcal{G}} \Big\{ \sum_{g \in \mathcal{J}} \eta_g \;\text{ s.t. }\; \text{Supp}(\mathbf{w}) \subseteq \bigcup_{g \in \mathcal{J}} g \Big\}, \tag{2}$$

where the η_g's are non-negative weights, and $\mathcal{J}$ is a subset of groups (of features) whose union covers the support of $\mathbf{w}$. When the weights η_g are well chosen, this penalty encourages solutions whose set of non-zero coefficients is in the union of a small number of groups, in other words the cardinality of $\mathcal{J}$ should be small. Note that computing this non-convex penalty $\varphi_{\mathcal{G}}(\mathbf{w})$ for a general group structure $\mathcal{G}$ is difficult since Eq. (2) is an NP-hard set cover problem [2]. Interestingly, we show later in this paper that with our "path coding" group structure $\mathcal{G}_p$ and particular weights η_g, the corresponding penalty $\varphi_{\mathcal{G}_p}$ can be computed in polynomial time.

Let us denote by η the vector $[\eta_g]_{g \in \mathcal{G}}$ in $\mathbb{R}^{|\mathcal{G}|}$, and by $\mathbf{N}$ the matrix in $\{0,1\}^{p \times |\mathcal{G}|}$ whose columns are indexed by the groups g in $\mathcal{G}$, such that the entry $\mathbf{N}_{jg}$ is equal to one when the index j is in the group g, and zero otherwise. We can now rewrite Equation (2) as a boolean linear program

$$\varphi_{\mathcal{G}}(\mathbf{w}) = \min_{\mathbf{x} \in \{0,1\}^{|\mathcal{G}|}} \left\{ \eta^\top \mathbf{x} \;\text{ s.t. }\; \mathbf{N}\mathbf{x} \geq \text{Supp}(\mathbf{w}) \right\},$$

where, with an abuse of notation, $\text{Supp}(\mathbf{w})$ is here a vector in $\{0,1\}^p$ such that its j-th entry is one if j is in the support of $\mathbf{w}$ and 0 otherwise. Let us denote by $|\mathbf{w}|$ the vector in $\mathbb{R}^p_+$ obtained by replacing the entries of $\mathbf{w}$ by their absolute value. We can now consider a convex relaxation of $\varphi_{\mathcal{G}}$:

$$\psi_{\mathcal{G}}(\mathbf{w}) \triangleq \min_{\mathbf{x} \in \mathbb{R}^{|\mathcal{G}|}_+} \left\{ \eta^\top \mathbf{x} \;\text{ s.t. }\; \mathbf{N}\mathbf{x} \geq |\mathbf{w}| \right\}. \tag{3}$$

The following lemma tells us that we have in fact obtained the penalty introduced by Jacob et al. [1].

Lemma 1 (Relation between $\psi_{\mathcal{G}}$ and the penalty of Jacob et al. [1].)
Suppose that any pattern in $\{0,1\}^p$ can be represented by a union of groups in $\mathcal{G}$. Then, the function $\psi_{\mathcal{G}}$ defined in Eq. (3) is equal to the penalty introduced by Jacob et al. in [1].[4]

[1] This issue was confirmed to us in a private communication with Laurent Jacob [1], and this was one of our main motivation for developing new algorithmic tools overcoming this problem.

[2] A path is defined as a sequence of vertices $(v_1, \ldots, v_k)$ such that for all $1 \leq i < k$, we have $(v_i, v_{i+1}) \in E$.

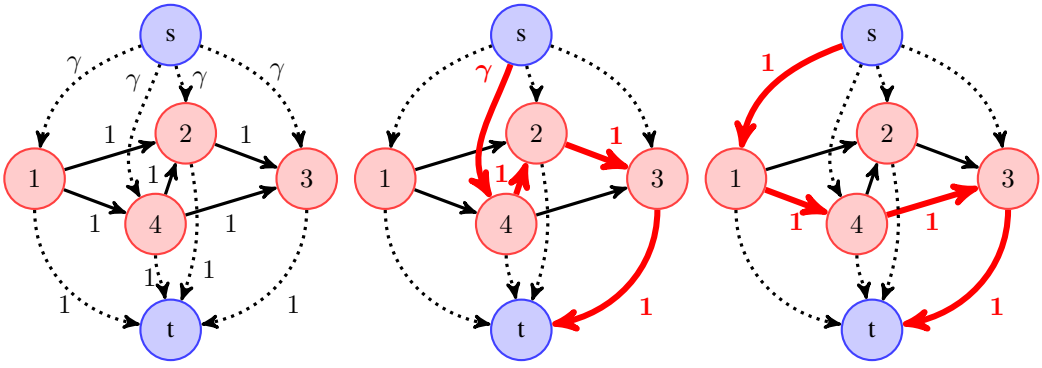
[3] Links of a different nature have been drawn between sparse estimation and network flows [3, 6], which neither involve the sparsity-inducing penalties of [1] and [2], nor path selection problems.

[4] In fact, the penalty of Jacob et al. [1] involves a sum of ℓ_2-norms, which needs to be replaced by ℓ_∞-norms for the lemma to be valid. We omit this detail here for space limitation reasons.

Note that Jacob et al. [1] have introduced their penalty from a different perspective and the link between Eq. (3) and their work is not obvious at first sight. For space limitation reasons, the proof of this lemma and the precise form of the penalty given in [1] are omitted here.

We are now interested in automatically selecting a small number of connected subgraphs from a directed acyclic graph $G = (V, E)$. We have already discussed in Section 1 group structures $\mathcal{G}$, and introduced $\mathcal{G}_p$ the *set of paths in* G. As a result, the path coding penalties $\varphi_{\mathcal{G}_p}$ and $\psi_{\mathcal{G}_p}$ encourage solutions which are sparse while forming a subgraph that can be covered by a small number of paths. To make this approach computationally tractable, we use network flow optimization [4] and an appropriate choice for the weights η_g. We define the weights as $\eta_g = \gamma + |g|$ for all g in $\mathcal{G}_p$, where γ is a new parameter allowing us to control the tradeoff between sparsity and connectivity of the solution. One can show that when $\gamma = 0$, the functions $\varphi_{\mathcal{G}_p}$ and $\psi_{\mathcal{G}_p}$ respectively become the ℓ_0- and the ℓ_1-penalties, therefore encouraging sparsity but not connectivity. On the other hand, when γ is large and the term $|g|$ is negligible, $\varphi_{\mathcal{G}_p}(\mathbf{w})$ simply "counts" the minimum number of paths to cover the support of $\mathbf{w}$, thereby encouraging connectivity regardless of the sparsity of $\mathbf{w}$.

A rationale for this choice of weights, which will allow us to deal with the exponential number of variables in Eq. (2) and (3), and later address problem (1), is that the weights $\eta_g = \gamma + |g|$ defined above can be seen as sums of costs along arcs of G when the latter is enriched with two additional nodes s and t, respectively dubbed "source" and "sink". We assume that there exist arcs between (s, j) and (j, t) for every vertex j in V with respective costs γ and 1, and that every arc in E has unit cost. We denote this new graph by $G' = (V', E')$, and represent it in Figure 1a.



(a) Graph G' with arcs costs. (b) Path $g = (4, 2, 3)$, $\eta_g = \gamma + 3$. (c) unit (s, t)-path flow.

Figure 1: (a) G' is obtained by adding a source s and sink t to a DAG with four nodes $\{1, 2, 3, 4\}$. (b) The weight η_g associated to the path $g = (4, 2, 3)$ on G is the sum of the costs on arcs along the path $(s, 4, 2, 3, t)$ in G'. (c) Example of a unit (s, t)-path flow along the path $g = (1, 4, 3)$.

3 Network Flow Optimization

The techniques presented in this section involve network flows, where a *flow* f on $G' = (V', E')$ is defined as a non-negative vector on arcs $[f_{uv}]_{(u,v)\in E'}$ that satisfies two constraints: (i) the value of the flow f_{uv} on an arc (u, v) in E' should satisfy lower and upper bounds $l_{uv} \leq f_{uv} \leq \delta_{uv}$, where l_{uv} and δ_{uv} are respectively called lower and upper capacities; (ii) the sum of incoming flow at a vertex is equal to the sum of outgoing flow except for the source and the sink (see [4]). The set of possible flows on G' is denoted by $\mathcal{F}$. We have defined in Section 2 some costs on the arcs of G', which we denote by c_{uv} for (u, v) in E'. A *minimum cost flow problem* consists of finding a flow f in $\mathcal{F}$ minimizing a linear cost, for example the total cost $\sum_{(u,v)\in E'} c_{uv} f_{uv}$. Solving this type of linear programs has been the topic of much research, which has led to efficient dedicated algorithms [4, 7].

For space limitation reasons, all proofs in this section are omitted and we only sketch the main ideas here. We first define an (s, t)-path flow as a flow vector in $\mathcal{F}$ carrying the same positive amount of flow from s to t along a path, as illustrated in Figure 1c. The main tool we use is the *flow decomposition theorem* (see [4]), which says that every flow vector can always be decomposed into a sum of (s, t)-path flows, and cycle flows (units of flow sent along a cycle, which in the case of DAGs do not exist). This apparently simple theorem has an interesting consequence, which is that

minimum cost flow problems can be seen from two equivalent viewpoints. One is either looking for the value f_{uv} of a flow on every arc (u,v) of the graph, or one should decide how much flow should be sent on every (s,t)-path and cycle [4]. We remark that the total cost of a flow sending one unit from s to t along a path g in G is exactly η_g, as illustrated in Figure 1b. This allows us to reformulate our optimization problems on paths in G as optimization problems on (s,t)-path flows in G', which in turn are equivalent to minimum cost flow problems and can be solved in polynomial time. Note that this equivalence does not hold when we have cycle flows, and this is the reason why we have assumed G to be acyclic. We also remark that it is possible to define constraints and/or costs for the amount of flow going through a vertex j in $V=\{1,\dots,p\}$, which we denote by $s_j(f)$. It is indeed easy to show that a vertex with a capacity/cost can be equivalently replaced in the network by two vertices, linked by an arc that carries the capacity/cost [4]. We can now state our propositions:

Proposition 1 (Computing $\varphi_{\mathcal{G}_p}$.)
Let $\mathbf{w}$ *be in* $\mathbb{R}^p$*, and consider the network* G' *with its costs defined in Section 2. Then,*

$$\varphi_{\mathcal{G}_p}(\mathbf{w}) = \min_{f\in\mathcal{F}} \left\{ \sum_{(u,v)\in E'} f_{uv}c_{uv} \;\; \textit{s.t.} \;\; s_j(f) \geq 1, \; \forall j \in \textit{Supp}(\mathbf{w}) \right\}, \tag{4}$$

which is a minimum cost flow problem with some lower-capacity constraints.

As already mentioned in Section 2, the penalty $\varphi_{\mathcal{G}}$ is NP-hard to compute for a general group structure $\mathcal{G}$. Interestingly, Proposition 1 shows that it is not the case for the path coding penalty $\varphi_{\mathcal{G}_p}$, despite the exponential number of groups. We now show that the same methodology applies to $\psi_{\mathcal{G}_p}$:

Proposition 2 (Computing $\psi_{\mathcal{G}_p}$.)
Let $\mathbf{w}$ *be in* $\mathbb{R}^p$*, and consider the network* G' *with its costs defined in Section 2. Then,*

$$\psi_{\mathcal{G}_p}(\mathbf{w}) = \min_{f\in\mathcal{F}} \left\{ \sum_{(u,v)\in E'} f_{uv}c_{uv} \;\; \textit{s.t.} \;\; s_j(f) \geq |\mathbf{w}_j|, \; \forall j \in \{1,\dots,p\} \right\}, \tag{5}$$

which is a minimum cost flow problem with some lower-capacity constraints.

We now address problem (1) with proximal gradient methods (see [5] for a review). These iterative schemes are designed to minimize objective functions of the same form as Eq. (1), when the function g is convex and differentiable with a Lipschitz continuous gradient. When Ω is convex ($\Omega=\psi_{\mathcal{G}_p}$ for example), these methods are known to converge to a solution and admit variants with optimal convergence rates among first-order methods [8]. When Ω is non-convex, e.g., $\Omega=\varphi_{\mathcal{G}_p}$, the guarantees are weak (finding the global optimum is out of reach, except in particular cases), but they can iteratively decrease the value of the objective function. The only requirement to use proximal gradient methods is to be able to efficiently compute the proximal operator defined as follows:

Definition 1 (Proximal Operator.)
The proximal operator associated with a regularization term $\lambda\Omega$*, which we denote by* $\textit{Prox}_{\lambda\Omega}$*, is the function that maps a vector* $\mathbf{u}\in\mathbb{R}^p$ *to the unique (by strong convexity) solution of*

$$\min_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{u}-\mathbf{w}\|_2^2 + \lambda\Omega(\mathbf{w}). \tag{6}$$

Computing efficiently this operator has been shown to be possible in many cases, notably for ℓ_0-, ℓ_1-, or other sparsity-inducing penalties (see [5, 6] and references therein). We now show that it is also the case for the path coding penalties $\varphi_{\mathcal{G}_p}$ and $\psi_{\mathcal{G}_p}$.

Proposition 3 (Computing the Proximal Operator of $\varphi_{\mathcal{G}_p}$.)
Let $\mathbf{u}$ *be in* $\mathbb{R}^p$*, and consider the network* G' *with its costs defined in Section 2. Let us define*

$$f^\star \in \arg\min_{f\in\mathcal{F}} \left\{ \sum_{(u,v)\in E'} f_{uv}c_{uv} + \sum_{j=1}^{p} \frac{1}{2}\max\left(\mathbf{u}_j^2(1-s_j(f)),0\right) \right\}, \tag{7}$$

which is a minimum cost flow problem, with piecewise linear costs. Denoting by $\mathbf{w}^\star \triangleq \textit{Prox}_{\varphi_{\mathcal{G}_p}}[\mathbf{u}]$*, we have for all* j *in* $V=\{1,\dots,p\}$ *that* $\mathbf{w}_j^\star = \mathbf{u}_j$ *if* $s_f(f^\star)>0$ *and* 0 *otherwise.*

Note that even though the formulation (6) is nonconvex when Ω is the function $\varphi_{\mathcal{G}_p}$, its global optimum can be found by solving the convex problem described in Eq. (7).

Proposition 4 (Computing the Proximal Operator of $\psi_{\mathcal{G}_p}$.)
Let $\mathbf{u}$ *be in* $\mathbb{R}^p$*, and consider the network* G' *with its costs defined in Section 2. Let us define*

$$f^\star \in \operatorname*{arg\,min}_{f \in \mathcal{F}} \left\{ \sum_{(u,v) \in E'} f_{uv} c_{uv} + \sum_{j=1}^{p} \frac{1}{2} \max(|\mathbf{u}_j| - s_j(f), 0)^2 \right\}, \tag{8}$$

which is a quadratic minimum cost flow problem, with piecewise quadratic costs. Denoting by $\mathbf{w}^\star \triangleq \mathit{Prox}_{\psi_{\mathcal{G}_p}}[\mathbf{u}]$*, we have for all* j *in* $V = \{1, \ldots, p\}$*,* $\mathbf{w}^\star_j = \operatorname{sign}(\mathbf{u}_j) \min(|\mathbf{u}_j|, s_j(f))$.

We have implemented the scaling push-relabel algorithm [7], which is designed to solve minimum cost flow problems with linear or piecewise linear costs, and one of its variants introduced in [9] to deal with the quadratic costs in Eq. (8). This algorithm runs in (weakly) polynomial time [7] and has proven to be efficient and scalable in practice for our experiments. We remark that we are also able to efficiently compute the dual norm of $\psi_{\mathcal{G}_p}$ (which is itself a norm), such that we can obtain duality gaps for problem (1) when $\Omega = \psi_{\mathcal{G}_p}$. For space limitation reasons, this is omitted here.

4 Experiments and Applications

We now present experiments on image and genomic data. All the algorithms have been implemented in C++ with a Matlab interface, they will be made available in an open-source software package.

Image Denoising. We compare here the penalties $\varphi_{\mathcal{G}_p}$ and $\psi_{\mathcal{G}_p}$ with the traditional ℓ_0- and ℓ_1-ones. We use a classical patch-based denoising scheme [10], which consists of the following steps: (i) extract all $p = e \times e$ overlapping patches from a noisy image; (ii) denoise independently each patch with a sparse approximation; (iii) since each pixel belongs to several patches, average the estimates to reconstruct the final image. We consider an orthogonal discrete cosine transform dictionary $\mathbf{X}$ in $\mathbb{R}^{p \times p}$ whose elements can be organized by vertical and horizontal frequencies on a grid. The DAG we consider follows this organization and is represented in Figure 2. Since $\mathbf{X}$ is orthogonal, denoising a patch $\mathbf{y}$ in $\mathbb{R}^p$ amounts to computing $\mathbf{X}\mathbf{w}^\star$, where $\mathbf{w}^\star = \operatorname{Prox}_{\lambda\Omega}[\mathbf{X}^\top \mathbf{y}]$. The dataset contains 12 classical images. We optimize the parameters e, λ, γ on the first 3 images, keeping the last 9 images as a test set and report denoising results in Table 1. For this task, we observe that nonconvex penalties perform better than convex ones. When the noise level σ is sufficiently high, the penalty $\varphi_{\mathcal{G}_p}$ significantly outperforms other ones. The optimization task is difficult and consists of solving a large number (several millions) of small-scale problems (typically $p \approx 100$). On a 1.2Ghz laptop CPU (core i3 330UM), for $\sigma = 10$ and $p = 64$ we denoise approximately $4\,000$ patches per second for the penalty $\varphi_{\mathcal{G}_p}$, and $1\,800$ for $\sigma = 50$ and $p = 196$. The penalty $\psi_{\mathcal{G}_p}$ was slower to use in practice: The above numbers respectively become 70 and 130.

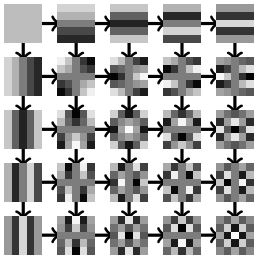


Figure 2: DCT dictionary

Table 1: Denoising results for the 9 test images. The numbers represent the average PSNR in dB (higher is better, a gain of 1dB approximately reduces the mean squared error by 20%.). Pixel values are scaled between 0 and 255 and sigma is between 5 and 100.

σ	5	10	15	20	25	50	100
ℓ_0	**37.04**	33.15	31.03	29.59	28.48	25.26	22.44
ℓ_1	36.42	32.28	30.06	28.59	27.51	24.48	21.96
$\varphi_{\mathcal{G}_p}$	37.01	**33.22**	**31.21**	**29.82**	**28.77**	**25.73**	**22.97**
$\psi_{\mathcal{G}_p}$	36.32	32.17	29.99	28.54	27.49	24.54	22.12

Breast Cancer Data. The goal of this experiment is to compare our approach with the one of Jacob et al. [1]—that is, the penalty $\psi_{\mathcal{G}}$ where the groups are all pairs of vertices linked by an arc, following a similar experiment as in their paper. The dataset consists of gene expression data from $p = 7\,910$ genes in $n = 295$ breast cancer tumors and the goal is to classify metastatic samples versus non-metastatic. We use the same predefined graph of genes used in [1], which we denote by G_0, and following [1], we try to select connected components involved in the metastatic nature of the cancer. Despite the fact that G_0 is not a DAG, we show here that *our approach can lead to better results than [1] in terms of prediction performance, connectivity of the solution and stability of the selected subnetwork.* For that we treat G_0 it as directed by choosing arbitrary directions on the arcs, and randomly removing some arcs along cycles, resulting in a DAG, which we denote by G.

We consider the formulation (1) where g is the logistic regression loss, and compare the penalties $\psi_{\mathcal{G}_p}$ and $\varphi_{\mathcal{G}_p}$ with other regularization functions. We proceed by randomly sampling 20% of the data as a test set, keeping 80% for training, selecting the parameters λ, γ using internal 5-fold cross validation on the training set, and we measure the balanced error rate between the two classes on the test set. We have repeated this experiment 20 times and report the averaged results in Table 2. We conclude that the goal of producing sparse, connected and stable solutions is clearly achieved by our penalty $\psi_{\mathcal{G}_p}$, outperforming Jacob et al. and other penalties in that aspect. In terms of error rate, our penalty seems to perform better than [1], but statistically assessing the difference is a difficult problem here: the data is very noisy and the number of samples is small, resulting in highly variable results, a problem also reported in [1] and other papers using this dataset.

As far as the computational performance of our approach is concerned, computing one proximal operator for the selected parameters was relatively fast, approximately 0.17 seconds on our 1.2GHz laptop CPU, making it easy to solve Eq. (1) using proximal methods, but tends to be significantly slower when the solution is less sparse, for instance with small values for λ.

Table 2: Experimental results on the breast cancer dataset. Row "error": average balanced error rate in percents with standard deviations. Row "sparsity": average number of selected genes. Row "connectivity": average number of selected connected components in G_0. Row "stability": percentage of selected genes which are consistently selected in at least 10 (out of 20) experimental runs.

	ℓ_2^2	ℓ_1	$\ell_1 + \ell_2^2$	[1]	$\varphi_{\mathcal{G}_p}$	$\psi_{\mathcal{G}_p}$
error	31.0 ± 6.1	36.0 ± 6.5	31.5 ± 6.7	35.9 ± 6.8	36.0 ± 6.8	30.2 ± 6.8
sparsity	7910.0	32.6	929.6	68.4	10.15	69.9
connectivity	58.0	30.90	355.2	13.15	1.6	1.3
stability	100.0	7.9	30.9	6.1	27.3	32.0

5 Conclusion

Our paper proposes a new form of structured penalty for supervised learning problems where predicting features are sitting on a DAG, and where one wishes to automatically select a small number of connected subgraphs of the DAG. The computational feasibility of this form of penalty is established by making a new link between supervised path selection problems and network flows. Our penalties are both non-convex and convex, which can be appropriate choices depending on the task, and they can control the connectivity of the solution, whether one wishes to encourage large or small connected components. We show on genomic and image data that our approach is fast and effective for solving different prediction problems, and able to model long-range interactions between variables. Interestingly, our penalties seem to empirically perform well on more general graphs, when heuristically removing cycles, and we would like in the future to find a way to properly handle them.

References

[1] L. Jacob, G. Obozinski, and J.-P. Vert. Group Lasso with overlap and graph Lasso. In *ICML*, 2009.

[2] J. Huang, Z. Zhang, and D. Metaxas. Learning with structured sparsity. *arXiv:0903.3002v2*, 2009.

[3] V. Cehver, M. Duarte, C. Hedge, and R. G. Baraniuk. Sparse signal recovery using markov random fields. In *NIPS*, 2008.

[4] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice Hall, 1993.

[5] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. *arXiv:1108.0775v1*, 2011.

[6] J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. Network flow algorithms for structured sparsity. In *NIPS*, 2010.

[7] A. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *J. Algorithm*, 22(1):1–29, 1997.

[8] Y. Nesterov. Gradient methods for minimizing composite objective function. CORE paper, 2007.

[9] R. K. Ahuja, D. S. Hochbaum, and J. B. Orlin. Solving the convex cost integer dual network flow problem. *Manage. Sci.*, 49(7):950–964, 2003.

[10] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE T. Image Process.*, 54(12):3736–3745, 2006.