A Fast Algorithm for Separated Sparsity via Perturbed Lagrangians

Aleksander Mądry[*]
MIT
madry@mit.edu

Slobodan Mitrović[*]
EPFL
slobodan.mitrovic@epfl.ch

Ludwig Schmidt[*]
MIT
ludwigs@mit.edu

Abstract

Sparsity-based methods are widely used in machine learning, statistics, and signal processing. There is now a rich class of structured sparsity approaches that expand the modeling power of the sparsity paradigm and incorporate constraints such as group sparsity, graph sparsity, or hierarchical sparsity. While these sparsity models offer improved sample complexity and better interpretability, the improvements come at a computational cost: it is often challenging to optimize over the (non-convex) constraint sets that capture various sparsity structures. In this paper, we make progress in this direction in the context of separated sparsity – a fundamental sparsity notion that captures exclusion constraints in linearly ordered data such as time series. While prior algorithms for computing a projection onto this constraint set required quadratic time, we provide a perturbed Lagrangian relaxation approach that computes provably exact projection in only nearly-linear time. Although the sparsity constraint is non-convex, our perturbed Lagrangian approach is still guaranteed to find a globally optimal solution. In experiments, our new algorithms offer a $10\times$ speed-up already on moderately-size inputs.

1 Introduction

Over the past two decades, sparsity has become a widely used tool in several fields including signal processing, statistics, and machine learning. In many cases, sparsity is the key concept that enables us to capture important structure present in real-world data while making the resulting problem computationally tractable and suitable for mathematical analysis. Among the many applications of sparsity are sparse linear regression, compressed sensing, sparse PCA, and dictionary learning.

The first wave of sparsity-based techniques focused on the standard notion of sparsity that only constrains the number of non-zeros. Over time, it became apparent that extending the notion of sparsity to encompass more complex structures present in real-world data can offer significant benefits. Specifically, utilizing such additional structure often improves the statistical efficiency in estimation problems and the interpretability of the final result. There is now a large body of work on structured sparsity that has introduced popular models such as group sparsity and hierarchical sparsity [1, 2, 3, 4, 5, 6, 7, 8, 9]. These statistical improvements, however, come at a computational cost: the resulting optimization problems are often much harder to solve. The key reason is that the combinatorial sparsity structures give rise to non-convex constraints. Consequently, many of the resulting algorithms have significantly worse running time than their "standard sparsity" counterparts. This trade-off raises an important question: can we design algorithms for structured sparsity that match the time complexity of commonly used algorithms for standard sparsity?

In this paper, we address this question in the context of the *separated sparsity* model, a popular sparsity model for data with a known minimum distance between large coefficients [10, 11, 12, 13, 14]. In the one-dimensional case, such as time series data, neuronal spike trains are a natural example. Here, a minimum refractory period ensures separation between consecutive spikes. In two dimensions, separation constraints arise in the context of astronomical images or super-resolution applications [12, 15].

We introduce new algorithms for separated sparsity that run in *nearly-linear* time. This significantly improves over prior work that required at least quadratic time, which is quickly prohibitive for large data sets. An important consequence of our fast running time is that it enables methods that utilize separated sparsity yet are essentially as fast as their counterparts based on standard sparsity only. For instance, when we instantiate our algorithm in compressive sensing, the running time of our method matches that of common methods such as IHT or CoSaMP.

[*]Authors ordered alphabetically.

Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84. Copyright 2018 by the author(s).

Our algorithms stem from a primal-dual linear programming (LP) perspective on the problem. Our theoretical findings reveal a rich structure behind the separated sparsity model, which we utilize to obtain efficient methods. Interestingly, our final algorithm has a very simple form that can be interpreted as a *Lagrangian relaxation* of the sparsity constraint. In spite of the non-convexity of the constraint, our algorithm is still guaranteed to find the globally optimal solution.

We also show that these algorithmic and theoretical contributions directly translate into empirical efficiency. Specifically, we demonstrate that, compared to the state of the art procedures, our methods yield an order of magnitude speed-up already on moderate-size inputs. We run experiments on synthetic data and real world neuronal spike train signals.

2 Separated sparsity and applications

In this section, we formally define separated sparsity and the corresponding algorithmic problems.

First, we briefly introduce our notation. As usual, $[d]$ denotes the set $\{1, \ldots, d\}$. We say that a vector $\theta \in \mathbb{R}^d$ is k-sparse if θ contains at most k non-zero coefficients. We define the support of θ as the set of indices corresponding to non-zero coefficients, i.e., $\text{supp}(\theta) = \{i \in [d] \mid \theta_i \neq 0\}$. We let $\|\theta\|$ denote the ℓ_2-norm of a vector $\theta \in \mathbb{R}^d$.

Separated sparsity. For a support $\Omega \subseteq [d]$, let $\text{sep}(\Omega) = \min_{i \neq j \in \Omega} |i - j|$ be the minimum separation of two indices in the support. We define the following two sets of supports: set $\mathbb{M}_\Delta = \{\Omega \subseteq [d] \mid \text{sep}(\Omega) \geq \Delta\}$, and Δ-*separated sparsity* supports $\mathbb{M}_{k,\Delta} = \{\Omega \in \mathbb{M}_\Delta \mid |\Omega| = k\}$. That is, $\mathbb{M}_{k,\Delta}$ is the set of support patterns containing k non-zeros with at least $\Delta - 1$ zero entries between consecutive non-zeros.

In order to employ separated sparsity in statistical problems, we often want to add constraints based on the support set $\mathbb{M}_{k,\Delta}$ to optimization problems such as empirical risk minimization. A standard way of incorporating constraints into gradient-based algorithms is via a *projection operator*. In the context of separated sparsity, this corresponds to the following problem.

Problem 1 *For a given input vector $x \in \mathbb{R}^d$, our goal is to project x onto the set $\mathbb{M}_{k,\Delta}$, i.e., to find a vector $\hat{x}$ such that*

$$\hat{x} \in \operatorname*{arg\,min}_{x' \in \mathbb{R}^d \,:\, \mathit{supp}(x') \in \mathbb{M}_{k,\Delta}} \|x - x'\| \; . \tag{1}$$

Problem 1 is the main algorithmic problem we address in this paper.

Sparse recovery. Structured sparsity has been employed in a variety of machine learning tasks. In order to keep the discussion coherent, we present our results in the context of the well-known sparse linear model:

$$y \;=\; X\theta^* + e \tag{2}$$

where $y \in \mathbb{R}^n$ are the observations/measurements, $X \in \mathbb{R}^{n \times d}$ is the design or measurement matrix, and $e \in \mathbb{R}^n$ is a noise vector. The goal is to find a good estimate $\hat{\theta}$ of the unknown parameters θ^* up to the noise level.

The authors of [3] give an elegant framework for incorporating structured sparsity into the estimation problem outlined above. They design a general recovery algorithm that relies on a model-specific *projection oracle*. In the case of separated sparsity, this oracle is required to solve precisely the Problem 1 stated above.

3 The algorithm and our results

Given an arbitrary vector $x \in \mathbb{R}^d$, Problem 1 requires us to find a vector $\hat{x}$ such that $\hat{x} \in \mathcal{M}_{k,\Delta}$ and $\|x - \hat{x}\|$ is minimized. We now slightly reformulate the problem. Let $c \in \mathbb{R}^d$ be a vector such that $c_i = x_i^2$ for all i. Then it is not hard to see that this problem is equivalent to finding a set of k entries in the vector c such that each of these entries is separated by at least Δ and the sum of these entries is maximized. Hence our main algorithmic problem is to find a set of k entries in an non-negative input vector c so that the entries are Δ-separated and their sum is maximized. More formally, our goal is to find a support $\hat{S}$ such that

$$\hat{S} \in \operatorname*{arg\,max}_{S \in \mathbb{M}_{k,\Delta}} \sum_{i \in S} c_i \; . \tag{3}$$

In the following, we also consider a relaxed version of (3) called ProjLagr, which is parametrized by a trade-off parameter λ and a vector $\tilde{c}$:

$$\text{ProjLagr}(\lambda, \tilde{c}) \;:=\; \operatorname*{arg\,max}_{S \in \mathbb{M}_\Delta} \sum_{i \in S} \tilde{c}_i + \lambda\,(k - |S|) \; .$$

Intuitively, ProjLagr represents a Lagrangian relaxation of the sparsity constraint in Equation (3).

3.1 Algorithm

Our main contribution is a new algorithm for Problem 1 that we call ***L****agrangian* ***A****pproach to the* ***S****eparated* ***S****parsity* ***P****roblem* (LASSP). The pseudo code is given in Algorithm 1.

LASSP is a *Las Vegas* algorithm: it always returns a correct answer, but the running time of the algorithm is randomized. Concretely, LASSP repeats a main loop until a stopping criterion is reached. Every iteration of LASSP first adds a small perturbation to the coefficients c (see Line 3). This perturbation has only a small effect on the solution but improves the "conditioning" of the corresponding non-convex Lagrangian relaxation ProjLagr so that it returns a globally optimal solution that almost satisfies the constraint. As we show in Section 5.2, we can solve this relaxation (Line 4) in nearly-linear time. After the algorithm has solved the Lagrangian relaxation, it obtains the final support $\hat{S}$ in line 5 by solving ProjLagr on a slightly shifted $\hat{\lambda}$ to ensure that the constraint is satisfied with good probability.

Remark: We assume that the bit precision γ required to represent the coefficients c is finite, and we provide our results as a function of γ. For practical purposes, γ is usually a constant. Since the solution to Equation (3) is invariant under scaling by a positive integer and γ is finite, without loss of generality we assume that $c \in \mathbb{Z}^d$.

Algorithm 1 LASSP:

```
Input: $c \in \mathbb{Z}^d$, $k \in \mathbb{N}_+$
Output: A solution $\hat{S}$ to (3)
1: repeat
2:     Let $X \in \mathbb{Z}^d$ be a vector such that $X_i$ is chosen uniformly at random from $\{0, \ldots, d^3 - 1\}$, $\forall i$
3:     Define vector $\tilde{c} := d^4 c + X$
4:     Let $\hat{\lambda} := \arg\min_{\lambda \in \mathbb{Z}} \textsc{ProjLagr}(\lambda, \tilde{c})$. In case of ties, maximize $\hat{\lambda}$.
5:     Choose $\hat{S} \in \textsc{ProjLagr}\left(\hat{\lambda} - \frac{1}{d+1}, \tilde{c}\right)$
6: until $\left|\hat{S}\right| = k$
7: return $\hat{S}$
```

3.2 Main results

As our main result, in the following theorem we show that LASSP runs in nearly linear time and solves Problem 1.

Theorem 1 *Let $c \in \mathbb{R}^d$, and let $\gamma \in \mathbb{N}_+$ be the maximal number of bits needed to store any c_i. There is an implementation of* LASSP *that for every c computes a solution $\hat{c}$ to Problem 1. With probability $1 - 1/d$, the algorithm runs in time $O(d(\gamma + \log d))$.*

Combined with the framework of [3], we get the following.

Corollary 1 *Let y, X, θ^*, and e be as in the sparse linear model in Equation (2). We assume that supp$(\theta^*) \in \mathbb{M}_{k,\Delta}$ and that X satisfies the model-RIP for $\mathbb{M}_{k,\Delta}$. There is an algorithm that for every y and e returns an estimate $\hat{\theta}$ such that*

$$\|\hat{\theta} - \theta^*\|_2 \;\le\; C\|e\|_2 \;.$$

Moreover, the algorithm runs in time $\widetilde{O}(T_X + d)$, where T_X is the time of multiplying the matrices X and X^T by a vector.

The corollary shows that, up to logarithmic factors, the running time is dominated by $T_X + d$. This matches the time complexity of standard sparse recovery and shows that we can utilize separated sparsity without a significant increase in time complexity. We validate these theoretical findings in Section 7 by showing that LASSP runs significantly faster than the state of the art algorithm used for sparse recovery with separation constraints.

Our algorithm LASSP is randomized. However, we also design a deterministic nearly-linear time algorithm and prove the following theorem. For clarity of exposition, the statement of the deterministic algorithm and the proof of the theorem are deferred to Appendix J.

Theorem 2 *Let $c \in \mathbb{R}^d$ be the input vector and let γ be as in Theorem 1. Then there is an algorithm that computes a solution $\hat{c}$ satisfying Equation* (1) *and runs in time $O(d(\gamma + \log k)\ \log d\ \log \Delta)$.*[†]

3.3 Further results

In Appendix C we present a dynamic programming approach that for a specific, but also natural, family of instances solves the separated sparsity problem in even linear time.

We also consider a natural extension to the 2D-variant of the separated sparsity projection problem and show that it is NP-hard in Appendix D. Moreover, in Appendix E, we extend our model to allow for blocks of separated variables and show that our algorithms for $\mathbb{M}_{k,\Delta}$ also applies to the more general variant. Finally, separated sparsity can be used to model signals in which a longer pattern is repeated multiple times so that any two patterns are at least Δ apart. This model is called *disjoint pulse streams* [12]. Again, the algorithmic core remains the same and algorithms for $\mathbb{M}_{k,\Delta}$ can also be used for this generalization.

3.4 Additional related work

The papers [10, 14] are closely related to our work. The paper [10] proposed the separated sparsity model, provided a sample complexity upper bound, and gave an LP-based model-projection algorithm. However, they resorted to a black-box approach for solving the LP, that lead to a fairly prohibitive $O(d^{3.5})$ time complexity. Recently, [14] provided a faster dynamic program for this problem with a time complexity of $O(d^2)$ and also showed a sample complexity lower bound. The algorithmic aspect of these papers is the main difference from our work: we exploit structure in both the primal and dual formulations of the LP and give an algorithm that *provably* runs in *nearly-linear* time.

Beside the papers addressing the core algorithmic question of projecting onto separated sparse vectors, there is much of work utilizing the sparsity model for applications in neural signal processing [11, 12, 13] and recovery with coherent dictionaries [16, 17]. In the latter application, the separated sparsity constraint enforces that the signal representation only consists of incoherent dictionary atoms. We expect that our algorithmic techniques will also lead to improvements in the context of these applications.

[†]After this work was done, it was pointed to us by Arturs Backurs and Christos Tzamos that this problem exhibits concave Monge property, which can be used to solve the problem in nearly-linear time with different algorithm. For more information, we point the reader to https://arxiv.org/abs/1802.06440.

4 Proof of correctness and the roadmap

We begin our analysis by proving that LASSP returns a correct result *if the algorithm terminates*. As we will see later, establishing termination is the crucial part of the analysis. The following lemma is a useful warm-up for understanding how the different pieces of our algorithm fit together.

Lemma 1 *When* LASSP *terminates, it outputs a support $\hat{S}$ such that x restricted to $\hat{S}$ is a solution to Problem 1.*

Proof. As we have argued above, the problems in Equations (1) and (3) are equivalent. So, we show that LASSP outputs a solution to the problem in Equation (3).

Let $\hat{S}$ be the set returned by LASSP. By the condition of the loop in Line 6, we have $|\hat{S}| = k$. So, as $\hat{S} \in \mathbb{M}_\Delta$ (see the definition of PROJLAGR), we have $\hat{S} \in \mathbb{M}_{k,\Delta}$.

Now, towards a contradiction, assume that support $\hat{S}$ is not a solution to the problem in Equation (3), while support $S^\star$ is. This implies that $\sum_{i \in S^\star} c_i > \sum_{i \in \hat{S}} c_i$. Now since, without loss of generality, we assumed that $c \in \mathbb{Z}^d$, the last inequality implies $\sum_{i \in S^\star} c_i \geq 1 + \sum_{i \in \hat{S}} c_i$, and hence

$$\sum_{i \in S^\star} d^4 c_i \geq d^4 + \sum_{i \in \hat{S}} d^4 c_i \ . \tag{4}$$

Observe that for any support $S \in \mathbb{M}_{k,\Delta}$, the term $\lambda\,(k - |S|)$ equals zero, and recall that $\hat{S}, S^\star \in \mathbb{M}_{k,\Delta}$. Furthermore, by the definition of the random vector X in line 2 and from (4)

$$\begin{aligned} \sum_{i \in S^\star} \left(d^4 c_i + X_i\right) \quad &\geq \quad \sum_{i \in S^\star} d^4 c_i \geq d^4 + \sum_{i \in \hat{S}} d^4 c_i \\ &> \quad \sum_{i \in \hat{S}} \left(d^4 c_i + X_i\right). \end{aligned}$$

Since $S^\star \in \mathbb{M}_\Delta$, this chain of inequalities contradicts Line 5 of LASSP which chooses $\hat{S}$ as an optimal solution to PROJLAGR $\left(\hat{\lambda} - 1/(d+1), d^4 c + X\right)$. This further implies that x restricted to $\hat{S}$ is a solution to Problem 1. □

4.1 Roadmap

Lemma 1 shows that LASSP outputs the right answer if it terminates. But does LASSP terminate on every input? Answering this question is the most intricate part of this paper. We split the proof in two main pieces. The first part is Section 5, where we provide an alternative view on the separated sparsity problem based on linear programming duality. The duality view paves the way towards proving our main results. In particular, we show that the subroutines in LASSP can be implemented quickly.

Lemma 2 *Single iteration of* LASSP *can be implemented to run in time $O(d(\gamma + \log d))$.*

The second part is Section 6, where we further study the duality view on separated sparsity. We show that after perturbing the input instance in Line 3 of LASSP, the support obtained with a shifted $\hat{\lambda}$ in Line 5 has cardinality k with high probability.

Lemma 3 *Algorithm* LASSP *runs only a single iteration with probability at least $1 - 1/d$.*

Together with Lemma 1, these results yield Theorem 1.

5 Part I – To duality and further

We now analyze the running time of a single iteration of LASSP. We provide a series of equivalences, as illustrated in Figure 1, in order to exploit structure in the separated sparsity problem. More precisely, we start with a linear programming (LP) view on separated sparsity. It has already been shown that this viewpoint yields a totally unimodular LP [10], which implies that the LP has an integral solution. Hence solving the LP solves the separated sparsity projection in Problem 1. However, prior work did not utilize this connection to reason about the power of the Lagrangian relaxation approach to the problem.

We begin our detailed analysis of the LP with the dual program $\mathcal{D}$. By strong duality, the value of $\mathcal{D}$ equals the value of the primal LP. Then we cast $\mathcal{D}$ as minimization of LP $\mathcal{D}_\lambda$ over λ. This reduction will play the central role in our analysis and connect $\mathcal{D}_\lambda$ to Line 4 of LASSP.

5.1 The LP Perspective

We start with a linear programming view on problem (3) by considering its LP relaxation denoted by $\mathcal{P}$:

$$\begin{array}{lll} \text{maximize} & c^T u & \\ \text{subject to} & \sum_{i=1}^{d} u_i = k & \\ & \sum_{j=i}^{\min\{i+\Delta-1,n\}} u_j \leq 1 & \forall i = 1 \dots d \\ & u_i \geq 0 & \forall i = 1 \dots d \end{array}$$

Given an LP $\mathcal{A}$ we use VAL $\mathcal{A}$ to denote its optimal objective value. As already noted, $\mathcal{P}$ is totally unimodular and thus there always exists an optimal solution to it that is integral [18]. For completeness, we provide the proof of total unimodularity in Appendix F.

Remark: This implies that a solution to $\mathcal{P}$ can be used to obtain a solution to the separated sparsity model projection: if $u^\star$ is an optimal solution to $\mathcal{P}$, we can derive the optimal support of (3) from the non-zero entries among the LP variables $u_1^\star, \dots, u_d^\star$. It is unclear, however, if there is a way to directly solve this LP fast, e.g. it is not known how to solve $\mathcal{P}$ directly in time matching the running time of our algorithm LASSP.

A key step in our approach is understanding the separated sparsity structure from the dual point of view. The dual LP

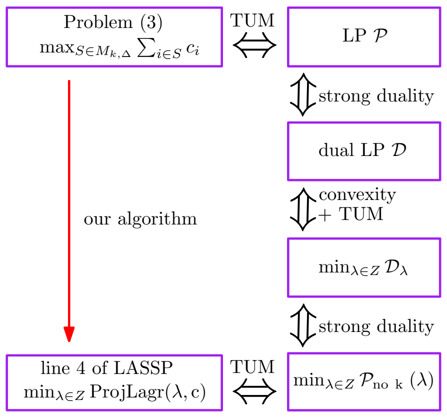
Problem (3)
$\max_{S\in M_{k,\Delta}} \sum_{i\in S} c_i$
TUM
LP $\mathcal{P}$
strong duality
dual LP $\mathcal{D}$
convexity + TUM
our algorithm
$\min_{\lambda\in Z} \mathcal{D}_\lambda$
strong duality
line 4 of LASSP
$\min_{\lambda\in Z}$ ProjLagr(λ, c)
TUM
$\min_{\lambda\in Z} \mathcal{P}_{\text{no k}}(\lambda)$

Figure 1: The values of the problems in the diagram are equal. Every equivalence relation carries structural information that we utilize in our analysis.

to $\mathcal{P}$, denoted by $\mathcal{D}$, is given as follows

$$\begin{array}{llll} \text{minimize} & w_0 k + \sum_{i=1}^{d} w_i & & \\ \text{subject to} & w_0 + \sum_{\substack{j\,:\,j\ge 1 \text{ and} \\ j\le i\le j+\Delta-1}} w_j \ge c_i & & \forall i = 1\dots d \\ & w_i \ge 0 & & \forall i = 1\dots d \\ & w_0 \in \mathbb{R} & & \end{array}$$

Then, as $\mathcal{P}$ is integral, so is $\mathcal{D}$.

Corollary 2 *For an integer k and a vector of integers c, there exists $\hat{w}$ such that $\hat{w}$ is an optimum of $\mathcal{D}$ and $\hat{w}_0 \in \mathbb{Z}$.*

We also define $\mathcal{D}_\lambda$ as the LP $\mathcal{D}$ in which the variable w_0 is set to λ. Now, it is not hard to show the following lemma, whose proof is deferred to Lemma 10 in Appendix F.

Lemma 4 *$\mathcal{D}_\lambda$ is convex with respect to λ.*

From the definition of $\mathcal{D}$, the following equality holds $\text{VAL}\,\mathcal{D} = \min_{\lambda\in\mathbb{R}} \text{VAL}\,\mathcal{D}_\lambda$. Furthermore, Corollary 2 implies that it is sufficient to consider λ in $\mathbb{Z}$ only, i.e.

$$\text{VAL}\,\mathcal{D} = \min_{\lambda\in\mathbb{Z}} \text{VAL}\,\mathcal{D}_\lambda. \tag{5}$$

Now, Lemma 4 implies that we can obtain VAL $\mathcal{D}$ by applying ternary search over λ on function VAL $\mathcal{D}_\lambda$.

5.2 Implementing one iteration of LASSP efficiently

We now derive the final connection between $\mathcal{D}$ and LASSP which will enable us to obtain $\hat{\lambda}$ at line 4 in nearly-linear time. To that end, consider $\mathcal{P}_{\text{no-}k}(\lambda)$ defined as

$$\begin{array}{llll} \text{maximize} & c^T u + \lambda(k - \mathbb{1}^T u) & & \\ \text{subject to} & \sum_{j=i}^{\min\{i+\Delta-1,d\}} u_j \le 1 & & \forall i = 1\dots d \\ & u_i \ge 0 & & \forall i = 1\dots d \end{array}$$

Observe that compared to $\mathcal{P}$, $\mathcal{P}_{\text{no-}k}(\lambda)$ does not contain the sparsity constraint. Furthermore, the LP $\mathcal{P}_{\text{no-}k}(\lambda)$ is a relaxed version of ProjLagr (λ, c). Also, as $\mathcal{P}$ is, then $\mathcal{P}_{\text{no-}k}(\lambda)$ is totally unimodular. Now this sequence of conclusions results in the following.

Corollary 3 *Problems $\mathcal{P}_{\text{no-}k}(\lambda)$ and* ProjLagr (λ, c) *are equivalent.*

To obtain the final connection, we consider $L_\mathcal{P}$ given by

$$L_\mathcal{P} := \min_{\lambda\in\mathbb{R}} \mathcal{P}_{\text{no-}k}(\lambda).$$

Next, note that the dual of $\mathcal{P}_{\text{no-}k}(\lambda)$ is $\mathcal{D}_\lambda$. Hence, the strong duality implies $\text{VAL}\,\mathcal{D}_\lambda = \text{VAL}\,\mathcal{P}_{\text{no-}k}(\lambda)$. That together with $\text{VAL}\,\mathcal{D} = \min_{\lambda\in\mathbb{R}} \text{VAL}\,\mathcal{D}_\lambda$ yields that $\mathcal{D}$ and $L_\mathcal{P}$ coincide as functions in λ. Furthermore, since we can solve $\mathcal{D}$ by applying ternary search over integral values of λ and $\mathcal{D}_\lambda$, we can solve $L_\mathcal{P}$ by applying ternary search over integral values of λ and function $\mathcal{P}_{\text{no-}k}(\lambda)$. But since $\mathcal{P}_{\text{no-}k}(\lambda)$ and ProjLagr (λ, c) are equivalent, we can also obtain $\hat{\lambda}$ at line 4 of LASSP by applying ternary search over λ.

Now, it is very easy to see that for an optimal solution $w^\star$ of $\mathcal{D}$ we have $w_0^\star \le \max_i |c_i|$. It is also not hard to show that there is an optimal solution such that $w_0^\star \ge -(k-1)\max_i |c_i|$ (see Lemma 18). Therefore, in order to find optimal $\hat{\lambda}$ it suffices to execute $O(\log \max_i |c_i| + \log k) = O(\gamma + \log d)$ iterations of ternary search.

Every iteration of the ternary search invokes ProjLagr, which can be implemented to run in linear time.

Lemma 5 *Given λ and $\hat{c} \in \mathbb{R}^d$, there is an algorithm that finds support $\hat{S} \in$* ProjLagr $(\lambda, \hat{c})$ *in time $O(d)$.*

Proof. Observe that for a fixed λ, solving ProjLagr $(\lambda, \hat{c})$ is equivalent to finding support $S' \in \mathbb{M}_\Delta$ that maximize $\sum_{i\in S'} (\hat{c}_i - \lambda)$. Hence, we can reinterpret ProjLagr $(\lambda, \hat{c})$ as follows: given a vector $\tilde{c} := \hat{c} - \lambda\mathbb{1}$, select a subset of $[d]$ of indices (not necessarily k of them) so that (i) every two indices are at least Δ apart, and (ii) the sum of the values of $\tilde{c}$ at the selected indices is maximized.

This task can be solved by standard dynamic programming in the following way. For every i, we define s_i to be the maximum value of the described task restricted to the first i indices of $\tilde{c}$. Then, it is easy to see that $s_{i+1} = \max\{s_i, s_{i+1-\Delta} + \tilde{c}_{i+1}\}$. Namely, we can either decide not to select index $i+1$, in which case the best is already contained in s_i; or, we can decide to select index $i+1$ which has value $\tilde{c}_{i+1}$ and for the rest we consider $s_{i+1-\Delta}$. Therefore, s_d can be obtained in time $O(d)$. Now it is easy to reconstruct the corresponding support in linear time. For completeness, we provide a full algorithm and a detailed proof in Lemma 12, Appendix G. □

Putting all together proves Lemma 2.

6 Part II – Active constraints

Note that the chain of equivalences present in Figure 1 shows that $\min_\lambda$ ProjLagr(λ, c) outputs the sum of coordinates of an optimal solution of problem 1. Prior to our work, it was not even known how to obtain this value in time faster than $O(dk)$, while our result shows we can compute it in nearly-linear time. So, it is natural to ask whether the same relaxation also outputs a support of size k? The answer is, unfortunately, no. To see that, consider the example: $c = (4, 7, 5, 0, 0, 5, 8, 5)$, $\Delta = 2$, and $k = 3$. For $\lambda > 2$ every solution $u^\star_\lambda$ to ProjLagr(λ, c) is such that $u^\star_\lambda$ has less than k non-zeros. On the other hand, for every $\lambda \le 2$ the exists a solution $u^\star_\lambda$ such that $u^\star_\lambda$ contains more than k non-zeros. Therefore, there is no λ, neither $\lambda - 1/(d+1)$, for which ProjLagr(λ, c) provably outputs a support of cardinality k. This also suggests that the perturbation we apply in lines 2-3 is essential!

Instead of studying lines 2-5 of LASSP directly, we shift our focus to $\mathcal{D}_\lambda$. In particular, we exhibit very close connection between its structure and the sparsity of the primal solution, which we present via the notion of "active constraints". Then we use these findings in our analysis to show that slight perturbation of the input instance, while not affecting the value of the solution, makes it possible to obtain a solution to problem 3 by applying Lagrangian relaxation.

6.1 Solving $\mathcal{D}_\lambda$

Observe that once we fixed the value of w_0, all remaining constraints in $\mathcal{D}_\lambda$ are “local” since they only affect a known interval of length Δ. They are also ordered in a natural way. As a result, we can solve $\mathcal{D}_\lambda$ by making a single pass over these variables. Starting with w_1 and all variables set to 0, we consider each constraint from left to right and increase the variables to satisfy these constraints in a lazy manner. That is, if in our pass we reach a constraint with index i that is still not satisfied, we increase the value of w_i until that constraint becomes satisfied and then move to the next constraint. Given c and λ, algorithm Dual-Greedy, i.e. Algorithm 2, formalizes this approach whose analysis appears in Appendix F.

Algorithm 2 Dual-Greedy:

Input: $c \in \mathbb{Z}^d$, $\lambda \in \mathbb{R}$
Output: an optimal solution w to $\mathcal{D}$ such that $w_0 = \lambda$

```
1: $w \leftarrow \mathbb{0}$;    $w_0 \leftarrow \lambda$
2: $sum_\Delta \leftarrow 0$
3: for $i := 1 \dots d$ do
4:     if $i - \Delta \ge 1$ then $sum_\Delta \leftarrow sum_\Delta - w_{i-\Delta}$
5:     $diff \leftarrow c_i - (w_0 + sum_\Delta)$
6:     if $diff > 0$ then $w_i \leftarrow diff$
7:     $sum_\Delta \leftarrow sum_\Delta + w_i$
8: return $w$
```

6.2 Tracking the change of $\mathcal{D}_\lambda$

Next we introduce the key concept that we need for relating the solution of dual to the sparsity of Lagrangian relaxation of the primal: the notion of *active constraints*. Let w be a vector obtained by Dual-Greedy(c, λ) and let w' be a vector obtained by Dual-Greedy$(c, \lambda - \varepsilon)$, for some fixed $\lambda \in \mathbb{Z}$ and small $\varepsilon \in (0, 1)$. Then, the set of coordinates that are for ε larger in w' than in w are called active constraints. Figure 2 provides an illustration of this concept. Intuitively, the active constraints correspond to those variables of $\mathcal{D}$ that increase when w_0 decreases by some small value. Hence, one can interpret active constraints as gradients of $\mathcal{D}_\lambda$ with respect to the variable λ. This concept appears to be very useful in characterizing the optimal solution of $\mathcal{D}$ in an alternative way. In particular, the following lemma holds.

Lemma 6 *Let $c \in \mathbb{Z}^d$, $\lambda \in \mathbb{Z}$, and $w \leftarrow$ Dual-Greedy(c, λ). Then, if w has exactly k active constraints the vector w is an optimal solution to $\mathcal{D}$.*

A full proof of a statement stronger than Lemma 6 along with its proof appears in Lemma 13, Appendix H, while in this section we provide a proof sketch. Let $w(\varepsilon) =$ Dual-Greedy$(c, \lambda - \varepsilon)$, for some small $\varepsilon \in (0, 1)$. By the definition of active constraints and the fact that w has k many, there are exactly k coordinates that are larger by ε in $w(\varepsilon)$ than in w. In addition, $w(\varepsilon)_0 = \lambda - \varepsilon$ and $w_0 = \lambda$. It is not hard to show that all the other coordinates of $w(\varepsilon)$ and w are the same, which we can express as $\sum_{i=1}^d w(\varepsilon)_i = k\varepsilon + \sum_{i=1}^d w_i$. Now, recall that the objective function of dual $\mathcal{D}$ with respect to vector w equals $w_0 k + \sum_{i=1}^d w_i$. Then we have

$$w_0 k + \sum_{i=1}^d w_i = (w_0 - \varepsilon)k + k\varepsilon + \sum_{i=1}^d w_i = w(\varepsilon)_0 k + \sum_{i=1}^d w(\varepsilon)_i.$$

Hence, the objective values of dual $\mathcal{D}$ for vectors $w(\varepsilon)$ and w are equal, for **all** the values $\varepsilon \in (0, 1)$. As $\mathcal{D}$ is convex in the value of variable w_0 and Dual-Greedy(c, λ) provides an optimal solution to $\mathcal{D}$ such that $w_0 = \lambda$, then w is an optimal solution to $\mathcal{D}$.

6.3 Wrapping up – perturbation and optimal sparsity

Now we use Lemma 6 to prove the following, which essentially justifies line 5 of LASSP.

Lemma 7 *Let $c \in \mathbb{Z}^d$, $\lambda \in \mathbb{Z}$, and $\tilde{w} \leftarrow$ Dual-Greedy(c, λ). Assume that $\tilde{w}$ has exactly k active constraints. Then, any optimal support $S^\star$ of* ProjLagr$(\lambda - \varepsilon, c)$*, for $0 < \varepsilon < 1/d$, has cardinality exactly k.*

So, if we produce λ as in Lemma 7, we will solve problem (3). These steps are implemented by lines 4-6 of LASSP. However, as illustrated in the beginning of the section, λ as in Lemma 7 might not exist. Intuitively, this

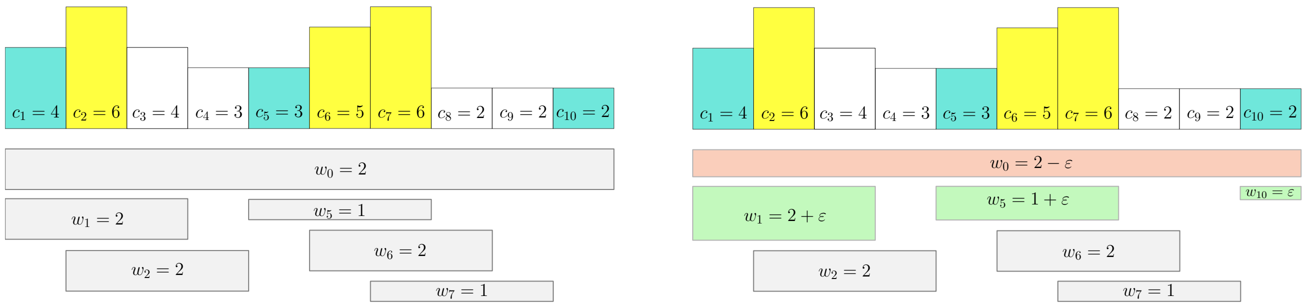

Figure 2: A sketch of an instance of $\mathcal{D}$: $c = (4, 6, 4, 3, 3, 5, 6, 2, 2, 2)$ and $\Delta = 3$. The left and right figure depicts w obtained by Dual-Greedy$(c, 2)$ and Dual-Greedy$(c, 2-\varepsilon)$ for $0 < \varepsilon < 1$, respectively. Constraints 1, 5, and 10 are active, i.e., if w_0 on the left is decreased by ε then only w_1, w_5, and w_{10} increase by ε, as shown on the right. Every w_i, for $i \geq 1$, covers Δ c-poles to its right.

situation happens when the number of active constraints jumps from a value smaller than k to a value larger than k for a very small change of λ. In such a case, we are unable to obtain λ as in Lemma 7. A key component of our analysis is showing that there is an efficient way of randomly altering c, and obtaining $\tilde{c}$, so that with high probability $\tilde{c}$ is such that: $\hat{\lambda}$ is obtained as at line 4; and, $\tilde{w} \leftarrow$ Dual-Greedy$(\tilde{c}, \hat{\lambda})$ has the same property as in Lemma 7. Lines 2 and 3 of LASSP implement this random perturbation. Intuitively, the perturbation achieved by X variables adds noise to our input instance so that the number of active constraints changes by at most one as λ slides over the integer domain. Following this intuition we obtain a proof of Lemma 3. However, due to the space limitation, we present its proof in Appendix I.

We conclude the section by giving a proof of Lemma 7.

Proof of Lemma 7. Recall that by our assumption there are exactly k active constraints defined by $\tilde{w}$. Then from Lemma 6 it follows that $\tilde{w}$ is a minimizer of $\mathcal{D}$, i.e. $\mathrm{VAL}\,\mathcal{D}_\lambda = \mathrm{VAL}\,\mathcal{D}$. By the integrality of $\mathcal{D}$ we have that $\mathrm{VAL}\,\mathcal{D} \in \mathbb{Z}$. Let $\lambda' = \lambda - \varepsilon$, for some $0 < \varepsilon < 1/d$. Then, it holds $\mathrm{VAL}\,\mathcal{D}_{\lambda'} = \mathrm{VAL}\,\mathcal{D}$ as only the k variables correponding to active constraints increased by ε while $w_0 = \lambda'$ decreased by ε. From the strong duality we also have $\mathrm{VAL}\,\mathcal{P}_{\text{no-}k}(\lambda') = \mathrm{VAL}\,\mathcal{D}_{\lambda'} = \mathrm{VAL}\,\mathcal{D}$.

Let $\tilde{u}$ be an integral optimal solution of $\mathcal{P}_{\text{no-}k}(\lambda')$. Following the definition we have

$$\mathrm{VAL}\,\mathcal{P}_{\text{no-}k}(\lambda') = (c^T - \lambda \mathbb{1}^T)\tilde{u} + \lambda k + \varepsilon(\mathbb{1}^T \tilde{u} - k).$$

Now we have the following properties: $\tilde{u}$ is integral; $0 < \varepsilon|\mathbb{1}^T\tilde{u} - k| < 1$ whenever $1 \leq |\mathbb{1}^T\tilde{u} - k| \leq d$; $c \in \mathbb{Z}^d$; $v \in \mathbb{Z}$; and $\mathrm{VAL}\,\mathcal{P}_{\text{no-}k}(\lambda') \in \mathbb{Z}$. Therefore, we have $\mathbb{1}^T\tilde{u} - k = 0$, and hence $\mathbb{1}^T\tilde{u} = k$. Since we showed the equivalence between $\mathcal{P}_{\text{no-}k}(\lambda')$ and $\textsc{ProjLagr}(\lambda', c)$ the lemma follows. □

7 Experiments

We empirically validate the claims outlined in the previous sections. To that end, we compare LASSP with the $O(dk)$-time dynamic program (DP) described in Section C as a baseline. Note that this DP already has a better time complexity than the best previously published algorithm from [14]. Both algorithms are implemented in the Julia programming language (version 0.5.0), which typically achieves performance close to C/C++ for combinatorial algorithms.

7.1 Synthetic data

We perform experiments with synthetic data in order to investigate how the algorithms scale as a function of the input size. We study two different setups: (i) the running time of the projection algorithms on their own, and (ii) the overall running time of a sparse recovery algorithm using the projection algorithms as a subroutine. For the latter, we use the structure-aware variant of the popular CoSaMP algorithm [19, 3].

Figures 3(a)-(b). For a problem of size d, we set the sparsity to $k = d/50$ and generate a random separated sparse vector with parameter $\Delta = (d - 5(k+1))/k - 1$. The non-zero coefficients are i.i.d. ± 1. For the projection-only benchmark, we add Gaussian noise with $\sigma = 1/10$ to all coordinates in order to make the problem non-trivial. For each problem size, we run 10 independent trials and report their mean.

Figure 3(a) shows the speed-up obtained by our nearly-linear time projection relative to the DP baseline. We observe that LASSP is up to 150× faster. This confirms our expectation that LASSP scales gracefully with the problem size, while the DP essentially becomes a quadratic-time algorithm.

Figure 3(b) compares the running times of CoSaMP with three different projection operators. The first variant makes no structural assumptions and uses hard thresholding as projection operator. The other two variants use a projection for the separated sparsity model, relying on the DP baseline and LASSP, respectively. The results show that the version of CoSaMP using LASSP instead of the DP is significantly faster. Moreover, CoSaMP with a simple sparse projection has similar running time to CoSaMP with our structured projection. Finally, we note that CoSaMP with separated

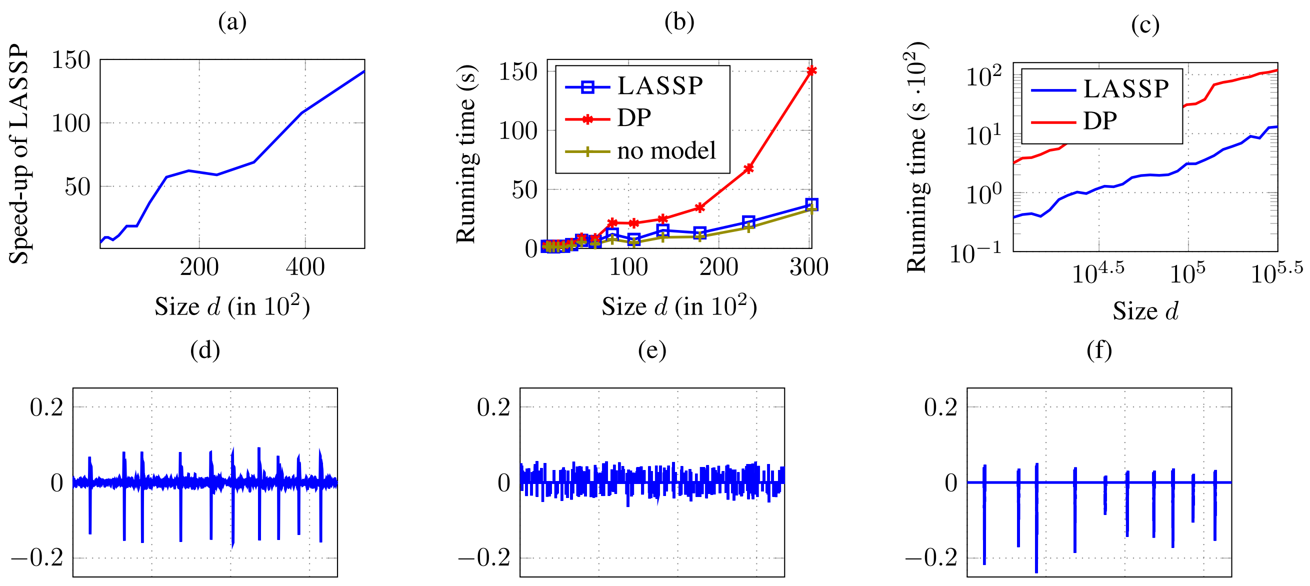

Figure 3: Separated sparsity experiments. In the top row we plot running times of our algorithm LASSP relative to the previous work. Plots (a) and (c) are obtained by projecting signals on the separated sparsity model. Plot (b) compares the running times of CoSaMP with three different projection operators. The variant "no-model" makes no structural assumptions and uses hard thresholding as projection operator. In the bottom row, in (e) we plot the recovery of the signal (d) obtained by the algorithm "no model". Plot (f) shows the recovery of LASSP. Both of the procedures use the same number of measurements.

sparsity requires $1.5\times$ fewer measurements to achieve the same recovery quality as CoSaMP with standard sparsity.

Figure 3(c). We fix the sparsity $k = 100$ and vary the length d of the signal. The signal is obtained in the same way as for plots Figure 3(a)-(b). We observe that our algorithm runs 10x faster than the baseline. Furthermore, the plot shows that the running times of both the baseline DP and LASSP scale linearly with the signal length d. This behavior is expected for the DP. On the other hand, our theoretical findings predict that the running time of LASSP scales as $d \log d$. This suggests that the empirical performance of our algorithm is even (slightly) better than what our proofs state.
In Appendix A we report results of additional experiments.

7.2 Neuronal signals

We also test LASSP on neuron spike train data from [12]. See Figure 3(d) for this input data. First, we run CoSaMP with a “standard sparsity” projection. The recovered signal is depicted in Figure 3(e). Next, we run the convolutional sparsity CoSaMP of [12] and use our fast projection algorithm. The recovered signal is given in Figure 3(f). For the both experiments we use $n = 250$ measurements.

We also run the convolutional sparsity CoSaMP on neuron spike train data of length 10^5, comparing the running time of LASSP and DP as projection operators. CoSaMP with LASSP runs $2\times$ faster in this context. We do not compare the running time relative to CoSaMP with a standard sparsity projection as it requires $10\times$ more measurements to achieve accurate recovery.

8 Conclusions

We have designed a nearly-linear time algorithm for projecting onto the set of separated sparse vectors. The core technique in our algorithm is Lagrangian relaxation. One of the key insights here is that even though there are separated sparsity instances for which the Lagrangian relaxation does not provide an optimal solution, it is still possible to obtain an optimal solution if the original input instance is only slightly perturbed.Furthermore, this perturbation does not change the final output, but rather drives the algorithm to choose an optimal solution of interest even after the hard sparsity constraint is relaxed. Our experiments show that our algorithm is not only of theoretical significance, but also outperforms the state of the art in practice. Exploring the power of perturbed Lagrangian relaxations for other non-convex constraint sets is an important direction for future work. We believe that our framework will enable simple and efficient algorithms for other problems as well.

Acknowledgments.

We thank Arturs Backurs for insightful discussions. L. Schmidt thanks Chinmay Hegde for providing a dataset for some of our experiments. A. Mądry was supported in part by an Alfred. P. Sloan Research Fellowship, Google Research Award and the NSF grant CCF-1553428. S. Mitrović was supported by Swiss NSF (grant number P1ELP2_161820). Part of this work was carried out while S. Mitrović was visiting MIT.

References

[1] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2006.

[2] Y. Eldar and M. Mishali, "Robust recovery of signals from a structured union of subspaces," *IEEE Transactions on Information Theory*, vol. 55, no. 11, pp. 5302–5316, 2009.

[3] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde, "Model-based compressive sensing," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1982–2001, 2010.

[4] J. Mairal, R. Jenatton, G. Obozinski, and F. Bach, "Convex and network flow optimization for structured sparsity," *The Journal of Machine Learning Research*, vol. 12, pp. 2681–2720, 2011.

[5] J. Huang, T. Zhang, and D. Metaxas, "Learning with structured sparsity," *The Journal of Machine Learning Research*, vol. 12, pp. 3371–3412, 2011.

[6] N. S. Rao, B. Recht, and R. D. Nowak, "Universal measurement bounds for structured sparse signal recovery." in *AISTATS*, ser. JMLR Proceedings, vol. 22, 2012, pp. 942–950.

[7] S. N. Negahban, P. Ravikumar, M. J. Wainwright, and B. Yu, "A unified framework for high-dimensional analysis of M-estimators with decomposable regularizers," *Statistical Science*, vol. 27, no. 4, pp. 538–557, 11 2012.

[8] B. Bah, L. Baldassarre, and V. Cevher, "Model-based sketching and recovery with expanders," in *SODA*, 2014, pp. 1529–1543.

[9] C. Hegde, P. Indyk, and L. Schmidt, "A nearly-linear time framework for graph-structured sparsity," in *ICML*. JMLR Workshop and Conference Proceedings, 2015, pp. 928–937.

[10] C. Hegde, M. F. Duarte, and V. Cevher, "Compressive sensing recovery of spike trains using a structured sparsity model," in *SPARS'09-Signal Processing with Adaptive Sparse Structured Representations*, 2009.

[11] E. L. Dyer, M. F. Duarte, D. H. Johnson, and R. G. Baraniuk, "Recovering spikes from noisy neuronal calcium signals via structured sparse approximation," in *LVA/ICA*, 2010, pp. 604–611.

[12] C. Hegde and R. G. Baraniuk, "Sampling and recovery of pulse streams," *IEEE Transactions on Signal Processing*, vol. 59, no. 4, pp. 1505–1517, 2011.

[13] E. L. Dyer, C. Studer, J. T. Robinson, and R. G. Baraniuk, "A robust and efficient method to recover neural events from noisy and corrupted data," in *6th International IEEE/EMBS Conference on Neural Engineering (NER)*, 2013, pp. 593–596.

[14] S. Foucart, M. F. Minner, and T. Needham, "Sparse disjointed recovery from noninflating measurements," *Applied and Computational Harmonic Analysis*, vol. 39, no. 3, pp. 558 – 567, 2015.

[15] Q. Huang and S. M. Kakade, "Super-resolution off the grid," in *Conference on Neural Information Processing Systems (NIPS)*, 2015, pp. 2665–2673.

[16] M. F. Duarte and R. G. Baraniuk, "Spectral compressive sensing," *Applied and Computational Harmonic Analysis*, vol. 35, no. 1, pp. 111 – 129, 2013.

[17] T. Needham, "Dictionary-sparse and disjointed recovery," in *International Conference on Sampling Theory and Applications (SampTA)*, 2015, pp. 278–282.

[18] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. Wiley-Interscience, 1988.

[19] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301–321, 2009.

[20] C. Hegde, P. Indyk, and L. Schmidt, "Approximation algorithms for model-based compressive sensing," *IEEE Transactions on Information Theory*, vol. 61, no. 9, 2015.

[21] ——, "Nearly linear-time model-based compressive sensing," in *ICALP*, 2014, vol. 8572.

[22] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto, "Optimal packing and covering in the plane are np-complete," *Information processing letters*, vol. 12, no. 3, pp. 133–137, 1981.

[23] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media, 2002, vol. 24.