Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

SADEGH JOKAR   MARC E. PFETSCH

Exact and Approximate Sparse Solutions of Underdetermined Linear Equations

Supported by the DFG Research Center MATHEON "Mathematics for key technologies" in Berlin.

ZIB-Report 07-05 (March 2007)

EXACT AND APPROXIMATE SPARSE SOLUTIONS OF UNDERDETERMINED LINEAR EQUATIONS

SADEGH JOKAR AND MARC E. PFETSCH

Abstract. In this paper, we empirically investigate the NP-hard problem of finding sparsest solutions to linear equation systems, i.e., solutions with as few nonzeros as possible. This problem has received considerable interest in the sparse approximation and signal processing literature, recently. We use a branch-and-cut approach via the maximum feasible subsystem problem to compute optimal solutions for small instances and investigate the uniqueness of the optimal solutions. We furthermore discuss five (modifications of) heuristics for this problem that appear in different parts of the literature. For small instances, the exact optimal solutions allow us to evaluate the quality of the heuristics, while for larger instances we compare their relative performance. One outcome is that the so-called basis pursuit heuristic performs worse, compared to the other methods. Among the best heuristics are a method due to Mangasarian and a bilinear approach.

1. Introduction

The area of sparse approximations has seen tremendous research progress in the recent years. At the core of many of these activities are results that ensure the efficient recovery of sparse solutions, i.e., solutions having few nonzeros, to linear systems $A\boldsymbol{x} = \boldsymbol{b}$ with $A \in \mathbb{R}^{m \times n}$, $\boldsymbol{b} \in \mathbb{R}^m$. The basic idea is to let A be a composition of several bases of $\mathbb{R}^m$ and hence $n > m$. This implies that $\boldsymbol{b}$ has several representations, often guaranteeing sparse solutions to the equation system: Under appropriate assumptions on A and if $\boldsymbol{b}$ is known to have a sufficiently sparse representation, Donoho and Huo [21] showed that the sparsest representation is unique. Moreover, a striking result states that, under slightly stronger assumptions, one can efficiently find this representation by solving a linear program (LP), which models the minimization of the ℓ_1-norm of $\boldsymbol{x}$ subject to $A\boldsymbol{x} = \boldsymbol{b}$. This fact was first investigated empirically by Chen, Donoho, and Saunders [13] and then theoretically studied by Donoho and Huo [21]. This approach is commonly called *basis pursuit*. Many strengthenings of this type of result have been obtained, see Section 1.3 for more details.

A generalization to the sparse representation problem is to allow $A\boldsymbol{x}$ deviate from $\boldsymbol{b}$ with respect to some ℓ_p-norm, i.e., to look for the sparsest solution $\boldsymbol{x}$ satisfying $\|A\boldsymbol{x} - \boldsymbol{b}\|_p \leq \delta$ for given $\delta > 0$, where $p \in [1, \infty]$. The literature investigates conditions on A and $\boldsymbol{b}$ such that a solution minimizing the ℓ_1-norm under such constraints is near a sparsest solution. Donoho,

---

Date: March 2007.

Supported by the DFG Research Center Matheon "Mathematics for key technologies" in Berlin.

Elad, and Temlyakov [20], Tropp [38], Donoho [16], Fuchs [25], and Candès, Romberg, and Tao [9] deal with the case $p = 2$. Donoho and Elad [19] and Fuchs [26] investigate $p = \infty$.

Another type of algorithm has its origin in the so-called *matching pursuit* approach, see Mallat and Zhang [28]. Similar results to the above mentioned can be proved for *orthogonal matching pursuit*, a variant of matching pursuit, developed by Pati, Rezaiifar, and Krishnaprasad [31], and Davis, Mallat, and Zhang [14]. For the results see Tropp [37], Donoho, Elad, and Temlyakov [20], and Tropp and Gilbert [36].

The sparse representation problem is closely connected to the maximum feasible subsystem problem (Max FS), in which one searches for a largest feasible subsystem of an infeasible linear inequality system; see Section 2 for a more detailed description and literature. In this context several heuristics have been proposed, e.g., a method using a bilinear formulation by Bennett and Bredensteiner [6] and the work of Mangasarian [29].

Unfortunately, the combinatorial optimization problem of finding a sparsest solution to an equation system is computationally hard, i.e., NP-hard (the same holds for Max FS). The above mentioned results investigate conditions under which sparse solutions can nevertheless be found efficiently and the analyses have become increasingly more exact. In practical applications, however, one needs sparse solutions also when the conditions investigated in the literature do not hold. One approach to this problem is to develop an exact combinatorial optimization method that can compute an optimal sparsest solution for practical problem sizes. The theoretical hardness results do not exclude such methods, since computational complexity captures worst-case and asymptotical behavior only. It seems, however, that the literature on sparse representations generally believes such an approach to be impossible. A second approach is to settle for algorithms that efficiently compute good approximate solutions to the above problem, i.e., they find solutions that have not much more nonzeros than the optimal solutions.

In this paper, we empirically investigate an exact branch-and-cut algorithm that was originally developed for Max FS. We furthermore investigate the practical behavior of five approximate methods: Besides the above mentioned basis pursuit and orthogonal matching pursuit, we propose a variant of orthogonal matching pursuit that improves upon its results, but is computationally more expensive. We also use the nonlinear optimization method due to Mangasarian, mentioned above. Finally, we adapt the bilinear optimization approach of Bennett and Bredensteiner to the case of finding sparse solutions. We consider sparse solutions satisfying the given equation system $A\boldsymbol{x} = \boldsymbol{b}$ and also solutions that satisfy $\|A\boldsymbol{x} - \boldsymbol{b}\|_\infty \leq \delta$ for a given $\delta > 0$.

The availability of the exact approach enables us to estimate the quality of heuristics via the optimum or the lower bound that is produced by the branch-and-cut approach. The exact algorithm can also be used to check whether the sparsest solution is unique, a key feature of the investigated problem.

The findings of this paper can be summarized as follows. We confirm the common pessimism about the optimal solvability of the exact sparse representation problem. It is – both theoretically and empirically – a very hard problem: the exact branch-and-cut algorithm can only solve very small instances to optimality, or instances in which the optimal solution has few nonzeros. On the positive side, the heuristics perform quite well, in particular, Mangasarian's approach. It is also a good idea to greedily decrease the support of a solution produced by a heuristic, if possible. Basis pursuit turns out to be the worst of the studied heuristics. If one wants to improve upon basis pursuit or orthogonal matching pursuit, however, one has to pay the price of solving several LPs.

The structure of this article is as follows. After fixing some notation and introducing the basic problems of this paper, in Section 1.3 we review results from the literature. This includes results on the computational complexity of the problems and examples of the representation theorems that can be proved in our context. In Section 2, we discuss our exact approach via the maximum feasible subsystem problem. Section 3 contains the introduction of the five heuristics mentioned above. In Section 4, we present computational experiments with the exact and approximate algorithms. We close with some final remarks.

1.1. Notation

We depict vectors in bold font, and for two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, we write $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y}$ for their inner product. We denote by $\mathbb{1}$ the vector of all ones and by $\mathbf{0}$ the zero vector of appropriate dimensions. For a positive integer n, we define $[n] := \{1, \dots, n\}$. For $\boldsymbol{x} \in \mathbb{R}^n$, we denote by $\operatorname{supp}(\boldsymbol{x}) = \{j \in [n] \,:\, x_j \neq 0\}$ the support of $\boldsymbol{x}$ and define the ℓ_0 quasi-norm to be the size of the support, i.e., $\|\boldsymbol{x}\|_0 := \#\{\operatorname{supp}(\boldsymbol{x})\}$. We will also use the ℓ_1, Euclidean ℓ_2, and max ℓ_∞-norm:

$$\|\boldsymbol{x}\|_1 = \sum_{j=1}^{n} |x_j|, \qquad \|\boldsymbol{x}\|_2 = \Big(\sum_{j=1}^{n} x_j^2\Big)^{\frac{1}{2}}, \quad \text{and} \quad \|\boldsymbol{x}\|_\infty = \max_{j\in[n]} |x_j|,$$

respectively.

1.2. Basic Problems

The basic problem of this paper is to find the sparsest solution to $A\boldsymbol{x} = \boldsymbol{b}$, where $A \in \mathbb{R}^{m\times n}$ is of full rank and $\boldsymbol{b} \in \mathbb{R}^m$. We therefore want to solve:

$$(P_0) \qquad \min\{\|\boldsymbol{x}\|_0 \,:\, A\boldsymbol{x} = \boldsymbol{b}\}.$$

We assume that $m < n$, and hence the solutions of $A\boldsymbol{x} = \boldsymbol{b}$ are underdetermined. We also consider the variant of this problem, where we allow deviations from equality, i.e., we allow solutions $\boldsymbol{x}$ that satisfy $\|A\boldsymbol{x} - \boldsymbol{b}\|_\infty \leq \delta$ for some $\delta \geq 0$. This problem can be posed as follows.

$$(P_0^\delta) \qquad \min\{\|\boldsymbol{x}\|_0 \,:\, \boldsymbol{b} - \delta\cdot\mathbb{1} \leq A\boldsymbol{x} \leq \boldsymbol{b} + \delta\cdot\mathbb{1}\}.$$

Clearly, problem (P_0) is equivalent to (P_0^0), but we will often distinguish the two, since specialized algorithms for (P_0) might be more efficient. For convenience, we define the sets of feasible points

$$Q := \{\boldsymbol{x} \in \mathbb{R}^n \,:\, A\boldsymbol{x} = \boldsymbol{b}\} \qquad \text{and}$$
$$Q^\delta := \{\boldsymbol{x} \in \mathbb{R}^n \,:\, \boldsymbol{b} - \delta\cdot\mathbb{1} \leq A\boldsymbol{x} \leq \boldsymbol{b} + \delta\cdot\mathbb{1}\}.$$

We will always assume that $Q^\delta \neq \varnothing$ (for $\delta = 0$ this is implied by the full rank of A). The above problems only become interesting if $\mathbf{0} \notin Q^\delta$. For $Q = Q^0$ this is equivalent to the assumption that $\boldsymbol{b} \neq \mathbf{0}$. For Q^δ we need $\delta < \max\{|b_1|, \dots, |b_m|\} = \|\boldsymbol{b}\|_\infty$.

Remark. We take the ℓ_∞-norm as a measure for deviations of equality in this paper, since it is the most suitable for linear programming based approaches. With some effort one could include deviations w.r.t. the ℓ_1-norm, but using the ℓ_2-norm would turn (P_0^δ) into a nonlinear mixed-integer problem, which would make our exact solution approach even more difficult. The ℓ_2-norm is usually used in the literature, see, for instance, Donoho, Elad, and Temlyakov [20], and Tropp [38].

Replacing the quasi-norm $\|\cdot\|_0$ by $\|\cdot\|_1$ in (P_0) and (P_0^δ) yields the following two problems.

$$(P_1) \qquad \min\{\|\boldsymbol{x}\|_1 \,:\, \boldsymbol{x} \in Q\}$$
$$(P_1^\delta) \qquad \min\{\|\boldsymbol{x}\|_1 \,:\, \boldsymbol{x} \in Q^\delta\}.$$

These problems are efficiently solvable by linear programming methods, see Section 3.1. The problems (P_1) and (P_1^δ) will be used several times later on.

1.3. Results from the Literature

In this section, we collect results from the literature concerning sparse solutions of $A\boldsymbol{x} = \boldsymbol{b}$. We first deal with the computational complexity properties of the problem. These seem hardly to be known in the sparse representation literature.

The decision version of (P_0) is one of the classical NP-complete problems, see Garey and Johnson [27], i.e., if P $\neq$ NP there exists no polynomial time algorithm that for every instance computes the optimal solution of (P_0). Since (P_0^δ) contains (P_0) as a special case, (P_0^δ) is NP-hard as well; Natarajan [30] presents an NP-hardness proof for $\delta > 0$.

Moreover, both problems are hard to approximate, i.e., if a very likely assumption is true (see below), there is no polynomial time algorithm that computes a solution with a support that is at most $2^{\log^{1-\rho} n}$ times as large as the support of an optimal solution, for any $\rho > 0$, see Amaldi and Kann [3] for (P_0) and Amaldi [2] for (P_0^δ).

The precise formulation is as follows. Let DTIME($s^{\operatorname{polylog} s}$) be the class of problems whose instances of size s can be solved in deterministic time $O(s^{\operatorname{polylog} s})$, where polylog s refers to any polynomial in $\log s$. The assumption NP $\not\subseteq$ DTIME($s^{\operatorname{polylog} s}$) is stronger than P $\neq$ NP, but it is also believed to be extremely likely; it amounts to the claim that not all problems in NP can be solved in quasi-polynomial time. Then, assuming NP $\not\subseteq$ DTIME($s^{\operatorname{polylog} s}$), (P_0) and (P_0^δ) cannot be approximated in polynomial time within a factor $2^{\log^{1-\rho} n}$, for any $\rho > 0$ (n is the number of columns of A). This shows that (P_0) and (P_0^δ) are indeed theoretically hard optimization problems.

Having the above negative results in mind, it is quite surprising that strong statements about efficient solutions to the sparse representation problem exist. We give a flavor of these results, for which we assume that $A = [B^1\ B^2]$, where $B^1 = [\boldsymbol{p}_1, \dots, \boldsymbol{p}_m]$ and $B^2 = [\boldsymbol{q}_1, \dots, \boldsymbol{q}_m]$ are two orthonormal bases of $\mathbb{R}^m$ (and hence $A \in \mathbb{R}^{m \times 2m}$). The following number, called *mutual incoherence*, is central for the results:

$$M := M(B^1, B^2) := \max\{|\boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{q}_j| \,:\, 1 \le i, j \le m\}.$$

Note that $M \le 1$, because B^1 and B^2 are normalized and $M \ge 1/\sqrt{m}$ (see, e.g., Donoho and Huo [21]). It turns out that the smaller the mutual incoherence, the nicer the properties that the matrix has with respect to sparse representation problems. Matrices with a minimum mutual incoherence are also called Grassmannian frames, see Strohmer and Heath [35].

Theorem 1 (Elad and Bruckstein [24])**.** *Let A be as above, $\boldsymbol{b} \in \mathbb{R}^m$, and let $\boldsymbol{x}^\star$ be an optimal solution of (P_0) with $\alpha := \|\boldsymbol{x}^\star\|_0 < \frac{1}{M}$. Then $\boldsymbol{x}^\star$ is the unique solution of $A\boldsymbol{x} = \boldsymbol{b}$ with $\|\boldsymbol{x}\|_0 \le \alpha$.*

If we slightly decrease the bound on the number of nonzeros, we can recover the unique optimal solution by solving (P_1).

Theorem 2 (Elad and Bruckstein [24])**.** *Let A be as above, $\boldsymbol{b} \in \mathbb{R}^m$, and let $\boldsymbol{x}^\star$ be the (unique) optimal solution of (P_0) with $\|\boldsymbol{x}^\star\|_0 < (\sqrt{2} - \frac{1}{2})/M$. Then the optimal solution of (P_1) coincides with $\boldsymbol{x}^\star$.*

This bound is an improvement of the bound $\frac{1}{2}(1+\frac{1}{M})$ obtained by Donoho and Huo [21]. The bound $(\sqrt{2} - \frac{1}{2})/M \approx 0.9142/M$ is only slightly smaller than the bound $\frac{1}{M}$ of Theorem 2.

The statement of Theorem 2 can be strengthened for random matrices. For instance, we have the following.

Theorem 3 (Donoho [17])**.** *Let $A \in \mathbb{R}^{m \times n}$, with $m < n$, be a matrix whose columns are unit vectors taken uniformly at random. Then there exists a constant $\rho(n)$ such that the following holds: Let $\boldsymbol{x}^\star$ be the optimal solution of (P_0) for the system $A\boldsymbol{x} = \boldsymbol{b}$, and assume that $\|\boldsymbol{x}^\star\|_0 < \rho(n) \cdot n$. Then the optimal solution of (P_1) coincides with $\boldsymbol{x}^\star$.*

Hence, this results shows that for random matrices one can find optimal solutions $\boldsymbol{x}^\star$ of (P_0) for appropriate $\alpha \in O(n)$ if $\|\boldsymbol{x}^\star\|_0 < \alpha$ by linear programming. In contrast, Theorem 2 only works for $\alpha = 1/M \in O(\sqrt{n})$. Thus, asymptotically Theorem 3 is stronger than Theorem 2. However, known bounds on $\rho(n)$ are very small, see also Candès and Tao [11].

For further extensions and strengthenings of these result see, for instance, Donoho and Elad [18], Candès and Romberg [7], Candès, Romberg, and Tao [8], and Donoho [15].

Remark. A complementary problem to (P_0) is as follows. Let $B \in \mathbb{R}^{r \times s}$ with $r > s$. Suppose that $\boldsymbol{b} := B\boldsymbol{x}' + \boldsymbol{e}$, where $\boldsymbol{x}'$ is some given vector and $\boldsymbol{e}$ is a vector of errors. The goal is to recover $\boldsymbol{x}'$, given B and $\boldsymbol{b}$. It turns out that, if B has full rank, this problem is equivalent to (P_0) and, similarly to (P_0), it can be solved by linear programming under certain assumptions on B and if $\boldsymbol{e}$ is sparse. For more information see Candès et al. [10], and Candès and Tao [11].

As mentioned in the introduction, there are several articles that deal with the problem $\min\{\|\boldsymbol{x}\|_0 \,:\, \|A\boldsymbol{x} - \boldsymbol{b}\|_2 \le \delta\}$. For the case with the ℓ_∞-norm instead of the ℓ_2-norm (i.e., (P_0^δ)) there are much fewer results. We cite the following.

Theorem 4 (Elad and Donoho [19]). *Let $A = [B_1\ B_2]$ be as above. For $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$ and $\boldsymbol{e} \in \mathbb{R}^m$, with $\|\boldsymbol{e}\|_\infty \leq \varepsilon$, define $\boldsymbol{b} := A\tilde{\boldsymbol{x}} + \boldsymbol{e}$. If for some $\gamma \geq 0$ and $\delta \geq \varepsilon$, we have*

$$\|\tilde{\boldsymbol{x}}\|_0 < \frac{1+M}{2M + 2\sqrt{m}(\varepsilon+\delta)/\gamma},$$

then the solution $\boldsymbol{x}^\star$ to (P_1^δ) satisfies $\|\boldsymbol{x}^\star - \tilde{\boldsymbol{x}}\|_1 \leq \gamma$.

In other words, if there exists a sparse solution $\tilde{\boldsymbol{x}}$ to the disturbed system, then the optimal solution to (P_1^δ) is not far from $\tilde{\boldsymbol{x}}$. For the ℓ_2-norm case mentioned above, it can also be proved that under appropriate conditions the support of $\boldsymbol{x}^\star$ is contained in the support of $\tilde{\boldsymbol{x}}$.

2. Integer Programming Approaches

The sparse representation problem (P_0^δ) can be formulated as a mixed integer linear program as follows:

$$\begin{aligned} \min\ & \mathbb{1}^{\mathrm{T}}\boldsymbol{z} \\ & \boldsymbol{b} - \delta\cdot\mathbb{1} \leq A\boldsymbol{x} \leq \boldsymbol{b} + \delta\cdot\mathbb{1} \\ & -Z\cdot\boldsymbol{z} \leq \ \boldsymbol{x}\ \leq Z\cdot\boldsymbol{z} \\ & \qquad\qquad \boldsymbol{z} \in \{0,1\}^n. \end{aligned}$$

If $Z > 0$ is large enough, then $z_j = 1$, $j \in [n]$, places no restrictions on x_j, while $z_j = 0$ forces $x_j = 0$. This shows that the above model indeed finds optimal solutions for (P_0^δ).

This approach, however, has usually suffers from serious numerical difficulties. On the one hand, Z has to be large enough to guarantee the correct behavior of the model. On the other hand, Z has to be as small as possible, since large values of Z can lead to wrong results, because of numerical problems. Furthermore, the larger Z is, the worse the LP-bounds in LP-based integer programming solvers; this has negative effects on the performance. While for special data A and $\boldsymbol{b}$, good bounds on $\|\boldsymbol{x}\|_\infty$ – and hence Z – may be possible, a priori bounds are impractically large, see e.g. Schrijver [34].

The problems with the previous approach can be avoided by transformation to the so-called *maximum feasible subsystem problem* (Max FS). Here, one is given an infeasible linear inequality system $D\boldsymbol{x} \leq \boldsymbol{d}$ and wants to find a feasible subsystem of maximal cardinality. See Amaldi, Pfetsch, and Trotter [4] or Pfetsch [32] for more information.

For the case $\delta = 0$, the transformation of an instance $A\boldsymbol{x} = \boldsymbol{b}$ of (P_0) to an instance of Max FS works as follows, see Amaldi and Kann [3]. Since A has full rank, we can use Gaussian elimination to obtain the equivalent system $\boldsymbol{u} + R\boldsymbol{v} = \boldsymbol{r}$, where $R \in R^{m\times(m-n)}$, $\boldsymbol{r} \in \mathbb{R}^m$, $\boldsymbol{u} \in \mathbb{R}^m$, and $\boldsymbol{v} \in \mathbb{R}^{m-n}$. We now consider the system

$$\boldsymbol{u} + R\boldsymbol{v} = \boldsymbol{r},\ \boldsymbol{u} = \boldsymbol{0},\ \boldsymbol{v} = \boldsymbol{0},$$

which is infeasible (recall that we assumed $\boldsymbol{b} \neq \boldsymbol{0}$ and hence $\boldsymbol{r} \neq \boldsymbol{0}$). Eliminating $\boldsymbol{u}$, we obtain $R\boldsymbol{v} = \boldsymbol{r}$, $\boldsymbol{v} = \boldsymbol{0}$. The Max FS instance is then

$$R\boldsymbol{v} \leq \boldsymbol{r},\ -R\boldsymbol{v} \leq -\boldsymbol{r},\ \boldsymbol{v} \leq \boldsymbol{0},\ -\boldsymbol{v} \leq \boldsymbol{0}. \tag{1}$$

This system has $4m - 2n$ inequalities and $m - n$ variables. Since every point $\boldsymbol{v}$ satisfies at least one of the opposite inequalities, at most one of each pair of inequalities has to be removed in order to obtain a feasible subsystem. Moreover, if k inequalities are removed to yield a feasible system, a solution $\boldsymbol{v}$ to the remaining system yields a solution $(\boldsymbol{u}, \boldsymbol{v})$ of $\boldsymbol{u} + R\boldsymbol{v} = \boldsymbol{r}$ having at most k nonzeros, where $\boldsymbol{u} = \boldsymbol{r} - R\boldsymbol{v}$. Conversely, a solution to $\boldsymbol{u}+R\boldsymbol{v} = \boldsymbol{r}$ with k nonzeros yields a feasible subsystem of (1) with k removed inequalities.

If $\delta > 0$, it seems that there exists no easy reduction to Max FS. We can, however, proceed as follows. We consider the infeasible system

$$A\boldsymbol{x} \leq \boldsymbol{b} + \delta \cdot \mathbb{1},\ -A\boldsymbol{x} \leq -\boldsymbol{b} + \delta \cdot \mathbb{1},\ \boldsymbol{x} \leq \boldsymbol{0},\ -\boldsymbol{x} \leq \boldsymbol{0},$$

where we declare the first $2m$ inequalities as mandatory and hence only the last $2n$ nonnegativity inequalities can be removed to obtain a feasible system. Similar to above, a feasible subsystem including the first $2m$ constraints and skipping k of the last $2n$ inequalities yields a solution $\boldsymbol{x} \in Q^{\delta}$ with at most k nonzeros. Conversely, as solution $\boldsymbol{x} \in Q^{\delta}$ with k nonzeros yields a feasible subsystem containing the first $2m$ constraints and skipping k of the last $2n$ inequalities.

Computing optimal solutions to Max FS can be done by a branch-and-cut algorithm. The handling of mandatory inequalities is straightforward. We refer to Pfetsch [33] for an extensive discussion of this approach.

In principle, a direct branch-and-cut algorithm for the sparse representation problem could avoid the doubling of variables arising in the transformation to Max FS. However, more research would be necessary to turn this into a practical approach.

3. Heuristic Approaches

In this section we present five heuristic approaches to solve (P_0^{δ}). We discuss basis pursuit, orthogonal matching pursuit, a nonlinear optimization approach due to Mangasarian, and a bilinear formulation.

3.1. Basis Pursuit

The approach to replace the ℓ_0 quasi-norm in (P_0) and (P_0^{δ}) by the more tractable ℓ_1-norm to obtain problems (P_1) and (P_1^{δ}) is called *basis pursuit* (BP) and was pioneered by Chen, Donoho, and Saunders [13], see also Chen [12].

There are several ways to rewrite these problems as linear programs. We will only present the different formulations for (P_1^{δ}) – the changes for (P_1) are straightforward. A first easy representation is

$$\begin{aligned} \min\ & \mathbb{1}^{\mathrm{T}}\boldsymbol{y} \\ & \boldsymbol{b} - \delta \cdot \mathbb{1} \leq A\boldsymbol{x} \leq \boldsymbol{b} + \delta \cdot \mathbb{1} \\ & \qquad\ \ -\boldsymbol{y} \leq\ \boldsymbol{x}\ \leq \boldsymbol{y}. \end{aligned}$$

Note that here $\boldsymbol{y}$ is automatically nonnegative. This formulation has $2n$ variables and $2(m + n)$ constraints. For many LP-solvers, e.g., simplex algorithm based solvers, the number of (ordinary) constraints is crucial for speed, while bounds on variables are implicitly handled and provide little additional computational cost. Therefore the following formulation, which we use in our implementations, should be more efficient.

$$\begin{aligned} \min\ & \mathbb{1}^{\mathrm{T}}(\boldsymbol{x}^+ + \boldsymbol{x}^-) \\ & A(\boldsymbol{x}^+ - \boldsymbol{x}^-) + \boldsymbol{s} = \boldsymbol{b} \\ & -\delta \cdot \mathbb{1} \leq \boldsymbol{s} \leq \delta \cdot \mathbb{1} \\ & \boldsymbol{x}^+, \boldsymbol{x}^- \geq \mathbf{0}. \end{aligned}$$

This system has $2n+m$ variables, m equations, and $2(n+m)$ bounds on the variables. It arises by splitting $\boldsymbol{x}$ into its positive part $\boldsymbol{x}^+$ and its negative part $\boldsymbol{x}^-$ and introducing slack variables $\boldsymbol{s}$. Note that in the optimum, for every $j \in [n]$, not both x_j^+ and x_j^- are positive; hence, we have the correspondence $x_j^+ - x_j^- = x_j$ and $x_j^+ + x_j^- = |x_j|$. Clearly, in a formulation for (P_0), variables $\boldsymbol{s}$ are not necessary.

Remark. A general problem with all linear programming based approaches is that the matrix A is usually dense in applications. This has negative effects on the performance of LP-solvers, since the speed of solving the equation systems in LP-solvers approaches its worst case cubic behavior, and techniques for handling sparse data do not help here. See Section 4 for more information and, for instance, Yarmish [39] for an implementation of a dense LP-solver.

3.2. Orthogonal Matching Pursuit

The *orthogonal matching pursuit* (OMP) approach to heuristically solve (P_0) was developed by Pati, Rezaiifar, and Krishnaprasad [31] and Davis, Mallat, and Zhang [14]. We will not discuss the earlier proposed matching pursuit introduced by Mallat and Zhang [28].

The idea of OMP to find a sparse solution of $A\boldsymbol{x} = \boldsymbol{b}$ is the following. At the ith iteration, one is given an approximate point $\boldsymbol{x}^i$ with residue $\boldsymbol{r}^i = \boldsymbol{b} - A\boldsymbol{x}^i$. We want to enlarge the support S^i of $\boldsymbol{x}^i$ by a single index and therefore compute

$$j^\star = \operatorname{argmax}\,\{|\boldsymbol{A}_j^{\mathrm{T}}\boldsymbol{r}^i| \,:\, j \in [n] \setminus S^i\},$$

where $\boldsymbol{A}_j$ is the jth column of A. Then one sets $S^{i+1} = S^i \cup \{j^\star\}$ and solves the least squares problem

$$\min_{\boldsymbol{\lambda} \in \mathbb{R}^n} \{\, \big\|\boldsymbol{b} - \sum_{j \in S^{i+1}} \lambda_j\, \boldsymbol{A}_j\big\|_2 \,:\, \lambda_j = 0 \text{ for } j \notin S^{i+1}\}. \tag{2}$$

The solution is taken as the new point $\boldsymbol{x}^{i+1}$ and one sets $\boldsymbol{r}^{i+1} = \boldsymbol{b} - A\boldsymbol{x}^{i+1}$. We stop the process if $\|\boldsymbol{r}^i\|_\infty$ is small enough. The number of iterations gives the number of nonzeros of the computed solution to $A\boldsymbol{x} = \boldsymbol{b}$.

To get a solution for (P_0^δ), we perform OMP as just explained, but stop as soon as $\|\boldsymbol{r}^i\|_\infty \leq \delta$.

Remark. We use the ℓ_∞-norm for the stopping criterion in order get a fair comparison with linear programming based approaches: In standard LP-solvers feasibility is checked w.r.t. the ℓ_∞-norm, using a feasibility tolerance of around $\varepsilon = 10^{-6}$. We use this tolerance in all implementations.

There are examples in which OMP produces solutions arbitrarily far away from the optimal solution of (P_0), see Chen, Donoho, and Saunders [13]. Nevertheless, one can prove interesting results about the quality of OMP solutions. For instance, Tropp and Gilbert [36] showed that if A consists of independent random unit vectors and $\boldsymbol{b}$ has a sparse enough representation, then OMP computes this representation (which is unique) with high probability.

Recently Donoho, Tsaig, Drori, and Starck [22] proposed *stagewise orthogonal matching pursuit*, a variant of OMP in which many coefficients can enter the model at each step.

3.2.1. *Generalizations of OMP.* Orthogonal matching pursuit can be generalized as follows. We replace (2) by the following problem with $S = S^{i+1}$.

$$\begin{aligned} \min\ & \mathbb{1}^{\mathrm{T}}(\boldsymbol{u}+\boldsymbol{w}) \\ & \boldsymbol{b} - \delta \cdot \mathbb{1} - A\boldsymbol{x} \leq \boldsymbol{u} \\ & A\boldsymbol{x} - \boldsymbol{b} - \delta \cdot \mathbb{1} \leq \boldsymbol{w} \\ & \boldsymbol{x}_j = \boldsymbol{0} \qquad j \notin S \\ & \boldsymbol{u},\ \boldsymbol{w} \geq \boldsymbol{0}. \end{aligned} \tag{3}$$

This LP computes a solution closest to Q^δ with respect to the ℓ_1-norm that uses only variables in S (note that in an optimal solution at most one of u_j and w_j is positive). The process is terminated if the objective function value of (3) is small enough.

Similarly, one can replace (2) by

$$\min_{\boldsymbol{\lambda} \in \mathbb{R}^n} \{\ \big\|\boldsymbol{b} - \sum_{j \in S^{i+1}} \lambda_j\, \boldsymbol{A}_j\big\|_\infty\ :\ \lambda_j = 0 \text{ for } j \notin S^{i+1}\}. \tag{4}$$

This leads to an LP that is similar to (3):

$$\begin{aligned} \min\ & z \\ & \boldsymbol{b} - \delta \cdot \mathbb{1} - A\boldsymbol{x} \leq z \cdot \mathbb{1} \\ & A\boldsymbol{x} - \boldsymbol{b} - \delta \cdot \mathbb{1} \leq z \cdot \mathbb{1} \\ & \boldsymbol{x}_j = \boldsymbol{0} \qquad j \notin S, \end{aligned}$$

where $S = S^{i+1}$ again denotes the current support set. The process is terminated when the optimal solution value is small enough.

In computational experiments it turned out that the results of these two methods do not significantly differ from the results produced by OMP. We therefore do not investigate these two variants further.

Another generalization yields a new method that we call *greedy orthogonal matching pursuit* (GOMP). We present it for the case $\delta > 0$ only; the modifications for $\delta = 0$ are straightforward. At the ith iteration, we are given an approximate solution $\boldsymbol{x}^i$ with support S^i and compute the best extension of S^i by solving (3) for each $S = S^i \cup \{j\}$, $j \notin S^i$. We choose $j^\star$ to be an index j that produces the smallest value in (3). The new point $\boldsymbol{x}^{i+1}$ is given by the solution of the corresponding LP. GOMP naturally increases the computational effort with respect to OMP, since it has to solve many LPs in each iteration. See Section 4 for more information.

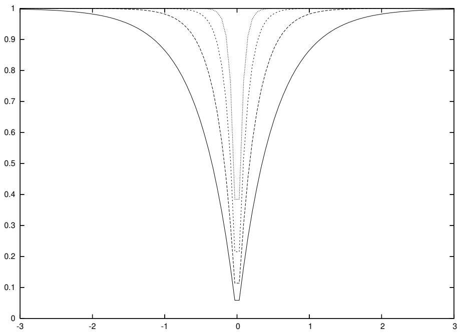


Figure 1: Plot of $f_2(x)$, $f_4(x)$, $f_8(x)$, and $f_{16}(x)$ for $n = 1$.

3.3. Mangasarian's Approach

We now present an adaptation of a general method due to Mangasarian [29]. The idea is to replace $\|\boldsymbol{x}\|_0$ by the approximation

$$f_\alpha(\boldsymbol{x}) = \sum_{j=1}^{n} \big(1 - e^{-\alpha|x_j|}\big),$$

for some $\alpha > 0$, see Figure 1. We clearly have $f_\alpha(\boldsymbol{x}) \leq \|\boldsymbol{x}\|_0$ and for $\boldsymbol{x} \in \mathbb{R}^n$,

$$\lim_{\alpha\to\infty} f_\alpha(\boldsymbol{x}) = \|\boldsymbol{x}\|_0.$$

For a fixed α and δ, we consider the problem

$$(M_\alpha^\delta) \quad \min\{f_\alpha(\boldsymbol{y}) \,:\, (\boldsymbol{x},\boldsymbol{y}) \in S(\delta)\},$$

where

$$S(\delta) := \{(\boldsymbol{x},\boldsymbol{y}) \in \mathbb{R}^n \times \mathbb{R}^n \,:\, \boldsymbol{b} - \delta\cdot\mathbb{1} \leq A\boldsymbol{x} \leq \boldsymbol{b} + \delta\cdot\mathbb{1},\ -\boldsymbol{y} \leq \boldsymbol{x} \leq \boldsymbol{y}\}.$$

Mangasarian [29] proved that a stationary point of (M_α^δ) can be reached by a successive linearization algorithm in finite time. Furthermore, there exists some α such that (M_α^δ) optimally solves (P_0^δ).

The successive linearization algorithm works as follows. Given a solution $(\boldsymbol{x}^i, \boldsymbol{y}^i) \in S(\delta)$, we find a (vertex) solution $(\boldsymbol{x}^{i+1}, \boldsymbol{y}^{i+1})$ of

$$\min\{\, \boldsymbol{c}^{\mathrm{T}}\boldsymbol{y} \,:\, (\boldsymbol{x},\boldsymbol{y}) \in S(\delta)\,\}, \qquad \text{where} \qquad c_j = \alpha\, e^{-\alpha y_j^i},\ j = 1,\dots,n.$$

Here, we replace $|x_j|$ by y_j and use that

$$\tfrac{d}{dy_j}(1 - e^{-\alpha\, y_j}) = \alpha e^{-\alpha\, y_j}.$$

In the above linear program the constant part of the linearization is skipped, because it is irrelevant. We terminate the process once $\|\boldsymbol{y}^{i+1} - \boldsymbol{y}^i\|_\infty$ is small enough.

Although it is guaranteed that there exists α such that (M_α^δ) computes an optimal solution to (P_0^δ), there is no efficient way to compute it (as long as P $\neq$ NP). We therefore use the following simple approach. We start with $\alpha = 0.1$ and double it after every of 20 iterations, each time solving the linearization with the successive linearization algorithm and storing the sparsest solution found.

3.4. A Bilinear Formulation

In this section, we introduce another formulation for the problem (P_0^δ) and a heuristic way to solve this model. This approach follows ideas of Bennett and Bredensteiner [6].

Lemma 5. *The minimization problem* (P_0^δ) *is equivalent to the following bilinear program (with equilibrium constraints):*

$$\begin{aligned} \min\ & \mathbb{1}^{\mathrm{T}}\boldsymbol{z} \\ & \boldsymbol{b} - \delta \cdot \mathbb{1} \leq A\boldsymbol{x} \leq \boldsymbol{b} + \delta \cdot \mathbb{1} \\ & x_j(1 - z_j) = 0,\ j \in [n] \\ & \boldsymbol{0} \leq \boldsymbol{z} \leq \mathbb{1}. \end{aligned}$$

Proof. Suppose that $\boldsymbol{x}^\star$ is an optimal solution of (P_0^δ) and $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}})$ is an optimal solution of the above bilinear program. We want to show that $\mathbb{1}^T\hat{\boldsymbol{z}} = \|\boldsymbol{x}^\star\|_0$. Defining $z_j^\star = 1$ if $x_j^\star \neq 0$ and $z_j^\star = 0$ if $x_j^\star = 0$, we have

$$\|\boldsymbol{x}^\star\|_0 = \|\boldsymbol{z}^\star\|_0 = \mathbb{1}^{\mathrm{T}}\boldsymbol{z}^\star \geq \mathbb{1}^T\hat{\boldsymbol{z}},$$

since $(\boldsymbol{x}^\star, \boldsymbol{z}^\star)$ is a feasible point for the above problem. Conversely, since if $\hat{x}_j \neq 0$ then $\hat{z}_j = 1$, we have $\|\hat{\boldsymbol{x}}\|_0 \leq \mathbb{1}^T\hat{\boldsymbol{z}}$. Hence, it follows that

$$\|\boldsymbol{x}^\star\|_0 \leq \|\hat{\boldsymbol{x}}\|_0 \leq \mathbb{1}^T\hat{\boldsymbol{z}},$$

since $\hat{\boldsymbol{x}}$ is a feasible solution for (P_0^δ). □

Note that the set of feasible points of the bilinear formulation is not convex anymore. Hence, there may exist many local optima.

A direct successive linearization method does not help for the above model, since fixing $\boldsymbol{x}$ or $\boldsymbol{z}$ forces hard bounds for the remaining variables that cannot be changed. We can, however, use a trick similar to the approach of Bennett and Bredensteiner [6]. The idea is to introduce a bound $\beta \in \mathbb{Z}_+$ on the objective function and move the bilinear constraints to the objective. To make the problem bounded, we note that $x_j(1 - z_j) = 0$ if and only if $|x_j|(1-z_j) = 0$. Hence, we again introduce variables $\boldsymbol{y}$ modeling the absolute values of $\boldsymbol{x}$ (see Section 3.1). This leads to the following formulation:

$$(B_\beta^\delta) \qquad \begin{aligned} \min\ & \boldsymbol{y}^{\mathrm{T}}(\mathbb{1} - \boldsymbol{z}) \\ & (\boldsymbol{x}, \boldsymbol{y}) \in S(\delta) \\ & \boldsymbol{0} \leq \boldsymbol{z} \leq \mathbb{1} \\ & \mathbb{1}^{\mathrm{T}}\boldsymbol{z} \leq \beta. \end{aligned}$$

Given a fixed $\beta \in \mathbb{Z}_+$, a natural successive linearization method for this formulation is as follows. At the ith iteration we have points $(\boldsymbol{x}^i, \boldsymbol{y}^i) \in S(\delta)$ and $\boldsymbol{z}^i \in [0,1]^n$. We then solve the linear program

$$(B_{\beta,1}^\delta) \qquad \begin{aligned} \min\ & \boldsymbol{y}^{\mathrm{T}}(\mathbb{1} - \boldsymbol{z}^i) \\ & (\boldsymbol{x}, \boldsymbol{y}) \in S(\delta) \end{aligned}$$

to obtain $\boldsymbol{x}^{i+1}$ and $\boldsymbol{y}^{i+1}$. Then we solve

$$(B_{\beta,2}^\delta) \qquad \begin{aligned} \min\ & (\boldsymbol{y}^{i+1})^{\mathrm{T}}(\mathbb{1} - \boldsymbol{z}) \\ & \boldsymbol{0} \leq \boldsymbol{z} \leq \mathbb{1} \\ & \mathbb{1}^{\mathrm{T}}\boldsymbol{z} \leq \beta, \end{aligned}$$

to get $\boldsymbol{z}^{i+1}$. The process is repeated as long as the objective function improves, i.e., $(\boldsymbol{y^{i+1}})^T(\mathbb{1} - \boldsymbol{z^{i+1}}) < (\boldsymbol{y^i})^T(\mathbb{1} - \boldsymbol{z^i})$.

In fact, the second linearization problem $(B^{\delta}_{\beta,2})$ has a simple closed form solution, see also [6]; we skip the easy proof.

Lemma 6. *Given the nonnegative vector $\boldsymbol{y}^{i+1} \in \mathbb{R}^n$, let $j_1, \dots, j_n$ be a permutation of $[n]$, such that $y^{i+1}_{j_1} \geq y^{i+1}_{j_2} \geq \cdots \geq y^{i+1}_{j_n}$. Then the vector defined by $z_{j_k} = 1$ for $k = 1, \dots, \beta$ and $z_{j_k} = 0$ for $k = \beta + 1, \dots, n$ is an optimal solution to $(B^{\delta}_{\beta,2})$.*

Bennett and Mangasarian [5] prove that (B^{δ}_{β}) has a vertex solution and then provide the following finiteness result for the successive linearization algorithm.

Theorem 7 (Bennett and Mangasarian [5])**.** *The successive linearization algorithm terminates in a finite number of steps at a global solution or a point $(\boldsymbol{x^{i+1}}, \boldsymbol{y^{i+1}}, \boldsymbol{z^i})$ that satisfies the minimum principle necessary optimality condition*

$$(\mathbb{1} - \boldsymbol{z}^i)^T(\boldsymbol{y} - \boldsymbol{y}^{i+1}) + (\boldsymbol{y}^{i+1})^T(\boldsymbol{z}^i - \boldsymbol{z}) \geq 0,$$

for all $(\boldsymbol{x}, \boldsymbol{y})$ feasible for $(B^{\delta}_{\beta,1})$ and $\boldsymbol{z}$ which is feasible for $(B^{\delta}_{\beta,2})$.

On top of the above linearization process one adds a binary search routine to determine the optimal β. We use the secant method described in [6].

3.5. Postprocessing

Having obtained a solution $\boldsymbol{x}^\star$ by a heuristic method, we can try to iteratively reduce the support of given points $\boldsymbol{x}^i \in Q^\delta$, where $\boldsymbol{x}^0 := \boldsymbol{x}^\star$. For this we let $S = \operatorname{supp}(\boldsymbol{x}^i)$ and check for each $k \in S$ whether

$$\{\boldsymbol{x} \in Q^\delta \,:\, x_j = 0,\ j \in [n] \setminus S,\ x_k = 0\} \neq \varnothing.$$

If this is the case for some k, we let $\boldsymbol{x}^{i+1}$ be the solution of this linear program. This approach sometimes helps to further reduce the support of the solution.

4. Computational Results

We conducted several computational experiments with the algorithms discussed in this paper. We use deterministic and random matrices and construct instances that are guaranteed to have small support.

The heuristics are implemented as described in Section 3 and the branch-and-cut algorithm as indicated in Section 2. All algorithms are implemented in `C/C++`. The linear programs for the heuristics are solved via the primal simplex implementation of CPLEX 10.11. The implementation of the branch-and-cut algorithm is described in detail in Pfetsch [33]; it is based on the framework SCIP (Solving Constraint Integer Programs) version 0.90 by Achterberg [1], using CPLEX 10.11 for solving the intermediate LPs. The computations were performed on a 3.4 GHz Pentium 4 machine with 3 GB of main memory and 2 MB cache, running Linux.

4.1. Experiments with Deterministic Matrices

In this section, we use deterministic matrices constructed as follows. We take $A = [I\ H] \in \mathbb{R}^{m\times 2m}$, where H is the normalized Hadamard matrix of size m, and I is the $m \times m$ identity matrix; this matrix was also used in Donoho, Elad, and Temlyakov [20]. By construction of the Hadamard matrix, m is a power of 2. The incoherence is $M := 1/\sqrt{m}$.

4.1.1. *The case* $m = 16$. We first investigate the case in which the number of rows of A is $m = 16$. Hence, the number of columns is $n = 32$, and the number of variables in the Max FS formulation is 64.

In a first experiment, we study the case in which we allow no deviation, i.e., $\delta = 0$. For each $N = 1, 2, \dots, 16$, we generated three random instances as follows. Take a random vector $\boldsymbol{x}_0$ with N nonzeros. The support entries are chosen uniformly at random, and the values are drawn from a normal distribution with zero mean and variance 1. Then take $\boldsymbol{b} = A\boldsymbol{x}_0$. Hence, it is guaranteed that $A\boldsymbol{x} = \boldsymbol{b}$ has a solution with N nonzeros, but it is possible that there exist solutions with fewer nonzeros – see the results below. Note that from $N = 16$ on, there trivially are solutions with at most 16 nonzeros.

Table 1 shows the results of the branch-and-cut algorithm and the heuristics. The columns have the following meaning: "nodes" gives the number of nodes/subproblems in the branch-and-cut algorithm, "time" is the CPU time of the branch-and-cut algorithm in seconds, "sols" lists the number of optimal solutions, "opt" gives the number of nonzeros in an optimal solution, "BP" shows the results of basis pursuit, "MG" of Mangasarian's approach, "OMP" of orthogonal matching pursuit, "GOMP" of the greedy OMP, and "BL" of the bilinear approach. Postprocessing never improved the solutions. Heuristical solutions that are optimal are marked in bold face. For computing the number of optimal solutions we set a time limit of 10 hours. Numbers appearing in italics in the corresponding column were obtained when the time limit was reached. The last row contains the geometric means over each column. The computation times of the heuristics are well below one second and are therefore not listed in the table. We now discuss these results in more detail.

The results show that the larger N gets, the more time the branch-and-cut algorithms needs. For $N \leq 13$ all instances can be solved in a couple of seconds, while for $N = 16$ the execution almost needs up to six hours. In fact, the solution time is correlated to the size of the optimal solution – if it is at least 15, the instances get very hard to solve. Note that for $N \leq 12$ the size of each optimal solution is N, while for larger N there often exists a solution of size smaller than N.

We used a modification of the branch-and-cut algorithm to count the number of optimal solutions. Table 1 shows that for $N \leq 11$ (with one exception for $N = 9$) the optimal solutions are unique, although Theorem 1 guarantees uniqueness only for $N \leq 1/M = \sqrt{16} = 4$. These results complement the experiments of Elad [23], who investigated uniqueness for matrices of size 8×20 by enumeration.

The heuristics perform very well for small N, usually finding the optimal solution; an exception is the second instance for $N = 6$, where all heuristics

Table 1: Results of the branch-and-cut algorithm (columns 2–5) and heuristics (BP, MG, OMP, GOMP, and BL) for *Hadamard matrices* with $m = 16$ and $\delta = 0$. Column "opt" gives the optimal solution, and numbers in bold are heuristical solutions that are optimal. The last row contains the geometric mean of the entries in each column.

N	nodes	time	sols	opt	BP	MG	OMP	GOMP	BL
1	1	0.01	1	1	**1**	**1**	**1**	**1**	**1**
1	1	0.00	1	1	**1**	**1**	**1**	**1**	**1**
1	1	0.00	1	1	**1**	**1**	**1**	**1**	**1**
2	1	0.00	1	2	**2**	**2**	**2**	**2**	**2**
2	2	0.00	1	2	**2**	**2**	**2**	**2**	**2**
2	1	0.01	1	2	**2**	**2**	**2**	**2**	**2**
3	1	0.01	1	3	**3**	**3**	**3**	**3**	**3**
3	2	0.01	1	3	**3**	**3**	**3**	**3**	**3**
3	2	0.01	1	3	**3**	**3**	**3**	**3**	**3**
4	4	0.01	1	4	**4**	**4**	**4**	**4**	**4**
4	1	0.00	1	4	**4**	**4**	**4**	**4**	**4**
4	2	0.01	1	4	**4**	**4**	**4**	**4**	**4**
5	1	0.01	1	5	**5**	**5**	**5**	**5**	**5**
5	1	0.01	1	5	**5**	**5**	**5**	**5**	**5**
5	4	0.03	1	5	**5**	**5**	**5**	**5**	**5**
6	1	0.02	1	6	8	**6**	**6**	**6**	**6**
6	17	0.15	1	6	14	11	9	9	11
6	1	0.02	1	6	**6**	**6**	**6**	**6**	**6**
7	1	0.04	1	7	**7**	**7**	**7**	**7**	**7**
7	4	0.08	1	7	**7**	**7**	**7**	**7**	**7**
7	15	0.09	1	7	**7**	**7**	**7**	**7**	**7**
8	13	0.18	1	8	**8**	**8**	9	**8**	**8**
8	14	0.20	1	8	11	**8**	**8**	10	**8**
8	53	0.34	1	8	10	9	**8**	**8**	9
9	167	0.56	1	9	**9**	**9**	**9**	12	**9**
9	123	0.48	8	9	12	**9**	**9**	**9**	**9**
9	227	1.09	1	9	13	12	11	11	12
10	915	4.58	1	10	14	11	12	12	11
10	633	3.40	1	10	15	13	13	13	14
10	296	1.42	1	10	13	12	**10**	13	12
11	2751	14.92	1	11	16	13	14	14	12
11	1539	10.37	1	11	16	16	16	12	16
11	719	3.50	1	11	14	**11**	15	14	**11**
12	8827	38.17	1	12	16	14	14	14	14
12	4772	28.90	8	12	16	16	16	15	15
12	2784	14.39	1	12	16	13	14	14	15
13	20505	63.26	48	13	14	14	16	14	**13**
13	3248	14.50	48	12	16	13	**12**	**12**	**12**
13	18007	51.11	288	13	15	15	16	16	15
14	178872	286.82	1296	14	16	15	15	**14**	**14**
14	190032	313.60	1296	14	16	16	16	16	16
14	171931	293.62	48	14	16	15	16	16	15
15	22553939	20881.82	*54609*	15	16	16	16	16	16
15	162437	320.93	1296	14	16	15	16	15	15
15	132454	220.60	970	14	16	16	16	15	16
16	23272940	21847.21	*63691*	15	**15**	**15**	16	16	**15**
16	22604125	21133.88	*55278*	15	16	16	16	16	16
16	19706327	18380.20	*50*	15	16	16	16	16	16
G	219.7	1.84	6.1	6.7	7.7	7.3	7.3	7.3	7.2

fail to find solutions close to optimal. Basis pursuit always finds the optimal solutions for $N \leq 5$. The maximum size of N for which this is guaranteed by Theorem 2 is $(\sqrt{2} - \frac{1}{2})/M \approx 0.91 \cdot 4 = 3.64$. Hence, in this case, the theoretical bound is quite close to the empirical bound. The other heuristics all improve upon basis pursuit and often find optimal solutions for $N \leq 9$. For large N, their performance does not show a clear picture; quite frequently they only find trivial solutions of size 16. There is no clear winner among the heuristics (excluding basis pursuit), although the bilinear method seems to be slightly better than the other heuristics (the geometric mean of its solution values is 7.2, compared to 7.3 for the others). Furthermore, the bilinear method finds 29 optimal solutions, compared to 22 for basis pursuit, 26 for Mangasarian's method, 26 for OMP, and 25 for GOMP.

In the second experiment, we allow a deviation of $\delta = 0.1$. Table 2 shows the results, the notation being the same as in Table 1. Column "nz" gives the number of nonzeros of the computed solution, and we additionally report in column "p", for each heuristic, the number of nonzeros obtained by the postprocessing method of Section 3.5. Unlike the $\delta = 0$ case, here this method can often improve the solution. Hence, it seems to be a good idea to use it after any of the heuristics. The computation times of the heuristics are again negligible.

Because of the allowed deviation, the optimal solution values are usually smaller than N. For instance, no optimal solution has more than 10 nonzeros. As a consequence, each instance can be solved by the branch-and-cut algorithm within two seconds.

The optimal solutions are unique only for $N \leq 5$. An explanation is that the admittance of solutions in a neighborhood around the solution with N nonzeros allows more candidates. On the other hand, the number of optimal solutions stays small. It seems that these solutions are more isolated, since their support is smaller (because of the deviation), and solutions to the equations with few nonzeros tend to be unique as observed above.

Again the heuristics perform quite well; for $N \leq 10$, at least one heuristic (with postprocessing) finds the optimal solution. When comparing the heuristics, it again turns out that basis pursuit, although its results can be considerably improved by postprocessing, in general produces worse solutions than the other heuristics. The best heuristic with respect to the geometric mean over the number of produced nonzeros is GOMP, followed by OMP, Mangasarian's method, the bilinear approach, and finally basis pursuit, but the numbers are quite close together for the first four. Note that the geometric mean of 5.3 for GOMP with postprocessing is not far from the geometric mean over all optimal solutions with 5.1, emphasizing the good quality of the produced solutions. Furthermore, basis pursuit finds 21 optimal solutions after postprocessing, Mangasarian's method 32, OMP 35, GOMP 39, and the bilinear method finds 27 optimal solutions after postprocessing. Hence, for these instances, GOMP performs better than the other heuristics, but as we will see later, it is also quite time consuming for larger instances.

4.1.2. *The case $m = 128$.* We experimented with the branch-and-cut algorithm for larger instances. It turned out that already for $m = 32$, it cannot solve the instances with large N to optimality within any reasonable

Table 2: Results of the branch-and-cut algorithm and heuristics for *Hadamard matrices* with $m = 16$ and $\delta = 0.1$. The notation is as in Table 1. Column "nz" gives the number of nonzeros in a solution and column "p" the number of nonzeros after the postprocessing method of Section 3.5.

					BP		MG		OMP		GOMP		BL	
N	nodes	time	sols	opt	nz	p	nz	p	nz	p	nz	p	nz	p
1	1	0.00	1	1	5	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
1	1	0.00	1	0	**0**	**0**	**0**	**0**	1	**0**	1	**0**	**0**	**0**
1	1	0.00	1	1	4	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
2	2	0.01	1	2	6	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**
2	2	0.01	1	2	6	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**
2	1	0.00	1	2	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**
3	2	0.01	1	2	6	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**
3	4	0.02	1	3	8	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**
3	1	0.02	1	3	7	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**
4	2	0.01	1	3	8	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**
4	2	0.02	1	4	7	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**
4	2	0.01	1	3	6	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**
5	4	0.01	1	3	5	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**
5	2	0.02	1	4	13	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**
5	2	0.01	1	2	6	**2**	3	3	3	**2**	**2**	**2**	4	3
6	4	0.03	10	5	9	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
6	4	0.03	3	5	9	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
6	10	0.05	2	6	12	9	8	**6**	**6**	**6**	**6**	**6**	9	8
7	4	0.03	1	5	10	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
7	31	0.11	2	6	13	7	7	7	7	7	**6**	**6**	10	8
7	4	0.02	1	4	11	7	**4**	**4**	**4**	**4**	**4**	**4**	7	7
8	4	0.09	1	7	10	10	9	**7**	9	9	**7**	**7**	9	**7**
8	15	0.07	9	6	8	7	7	7	**6**	**6**	**6**	**6**	7	7
8	31	0.10	1	6	12	11	8	8	9	8	**6**	**6**	11	11
9	11	0.05	2	6	8	**6**	**6**	**6**	7	**6**	**6**	**6**	**6**	**6**
9	27	0.11	2	7	11	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**
9	5	0.15	3	7	12	12	**7**	**7**	**7**	**7**	**7**	**7**	10	9
10	15	0.10	8	7	13	10	**7**	**7**	8	**7**	**7**	**7**	**7**	**7**
10	165	0.53	42	9	15	10	**9**	**9**	10	**9**	**9**	**9**	10	**9**
10	35	0.21	5	8	12	9	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**
11	27	0.09	11	7	13	10	**7**	**7**	**7**	**7**	**7**	**7**	9	**7**
11	27	0.09	1	7	11	9	9	8	8	8	8	8	9	8
11	62	0.32	2	8	13	9	10	10	10	10	9	9	9	9
12	176	0.52	10	9	14	13	13	11	10	**9**	**9**	**9**	11	11
12	185	0.56	1	8	15	12	9	9	9	9	9	9	11	11
12	431	1.44	45	10	11	**10**	**10**	**10**	11	11	**10**	**10**	**10**	**10**
13	183	0.57	3	9	12	11	10	10	11	11	11	11	11	11
13	388	1.56	87	10	16	13	15	14	12	12	11	11	11	11
13	103	0.45	6	9	14	11	**9**	**9**	10	**9**	**9**	**9**	13	11
14	377	1.73	44	10	15	13	**10**	**10**	12	**10**	**10**	**10**	11	**10**
14	430	1.69	78	10	14	11	11	11	11	**10**	**10**	**10**	11	11
14	160	0.49	3	9	13	13	10	10	11	**9**	10	10	12	12
15	181	0.57	2	9	15	13	10	10	11	10	10	10	12	12
15	352	1.47	12	10	16	13	12	12	11	11	**10**	**10**	13	12
15	369	1.22	18	10	14	12	11	**10**	**10**	**10**	11	11	**10**	**10**
16	153	0.72	2	9	14	13	13	12	10	**9**	10	**9**	11	11
16	463	1.78	76	10	16	11	**10**	**10**	11	11	**10**	**10**	11	11
16	375	1.42	32	10	16	14	11	11	12	11	11	11	11	11
G	18.5	0.14	3.3	5.1	9.5	6.1	5.6	5.5	5.6	5.4	5.3	5.3	5.9	5.7

	BP		MG		OMP			GOMP			BL	
N	nz	time	nz	time	nz	time	p	nz	time	p	nz	time
10	10.0	(0.07)	10.0	(0.10)	10.0	(0.01)	10.0	10.0	(0.11)	10.0	10.0	(0.09)
15	15.0	(0.06)	15.0	(0.10)	15.0	(0.01)	15.0	15.0	(0.33)	15.0	15.0	(0.10)
20	20.0	(0.07)	20.0	(0.12)	20.0	(0.01)	20.0	20.0	(0.72)	20.0	20.0	(0.11)
25	25.0	(0.07)	25.0	(0.12)	25.0	(0.02)	25.0	25.0	(1.31)	25.0	25.0	(0.12)
30	30.0	(0.07)	30.0	(0.12)	30.0	(0.02)	30.0	30.0	(2.13)	30.0	30.0	(0.12)
35	35.0	(0.08)	35.0	(0.12)	35.0	(0.03)	35.0	35.0	(3.24)	35.0	35.0	(0.13)
40	40.0	(0.08)	40.0	(0.13)	40.0	(0.03)	40.0	40.0	(4.63)	40.0	40.0	(0.14)
45	55.7	(0.08)	45.0	(0.13)	45.0	(0.04)	45.0	45.0	(6.33)	45.0	45.0	(0.13)
50	70.1	(0.09)	50.9	(0.15)	50.0	(0.05)	50.0	50.1	(8.41)	50.0	50.0	(0.15)
55	100.3	(0.09)	55.0	(0.15)	56.7	(0.07)	56.0	56.6	(11.62)	55.9	56.0	(0.15)
60	127.2	(0.08)	60.0	(0.17)	60.4	(0.08)	60.0	60.3	(13.61)	60.0	65.6	(0.16)
65	127.9	(0.08)	67.7	(0.18)	80.5	(0.20)	79.5	78.0	(28.87)	77.2	69.7	(0.14)
70	128.0	(0.08)	74.9	(0.22)	91.2	(0.27)	89.6	99.9	(50.85)	98.8	94.2	(0.18)
75	128.0	(0.08)	118.8	(0.33)	117.8	(0.46)	117.2	113.2	(65.26)	113.0	123.4	(0.18)
80	128.0	(0.08)	113.7	(0.33)	122.5	(0.51)	121.9	120.2	(73.78)	119.8	127.0	(0.17)
G	51.1	(0.08)	41.1	(0.15)	42.4	(0.05)	42.2	42.4	(5.26)	42.3	42.5	(0.14)

Table 3: Results of the heuristics for *Hadamard matrices* with $m = 128$ and $\delta = 0$. Columns "p" give number of nonzeros after postprocessing. The entries are averages over ten instances. The last row contains the geometric mean of the entries in each column.

time. Furthermore, the dual bounds can only provide meaningful information for very small N (approx. $N \leq 35$). We therefore decided to test only the heuristics on larger instances and present results for $m = 128$ and $N = \{10, 15, 20, \dots, 80\}$ for $\delta = 0$ and $N = \{10, 15, \dots, 100\}$ for $\delta = 0.1$. For each N we generated *ten* instances (i.e., random right hand sides). The entries in the tables report average results.

Table 3 shows the results of the heuristics for $\delta = 0$. We skipped the columns for the postprocessing method, in which no result could be improved. The last row again contains the geometric mean of each column.

We observe that all heuristics could find a solution of value N for $N \leq 40$. It turns out that Mangasarian's approach produces the sparsest solutions in geometric mean, closely followed by OMP, GOMP, the bilinear method, and finally basis pursuit. Note that GOMP does not improve upon OMP in the geometric mean, but this is mainly because of the bad performance for $N = 70$. For all other N, GOMP is better than OMP on average. GOMP, however, needs much larger computation times. Note also that basis pursuit only produces useless results for $N \geq 55$. Furthermore, the geometric mean of Mangasarian's approach with 41.1 is not too far from the geometric mean over all N with 38.6, indicating the good performance of this heuristic.

Table 4 shows the results for the case $\delta = 0.1$. Because of space constraints, we skipped the information about the running times, except for GOMP; the other running times are similarly small as in Table 3. The results again show that postprocessing pays off. Altogether, GOMP produces the sparsest solutions in geometric mean after postprocessing, followed Mangasarian's approach, OMP, and the bilinear method; basis pursuit performs clearly worse than the other methods. All heuristics produce solutions whose number of nonzeros in the geometric mean are below the geometric mean over all N with 46.4.

	BP		MG		OMP		GOMP			BL	
N	nz	p	nz	p	nz	p	nz	time	p	nz	p
10	12.7	7.2	7.3	7.2	8.4	7.3	7.3	(0.51)	7.3	8.1	7.4
15	20.2	10.8	11.1	10.9	12.6	11.2	10.9	(0.94)	10.8	11.0	10.9
20	26.7	16.4	15.6	15.6	18.0	15.7	15.2	(1.55)	15.2	15.5	15.2
25	34.4	20.1	20.7	19.9	22.0	19.5	19.6	(2.33)	19.6	21.0	20.2
30	42.0	23.7	23.9	23.4	26.3	23.5	23.1	(2.83)	23.1	23.7	23.4
35	53.8	31.5	29.3	29.3	32.0	29.0	28.5	(4.55)	28.5	29.1	28.7
40	57.9	35.3	31.6	31.3	34.3	30.9	30.2	(4.94)	30.2	30.7	30.2
45	67.8	42.9	35.9	35.5	39.4	35.5	35.0	(6.32)	35.0	37.8	36.8
50	76.4	50.6	39.2	38.9	43.1	39.1	38.3	(8.03)	38.1	39.6	39.2
55	82.1	53.2	43.2	42.7	48.9	43.9	41.8	(8.89)	41.7	43.4	42.6
60	91.4	65.3	48.6	48.3	51.8	47.4	47.3	(11.07)	47.2	50.5	49.8
65	96.5	70.7	53.2	53.0	61.8	56.3	52.1	(12.68)	51.9	55.2	54.6
70	100.4	78.6	57.8	57.4	65.9	59.1	59.4	(15.09)	59.1	61.9	60.5
75	103.5	84.9	62.9	62.6	70.4	65.4	62.1	(15.71)	62.1	69.1	68.3
80	105.1	85.6	65.8	65.8	71.4	66.7	63.6	(17.72)	63.6	68.6	67.3
85	108.0	87.0	70.8	70.3	77.2	71.7	68.3	(17.83)	68.3	73.6	72.3
90	109.7	89.8	72.7	72.6	76.7	71.4	69.7	(19.40)	69.5	75.7	75.0
95	112.2	90.5	73.1	72.6	78.3	72.9	71.7	(20.46)	71.5	76.1	75.3
100	114.0	93.9	72.8	72.7	78.7	73.4	72.6	(19.79)	72.6	77.4	76.5
G	63.7	43.5	36.9	36.6	40.7	36.9	36.0	(6.63)	36.0	38.1	37.3

Table 4: Results of the heuristics for *Hadamard matrices* with $m = 128$ and $\delta = 0.1$.

	BP		MG		OMP			GOMP			BL	
N	nz	time	nz	time	nz	time	p	nz	time	p	nz	time
10	10.0	(0.16)	10.0	(0.29)	10.0	(0.00)	10.0	10.0	(0.11)	10.0	10.0	(0.24)
15	15.0	(0.16)	15.0	(0.29)	15.0	(0.01)	15.0	15.0	(0.33)	15.0	15.0	(0.24)
20	20.0	(0.18)	20.0	(0.35)	20.1	(0.01)	20.0	20.1	(0.72)	20.0	20.0	(0.31)
25	25.0	(0.17)	25.0	(0.31)	25.3	(0.01)	25.0	25.2	(1.32)	25.0	25.0	(0.28)
30	30.0	(0.18)	30.0	(0.31)	30.4	(0.02)	30.0	30.1	(2.16)	30.0	30.0	(0.28)
35	35.0	(0.19)	35.0	(0.34)	35.9	(0.02)	35.0	35.2	(3.25)	35.0	35.0	(0.31)
40	40.0	(0.20)	40.0	(0.36)	41.6	(0.03)	40.0	41.5	(5.05)	40.0	40.0	(0.32)
45	45.0	(0.21)	45.0	(0.40)	48.0	(0.05)	45.0	47.6	(7.38)	45.0	45.0	(0.36)
50	89.0	(0.21)	50.0	(0.38)	59.8	(0.10)	57.7	52.4	(9.42)	50.0	50.0	(0.36)
55	120.7	(0.20)	55.0	(0.43)	82.4	(0.22)	76.4	65.2	(19.11)	62.1	62.2	(0.44)
60	128.0	(0.20)	60.0	(0.45)	108.7	(0.40)	106.9	102.2	(53.99)	99.3	80.3	(0.42)
65	128.0	(0.20)	71.3	(0.57)	121.3	(0.49)	121.0	125.9	(78.55)	125.9	96.3	(0.42)
70	128.0	(0.20)	81.6	(0.67)	127.2	(0.53)	127.2	122.0	(73.96)	120.1	104.6	(0.44)
75	128.0	(0.20)	106.8	(0.87)	127.3	(0.54)	127.3	126.0	(78.63)	126.0	117.4	(0.45)
80	128.0	(0.20)	113.5	(0.92)	127.3	(0.53)	127.3	125.7	(78.24)	125.7	127.7	(0.44)
G	51.8	(0.19)	41.2	(0.43)	49.0	(0.09)	48.0	47.4	(6.81)	46.6	44.5	(0.35)

Table 5: Results of the heuristics for *random matrices* with $m = 128$ and $\delta = 0$.

4.2. Experiments with Random Matrices

In our final two experiments, we constructed random matrices by taking 256 unit random vectors of size 128. Similar random matrices and experiments have been applied in many articles, see, e.g., Candès and Tao [11]. We ran the heuristics for *ten* instances each and report the averages in Table 5 for the case of $\delta = 0$ and in Table 6 for $\delta = 0.1$. The notation is similar to Tables 3 and 4.

	BP		MG		OMP		GOMP			BL	
N	nz	p	nz	p	nz	p	nz	time	p	nz	p
10	13.7	5.8	6.0	5.6	5.3	5.1	5.3	(1.53)	5.3	6.4	5.4
15	21.6	10.5	7.7	7.6	8.4	7.6	7.6	(2.20)	7.6	7.8	7.7
20	30.3	15.0	11.1	10.9	12.5	10.6	10.8	(3.38)	10.6	11.0	10.7
25	37.3	25.3	14.9	14.9	17.8	15.2	15.3	(4.97)	15.2	15.8	15.3
30	43.3	27.8	17.9	17.6	20.7	18.2	17.9	(6.49)	17.9	19.2	18.1
35	47.0	34.1	20.6	20.2	24.3	21.4	21.0	(8.33)	20.7	21.5	20.7
40	54.9	40.0	23.4	23.1	29.1	24.6	24.2	(10.37)	24.0	24.6	24.3
45	60.2	45.1	27.6	26.9	33.6	29.3	29.0	(12.97)	28.8	29.5	28.3
50	62.9	50.5	30.6	30.4	36.4	31.3	32.8	(15.88)	31.9	32.6	31.4
55	70.2	54.1	32.2	31.9	44.2	36.8	36.3	(19.12)	35.8	34.6	33.4
60	75.9	60.5	36.6	36.2	51.7	44.4	38.8	(21.44)	38.6	38.1	37.7
65	78.8	61.9	38.9	38.6	54.6	46.6	44.3	(26.54)	43.4	40.2	39.6
70	80.4	63.4	42.6	42.3	60.6	51.6	49.6	(31.65)	48.6	44.8	43.9
75	83.2	67.8	44.3	44.2	60.6	53.0	50.0	(31.85)	49.7	46.3	45.0
80	86.0	70.3	46.4	46.1	63.2	55.1	51.7	(34.56)	50.6	49.1	48.7
85	90.2	74.3	47.0	46.4	66.8	57.7	55.3	(37.69)	53.7	51.0	50.1
90	89.1	69.8	48.9	48.7	66.8	57.1	53.6	(36.07)	52.7	51.2	50.3
95	90.8	73.1	51.0	50.5	66.0	58.7	55.4	(38.13)	53.9	53.8	52.6
100	94.3	75.8	53.1	52.7	71.1	60.5	56.4	(40.34)	55.4	54.1	53.3
G	57.1	40.5	26.8	26.5	33.7	29.3	28.5	(13.89)	28.1	28.2	27.3

Table 6: Results of the heuristics for *random matrices* with $m = 128$ and $\delta = 0.1$.

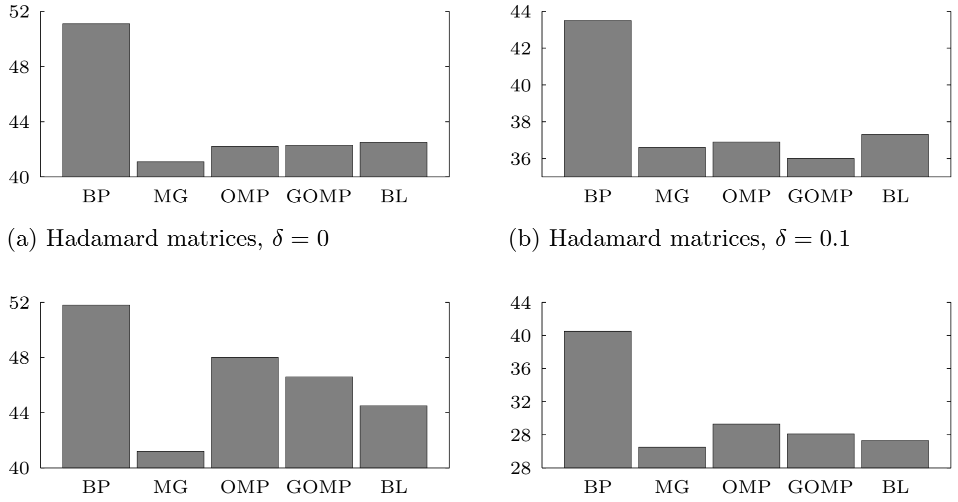
The general outcome of the experiments for $\delta = 0$ are similar to the deterministic case. Basis pursuit fails to recover all solutions of size N beginning at $N = 50$, while Mangasarian's approach yields perfect recovery for $N \leq 60$. The best performance in the geometric mean is achieved by Mangasarian's approach. Recall that the geometric mean over all N is 38.6, and hence Mangasarian's approach with 41.2 yields good results.

The results for $\delta = 0.1$ confirm the above results. Clearly the best heuristic for these random instances is Mangasarian's approach, followed by the bilinear method, GOMP (which is again much more expensive than the other heuristics), OMP, and, clearly the worst heuristic, basis pursuit. A comparison to the geometric mean of 46.4 over all N shows that the solutions of the heuristics are of considerable smaller size than N, in the geometric mean. The results also show that the heuristics, in general, produce very good results (up to about $N = 70$ for $\delta = 0$ and generally for $\delta = 0.1$).

Comparing the results for Hadamard and random matrices, the picture is not clear. Surprisingly, for $\delta = 0$, the heuristics produce solutions with more nonzeros in the geometric mean for random matrices than for Hadamard matrices, while for $\delta = 0.1$ the reverse holds. The results for $\delta = 0$ are not in conflict with theory, since the known good behavior of random matrices, for instance in Theorem 3, hold only asymptotically and with high probability.

5. Conclusions

The results of this paper can be summarized as follows. An exact solution of the sparse representation problem by branch-and-cut seems to be hard, especially if the optimal solutions have large support. Nevertheless, the optimal solutions obtained in this way allow to evaluate the performance



(a) Hadamard matrices, $\delta = 0$
(b) Hadamard matrices, $\delta = 0.1$
(c) Random matrices, $\delta = 0$
(d) Random matrices, $\delta = 0.1$

Figure 2: Comparison of the geometric means for the experiments of Section 4.1.2 and 4.2 with $m = 128$.

of heuristics. Furthermore, this exact approach can be used to investigate the uniqueness of the optimal solutions. The results show that uniqueness occurs for instances with a much larger number of nonzeros than guaranteed by theory.

Concerning the heuristics, Figure 2 shows a comparison of the results of all experiments with $m = 128$. It turns out that basis pursuit consistently is the worst heuristic. In total, Mangasarian's approach seems to be the best heuristic. GOMP improves the results of OMP, but is computationally too expensive for larger instances. OMP itself yields quite good results as well. The bilinear method is somewhat in between the other approaches; for random instances it works better than for Hadamard matrices. If deviations are allowed, postprocessing usually helps to reduce the number of nonzeros for all heuristics.

Finally, the computation times of the heuristics (except GOMP) are quite reasonable for the problem sizes we consider. This of course might change for larger practically relevant instances. OMP, which does not use linear programming, is in general faster than the other approaches and might be of use for these cases. But OMP is also a bit worse in terms of quality of solutions.

References

[1] T. Achterberg, *SCIP – A framework to integrate constraint and mixed integer programming*, Report 04–19, ZIB, 2004. `http://www.zib.de/Publications/abstracts/ZR-04-19/`.

[2] E. Amaldi, *On the complexity of designing compact perceptrons and some consequences*, in Electronic Proc. Fifth International Symposium on Artificial Intelligence and Mathematics, Fort Lauderdale, Florida, E. Boros and R. Greiner, eds., RUTCOR, 1999.

[3] E. Amaldi and V. Kann, *On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems*, Theor. Comput. Sci. **209**, no. 1–2 (1998), pp. 237–260.

[4] E. Amaldi, M. E. Pfetsch, and L. E. Trotter, Jr., *On the maximum feasible subsystem problem, IISs, and IIS-hypergraphs*, Math. Program. **95**, no. 3 (2003), pp. 533–554.

[5] K. Bennett and O. Mangasarian, *Bilinear separation of two sets in n-space*, Comput. Optim. Appl. **2** (1993), pp. 207–227.

[6] K. P. Bennett and E. J. Bredensteiner, *A parametric optimization method for maschine learning*, INFORMS J. Comput. **9**, no. 3 (1997), pp. 311–318.

[7] E. J. Candès and J. Romberg, *Quantitative robust uncertainty principles and optimally sparse decompositions*, Found. Comput. Math. **6**, no. 2 (2006), pp. 227–254.

[8] E. J. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Trans. Inform. Theory **52**, no. 2 (2006), pp. 489–509.

[9] E. J. Candès, J. Romberg, and T. Tao, *Stable signal recovery from incomplete and inaccurate measurements*, Comm. Pure Appl. Math. **59**, no. 8 (2006), pp. 1207–1223.

[10] E. J. Candés, M. Rudelson, T. Tao, and R. Vershynin, *Error correction via linear programming*, in Proc. 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS05), IEEE, 2005, pp. 295–308.

[11] E. J. Candès and T. Tao, *Decoding by linear programming*, IEEE Trans. Inform. Theory **51**, no. 12 (2005), pp. 4203–4215.

[12] S. S. Chen, *Basis Pursuit*, PhD thesis, Standford University, 1995.

[13] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decmposition by basis pursuit*, SIAM J. Sci. Comput. **20**, no. 1 (1999), pp. 33–61.

[14] G. Davis, S. Mallat, and Z. Zhang, *Adaptive time-frequency decompositions with matching pursuits*, Opt. Eng. **33**, no. 7 (1994), pp. 2183–2191.

[15] D. L. Donoho, *Compressed sensing*, IEEE Trans. Inform. Theory **52**, no. 4 (2006), pp. 1289–1306.

[16] D. L. Donoho, *For most large underdetermined systems of equations, the minimal ℓ_1-norm near-solution approximates the sparsest near-solution*, Comm. Pure Appl. Math. **59**, no. 7 (2006), pp. 907–934.

[17] D. L. Donoho, *For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution*, Comm. Pure Appl. Math. **59**, no. 6 (2006), pp. 797–829.

[18] D. L. Donoho and M. Elad, *Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization*, Proc. Natl. Acad. Sci. USA **100**, no. 5 (2003), pp. 2197–2202.

[19] D. L. Donoho and M. Elad, *On the stability of the basis pursuit in the presence of noise*, Signal Process. **86** (2006), pp. 511ŋ–532.

[20] D. L. Donoho, M. Elad, and V. Temlyakov, *Stable recovery of sparse overcomplete representations in the presence of noise*, IEEE Trans. Inform. Theory **52**, no. 1 (2006), pp. 6–18.

[21] D. L. Donoho and X. Huo, *Uncertainty principles and ideal atomic decomposition*, IEEE Trans. Inform. Theory **47**, no. 7 (2001), pp. 2845–2862.

[22] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, *Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit*, Tech. Report 2006-02, Standford, Department of Statistics, 2006.

[23] M. Elad, *Sparse representations are most likely to be the sparsest possible*, EURASIP J. Appl. Signal Process. **2006** (2006), pp. 1ŋ–12.

[24] M. Elad and A. M. Bruckstein, *A generalized uncertainty principle and sparse representation in pairs of bases*, IEEE Trans. Inform. Theory **48**, no. 9 (2002), pp. 2558–2567.

[25] J. J. Fuchs, *Recovery of exact sparse representations in the presence of bounded noise*, IEEE Trans. Inform. Theory **51**, no. 10 (2005).

[26] J.-J. FUCHS, *Recovery conditions of sparse representations in the presence of noise*, in Proc. ICASSP 2006, IEEE, 2006, pp. 337–340.
[27] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.
[28] S. MALLAT AND Z. ZHANG, *Matching pursuit in a time-frequency dictionary*, IEEE Trans. Inform. Theory **41** (1993), pp. 3397–3415.
[29] O. L. MANGASARIAN, *Minimum-support solutions of polyhedral concave programs*, Optimization **45** (1999), pp. 149–162.
[30] B. K. NATARAJAN, *Sparse approximate solutions to linear systems*, SIAM J. Comput. **24**, no. 2 (1995), pp. 227–234.
[31] Y. C. PATI, R. REZAIIFAR, AND P. S. KRISHNAPRASAD, *Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decompositions*, in Proc. 27th Asilomar Conference on Signals, Systems and Computers, A. Singh, ed., 1993.
[32] M. E. PFETSCH, *The Maximum Feasible Subsystem Problem and Vertex-Facet Incidence of Polyhedra*, PhD thesis, TU Berlin, 2002.
[33] M. E. PFETSCH, *Branch-and-cut for the maximum feasible subsystem problem*, ZIB-Report 05-46, Zuse Institute Berlin, 2005.
[34] A. SCHRIJVER, *Theory of linear and integer programming*, John Wiley & Sons, Chichester, 1986. Reprint 1998.
[35] T. STROHMER AND J. ROBERT W. HEATH, *Grassmannian frames with applications to coding and communication*, Appl. Comput. Harmon. Anal. **14**, no. 3 (2003), pp. 257–275.
[36] J. TROPP AND A. C. GILBERT, *Signal recovery from partial information via orthogonal matching pursuit*. Preprint, Apr. 2005.
[37] J. A. TROPP, *Greed is good: Algorithmic results for sparse approximations*, IEEE Trans. Inform. Theory **50**, no. 10 (2004), pp. 2231–2242.
[38] J. A. TROPP, *Just relax: Convex programming methods for identifying sparse signals in noise*, IEEE Trans. Inform. Theory **52**, no. 3 (2006), pp. 1030–1051.
[39] G. YARMISH, *Wavelet decomposition via the standard tableau simplex method of linear programming*. Preprint, available at `www.optimization-online.org`, 2006.

SADEGH JOKAR, INSITUTE OF MATHEMATICS, TU BERLIN, STRASSE DES 17. JUNI 136, 10623 BERLIN, GERMANY
E-mail address: `jokar@math.tu-berlin.de`

MARC E. PFETSCH, ZUSE INSTITUTE BERLIN, TAKUSTR. 7, 14195 BERLIN, GERMANY
E-mail address: `pfetsch@zib.de`