



# 1 Practical performance of Random Projections in 2 Linear Programming

3 Leo Liberti   

4 LIX CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, 91128 Palaiseau, France

5 Benedetto Manca  

6 Dip. Matematica e Informatica, Università degli Studi di Cagliari, Via Ospedale 72, 09124 Cagliari,  
7 Italy

8 Pierre-Louis Poirion  

9 RIKEN Center for Advanced Intelligence Project, Tokyo, Japan

## 10 — Abstract —

11 The use of random projections in mathematical programming allows standard solution algorithms  
12 to solve instances of much larger sizes, at least approximately. Approximation results have been  
13 derived in the relevant literature for many specific problems, as well as for several mathematical  
14 programming subclasses. Despite the theoretical developments, it is not always clear that random  
15 projections are actually useful in solving mathematical programs in practice. In this paper we  
16 provide a computational assessment of the application of random projections to linear programming.

17 **2012 ACM Subject Classification** Mathematics of computing → Combinatorial optimization/Prob-  
18 abilistic algorithms

19 **Keywords and phrases** Linear Programming, Johnson-Lindenstrauss Lemma, Computational testing

20 **Digital Object Identifier** 10.4230/LIPIcs.SEA.2022.21

21 **Category** contributed paper

22 **Funding** *Benedetto Manca*: Partly supported by grant STAGE, Fondazione Sardegna 2018.

## 23 **1** Introduction

24 This paper is about applying Random Projections (RP) to Linear Programming (LP)  
25 formulations. RPs are dimensional reduction operators that usually apply to data. The  
26 point of applying RPs to LPs is to obtain an approximate solution of the high-dimensional  
27 formulation by solving a related lower-dimensional one. The main goal of this paper is to  
28 discuss the pros and cons of this technique from a computational (practical) point of view.

### 29 **1.1** Random Projections

30 In general, RPs are functions, sampled randomly from certain distributions, that map a  
31 vector in  $\mathbb{R}^m$  to one in  $\mathbb{R}^k$ , where  $k \ll m$ . In this paper we restrict our attention to linear  
32 RPs, which are  $k \times m$  random matrices  $T$ . The most famous result about RPs is the  
33 Johnson-Lindenstrauss Lemma [12], which we recall here in its probabilistic form. Given a  
34 finite set  $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^m$  and an  $\epsilon \in (0, 1)$ , there exists a  $\delta = O(e^{-\mathcal{C}\phi(k)})$  (with  $\phi$   
35 usually linear and  $\mathcal{C}$  a universal constant not depending on input data) and an RP  $T$  with  
36  $k = O(\epsilon^{-2} \ln n)$  such that

$$37 \quad \mathbf{Prob}(\forall i < j \leq n (1 - \epsilon)\|x_i - x_j\|_2 \leq \|Tx_i - Tx_j\|_2 \leq (1 + \epsilon)\|x_i - x_j\|_2) \geq 1 - \delta. \quad (1)$$

38 If  $T$  is sampled componentwise from the normal distribution  $N(0, 1/\sqrt{k})$ , Eq. (1) holds (note  
39 that other distributions also work). The JLL is not the only result worth mentioning in RP  
40 [20, 10, 17], but it is the object of interest in this paper.



© Leo Liberti and Benedetto Manca and Pierre-Louis Poirion;  
licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Experimental Algorithms (SEA 2022).

Editors: Christian Schulz and Bora Uçar; Article No. 21; pp. 21:1–21:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

41 The JLL directly applies to all problems involving the Euclidean distance between points  
 42 in a Euclidean space of high dimension, e.g. the design of an efficient nearest-neighbor data  
 43 structure (i.e. given  $X \subset \mathbb{R}^m$  and  $q \in \mathbb{R}^m$  quickly return  $x \in X$  closest to  $q$ ) [11].

44 More in general, the JLL shows that RPs can transform the point set  $X$  to a lower  
 45 dimensional set  $TX$  such that  $X$  and  $TX$  are “approximately congruent”: the pairwise  
 46 distances in  $X$  are approximately the same (multiplicatively) as the corresponding pairwise  
 47 distances in  $TX$ , even if  $X$  has  $m$  dimensions and  $TX$  only  $k$  (proportional to  $\epsilon^{-2} \ln |X|$ ).  
 48 Since “approximately congruence” means “almost the same, aside from translations, rotations,  
 49 and reflections”, it is reasonable to hope that RPs might apply to other constructs than just  
 50 sets of points, and still deliver a theoretically quantifiable approximation. In this paper we  
 51 consider LP.

## 52 1.2 Applying RPs to Linear Programming

53 In this paper we are interested in the application of the JLL to LP in standard form:

$$54 \quad \left. \begin{array}{l} \min_x \quad c^\top x \\ Ax = b \\ x \geq 0, \end{array} \right\} \text{ (LP)}$$

55 where  $x = (x_1, \dots, x_n)$ ,  $A$  is an  $m \times n$  matrix, and  $b \in \mathbb{R}^m$ .

56 There are several issues in applying RPs to Mathematical Programs (MP) in general.  
 57 The three foremost are:

- 58 1. RPs project vectors rather than decision variables and constraint functions;
- 59 2. RPs ensure approximate congruence of the input vectors in the lower-dimensional output:  
 60 but approximation arguments in LP must instead be based on optimality and feasibility  
 61 (unrelated to the  $\ell_2$  norm);
- 62 3. RPs only apply to finite point sets, whereas LP decision variables represent infinite point  
 63 sets.

64 These issues pose nontrivial theoretical challenges, and the proof techniques vary consid-  
 65 erably depending on the MP subclass being considered. The first issue mentioned above is  
 66 addressed by applying RPs to the problem parameters (the input data); in the LP case, we  
 67 project the linear system  $Ax = b$ . We speak of the *original* formulation  $P$  and the *projected*  
 68 formulation  $TP$ . This yields a fourth issue: the solution of  $TP$  may be infeasible in  $P$ : in  
 69 such cases, a *solution retrieval* phase is necessary in order to construct a feasible solution of  
 70  $P$  from that of  $TP$ .

71 The second and third issues are addressed in [23], leading to statements similar to the  
 72 JLL, but concerning approximate LP feasibility and optimality. If  $E(P, T)$  is a statement  
 73 about the feasibility or optimality error between the LP formulations  $P$  and  $TP$ , the general  
 74 structure of these results is similar to the probabilistic version of the JLL:

$$75 \quad \mathbf{Prob}(E(P, T)) \geq 1 - \delta, \tag{2}$$

76 where  $\delta$  usually depends on  $\epsilon$ ,  $k$  and possibly even the solution of  $P$ . We shall recall the  
 77 statements of these results more precisely in Sect. 2.

## 78 1.3 Relevant literature

79 The main reference for RPs and LP in standard form is [23], which presents the theory  
 80 addressing the above issues, and a computational study focussing on dense random LP

81 instances. RPs were also applied to some specific LP problems: PAC learning [18] and  
 82 quantile regression [24], with dimensional reduction techniques tailored to the corresponding  
 83 LP structure. Other works in applying RPs to different types of MP subclasses are [22]  
 84 (quadratic programs with a ball constraint), [3] (general quadratic programs), [16] (conic  
 85 programs including second-order cone and semidefinite programs).

## 86 1.4 Contributions of this paper

87 Although some of the relevant literature carries computational results, we think that, compu-  
 88 tationally, the application of RPs to LPs is still experimental: in practice the output on a  
 89 given instance can range from accurate all the way to catastrophic.

90 One of the difficulties is that, in writing  $k = O(\epsilon^{-2} \ln n)$ , we are neglecting a constant  
 91 multiplicative coefficient  $C$  related to the “big oh”, the appropriate value of which is usually  
 92 the fruit of guesswork. Another difficulty is that the theoretical results in this area apply  
 93 to “high dimensions”, without specifying a minimum dimension above which they hold. In  
 94 catastrophic cases, the theory ensures that results would improve for larger instance sizes,  
 95 but just how large is unknown. At this time, in our opinion, no-one is able to justifiably  
 96 foresee whether RPs will be useful or not on a given LP instance. The only existing work  
 97 about practical RP usage is [21], which only focusses on computational testing of different  
 98 RP matrices.

99 This paper will provide a computational analysis of LP cases where RPs work reasonably  
 100 well, and others where they do not, and attempt to derive some guidelines for choosing  
 101 appropriate values for the most critical unknown parameters. On the theoretical side, we  
 102 tighten two results of [16] when applied to the LP case.

103 The rest of this paper is organized as follows. In Sect. 2 we recall the main theoretical  
 104 results relative to the application of RPs to LP, and state the two new tightened results. In  
 105 Sect. 3 we illustrate the benchmark goal, the LP structures we test, and the methodology. In  
 106 Sect. 4 we discuss the benchmark results.

## 107 2 Summary of theoretical results

108 We apply RPs to the original formulation (LP) by reducing the number  $m$  of constraints.  
 109 Let  $T$  be a  $k \times m$  RP matrix. The projected formulation is:

$$110 \quad \min\{c^\top x \mid TAx = Tb \wedge x \geq 0\} \quad (TLP).$$

111 We first discuss feasibility. We note that the geometric interpretation of the feasible  
 112 set  $F = \{x \mid Ax = b \wedge x \geq 0\}$  of (LP) is that  $F$  is the set of conic combinations of the  
 113 columns  $A^j$  of  $A$ , i.e.  $F = \text{cone}(A)$ . We also let  $\text{conv}(A)$  the convex hull of the columns of  
 114  $A$ , and  $\|x\|_A = \min\{\sum_j \lambda_j \mid x = \sum_j \lambda_j A^j\}$  be the  $A$ -norm of  $x \in \text{cone}(A)$ . Is  $F$  invariant  
 115 w.r.t. the application of  $T$  to (LP)? If  $x \in F$  then  $TAx = Tb$  by linearity of  $T$ . On the  
 116 other hand, it is generally false that if  $x \geq 0$  but  $x \notin F$ , then  $TAx \neq Tb$ . The following  
 117 approximate feasibility statement

$$118 \quad b \notin \text{cone}(A) \Rightarrow \mathbf{Prob}(Tb \notin \text{cone}TA) \geq 1 - 2(n+1)(n+2)e^{-\mathcal{C}(\epsilon^2 - \epsilon^3)k} \quad (3)$$

119 is proved in [23, Thm. 3] for all  $\epsilon \in (0, \Delta^2/(\mu_A + 2\mu_A\sqrt{1 - \Delta^2} + 1))$ , where  $\mathcal{C}$  is the universal  
 120 constant of the JLL,  $\mu_A = \max\{\|x\|_A \mid x \in \text{cone}(A) \wedge \|x\|_2 \leq 1\}$ , and  $\Delta$  is a lower bound to  
 121  $\min_{x \in \text{conv}(A)} \|b - x\|_2$ .

122 Let  $\text{val}(\cdot)$  indicate the optimal objective function value of a MP formulation. The  
 123 approximate optimality statement for (LP) derived in [23, Thm. 4] is conditional to the LP

## 21:4 Practical random projections for LP

124 formulation being feasible and bounded, so that, if  $x^*$  is an optimal solution, there is  $\theta$   
 125 (assumed w.l.o.g.  $\geq 1$ ) such that  $\sum_j x_j^* < \theta$ . Given  $\gamma \in (0, \text{val}(\text{LP}))$ ,

$$126 \quad \mathbf{Prob}(\text{val}(\text{LP}) - \gamma \leq \text{val}(\text{TLP}) \leq \text{val}(\text{LP})) \geq 1 - \delta, \quad (4)$$

127 where  $\delta = 4ne^{-\mathcal{C}(\epsilon^2 - \epsilon^3)^k}$ ,  $\epsilon = O(\gamma/(\theta^2 \|y^*\|_2))$ , and  $y^*$  is an optimal dual solution of (LP).  
 128 Like other approximate optimality results in this field, some quantities in the probabilistic  
 129 statement depend on the norm of a dual optimal solution. This adds a further difficulty to  
 130 computational evaluations, since they cannot be computed prior to solving the problem.

131 Let  $\bar{x}$  be a *projected solution*, i.e. an optimal solution of the projected formulation. In [23,  
 132 Prop. 3], it is proved that  $\bar{x}$  is feasible in the original formulation with zero probability. We  
 133 therefore need to provide a solution retrieval method. A couple were proposed in [23], but  
 134 the one found in [16, Eq. (6)] comes with an approximation guarantee and a good practical  
 135 performance. The retrieved solution  $\tilde{x}$  is defined as the projection of  $\bar{x}$  on the affine subspace  
 136  $Ax = b$ , and computed using the pseudoinverse:

$$137 \quad \tilde{x} = \bar{x} + A^\top(AA^\top)^{-1}(b - A\bar{x}). \quad (5)$$

138 The fact that we only project on  $Ax = b$  without enforcing  $x \geq 0$  is necessary, since otherwise  
 139 we would need to solve the whole high-dimensional LP. On the other hand, it causes potential  
 140 infeasibility errors w.r.t.  $x \geq 0$ . A probabilistic bound on this error is cast in general terms  
 141 for conic programs in [23]. Let  $\kappa(A)$  be the condition number of  $A$ ; applying [23, Thm. 4.4] to  
 142 LP, we obtain the following result, which bound the (negativity of) the smallest component  
 143 of  $\tilde{x}$  in terms of that of  $\bar{x}$ .

144 ► **Proposition 1.** *There is a universal constant  $\mathcal{C}_2$  such that, for any  $u \geq 0$ , we have:*

$$145 \quad \mathbf{Prob}\left(\min_{j \leq n} \tilde{x}_j \geq \min_{j \leq n} \bar{x}_j - \epsilon \theta \kappa(A) (\mathcal{C}_2 + u \sqrt{2/\ln(n)})\right) \geq 1 - 2e^{-u^2}.$$

146 The proof is based on an improvement of [16, Eq. (7)] based on computing the Gaussian  
 147 width and diameter of  $\{x \geq 0 \mid \langle \mathbf{1}, x \rangle \leq 1\}$ . As a corollary, we also have the following  
 148 result about the difference between objective function values of the retrieved and projected  
 149 solutions.

150 ► **Corollary 2.** *Let  $\tilde{f}$  be the objective function value of the retrieved solution  $\tilde{x}$ , and  $\bar{f}$  be the  
 151 optimal objective function value of the projected formulation. There is a universal constant  
 152  $\mathcal{C}_2$  such that, for any  $u \geq 0$ , we have:*

$$153 \quad \mathbf{Prob}(|\tilde{f} - \bar{f}| \leq \epsilon \theta \kappa(A) \|c\|_2 (\mathcal{C}_2 + u \sqrt{2/\ln(n)})) \geq 1 - 2e^{-u^2}.$$

### 154 **3** What we establish and how

155 Upon receiving an LP instance to be solved using RPs, one has to at least know how to decide  
 156  $k$  (the projected dimension) so that the solution of the projected formulation is reasonably  
 157 close to that of the original one.

158 Ideally, one would like to estimate all unknown parameters:  $k, \epsilon, \mathcal{C}, \mathcal{C}$  in function of  $\gamma$   
 159 and  $\delta$ . This is theoretically hopeless because the theoretical bounds derived for “all LPs”  
 160 are far from tight. We shall see below that it is also computationally hopeless. In practice,  
 161 moreover, one might be much more interested in finding a good retrieved solution (i.e. almost  
 162 feasible in the original problem), rather than finding a good approximation to the optimal

163 objective function value, since a feasible solution can be improved by local methods, while  
 164 an approximate optimal value may at best be useful as an objective cut.

165 Our approach will accordingly be based on solving sets of uniformly sampled LP instances  
 166 (from different applications) using a standard solver, and analyse the output in terms of how  
 167 the feasibility and optimality errors of the retrieved solution vary with problem size and  $\epsilon$ .

### 168 3.1 The RP matrix

169 All componentwise sampled sub-Gaussian distributions [6] can be used to ensure the results  
 170 cited in this paper. Some sparse variants also exist, along the lines of [1, 14]. We use the  
 171 sparse RPs described in [3, §5.1]. For a given density  $\sigma \in (0, 1)$  and standard deviation  
 172  $\sqrt{1/(k\sigma)}$ , with probability  $\sigma$  we sample a component of the  $k \times m$  RP  $T$  from the distribution  
 173  $N(0, \sqrt{1/(k\sigma)})$ , and set it to zero with probability  $1 - \sigma$ . In our computational study, we set  
 174  $\sigma = d_A/2$ , where  $d_A$  is the density of the constraint matrix  $A$ .

### 175 3.2 LP structures

176 We consider randomly generated LPs of the following four classes: MAX FLOW problems [7],  
 177 DIET problems [5], QUANTILE REGRESSION problems [15], and BASIS PURSUIT problems from  
 178 sparse coding [2]. This choice yields a set of LP problems going from extremely sparse (MAX  
 179 FLOW) to completely dense (BASIS PURSUIT), with the DIET and QUANTILE REGRESSION  
 180 providing cases of various intermediate densities. These four test cases arise from a diverse  
 181 range of application settings: combinatorial optimization, continuous optimization, statistics,  
 182 data science.

#### 183 3.2.1 Maximum flow

184 The MAX FLOW formulation is defined on a weighted digraph  $G = (N, \mathcal{A}, u)$  with a source  
 185 node  $s \in N$ , a target node  $t \in N$  (with  $s \neq t$ ) and  $u : \mathcal{A} \rightarrow \mathbb{R}_+$ , as follows:

$$\left. \begin{array}{l}
 \max_{x \in \mathbb{R}_+^{|\mathcal{A}|}} \quad \sum_{\substack{i \in N \setminus \{s\} \\ (s,i) \in \mathcal{A}}} x_{si} - \sum_{\substack{i \in N \setminus \{s\} \\ (i,s) \in \mathcal{A}}} x_{is} \\
 \forall i \in N \setminus \{s, t\} \quad \sum_{\substack{j \in N \\ (i,j) \in \mathcal{A}}} x_{ij} = \sum_{\substack{j \in N \\ (j,i) \in \mathcal{A}}} x_{ji} \\
 \forall (i, j) \in \mathcal{A} \quad 0 \leq x_{ij} \leq u_{ij}.
 \end{array} \right\} \quad (\text{MF})$$

187 We generate random weighted digraphs  $G = (N, \mathcal{A}, u)$  with the property that a single  
 188 (randomly chosen) node  $s$  is connected (through paths) to all of the other nodes: we first  
 189 generate a random tree on  $N \setminus \{t\}$ , orient it so that  $s$  is the root, add a node  $t$  with the same  
 190 indegree as the outdegree of  $s$ , and then proceed to enrich this digraph with arcs generated at  
 191 random using the Erdős-Renyi model with probability 0.05. We then generate the capacities  
 192  $u$  uniformly from  $[0, 1]$ . Finally, we compute the digraph's incidence matrix  $A$ , which has  
 193  $m = |N| - 2$  rows and  $|\mathcal{A}|$  columns. Instances are feasible because the graph always has a  
 194 path from  $s$  to  $t$  by construction, and the zero flow is always feasible.

195 Although (MF) is an LP, it is not in standard form, because of the upper bounding  
 196 constraints  $x \leq u$ . But, by [23, §4.2], we can devise a block-structured RP matrix that only  
 197 projects the equations  $Ax = b$ , leaving the inequalities  $x \leq u$  alone. In this case,  $A$  is a flow  
 198 matrix with two nonzeros per column, one set to 1 the other to  $-1$ , aside from columns  
 199 referring to source and target nodes  $s, t$  that only have one nonzero; and  $b = 0$ . The density  
 200 of  $A$  is  $d_A = \frac{2|\mathcal{A}| - 2}{(m-2)|\mathcal{A}|} \approx 2/m$ .

201 For our random (MF) instances,  $\theta = |\mathcal{A}|$  is a valid upper bound to  $\sum_{(i,j) \in \mathcal{A}} x_{ij}^*$ , since  
 202  $0 \leq x_{ij} \leq u_{ij} \leq 1$  for all  $(i, j) \in \mathcal{A}$ .

### 203 3.2.2 Diet problem

204 The DIET formulation is defined on an  $m \times n$  nutrient-food matrix  $D$ , a food cost vector  
 205  $c \in \mathbb{R}_+^n$ , and a nutrient requirement vector  $b \in \mathbb{R}^m$ , as follows:

$$206 \left. \begin{array}{l} \min_{q \in \mathbb{R}_+^{mn}} c^\top q \\ Dq \geq b. \end{array} \right\} \text{ (DP)}$$

207 We sample  $c, D, b$  uniformly componentwise in  $[0, 1]$ , and set the density of  $D$  to  $d_D = 0.5$ .  
 208 Instances are feasible because one can always buy enough food to satisfy all nutrient  
 209 requirements. If  $\|D_i\|_0 = |\text{nonzeros of row } D_i|$ , then  $\hat{q} = (\max_{i \leq m} (b_i / (\|D_i\|_0 D_{ij})) \mid j \leq n)$  is a  
 210 feasible solution.

211 Again, (DP) is not in standard form, but the transformation is immediate using slack  
 212 variables  $r_i \geq 0$  for  $i \leq m$ . We let  $A = (D \mid -I)$ , where  $I$  is  $m \times m$ . The decision variable  
 213 vector is  $x = (q, r)$ . The density of  $A$  is  $d_A = (d_D mn + m) / (m(n + m)) = (d_D n + 1) / (n + m)$ .

214 For (DP), the upper bounding solution  $\hat{q}$  yields slack values  $\hat{r}_i = D_i \hat{q} - b_i$  for all  $i \leq m$ ,  
 215 where  $D_i$  is the  $i$ -th row of  $D$ . So we let  $\theta = \sum_j \hat{q}_j + \sum_i \hat{r}_i$  be an upper bound for  $\sum_j x_j^*$ .

### 216 3.2.3 Quantile regression

217 The QUANTILE REGRESSION formulation, for a quantile  $\tau \in (0, 1)$ , is defined over a database  
 218 table  $D$  having density  $d_D$  with  $m$  records and  $p$  fields, and a further column field  $b$ . We  
 219 make a statistical hypothesis  $b = \sum_j \beta_j D^j$ , and aim at estimating  $\beta = (\beta_j \mid j \leq p)$  from the  
 220 data  $b, D$  so that errors from the  $\tau$ -quantile are minimized. Instances may only have nonzero  
 221 optimal value if  $m > p$ , as is clear from the constraints of the formulation below:

$$222 \left. \begin{array}{l} \min_{\substack{\beta \in \mathbb{R}^p \\ u^+, u^- \in \mathbb{R}_+^m}} \tau \mathbf{1}^\top u^+ + (1 - \tau) \mathbf{1}^\top u^- \\ D\beta + Iu^+ - Iu^- = b, \end{array} \right\} \text{ (QR)}$$

223 where the constraint system  $Ax = b$  has  $A = (D \mid I \mid -I)$ ,  $x = (\beta, u^+, u^-)$ , and  $\tau$  (the quantile  
 224 level) is given, and fixed at 0.2 in our experiments. The data matrix  $(D, b)$  is sampled  
 225 uniformly componentwise from  $[-1, 1]$ , with  $d_D = 0.8$ . Instances are all feasible because  
 226 the problem reduces to solving the overconstrained linear system  $D\beta = b$  with a “skewed”  
 227 version of an  $\ell_1$  error function.

228 We note that (QR) is not in standard form, since the components of  $\beta$  are unconstrained;  
 229 but this is not an issue, insofar as the problem is bounded (since it is feasible and it minimizes  
 230 a weighted sum of non-negative variables), and this is enough to have the results in [23] hold.  
 231 On the contrary, the lack of non-negative bounds on  $\beta$  is an advantage, since we need not  
 232 worry about negativity errors in the  $\beta$  components of the retrieved solution (Prop. 1). The  
 233 density of  $A$  is  $d_A = (d_D mp + 2m) / (mp + 2m^2) = (d_D p + 2) / (p + 2m)$ .

234 For (QR), given that all data is sampled uniformly from  $[-1, 1]$ , no optimum can ever  
 235 have  $|\beta_j| > 1$ . As for  $u^+, u^-$ , we note that any feasible  $\beta$  yields an upper bound to the  
 236 optimal objective function value, which only depends on  $u^+, u^-$ : we can therefore choose  
 237  $\beta = 0$ , and obtain  $u_i^+ - u_i^- = b_i$  for all  $i \leq m$ ; we then let  $u_i^+ = b_i \wedge u_i^- = 0$  if  $b_i > 0$ , and  
 238  $u_i^+ = 0 \wedge u_i^- = -b_i$  otherwise. This yields an upper bound estimate  $\theta = p + \sum_i |b_i|$  to  $\sum_j x_j^*$ .

### 3.2.4 Basis pursuit

The BASIS PURSUIT formulation aims at finding the sparsest vector  $x$  satisfying the underdetermined linear system  $Ax = b$  by resorting to a well-known approximation of the zero-norm by the  $\ell_1$  norm [2]:

$$\left. \begin{array}{l} \min_{x, s \in \mathbb{R}^n} \quad \mathbf{1}^\top s \\ Ax = b \\ \forall j \leq n \quad -s_j \leq x_j \leq s_j. \end{array} \right\} \text{ (BP)}$$

According to sparse coding theory [4], we work with a fully dense  $m \times n$  matrix  $A$  sampled componentwise from  $\mathbf{N}(0, 1)$  (with density  $d_A = 1$ ), a random message obtained as  $z/Z$  from a sparse  $z \in (\mathbb{Z} \cap [-Z, Z])^n$  (with density 0.2) and  $Z = 10$ , and compute the encoded message  $b = Az$ . We then solve (BP) in order to recover the sparsest solution of the underconstrained system  $Ax = b$ , which should provide an approximation of  $z$ .

Similarly to (MF), in (BP) we can partition the constraints into equations  $Ax = b$  and inequalities  $-s \leq x \leq s$ . Again by [23, §4.2], we devise a block-structured RP matrix which only projects the equations.

As in Sect. 3.2.3, (BP) is not in standard form, since none of the variables are non-negative. In this case, moreover, it is not easy to establish a bound  $\theta$  on  $\sum_j (x_j^* + s_j^*)$ , since  $A$  is sampled from a normal distribution. On the other hand, for  $A_{ij} \sim \mathbf{N}(0, 1)$  we have  $\mathbf{Prob}(A_{ij} \in [-3, 3]) = 0.997$ . By construction, we have  $b \in [-3n, 3n]^m$ , which implies a defining interval  $[-n, n]$  on the components of optimal solutions, yielding  $\theta = 2n^2$  with probability 0.997.

## 3.3 Methodology

The goal of this paper is to provide a computational assessment of RPs applied to LP.

As discussed at the beginning of Sect. 3, the actual determination of all relevant parameters is theoretically hopeless. We can certainly simplify the task a little by noting that the coefficient  $C$  can be removed since it suffices to decide a value for  $\epsilon$  in order to decide  $k$ . Ideally we would like to decide  $\gamma$  first (see Eq. (4)), then compute  $\epsilon$  as  $O(\gamma/(\theta^2 \|y^*\|_2))$ , and sample an appropriate RP. Unfortunately, estimating  $\theta$  and  $\|y^*\|_2$  prior to solving the original LP leads to tiny values for  $\epsilon$  (e.g.  $10^{-i}$  for  $i \in \{2, \dots, 11\}$  in some preliminary tests), which would require the rows of  $A$  to be at least  $O(10^{i^2})$  in order to yield a useful projection. Since we are interested in applying RPs to LPs with  $O(10^2)$  and  $O(10^3)$  rows, this “ideal” approach is inapplicable.

Instead, we repeatedly solve sets of instances of each LP structure. Each projected instance is solved with different values of  $\epsilon \in \mathcal{E} = \{0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$  (these values have been found to be the most relevant in preliminary computational experiments performed over several years). Moreover, to mitigate the effect of randomness, we solve each instance with each  $\epsilon$  multiple times. For each instance and  $\epsilon$  we collect performance measures on objective function values, infeasibility errors, and CPU time. This allows us to illustrate the co-variability of  $\epsilon$  and instance size with the performance measures.

## 4 The benchmark

The solution pipeline is based on Python 3 [19] and the libraries `scipy` [13] and `amplpy` [8] (besides other standard python libraries). For each problem type, we loop over instances (based on row size of the equality constraint system, varying in  $\mathcal{S}$ , see below), over  $\epsilon \in \mathcal{E}$ ,



280 and over 5 different runs for each instance and  $\epsilon$  in order to amortize the result randomness  
 281 depending on the choice of  $T$ . We solve all of the original and projected instances using  
 282 CPLEX 20.1 [9]. We use the barrier solver, because we found this to be more efficient with  
 283 large dense LPs than the simplex-based solvers in CPLEX. Our code can be downloaded  
 284 here.<sup>1</sup> All tests have been carried out on a MacBook 2017 with a 1.4GHz dual-core Intel Core  
 285 i7 with 16GB RAM.

## 286 4.1 Choice of instances

287 In the case of DIET, QUANTILE REGRESSION, and BASIS PURSUIT, we generated instances  
 288 so that the number of rows of the equality constraint system  $Ax = b$  is in the set  $\mathcal{S} =$   
 289  $\{500p \mid 1 \leq p \leq 5 \wedge p \in \mathbb{N}\}$ . For MAX FLOW we used  $\mathcal{S}' = \mathcal{S} \setminus \{2500\}$  because the larger  
 290 size triggered a RAM-related error in a part of the solution pipeline involving the AMPL [8]  
 291 interpreter.

### 292 4.1.1 The variable space

293 The space of original, projected, and retrieved variable values is identical for MAX FLOW,  
 294 QUANTILE REGRESSION, and BASIS PURSUIT, since these three structures are originally cast  
 295 in an equality constraint form  $Ax = b$ . This desirable property fails to hold for DIET, which  
 296 deserves a separate discussion.

297 The original formulation (DP) of DIET is in inequality form  $Dq \geq b$ , but the projected  
 298 formulation is derived from the constraints  $Ax = b$  in standard form, where  $A = (D \mid -I)$ .

299 The theoretical results in Sect. 2 justify a fair comparison only between original and  
 300 projected solutions in standard form. Since this paper is about a *practical* comparison,  
 301 however, and since no-one would convert (DP) to standard form before solving it (because the  
 302 solver would do it as needed), we chose to compute objective function values and feasibility  
 303 errors of the projected formulation on the space of the original formulation variables  $q$ . Thus,  
 304 for a retrieved solution  $\tilde{x} = (\tilde{q}, \tilde{r})$  we only considered  $\tilde{q}$  in order to compute the objective  
 305 function value of  $\tilde{x}$ .

306 Considering only the  $q$  variables is unproblematic if applied to the optimal solution  $x^*$   
 307 of the original formulation in standard form, because  $s^* \geq 0$  and  $A = (D \mid -I)$  ensure that  
 308  $q^*$  is a feasible solution in  $Dq \geq b$ . When applied to the projected formulation, however,  
 309  $TA = (TD \mid -TI)$  yields a block matrix  $TI$  with both positive and negative entries (since  $T$   
 310 is sampled from a normal distribution). Thus, it often happens that the underdetermined  
 311  $k \times m$  system  $TI = Tb$  has solutions. In this case, since the objective tends to minimize  $c^\top q$ ,  
 312 the projected solution  $\tilde{x} = (\tilde{q}, \tilde{s})$  will have  $\tilde{q} = 0$ , yielding zero projected objective function  
 313 value. This, in turn, may yield  $D\tilde{q} \not\geq b$ . The application of RPs to DIET is therefore less  
 314 successful than for other structures.

## 315 4.2 Performance measures

316 At the end of each solver call we record: the optimal objective function  $f^*$  of the original  
 317 problem, the optimal objective function  $\tilde{f}$  of the projected problem, the objective function  
 318 value  $\tilde{f}$  of the retrieved solution  $\tilde{x}$ , the feasibility error w.r.t. equation constraints  $Ax = b$   
 319 (eq) and inequalities  $x \geq 0$  (in), the CPU time  $t^*$  taken to solve the original formulation,  
 320 and the CPU time  $\tilde{t}$  taken to solve the projected formulation.

---

<sup>1</sup> The URL is [https://mega.nz/file/p8MQhbpT#0TJBUVgaBf4KPVk2fu\\_5k05cMy2VozJk-OfQ1PzdJ0U](https://mega.nz/file/p8MQhbpT#0TJBUVgaBf4KPVk2fu_5k05cMy2VozJk-OfQ1PzdJ0U).



321 The CPU time  $t^*$  takes into account: reading the instance, constructing the original  
 322 formulation, and solving it. The CPU time  $\bar{t}$  takes into account: reading the instance,  
 323 sampling the RP, projecting the instance data, constructing the projected formulation,  
 324 solving it, and performing solution retrieval.

325 The benchmark considers: the average objective function ratios  $\bar{f}/f^*$ ,  $\tilde{f}/f^*$ , the average  
 326 errors `avgeq`, `avgin` for  $Ax = b$  and  $x \geq 0$ , the ratio  $k/m$ , the average CPU ratio  $\bar{t}/t^*$ : all  
 327 averages are computed over 5 solution runs over a given instance size and  $\epsilon$  value.

### 328 4.3 RP performance on Max Flow

329 The application of RPs to the MAX FLOW problem looks like a success story: the ratio  
 330 of projected to original optimal objective function value is very close to 1.0 and constant  
 331 w.r.t.  $\epsilon$  ( $\bar{f}/f^* \geq 1$  is normal insofar as MAX FLOW is a maximization problem, and TLP is  
 332 a relaxation of LP). The feasibility error of the retrieved solution related to the equality  
 333 constraints  $Ax = b$  is very close to zero, and the error w.r.t.  $x \geq 0$  decreases as  $m$  increases  
 334 (a healthy behaviour in RPs) and also as  $\epsilon$  increases (implying that maximum negativity  
 335 error increases more slowly than the number of variables). The CPU time ratio decreases  
 336 proportionally to  $k/m$ , as expected. The only issue is that the objective function value at  
 337 the retrieved solution is only around 0.5 of the optimum.

### 338 4.4 RP performance on Diet

339 As mentioned in Sect. 4.1.1, the practical application of RPs to the DIET problem is not  
 340 successful, as shown by the plots in Fig. 2. The projected cost is almost always zero, because  
 341 the constraint projection allowed the solver to satisfy  $(D|-I)(q,r)^\top = b$  using slack variables  
 342 only. This causes sizable errors in the retrieved solutions. As expected, the CPU time  
 343 taken to solve the projected formulation is a tiny fraction of the time to solve the original  
 344 formulation.

345 We tried to experiment with a modified projected objective ( $c|\mathbf{1}$ ) so that we would  
 346 minimize the sum of the projected slack variables. This yielded quantitatively better results,  
 347 as shown in Fig. 3; qualitatively, the results still look like a failure.

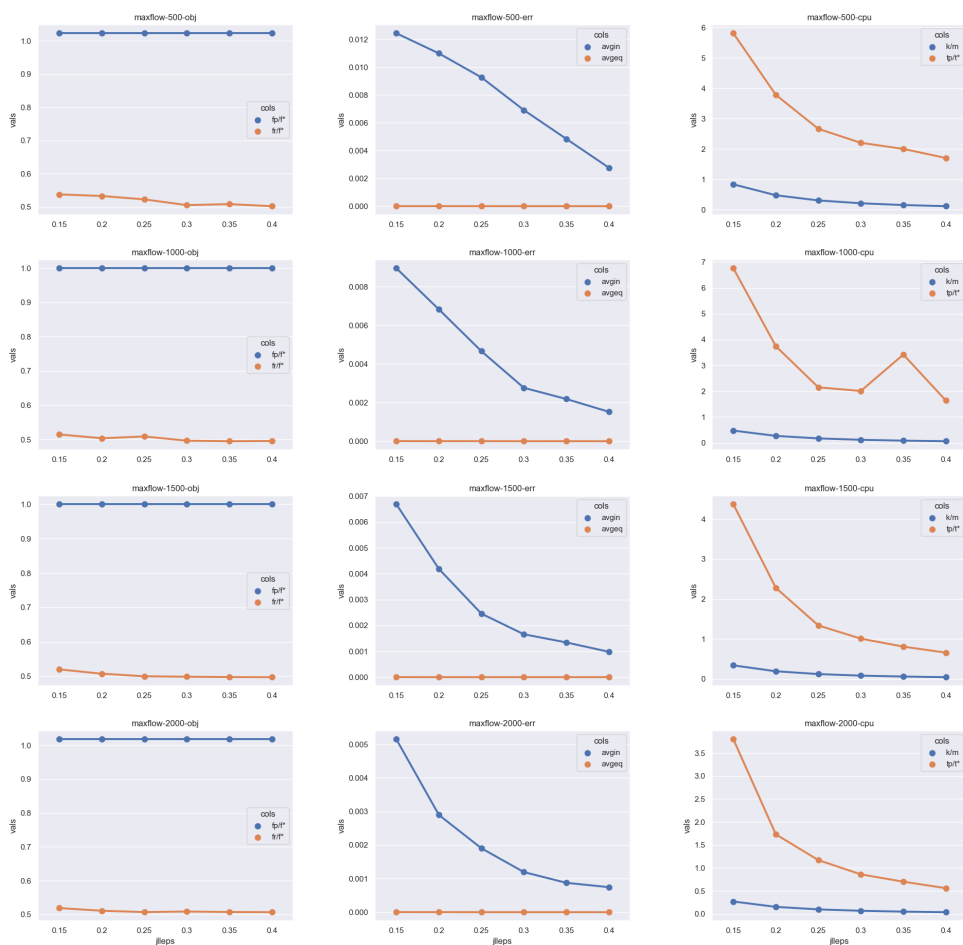
### 348 4.5 RP performance on Quantile Regression

349 The results quality on QUANTILE REGRESSION is mixed. The ratio  $\bar{f}/f^*$  is rather low, but  
 350 we note that it is higher (better) for low sizes and low  $\epsilon$  values, which is a sign that  $\epsilon$  should  
 351 be further decreased for all (and specially large) sizes. Interestingly, the objective value of  
 352 the retrieved solution  $\tilde{x}$  has better quality. The feasibility errors of  $\tilde{x}$  are zero for  $Ax = b$ ,  
 353 and not negligible (around 0.2, with one outlier) for  $x \geq 0$ : the trend, unfortunately, is not  
 354 decreasing, either with  $\epsilon$  or  $m$  increasing. CPU time ratios are good.

355 To see whether increasing sizes and decreasing  $\epsilon$  improved performances, we solved an  
 356 instance with  $m = 5000$  and  $p = 100$  with  $\epsilon = 0.1$ , obtaining the following results.

$\epsilon$	$\bar{f}/f^*$	$\tilde{f}/f^*$	<code>avgin</code>	<code>avgeq</code>	$k/m$	$\bar{t}/t^*$
quantreg-5000						
0.10	0.1460	0.3839	0.1784	0.0000	0.18	4.43

358 We can see that the objective function ratios of this instance provide a definite improvement  
 359 with respect to the three largest instances in Fig. 4 ( $m \in \{1500, 2000, 2500\}$ ). The negativity  
 360 error is, however, of the same magnitude as before.



■ **Figure 1** MAX FLOW plots (increasing  $\epsilon$  on abscissae): instances of growing size on rows, objective function ratios on the first column, feasibility errors on the second,  $k/m$  and CPU time ratio on the third.

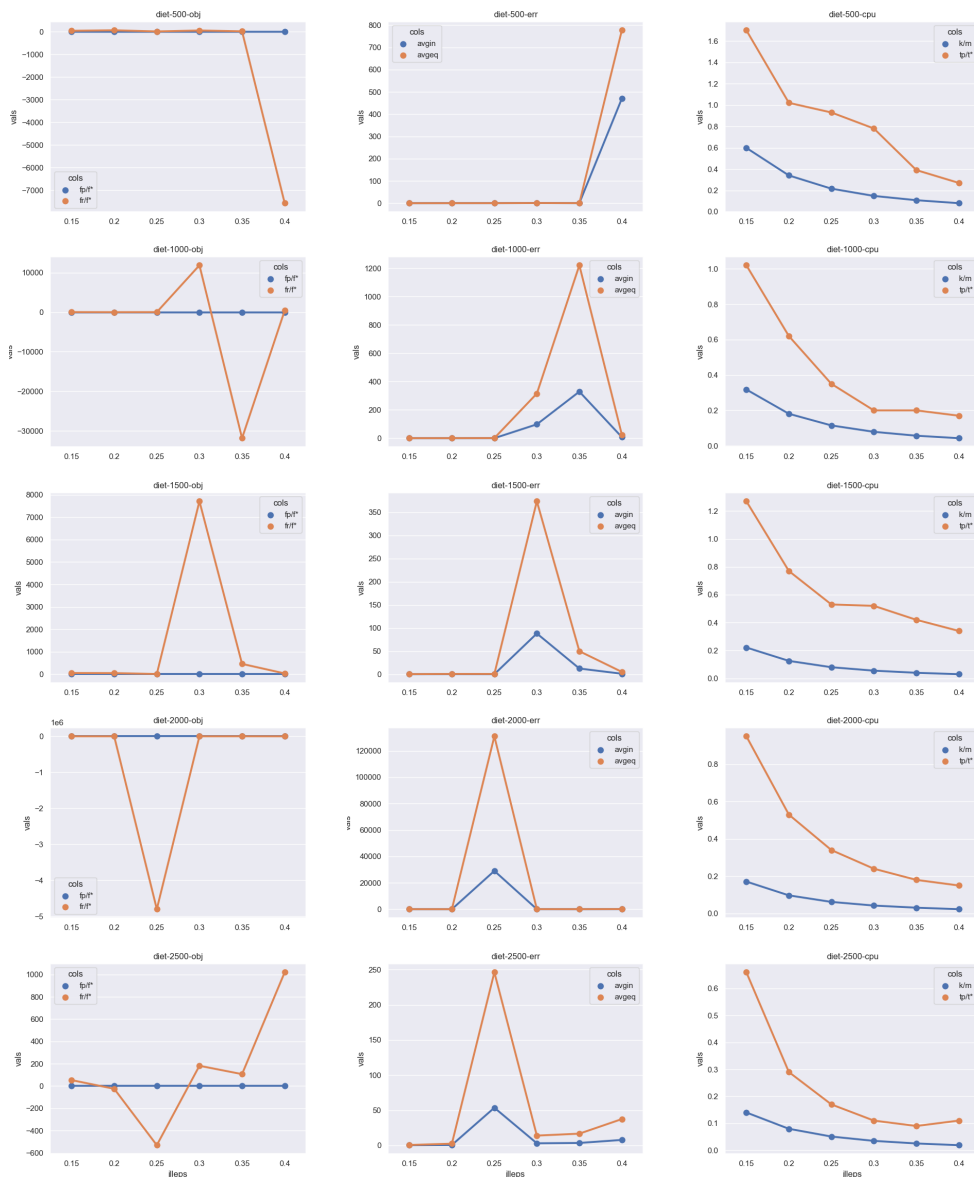
### 361 4.6 RP performance on Basis Pursuit

362 In the BASIS PURSUIT problem we see an encouraging trend of the ratio  $\bar{f}/f^*$ , which starts  
 363 off at 0.8 for  $m = 500$  and  $\epsilon = 0.15$ , and indicates that  $\epsilon$  should be decreased for larger sizes.  
 364 The retrieved solution was not computed on the “sandwich” variables  $s$  (see Eq. (BP)), but  
 365 as the  $\ell_1$  norm of  $\tilde{x}$ . Since there are fewer constraints in the encoding matrix  $A$ , it follows  
 366 from compressed sensing theory that the sparsest solution is found less often, a fact that  
 367 increases the objective value of the retrieved solution. The feasibility errors are always zero  
 368 (for  $Ax = b$  and  $x \geq 0$ ), which happens because the variables  $x$  are unbounded. The CPU  
 369 time ratio is not as regular as for the other structures, but still denotes a remarkable time  
 370 saving when solving projected formulations.

371 To see whether increasing sizes and decreasing  $\epsilon$  improved performances, we solved an  
 372 instance with  $m = 5000$  and  $n = 6000$  with  $\epsilon = 0.1$ , obtaining the following results.

373

$\epsilon$	$\bar{f}/f^*$	$\tilde{f}/f^*$	avgin	avgeq	$k/m$	$\bar{t}/t^*$
basispursuit-5000						
0.10	0.4925	1.5395	0.0000	0.0000	0.17	0.09



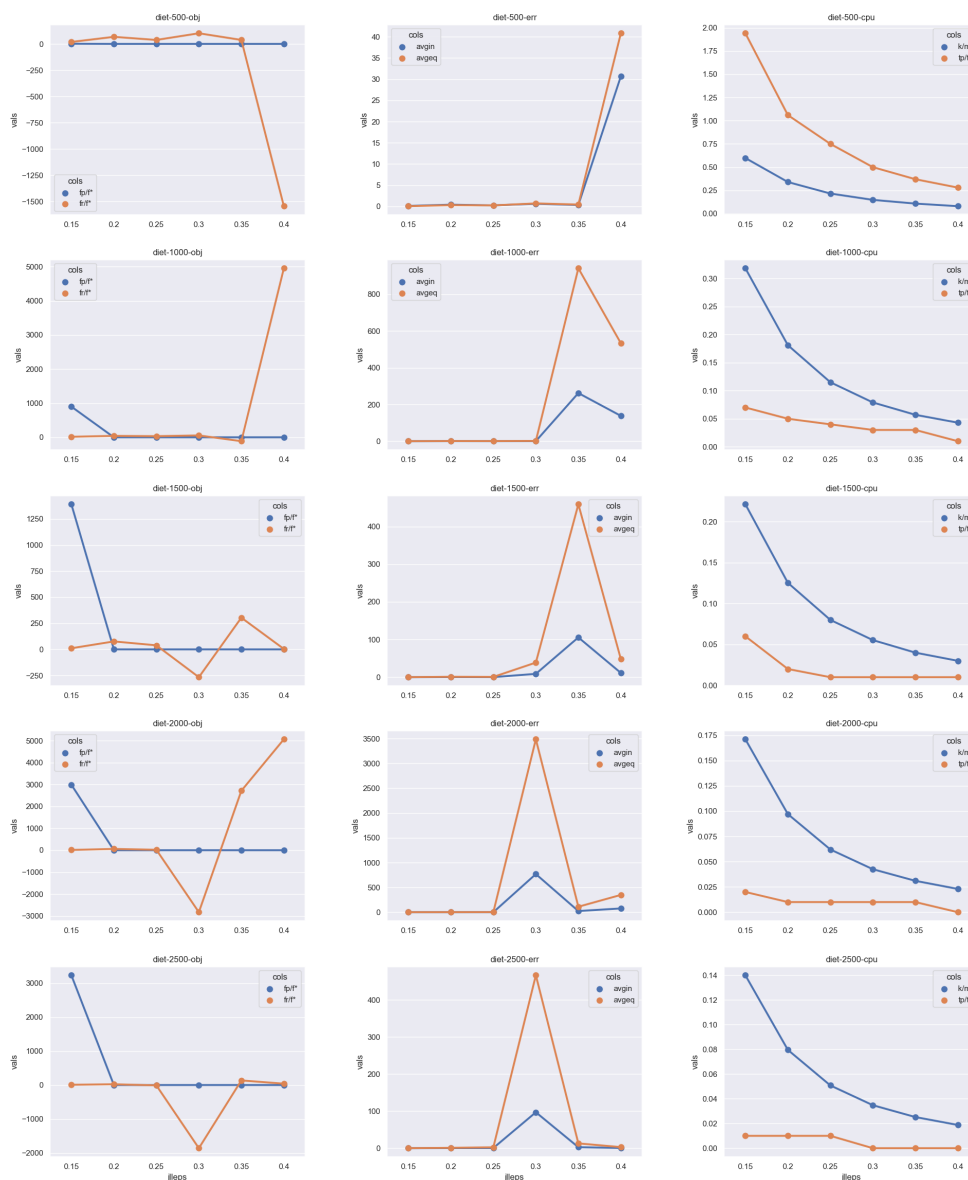
■ **Figure 2** DIET plots (increasing  $\epsilon$  on abscissae): instances of growing size on rows, objective function ratios on the first column, feasibility errors on the second,  $k/m$  and CPU time ratio on the third.

374 An improvement with respect to the three largest instances in Fig. 5 ( $m \in \{1500, 2000, 2500\}$ )  
 375 is present, which points to the correct trend, albeit not substantial.

## 376 5 Conclusion

377 In this paper we have pursued a computational study of the application of random projections  
 378 to linear program data, based on solving original and projected formulations linear program  
 379 instances of various structures and sizes. We found that original formulations only involving  
 380 inequalities are particularly challenging, but those that natively involve equations behave

## 21:12 Practical random projections for LP

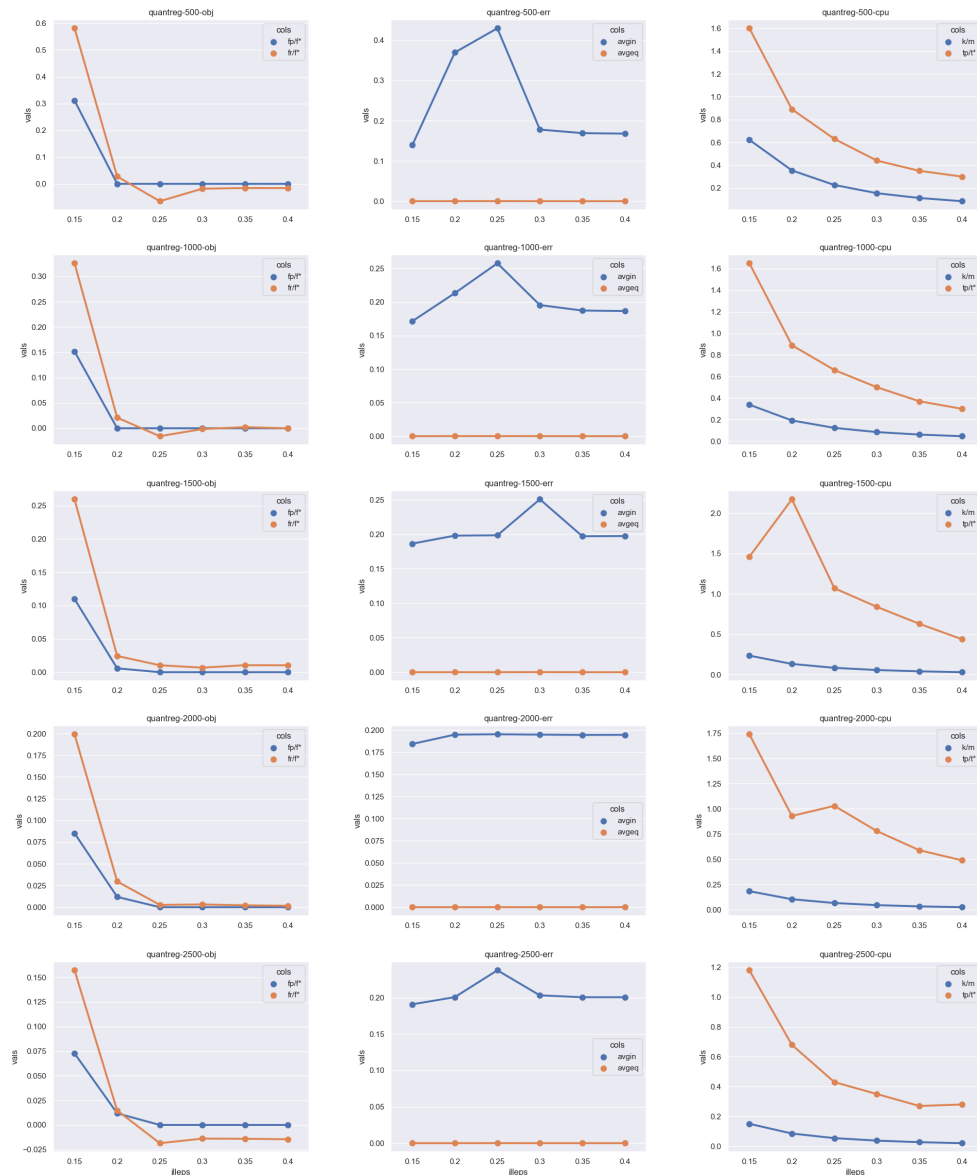


■ **Figure 3** DIET plots with modified objective attempting to drive the slack variables to zero.

381 better. The sparsity of the constraint matrix does not appear to pose issues, as long as  
 382 sparse RPs are used. Lastly, the sizes we considered here are possibly at the lower end of the  
 383 range allowed by RPs: better results should be obtained with larger sizes and smaller values  
 384 of  $\epsilon$ , which in turn imply larger CPU times.

### 385 — References —

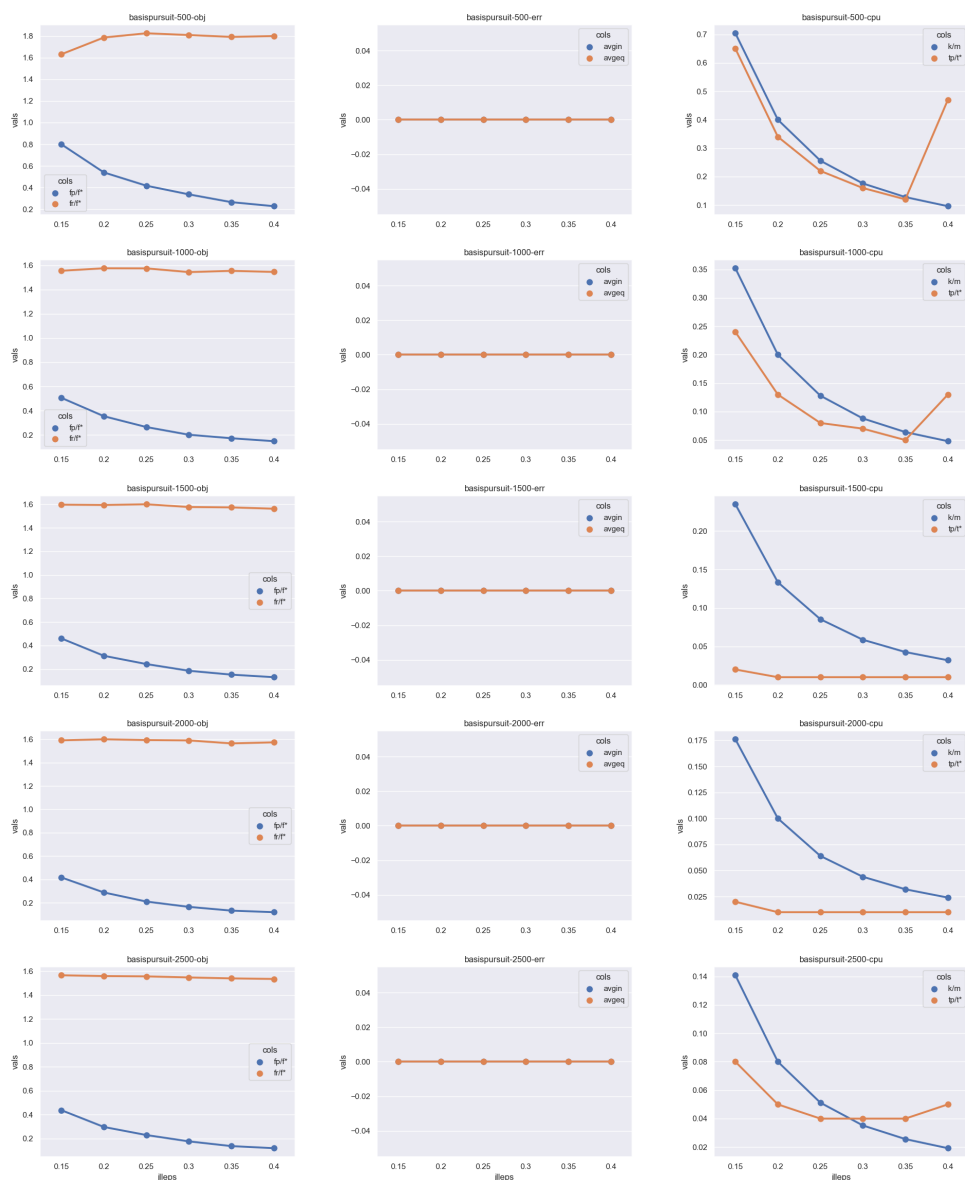
- 386 1 D. Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary  
 387 coins. *Journal of Computer and System Sciences*, 66:671–687, 2003.
- 388 2 E. Candès and T. Tao. Decoding by Linear Programming. *IEEE Transactions on Information*  
 389 *Theory*, 51(12):4203–4215, 2005.



■ **Figure 4** QUANTILE REGRESSION plots (increasing  $\epsilon$  on abscissae): instances of growing size on rows, objective function ratios on the first column, feasibility errors on the second,  $k/m$  and CPU time ratio on the third.

390 3 C. D’Ambrosio, L. Liberti, P.-L. Poirion, and K. Vu. Random projections for quadratic  
 391 programs. *Mathematical Programming B*, 183:619–647, 2020.  
 392 4 S. Damelin and W. Miller. *The mathematics of signal processing*. CUP, Cambridge, 2012.  
 393 5 G. Dantzig. The Diet Problem. *Interfaces*, 20(4):43–47, 1990.  
 394 6 S. Dirksen. Dimensionality reduction with subgaussian matrices: A unified theory. *Foundations*  
 395 *of Computational Mathematics*, 16:1367–1396, 2016.  
 396 7 L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ,  
 397 1962.  
 398 8 R. Fourer and D. Gay. *The AMPL Book*. Duxbury Press, Pacific Grove, 2002.  
 399 9 IBM. *ILOG CPLEX 20.1 User’s Manual*. IBM, 2020.

## 21:14 Practical random projections for LP



■ **Figure 5** BASIS PURSUIT plots (increasing  $\epsilon$  on abscissae): instances of growing size on rows, objective function ratios on the first column, feasibility errors on the second,  $k/m$  and CPU time ratio on the third.

- 400 **10** P. Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Foundations of*  
401 *Computer Science*, volume 42 of *FOCS*, pages 10–33, Washington, DC, 2001. IEEE.
- 402 **11** P. Indyk and A. Naor. Nearest neighbor preserving embeddings. *ACM Transactions on*  
403 *Algorithms*, 3(3):Art. 31, 2007.
- 404 **12** W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In  
405 G. Hedlund, editor, *Conference in Modern Analysis and Probability*, volume 26 of *Contemporary*  
406 *Mathematics*, pages 189–206, Providence, RI, 1984. AMS.
- 407 **13** E. Jones, T. Oliphant, and P. Peterson. *SciPy: Open source scientific tools for Python*, 2001.  
408 [Online; accessed 2016-03-01]. URL: <http://www.scipy.org/>.

- 409 **14** D. Kane and J. Nelson. Sparser Johnson-Lindenstrauss transforms. *Journal of the ACM*,  
410 61(1):4, 2014.
- 411 **15** R. Koehler. *Quantile regression*. CUP, Cambridge, 2005.
- 412 **16** L. Liberti, P.-L. Poirion, and K. Vu. Random projections for conic programs. *Linear Algebra*  
413 *and its Applications*, 626:204–220, 2021.
- 414 **17** L. Liberti and K. Vu. Barvinok’s naive algorithm in distance geometry. *Operations Research*  
415 *Letters*, 46:476–481, 2018.
- 416 **18** D. Pucci de Farias and B. Van Roy. On constraint sampling in the Linear Programming  
417 approach to approximate Dynamic Programming. *Mathematics of Operations Research*,  
418 29(3):462–478, 2004.
- 419 **19** G. van Rossum and *et al.* *Python Language Reference, version 3*. Python Software Foundation,  
420 2019.
- 421 **20** S. Vempala. *The Random Projection Method*. Number 65 in DIMACS Series in Discrete  
422 Mathematics and Theoretical Computer Science. AMS, Providence, RI, 2004.
- 423 **21** S. Venkatasubramanian and Q. Wang. The Johnson-Lindenstrauss transform: An empirical  
424 study. In *Algorithm Engineering and Experiments*, volume 13 of *ALLENEX*, pages 164–173,  
425 Providence, RI, 2011. SIAM.
- 426 **22** K. Vu, P.-L. Poirion, C. D’Ambrosio, and L. Liberti. Random projections for quadratic  
427 programs over a Euclidean ball. In A. Lodi and *et al.*, editors, *Integer Programming and*  
428 *Combinatorial Optimization (IPCO)*, volume 11480 of *LNCS*, pages 442–452, New York, 2019.  
429 Springer.
- 430 **23** K. Vu, P.-L. Poirion, and L. Liberti. Random projections for linear programming. *Mathematics*  
431 *of Operations Research*, 43(4):1051–1071, 2018.
- 432 **24** J. Yang, X. Meng, and M. Mahoney. Quantile regression for large-scale applications. *SIAM*  
433 *Journal of Scientific Computing*, 36(5):S78–S110, 2014.