# On Augmented Lagrangian for LP Feasibility

Pablo Guerrero-García[1], Eligius M. T. Hendrix[2], and Ana Maria A. C. Rocha[3(✉)]

[1] Applied Mathematics, Universidad de Málaga, 29071 Málaga, Spain
pablito@ctima.uma.es
[2] Computer Architecture, Universidad de Málaga, 29071 Málaga, Spain
eligius@uma.es
[3] ALGORITMI Research Centre/LASI, University of Minho, Campus de Gualtar, 4710-057 Braga, Portugal
arocha@dps.uminho.pt

**Abstract.** Finding an initial feasible point which is both accurate and sparse for a linear programming (LP) problem has been known as the Phase I problem. Literature provides various ideas based on Lagrange relaxation that imply solving a sequence of convex quadratic subproblems with nonnegativity constraints. This paper provides a small elaboration of several ideas and adds a novel approach to generate a feasible point or certify that it does not exist. Moreover, we perform a comparative computational study on the smallest Netlib sparse LP problems to generate the performance profiles on five metrics where subproblems are solved to optimality with FMINCON from MATLAB® Optimization Toolbox.

**Keywords:** linear programming · Phase I · quadratic optimization · augmented Lagrangian

## 1 Introduction

The concept of linking constraints to an objective in optimization problems was found due to Lagrange in the $19^{th}$ century. The so-called Lagrange multipliers were related to duality theory in the $20^{th}$ century. At the same time, ideas from penalty methods were linked with ideas from Lagrange multipliers, resulting in the so-called Augmented Lagrangian (AL) methods. The concepts have been applied to nonlinear programming (NLP), global optimization and to linear programming (LP) [3].

Our focus is on an algorithm from late 90 s (cf. [6,7] and references therein) based on a specific active-set strategy to solve a generic sparse LP problem

$$(P) \quad \begin{array}{ll} \min & c^T x \\ \text{s.t.} & A^T x \geq b \end{array}, \; x \in \mathbb{R}^n \; < > \; (D) \quad \begin{array}{ll} \max & b^T y \\ \text{s.t.} & Ay = c \,, \; y \geq 0 \end{array}, \; y \in \mathbb{R}^m \qquad (1)$$

This work has been supported by FCT - Fundação para a Ciência e Tecnologia within the R&D Unit Project of ALGORITMI Centre.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15888, pp. 167–179, 2026.
https://doi.org/10.1007/978-3-031-97596-7_11

where $A \in \mathbb{R}^{n \times m}$ with $m \geq n$ and $\text{rank}(A) = n$. The vectors $c$ and $b$ denote the gradient of the objective function of the primal and dual respectively. The data $a_j$ and $b_j$ represent the coefficient vector and right-hand side value $j$ of primal constraint $a_j^T x - b_j \geq 0$. Experiments have been reported in [7] on a specific active set LP algorithm. It requires an initial feasible point of the dual $(D)$. Our research question is whether an augmented Lagrangian methodology can be applied to generate such a dual feasible point. Specifically, we are interested in a procedure that finds either a dual feasible point $y$ or a direction $d$ of the primal feasible region to certify that the dual $(D)$ does not have a feasible point (i.e., that $(P)$ has an unbounded objective function).

To investigate the question, we introduce the notation in Sect. 2 and discuss several algorithms in Sect. 3. We sketch the behaviour of the algorithms and compare them computationally in Sect. 4 and Sect. 5 summarizes our findings.

## 2 Mathematical Description and Literature Embedding

### 2.1 Nonlinear Optimization View on Augmented Lagrangians

The handbook [9] provides an introduction to the concepts of nonlinear optimization including concept of Lagrangian relaxation and penalty terms up to what is called Augmented Lagrangians (AL). First of all, consider a NLP description with objective function $g(y)$ and set of equalities written as $s(y) = 0$. These are similar to the constraints of the dual $(D)$:

$$\begin{array}{ll} \min & g(y) \qquad\qquad\qquad\quad , y \in \mathbb{R}^m \\ \text{s. t.} & s_i(y) = 0, i = 1, \ldots, n \ , y \geq 0. \end{array} \tag{2}$$

Let $x$ be an estimate of the dual solution of (2) (so a solution of the primal) and $\rho$ a penalty parameter. A quadratic penalty based AL taking the equality constraints into account and leaving out the nonnegativity constraints is

$$\mathcal{L}_D(y; x, \rho) := g(y) - x^T s(y) + \frac{\rho}{2} s(y)^T s(y). \tag{3}$$

Notice that to be consistent with the format of the problems, we write the Lagrangian term with a minus sign. A general AL algorithm can be based on minimizing $\min_{y \geq 0} \mathcal{L}_D(y; x_k, \rho_k)$ slowly increasing the penalty parameter $\rho_k$ and updating the dual estimate according to $x_{k+1} = x_k - \rho_k s(y_k)$.

As outlined in [9], the hard part from a point of view of smooth optimization is not the equality constraints, but the inequality constraints; in fact, the problem is related to $\min_{y \geq 0} \mathcal{L}_D(y; x_k, \rho_k)$. This could be promoted by adding to (3) a nonsmooth term

$$\frac{\rho}{2} \sum_j (y_j^-)^2, \tag{4}$$

where the expression $z^- := \max\{-z, 0\}$. There exist several variants of penalty functions like using the $1-$norm rather than the 2-norm and exponential terms.

### 2.2 Derived Augmented Lagrangian Steps for Linear Programming

It may be clear that the generic NLP expressions can be translated directly for problems $(P)$ and $(D)$ to derive algorithms, which will be outlined in Sect. 3 in the shape of pseudocodes. First of all, the objective function of the dual is $g(y) = -b^T y$ and the slack is $s(y) = c - Ay$.

To find an initial feasible point of the dual $(D)$, our research question is whether an augmented Lagrangian methodology can be applied for that. Specifically, we can set the objective vector $b = 0$ and investigate whether algorithms from the literature reach either a dual feasible point $y$ or a direction $d$ of the primal feasible region which certifies that the dual feasible region is empty. The question is whether and how to use the theory to derive a dual feasible point solving several nonnegativity bound constrained sub-problems.

One of the algorithms we investigate for generating an initial point $y$ of problem $(D)$ has been an application of the concept for LP in the software `Clp` in the open source COIN-OR software. The corresponding so-called "idiot crash" algorithm has been outlined in [3]. The basic iterate direction $d_k$ in the primal space relates to the AL (3) of the dual:

$$y_{k+1} = \text{argmin}_{y \geq 0} \left( -b^T y - d_k^T (c - Ay) + \frac{\rho_k}{2} (c - Ay)^T (c - Ay) \right), \qquad (5)$$

where looking for a feasible point of $(D)$, we put $b = 0$ and iteratively solve the quadratic problem under nonnegativity constraints

$$y_{k+1} = \text{argmin}_{y \geq 0} \left( -d_k^T (c - Ay) + \frac{\rho_k}{2} (c - Ay)^T (c - Ay) \right). \qquad (6)$$

We will see that such ideas have already been implemented in algorithms in 1972 due to [10].

## 3 Algorithms

One of the investigations on providing an initial point is due to the so-called "idiot crash" algorithm. The motivation of the research is due to its use in the LP implementation `Clp` by Forrest in the open source COIN-OR software. We use the description of Galabova and Hall in [3] (see Algorithm1). Therefore, in the experiments we will refer to it as the FoGaHa algorithm. The idea is to force a basic solution "idiot crash" by having the objective $-b^T y$ in the further quadratic objective function. A surprising aspect is that the penalty parameter $\rho_k \to \infty$ is used in division mode to determine the Lagrange multipliers $d_k$ in line 9, such that the primal direction is getting smaller during the iterations. Notice that in our description, we keep a minus sign to be consistent with having a direction $d$ of the primal feasible region.

A second question in the investigation of `Clp` by [3] deals with the criterion used in line 4 to decide on the update of $\rho_k$. For the criterion we follow the description in [3] and update $\rho_k$ every three iterations.

**Algorithm 1.** Idiot crash algorithm $(A, c, b, \omega)$

**Require:** $A$: matrix
$c$ and $b$: vectors
$\omega > 1$: multiplication parameter
1: Initialize $k = 0$, $d_0 = 0, y_0 = 0, \rho_0 > 0$
2: **repeat**
3: $y_{k+1} \in \operatorname{argmin}_{y \geq 0} \left(-b^T y - d_k^T (c - Ay) + \frac{\rho_k}{2}(c - Ay)^T (c - Ay)\right)$
4: **if** a criterion is satisfied **then**
5: $d_{k+1} = d_k$
6: $\rho_{k+1} = \omega \rho_k$ ▶ update $\rho$
7: **else**
8: $\rho_{k+1} = \rho_k$
9: $d_{k+1} = -\frac{1}{\rho_k}(c - Ay_{k+1})$
10: $k = k + 1$
11: **until** stopping criterion
12: **return** $y_k$ or $d_k$ with $A^T d_k \geq 0$ and $c^T d_k < 0$

We are having a challenge for cases where no feasible point exists and consequently the primal problem $(P)$ is unbounded. A certificate is a direction $d$ of the primal feasible region with $A^T d \geq 0$ and $c^T d < 0$. This is not directly provided by the algorithm. Therefore, in the sequel, we will provide a first stage which first focuses on the feasibility question, before looking for a basic (vertex) solution. It should be noticed again that actually the algorithm does not guarantee to generate a basic solution and requires a post-processing with a crossover procedure. This question goes beyond the investigation of this paper.

Notice that in fact the minimizer of the augmented Lagrangian in line 3 is not necessarily unique. The Hessian $\rho A^T A$ is positive semi-definite. This is why in contrast to other papers, we see argmin as a set of minimum points. Actually, we realized this after running several optimization routines and comparing results.

**Algorithm 2.** Polyak-Tretiakov $(A, c, b, \omega)$

**Require:** $A$: matrix
$c$ and $b$: vectors
$\omega > 1$: multiplication parameter
1: Initialize $k = 0$, $d_0 = 0, y_0 = 0, \rho_0 > 0$
2: **repeat**
3: $y_{k+1} \in \operatorname{argmin}_{y \geq 0} \left(-b^T y - d_k^T (c - Ay) + \frac{\rho_k}{2}(c - Ay)^T (c - Ay)\right)$
4: $d_{k+1} = d_k - \rho_k (c - Ay_{k+1})$
5: $\rho_{k+1} = \omega \rho_k$ ▶ update $\rho$
6: $k = k + 1$
7: **until** stopping criterion
8: **return** $y_k$ or $d_k$ with $A^T d_k \geq 0$ and $c^T d_k < 0$

A second algorithm is due to [10] published already in 1972. We studied the description and implemented the interpretation described in [8] similar to Algorithm 2. It has been shown that if an optimal primal and dual solution exists, the procedure converges in a finite number of steps. The penalty parameter is updated in each iteration. The update of the Lagrange multipliers $d_k$ follows a traditional scheme which includes the penalty parameter. One of the challenges is that if the instance to be solved does not have a feasible dual solution, then no certificate of infeasibility is generated.

Therefore, we developed a first stage algorithm to check on the existence of a feasible point solving a nonnegative convex quadratic optimization problem which generates not necessarily a vertex solution in Algorithm 3. If no feasible point exists, it provides a direction $d$ of the primal problem with $A^T d \geq 0$ and $c^T d < 0$ as a certificate of infeasibility of the dual. If a feasible point $y_0$ exists, it is used as a starting point in the AL minimization in a Polyak-Tretiakov like iterative scheme generating a basic solution in a finite number of steps.

**Algorithm 3.** First Stage added $(A, c, b, \omega)$

**Require:** $A$: matrix
$c$ and $b$: vectors
$\omega > 1$: multiplication parameter
$y_0 \in \operatorname{argmin}_{y \geq 0} \frac{1}{2}(c - Ay)^T(c - Ay)$
$d_0 = -(c - Ay_0)$
**if** $\|d_0\| \geq 10^{-6}$ **then** ▶ no feasible point exists
  **return** $d_0$
**else** ▶ $y_0$ is feasible, but not necessarily basic solution
  Initialize $k = 0, \rho_0 > 0$ ▶ Use $y_0$ as starting value for the AL optimization
  **repeat**
    $y_{k+1} \in \operatorname{argmin}_{y \geq 0} \left(-b^T y - d_k^T(c - Ay) + \frac{\rho_k}{2}(c - Ay)^T(c - Ay)\right)$
    $d_{k+1} = d_k - \rho_k(c - Ay_{k+1})$
    $\rho_{k+1} = \omega \rho_k$ ▶ update $\rho$
    $k = k + 1$
  **until** $\|y_k - y_{k-1}\| < \varepsilon$
  **return** $y_k$

In our experiments, we used FMINCON to minimize the quadratic function on the positive orthant. Actually, for the specific case of minimizing a sum of squares on nonnegativity constraints, we have been experimenting with a gradient projection approach from handbook [9, Sect. 16.7], but realized that for reproduction purposes it is easier to use a standard optimization routine.

## 4 Numerical Experiments

We first illustrate the approaches with two small examples from literature in Sect. 4.1. Then we use several benchmark problems in Sect. 4.2 to compare

behaviour. In all experiments, we used FMINCON to minimize the quadratic AL function over nonnegativity constraints with the default setting. We provided the expressions for the gradient to reduce the number of function evaluations. The final accuracy was $\varepsilon = 1.5\ 10^{-8}$.

### 4.1 Numerical Illustration

Consider the following instance of an LP problem

$$A = \begin{pmatrix} -1 & -1 & -1 & 1 & 0 \\ 0 & -1 & 1 & 0 & -1 \\ -1 & -1 & -2 & 0 & 0 \end{pmatrix}, c = \begin{pmatrix} -4 \\ -2 \\ -6 \end{pmatrix}, b^T = (-3, -2, -1, 0, 0),$$

with optimal primal solution $x^* = (3, 0, -1)$ and the dual $y^* = (0, 2, 2, 0, 2)$.

We start inspecting the idiot crash variant as outlined in Algorithm 1 called FoGaHa here. We use the multiplication factor $\omega = 3$ as suggested in [3] and use as initial penalty value $\rho = 1.0$. Their pseudocode suggests that subproblems are approximately solved. However, no details are given. Their discussion in an appendix suggests they solve to optimality, which we adopted in our experiments. The algorithm converges in 13 iterations to an approximation of $y^*$. It is typical that the approximation differs $10^{-5}$ from the optimum. Notice that the FMINCON is also fed with the explicit gradient expression and requires 427 function evaluations. However, the slack $c - Ay$ is still order of magnitude of $5 \times 10^{-4}$ in norm off from zero. Using the same accuracy would require 34 iterations and 2234 function evaluations to get closer to the exact optimal value $y^*$.

The Polyak-Tretiakov variant called PolTre according to Algorithm 2 was also run with a multiplication factor $\omega = 3$. We observe a convergence to the exact solution $y^*$ with an accuracy of $10^{-6}$ after 4 iterations requiring 79 function evaluations. The order of magnitude of $\|c - Ay\|$ is $8 \times 10^{-8}$. Algorithm 3, which we will call GuHeRo, runs first a first stage to find a non-basic (even interior) solution $y_0 = (1.7176, 1.8197, 1.2313, 0.7867, 1.4116)$ using 14 function evaluations corresponding to $\|c - Ay_0\|$ of $3 \times 10^{-8}$. From there it starts the second stage which requires 3 iterations and 57 function evaluations to approach $y^*$ with an accuracy of $10^{-6}$ and $\|c - Ay_4\|$ of $3 \times 10^{-7}$. It seems for this instance that running a first stage improves solving the problem compared to running Algorithm 2, called PolTre, without the first stage.

Now we consider a small instance with an unbounded primal problem and consequently an infeasible dual problem. This means we have to find a certificate direction $d$ for the primal problem. Observe that the second and third column are linearly dependent.

$$A = \begin{pmatrix} 1 & 2 & -2 & 1 & 1 & 0 & 0 \\ -2 & 1 & -1 & 0 & 0 & 1 & 0 \\ 1 & 2 & -2 & -3 & 0 & 0 & 0 \\ -1 & -2 & 2 & 1 & 0 & 0 & 1 \end{pmatrix}, c = \begin{pmatrix} 1 \\ 4 \\ 0 \\ -1 \end{pmatrix}, b^T = (-2, 4, -4, 2, 0, 0, 0)$$

Running the idiot crash provides a point $y$ after 13 iterations and 458 function evaluations towards $y = (0, \alpha, \alpha, \frac{1}{4}, 0, 3.54, 0)^T$, where $\alpha$ is an arbitrary big number corresponding to an infeasible slack of $(c - Ay) = (-\frac{1}{6}, 0, -\frac{1}{6}, -\frac{1}{3})$. However,

the direction $d = -\frac{1}{\rho_k}(c - Ay)$ is yielding very small values due to the division by the penalty parameters. Using simply $(c - Ay)$ provides a direction of the feasible area of $(P)$ with a positive objective value $c^T(c - Ay) = \frac{1}{6}$ so does not certify unboundedness of the primal and with that infeasibility of the dual.

Algorithm 2 (Polyak-Tretiakov) generates after two iterations with 40 function evaluations infeasible point $y = (0, 2.3, 1, 84, \frac{1}{4}, 0, 3.54, 0)^T$. Direction $d = -(4, 0, 2, 4)$ does not provide a direction of the feasible area of $(P)$.

Algorithm 3, called GuHeRo, uses the first phase to detect the infeasibility. Basically, the minimum objective function value is $\frac{1}{6} > 0$. It generates a direction into the feasible area of $d = (c - A^T y) = (\frac{1}{6}, 0, \frac{1}{6}, \frac{1}{3})^T$ with an objective $c^T d = -\frac{1}{6} < 0$. As $A^T d > 0$ and $c^T d < 0$ we have a certificate that the primal is unbounded and the dual infeasible.

**Table 1.** NETLIB test problem instances; #: Bixby number [1]; nnz: number of nonzeros; $m$ and $n$ the number of rows and columns of matrix $A$.

| # | name | optimal value | $m$ | $n$ | nnz($A$) | nnz($b$) | nnz($c$) | nnz(LP) |
|---|---|---|---|---|---|---|---|---|
| 6 | ADLITTLE | −0.22549496316e+6 | 138 | 56 | 424 | 82 | 37 | 543 |
| 1 | AFIRO | 0.46475314286e+3 | 51 | 27 | 102 | 5 | 7 | 114 |
| 28 | AGG | 0.35991767287e+8 | 615 | 488 | 2862 | 131 | 432 | 3425 |
| 29 | BANDM | 0.15862801845e+3 | 472 | 305 | 2494 | 165 | 118 | 2777 |
| 9 | BLEND | 0.30812149846e+2 | 114 | 74 | 522 | 30 | 8 | 560 |
| 22 | BRANDY | −0.15185098965e+4 | 303 | 220 | 2202 | 2 | 54 | 2258 |
| 16 | DUISRAEL | −0.89664482186e+6 | 316 | 142 | 2411 | 171 | 89 | 2671 |
| 30 | E226 | 0.18751929066e+2 | 472 | 223 | 2768 | 189 | 99 | 3056 |
| 23 | ISRAEL | 0.89664482186e+6 | 316 | 174 | 2443 | 89 | 171 | 2703 |
| 14 | LOFTI | 0.25264706062e+2 | 366 | 153 | 1136 | 8 | 49 | 1193 |
| 3 | SC50A | 0.64575077059e+2 | 78 | 50 | 160 | 1 | 10 | 171 |
| 2 | SC50B | 0.70000000000e+2 | 78 | 50 | 148 | 1 | 5 | 154 |
| 4 | SC105 | 0.52202061212e+2 | 163 | 105 | 340 | 1 | 20 | 361 |
| 10 | SC205 | 0.52202061212e+2 | 317 | 205 | 665 | 1 | 38 | 704 |
| 7 | SCAGR7 | 0.23313898243e+7 | 185 | 129 | 465 | 133 | 53 | 651 |
| 19 | SCAGR25 | 0.14753433061e+8 | 671 | 471 | 1725 | 475 | 179 | 2379 |
| 31 | SCFXM1 | −0.18416759028e+5 | 600 | 330 | 2732 | 23 | 116 | 2871 |
| 17 | SCORPION | −0.18781248227e+4 | 466 | 388 | 1534 | 282 | 76 | 1892 |
| 26 | SCSD1 | −0.86666666743e+1 | 760 | 77 | 2388 | 760 | 1 | 3149 |
| 20 | SCTAP1 | −0.14122500000e+4 | 660 | 300 | 1872 | 360 | 154 | 2386 |
| 15 | SHARE1B | 0.76589318579e+5 | 253 | 117 | 1179 | 31 | 103 | 1313 |
| 12 | SHARE2B | 0.41573224074e+3 | 162 | 96 | 777 | 36 | 24 | 837 |
| 8 | STOCFOR1 | 0.41131976219e+5 | 165 | 117 | 501 | 27 | 8 | 536 |

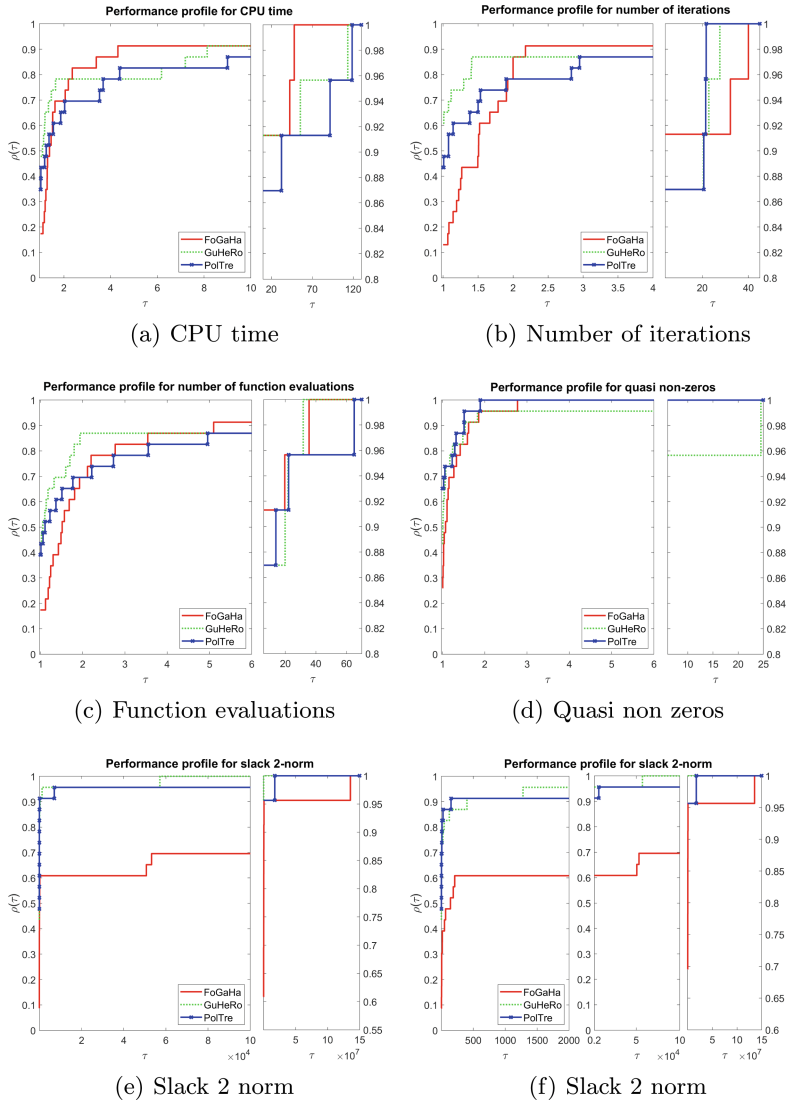
**Table 2.** A comparison of three algorithms, MATLAB® R2024b. #: Bixby number; NZ: number of nearly nonzero $|y_i| \geq \varepsilon$; Feas acc: $\|c - Ay\|_2$; Iters: number of iterations; Feval: number of function evaluations; Time: CPU time in seconds.

| # | IDIOT CRASH (FOGAHA) | | | | | FIRST STAGE ADDED (GUHERO) | | | | | POLYAK-TRETIAKOV (POLTRE) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NZ | Feas acc | Iters | Feval | Time | NZ | Feas acc | Iters | Feval | Time | NZ | Feas acc | Iters | Feval | Time |
| 6 | 102 | 1.44e−04 | 25 | 12673 | 17.4 | 71 | 1.96e−09 | 13 | 4072 | 7.4 | 71 | 5.37e−10 | 14 | 3579 | 7.5 |
| 1 | 50 | 2.22e−05 | 33 | 4590 | 1.6 | 34 | 1.18e−07 | 27 | 2463 | 0.9 | 35 | 3.11e−07 | 27 | 2086 | 0.8 |
| 28 | 601 | 2.26e+02 | 480 | 853573 | 45976.4 | 593 | 4.32e−02 | 12 | 23985 | 957.3 | 615 | 1.78e−03 | 34 | 65450 | 1948.3 |
| 29 | 306 | 6.14e−01 | 483 | 827354 | 28685.4 | 421 | 1.15e−05 | 15 | 41981 | 674.0 | 383 | 8.19e−02 | 322 | 582682 | 21887.8 |
| 9 | 93 | 3.54e−03 | 13 | 8311 | 7.4 | 93 | 8.68e−06 | 14 | 4591 | 5.5 | 114 | 6.97e−08 | 14 | 4654 | 5.1 |
| 22 | 303 | 1.31e−02 | 13 | 13611 | 99.5 | 303 | 2.53e−05 | 12 | 11482 | 77.5 | 212 | 1.98e−08 | 18 | 14075 | 120.2 |
| 16 | 114 | 1.27e+01 | 478 | 828634 | 8329.7 | 117 | 2.56e−01 | 320 | 560789 | 6465.4 | 286 | 2.20e+00 | 317 | 546809 | 7622.4 |
| 30 | 472 | 2.37e−05 | 25 | 14281 | 395.8 | 472 | 7.69e−06 | 14 | 11491 | 244.5 | 446 | 4.33e−06 | 16 | 25469 | 456.0 |
| 23 | 257 | 2.29e+00 | 16 | 15394 | 128.5 | 279 | 3.45e−02 | 221 | 290490 | 667.8 | 254 | 5.20e−02 | 8 | 9157 | 92.8 |
| 14 | 366 | 5.49e−03 | 335 | 384864 | 1186.7 | 366 | 7.79e−03 | 224 | 258597 | 930.4 | 366 | 6.89e−03 | 224 | 274958 | 782.8 |
| 3 | 78 | 2.12e−05 | 37 | 35859 | 14.5 | 78 | 1.41e−06 | 22 | 7030 | 3.4 | 78 | 1.41e−05 | 17 | 24973 | 11.8 |
| 2 | 75 | 2.84e−06 | 31 | 13632 | 6.2 | 75 | 4.86e−07 | 29 | 18041 | 5.6 | 78 | 3.60e−06 | 26 | 20586 | 6.9 |
| 4 | 163 | 1.26e−05 | 25 | 44915 | 61.4 | 163 | 6.26e−06 | 20 | 16210 | 18.1 | 163 | 1.71e−05 | 59 | 80368 | 79.6 |
| 10 | 317 | 1.51e−05 | 353 | 474825 | 1187.0 | 317 | 1.11e−04 | 232 | 320304 | 989.8 | 315 | 2.56e−06 | 234 | 302754 | 3644.8 |
| 7 | 144 | 1.88e−01 | 16 | 11137 | 26.8 | 139 | 1.39e−09 | 15 | 8558 | 20.6 | 129 | 2.44e−02 | 326 | 556301 | 1883.6 |
| 19 | 476 | 3.40e−01 | 16 | 27285 | 902.9 | 671 | 3.56e−03 | 328 | 596990 | 102249.3 | 543 | 1.62e−03 | 331 | 605398 | 107054.1 |
| 31 | 600 | 2.75e−01 | 10 | 17736 | 481.9 | 589 | 1.20e−06 | 14 | 28371 | 705.3 | 600 | 1.83e−04 | 19 | 31355 | 658.1 |
| 17 | 466 | 3.51e−05 | 351 | 405276 | 2336.9 | 442 | 1.41e−02 | 328 | 550287 | 19010.2 | 297 | 5.96e−05 | 233 | 284138 | 21040.0 |
| 26 | 550 | 3.81e−05 | 40 | 32103 | 797.8 | 760 | 2.93e−07 | 21 | 27293 | 2250.6 | 116 | 2.67e−07 | 29 | 15176 | 364.2 |
| 20 | 660 | 9.81e−05 | 28 | 54685 | 1330.5 | 660 | 1.39e+00 | 320 | 564598 | 48527.5 | 660 | 2.42e−05 | 14 | 28381 | 874.9 |
| 15 | 253 | 6.64e−04 | 19 | 31162 | 225.5 | 253 | 4.07e−05 | 15 | 27779 | 259.4 | 253 | 1.09e−03 | 23 | 38017 | 193.7 |
| 12 | 162 | 6.38e−03 | 16 | 11954 | 22.8 | 162 | 1.02e−07 | 14 | 20288 | 37.4 | 162 | 1.67e−08 | 14 | 13267 | 23.1 |
| 8 | 124 | 3.80e+00 | 10 | 5300 | 8.2 | 128 | 7.44e−05 | 6 | 4813 | 7.5 | 121 | 2.34e−05 | 6 | 4356 | 6.6 |

### 4.2 Benchmark Problems

We run computational experiments using the first 31 smallest NETLIB problems [4] in Table 1. From them we have selected only those formulated in standard form and/or canonical form. This defines a test benchmark of 23 LPs in pure inequality form.

Note that DUISRAEL is relatively new. Following the reasoning of the Bixby numbers it has Bixby number 16 due to the number of nonzeros. Moreover, according to [1], we are aware of the fact that the order used in Table 1 is not most logical, but we guess it is easier to find the details of a particular LP by only focusing on its name. Table 2 shows the results obtained by the three algorithms comparing each instance individually. For example one can observe that for instances BANDM and SCAGR7, the algorithm GuHeRo provides far more accuracy with much fewer function evaluations than the other algorithms. To have a more global comparison of the performance of the three algorithms, we use performance profiles as described in Dolan and Moré's paper [2].




(a) CPU time (b) Number of iterations

(c) Function evaluations (d) Quasi non zeros

(e) Slack 2 norm (f) Slack 2 norm

**Fig. 1.** Performance profiles, 5 metrics, 3 algorithms: FoGaHa in red; GuHeRo in green, PolTre in blue (Color figure online)

### 4.3 Comparison Based on Performance Profiles

The performance profiles are based on minimizing a certain performance metric to compare the performance of solvers. These are based on observations of running algorithm $s \in \mathcal{S}$ for solving problem instance $p \in \mathcal{P}$. In our case, the metric data are provided by Table 2. Let $m_{p,s}$ be the value of the metric when solving problem instance $p \in \mathcal{P}$ by solver $s \in \mathcal{S}$. The comparison is based on performance ratios defined by

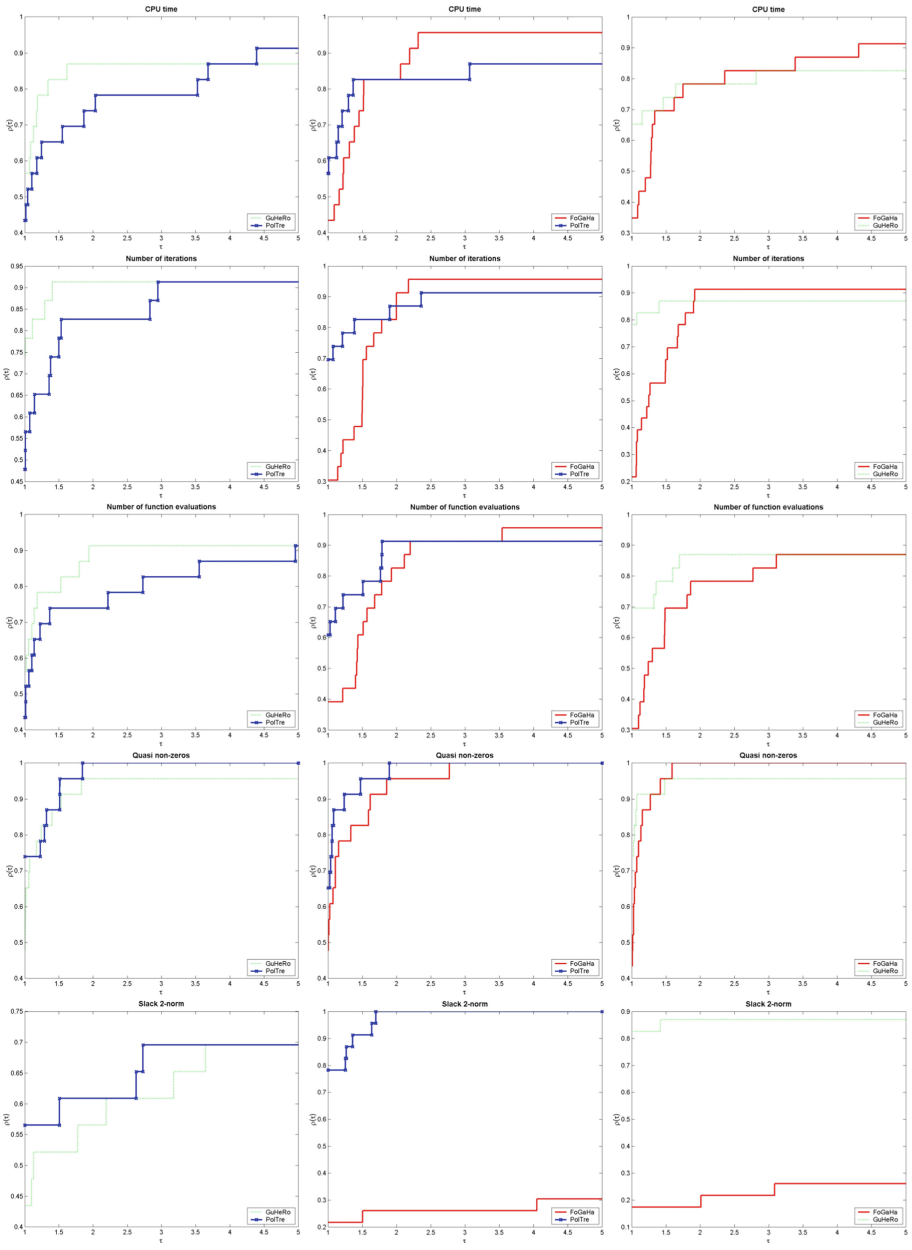



**Fig. 2.** Performance profiles one-versus-one comparison, 5 metrics, three algorithms: FoGaHa in red; GuHeRo in green, PolTre in blue (Color figure online)

$$r_{p,s} = \frac{m_{p,s}}{\min\{m_{p,s} : s \in \mathcal{S}\}}.$$

The overall assessment of the performance of the solver $s$ is given by

$$\rho_s(\tau) = \frac{\text{number of instances with } r_{p,s} \leq \tau}{\text{total number of instances}}.$$

Following a probabilistic perspective, $\rho_s(\tau)$ gives the probability (for $s \in \mathcal{S}$) that $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio.

The value of $\rho_s(1)$ gives the probability that a particular solver, $s$, will have a better performance than the others. Thus, to see which solver has the least value of the metric most of the times, $\rho_s(1)$ should be compared for all the solvers. The higher the value of $\rho_s$ the better the solver is. In this interpretation, $\rho_s(\tau)$ measures for large values of $\tau$ the solver robustness.

Figure 1 shows the performance profiles based on the selected metrics. We do not only focus on efficiency metrics as computational effort in terms of CPU time, number of iterations and objective function evaluations, but also the effectiveness measured as quality of the feasible point obtained in terms of accuracy and sparsity. The hypothesis is that the "idiot crash" algorithm as described by Forrest, Galabova and Hall in their pseudocode can be outperformed when subproblems are solved to optimality rather than approximately. Moreover, we want to measure the importance of a good initialization step in a first stage when added to the Polyak-Tretiakov algorithm.

Consider first the computational effort. From Fig. 1(a) we can observe that GuHeRo is faster than PolTre in the whole test set. There are some instances for which FoGaHa is faster. However, having a more detailed look at the results of FoGaHa for instances SCFXM1 (#31) and SCAGR25 (#19) in Table 2, one can observe that indeed computation is faster, but FoGaHa reaches an accuracy less than at least two orders of magnitude compared to GuHeRo. From Fig. 1(b) we can observe that GuHeRo requires less iterations than PolTre in the whole test set. FoGaHa only performs better for several instances against reaching a lower accuracy as described above. From Fig. 1(c) we can observe that GuHeRo requires less function evaluations than both PolTre and FoGaHa in the whole test set. FoGaHa performs better than PolTre on half the test set. Furthermore, GuHeRo appears slightly robuster than FoGaHa when considering this metric.

We now focus on the quality of the obtained feasible point. Figure 1(d) illustrates that PolTre reaches sparser feasible points i.e., those with a smaller number of nonzeros. Notice that the addition of the first stage of GuHeRo does not improve this performance. However, GuHeRo reaches sparser results than FoGaHa for nearly the whole test set. Figures 1(e) and 1(f) focus on the Euclidean norm of the slack $s := c - Ay$. We can observe that both GuHeRo and PolTre reach tighter solutions than FoGaHa. Focusing further on the difference between GuHeRo and PolTre, we provide both a 2-zoom and a 3-zoom performance profile, which illustrate that GuHeRo is most robust according to this metric.

To strengthen the observations before, we also provide 15 performance profiles for the 5 metrics considered for one-versus-one comparisons as suggested by

[5], see Fig. 2. In this case we have restricted the analysis to a factor $\tau \leq 5$ to improve readability, since the limiting behaviour was already analyzed before. The second and third column show that PolTre and GuHeRo provide better performance than FoGaHa in all metrics. The first column suggests a smaller computational effort of GuHeRo, whereas PolTree slightly outperforms GuHeRo with respect to the quality of the obtained feasible point.

## 5 Conclusions

This paper focuses on the generation of a feasible dual point of a potentially sparse LP problem. As a basis, we took two algorithms from literature, the Polyak and Tretiakov algorithm from early 70 s and the so-called "idiot crash" algorithm of Forrest analysed recently by Galabova and Hall. As an alternative, we added to the first algorithm a first stage initialization to generate a starting point. This first stage initialization step has the advantage that it generates a certificate of infeasibility when the instance to be solved has an unbounded solution and the corresponding dual is infeasible.

First of all, the described algorithms have been embedded in a common framework. Two illustrative small examples show the difference among the algorithms. A comparative computational study on the smallest Netlib sparse LP problems has been used to generate performance profiles on five metrics; CPU time, number of iterations, number of function evaluations, the accuracy of the solution as the Euclidean size of the slack and the sparsity. Subproblems are solved up to optimality with FMINCON from the MATLAB® Optimization Toolbox. The experiments show the advantage of the introduced algorithm. For most cases, the novel algorithm is faster than the original Polyak Tretjakov algorithm and provides better (more accurate) solutions than the idiot crash algorithm.

## References

1. Bixby, R.E.: Implementing the simplex method: the initial basis. ORSA J. Comput. **4**(3), 267–284 (1992)
2. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
3. Galabova, I., Hall, J.: The 'idiot' crash quadratic penalty algorithm for linear programming and its application to linearizations of quadratic assignment problems. Optim. Methods Softw. **35**(3), 488–501 (2020)
4. Gay, D.M.: Electronic mail distribution of linear programming test problems. Committee on Algorithms (COAL) Newslett. **13**, 10–12 (1985)
5. Gould, N., Scott, J.: A note on performance profiles for benchmarking software. ACM Trans. Math. Softw. **43**(2), 15(1–5) (2017)
6. Guerrero-García, P., Hendrix, E.M.T.: On active-set LP algorithms allowing basis deficiency. In: Gervasi, O., Murgante, B., Misra, S., Rocha, A., Garau, C. (eds.) Computational Science and Its Applications–Proceedings of the ICCSA 2022 Workshops, Part II, LNCS, vol. 13378, pp. 174–187. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-10562-3_13

7. Guerrero-García, P., Hendrix, E.M.T.: Experiments with active-set LP algorithms allowing basis deficiency. Computers **12**(3), 1–23 (2023). https://doi.org/10.3390/computers12010003
8. Güler, O.: Augmented Lagrangian algorithms for linear programming. J. Optim. Theory Appl. **75**, 445–470 (1992)
9. Nocedal, J., Wright, S.J.: Numerical Optimization, 2e edn. Springer, New York, NY, USA (2006). https://doi.org/10.1007/978-0-387-40065-5
10. Polyak, B.T., Tretjakov, N.V.: An iterative method of linear programming and its economic interpretation. Ekon. Mat. Metody **8**(5), 740–751 (1972). in Russian