The "Idiot" crash quadratic penalty algorithm for linear programming and its application to linearizations of quadratic assignment problems

Citation for published version:
Galabova, I & Hall, J 2019, 'The "Idiot" crash quadratic penalty algorithm for linear programming and its application to linearizations of quadratic assignment problems', *Optimization Methods and Software*. https://doi.org/10.1080/10556788.2019.1604702

Digital Object Identifier (DOI):
10.1080/10556788.2019.1604702

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Optimization Methods and Software

The "Idiot" crash quadratic penalty algorithm for linear programming and its application to linearizations of quadratic assignment problems

I. L. Galabova J. A. J. Hall

University of Edinburgh
School of Mathematics and Maxwell Institute for Mathematical Sciences
James Clerk Maxwell Building
Peter Guthrie Tait Road, Edinburgh, EH9 3FD, UK
Email J.A.J.Hall@ed.ac.uk

24 April 2018

Abstract

We provide the first meaningful documentation and analysis of the "Idiot" crash implemented by Forrest in Clp that aims to obtain an approximate solution to linear programming (LP) problems for warm-starting the primal simplex method. The underlying algorithm is a penalty method with naive approximate minimization in each iteration. During initial iterations an approach similar to augmented Lagrangian is used. Later the technique corresponds closely to a classical quadratic penalty method. We discuss the extent to which it can be used to obtain fast approximate solutions of LP problems, in particular when applied to linearizations of quadratic assignment problems.

1 Introduction

The efficient solution of linear programming (LP) problems is crucial for a wide range of practical applications, both as problems modelled explicitly, and as subproblems for discrete and nonlinear optimization problems. Finding an approximate solution rapidly is valuable as a "crash start" to an exact solution method. There are also applications where it is preferable to trade solution accuracy for a significant increase in speed.

The "Idiot" crash within the open source LP solver `Clp` [2] of Forrest aims to find an approximate solution of an LP problem prior to application of the primal revised simplex method. In essence, the crash replaces minimization of the linear objective subject to linear constraints by minimization of the objective plus a multiple of a quadratic function of constraint violations.

Section 2 sets out the context of the Idiot crash within `Clp` and the very limited documentation and analysis that exists. Since the Idiot crash is later discussed in relation to the quadratic penalty and augmented Lagrangian methods, a brief introduction to these established techniques is also given. The algorithm used by the Idiot crash is set out in Section 3, together with results of experiments on representative LP test problems and a theoretical analysis of its properties. The extent to which the Idiot crash can be used to obtain fast approximate solutions of LP problems, in particular when applied to linearizations of quadratic assignment problems (QAPs), is explored in Section 4. Conclusions are set out in Section 5.

2 Background

For convenience, discussion and analysis of the algorithms in this paper are restricted to linear programming (LP) problems in standard form:

$$\text{minimize } f = \boldsymbol{c}^T\boldsymbol{x} \quad \text{subject to } A\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} \geq \boldsymbol{0}, \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{c} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$ and $m < n$. In problems of practical interest, the number of variables and constraints can be large and the matrix A is sparse. It can also be assumed that A has full rank of m. The algorithms, discussion and analysis below extend naturally to more general LP problems.

The Idiot crash was introduced into the open source LP solver `Clp` [2] in 2002 and aims to find an approximate solution of an LP problem prior to application of the primal revised simplex method. Beyond its definition as source code [6], a 2014 reference by Forrest to having given "a bad talk on it years ago" [5], as well as a few sentences of comments in the source code, documentation for the mixed-integer programming solver `Cbc` [7], and a public email [5], the Idiot crash lacks documentation or analysis. Forrest's comments stress the unsophisticated nature of the crash and only hint at its usefulness for preceding the primal simplex algorithm. However, for several test problems used in the Mittelmann benchmarks [12], `Clp` is significantly faster than at least one of the three major commercial solvers (`Cplex`, `Gurobi` and `Xpress`), and experiments in Section 3.2 show that the Idiot crash is a major factor in this relative performance. For three of these test problems, NUG12, NUG15 and QAP15, which are quadratic assignment problem (QAP) linearizations, it is shown to be particularly effective. This serves as due motivation for studying the algorithm, understanding why it performs well on certain LP problems, notably QAPs, and how it might be of further value.

The Idiot crash terminates at a point that has no guaranteed properties other than satisfying the bounds $\boldsymbol{x} \geq \boldsymbol{0}$. In particular, it satisfies no known bound on the residual $\|A\boldsymbol{x} - \boldsymbol{b}\|_2$ or distance (positive or negative) from the optimal objective value. Although some variables may be at bounds, there is no reason why the point should be a vertex solution. Thus, within the context of `Clp`, before the primal simplex method can be used to obtain an optimal solution to the LP problem, a "crossover" procedure is required to identify a basic solution from the point obtained by the Idiot crash. We believe that `Clp` uses the same crossover code as is used to get a basic solution after the `Clp` interior point solver, but one crucial difference in the case of the Idiot crash is that it yields no dual values. How this absence is accommodated is beyond the scope of this paper. Since the

Idiot crash seeks a primal feasible point and has nothing corresponding to dual values, it would seem more appropriate to use it to warm-start the primal simplex method rather than dual simplex.

2.1 Penalty function methods

Although Forrest states that the Idiot crash minimizes a multiple of the LP objective plus a sum of squared primal infeasibilities [5], a more general quadratic function of constraint violations is minimized in `Clp`. This includes a linear term, making the Idiot crash objective comparable with an augmented Lagrangian function. For later reference, these two established penalty function methods are outlined below.

The quadratic penalty method

For the nonlinear equality problem

$$\text{minimize } f(\boldsymbol{x}) \quad \text{subject to } \boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{0}, \tag{2}$$

the quadratic penalty method minimizes

$$\phi(\boldsymbol{x}, \mu) = f(\boldsymbol{x}) + \frac{1}{2\mu}\boldsymbol{r}(\boldsymbol{x})^T\boldsymbol{r}(\boldsymbol{x}), \tag{3}$$

for a decreasing sequence of positive values of μ. If $\boldsymbol{x}^k$ is the global minimizer of $\phi(\boldsymbol{x}, \mu^k)$ and $\mu^k \downarrow 0$, Nocedal and Wright [13] show that every limit point $\boldsymbol{x}^*$ of the sequence $\{\boldsymbol{x}^k\}$ is a global solution of (2). The subproblem of minimizing $\phi(\boldsymbol{x}, \mu^k)$ is known to be increasingly ill-conditioned as smaller values of μ^k are used [13] and this is one motivation for use of the augmented Lagrangian method, for which μ would not need to be as small as machine precision.

The augmented Lagrangian method

The augmented Lagrangian method, outlined in Algorithm 1, was originally presented as an approach to solving nonlinear programming problems like (2). It was first proposed by Hestenes in his survey of multiplier and gradient methods [9] and then fully interpreted and analysed, first by Powell [15] and then by Rockafellar [17]. The augmented Lagrangian function (4) is a combination of the Lagrangian function and the quadratic penalty function [13]. It is the quadratic penalty function with an explicit estimate of the Lagrange multipliers $\boldsymbol{\lambda}$:

$$\mathcal{L}_A(\boldsymbol{x}, \boldsymbol{\lambda}, \mu) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T\boldsymbol{r}(\boldsymbol{x}) + \frac{1}{2\mu}\boldsymbol{r}(\boldsymbol{x})^T\boldsymbol{r}(\boldsymbol{x}). \tag{4}$$

Although originally intended for nonlinear programming problems, the augmented Lagrangian method has also been applied to linear programming problems [4, 8]. However, neither article assesses its performance on large-scale practical LP problems.

Algorithm 1 The augmented Lagrangian algorithm for problem (2)

Initialize $\boldsymbol{x}^0 \geq \boldsymbol{0}$, μ^0, $\boldsymbol{\lambda}^0$ and a tolerance τ^0
For $k = 0, 1, 2, \ldots$
  Find an approximate minimizer $\boldsymbol{x}^k$ of $\mathcal{L}_A(.;\boldsymbol{\lambda}^k, \mu^k)$, starting at $\boldsymbol{x}^k$
    and terminating when $\|\nabla_{\boldsymbol{x}}\mathcal{L}_A(\boldsymbol{x}^k, \boldsymbol{\lambda}^k, \mu^k)\| \leq \tau^k$
  If a convergence test for (2) is satisfied
    stop with an approximate solution $\boldsymbol{x}^k$
  End if
  Update Lagrange multipliers $\boldsymbol{\lambda}$
    Set $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \mu^k \boldsymbol{r}(\boldsymbol{x}^k)$
  Choose new penalty parameter μ^{k+1} so that $0 \leq \mu^{k+1} \leq \mu^k$
  Choose new tolerance τ^{k+1}
End

3 The Idiot crash algorithm

This section presents the Idiot crash algorithm (ICA) in `Clp`, followed by some practical and mathematical analysis of its behaviour. Experiments assess the extent to which ICA accelerates the solution of representative LP test problems using the primal simplex method, and can be used to find a feasible and near-optimal solution of the problems. Theoretical analysis of the limiting behaviour of the algorithm shows that it will solve any LP problem that has an optimal solution.

3.1 The algorithm

ICA approximately minimizes the function

$$h(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\lambda}^T\boldsymbol{r}(\boldsymbol{x}) + \frac{1}{2\mu}\boldsymbol{r}(\boldsymbol{x})^T\boldsymbol{r}(\boldsymbol{x}), \quad \text{where} \quad \boldsymbol{r}(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b}, \tag{5}$$

subject to the bounds $\boldsymbol{x} \geq \boldsymbol{0}$ for sequences of values of parameters $\boldsymbol{\lambda}$ and μ. The minimization is performed with respect to each component of $\boldsymbol{x}$ in turn, with the starting index of this loop over all components being chosen randomly. The general structure of the algorithm is set out in Algorithm 2. Except for the alternative expression for updating $\boldsymbol{\lambda}^k$ and the component-wise minimization of $h(\boldsymbol{x})$, this algorithm is very close to that of `LANCELOT` [3] applied to problem (1). The number of ICA iterations is determined heuristically according to the size of the LP and progress of the algorithm. Unless ICA is abandoned after around 20 "sample" iterations (see below), the number of iterations performed ranges between 30 and 200. The value of μ^0 ranges between 0.001 and 1000, again according to the LP dimensions. When μ is changed, the factor by which it is reduced is typically $\omega = 0.333$. The final value of μ is typically a little less than machine precision.

The version of ICA implemented in `Clp` has several additional features. If $\boldsymbol{x}^0$ is feasible then the algorithm is not performed and the value $\boldsymbol{x} = \boldsymbol{x}^0$ is returned. Otherwise,

Algorithm 2 The Idiot crash algorithm for problem (1), with $\boldsymbol{r}(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b}$

Initialize $\boldsymbol{x}^0 = \boldsymbol{0}$, μ^0, $\boldsymbol{\lambda}^0 = \boldsymbol{0}$
Set $\mu^1 = \mu^0$ and $\boldsymbol{\lambda}^1 = \boldsymbol{\lambda}^0$
For $k = 1, 2, 3, \ldots$

$$\boldsymbol{x}^k \approx \arg\min_{\boldsymbol{x} \geq \boldsymbol{0}} h(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\lambda}^{k^T}\boldsymbol{r}(\boldsymbol{x}) + \frac{1}{2\mu^k}\boldsymbol{r}(\boldsymbol{x})^T\boldsymbol{r}(\boldsymbol{x})$$

If a criterion is satisfied (see 3.3.1) update μ^k:
$\mu^{k+1} = \mu^k/\omega$, for some factor $\omega > 1$
$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k$
Else update $\boldsymbol{\lambda}^k$:
$\mu^{k+1} = \mu^k$
$\boldsymbol{\lambda}^{k+1} = \mu^k\boldsymbol{r}(\boldsymbol{x}^k)$
End

initially, the approximate component-wise minimization is performed twice. If a 10% decrease in primal infeasibility is not observed in about 30 iterations, it is considered that ICA would not be beneficial and the value $\boldsymbol{x} = \boldsymbol{x}^0$ is returned. Otherwise, ICA continues but the mechanism for approximate minimization is adjusted. During each subsequent iteration, the function $h(\boldsymbol{x})$ is minimized componentwise 105 times. There is no indication why this particular value was chosen. However, one of the features is the option to decrease this number. From the 50$^{\text{th}}$ minimization onward, a check is performed after the function is minimized componentwise 10 times. Progress is measured with a moving average of expected progress. If it is considered that not enough progress is being made, the function is not minimized any longer for the same values of the parameters. Instead, either μ or $\boldsymbol{\lambda}$ is updated and the next iteration is performed. Thus, in the cases when it is likely that the iteration would not be beneficial, not much unnecessary time is spent. Another feature is that in some cases there is a limit on the step size for the update of each x_j. Additionally, there is a statistical adjustment of the values of $\boldsymbol{x}$ at the end of each iteration. These features are omitted from this paper because experiments showed that they have little effect on performance. Depending on the problem size and structure, the weight parameter (μ) is updated either every 3 iterations or every 6. Again, there is no indication why these values are chosen. To a large extent it must be assumed that the algorithm has been tuned to achieve a worthwhile outcome when possible, and terminate promptly when not. The dominant computational cost for each component-wise minimization of $h(\boldsymbol{x})$ is about the same as a matrix-vector product $A\boldsymbol{v}$.

Relation to augmented Lagrangian and quadratic penalty function methods

In form, the augmented Lagrangian function (4) and the ICA function (5) are identical for LP problems and in both methods the penalty parameter μ is reduced over a sequence of iterations. However, they differ fundamentally in the update of $\boldsymbol{\lambda}$. For ICA, new values of $\boldsymbol{\lambda}$ are given by $\boldsymbol{\lambda}^{k+1} = \mu^k\boldsymbol{r}(\boldsymbol{x}^k)$. Since μ is reduced to around machine precision and the aim is to reduce $\boldsymbol{r}(\boldsymbol{x})$ to zero, the components of $\boldsymbol{\lambda}$ become small. Contrast this with

Table 1: Test problems, the speed-up of the `Clp` primal simplex solver when ICA is used, and the percentage of solution time accounted for by ICA. Only for the problem names in bold does default `Clp` use ICA and the primal simplex solver.

Model	Speed-up	Idiot (%)	Model	Speed-up	Idiot (%)
CRE-B	2.6	28.9	PDS-40	1.3	5.0
DANO3MIP	1.4	3.6	PDS-80	1.0	0.1
DBIC1	1.5	40.6	PILOT87	1.3	2.5
DFL001	1.0	0.1	**QAP12**	2.5	0.6
FOME12	1.1	0.1	**QAP15**	4.0	0.1
FOME13	1.9	3.3	**SELF**	6.1	22.7
KEN-18	1.0	0.7	SGPF5Y6	1.4	4.8
L30	1.9	1.4	**STAT96V4**	1.7	1.2
LINF_520C	9.4	8.2	STORM_1000	4.5	0.8
LP22	1.4	1.9	STORM-125	4.1	10.1
MAROS-R7	0.9	7.8	STP3D	6.5	0.9
MOD2	1.4	2.7	TRUSS	0.8	17.1
NS1688926	1.4	1.0	WATSON_1	1.8	8.9
NUG15	4.2	0.1	WATSON_2	1.1	4.4
PDS-100	2.5	5.4	WORLD	1.3	2.0

the values of $\boldsymbol{\lambda}$ in the augmented Lagrangian method, as set out in Algorithm 1. These are updated by the value $\mu^k \boldsymbol{r}(\boldsymbol{x}^k)$ and converge to the (generally non-zero) Lagrange multipliers for the equations.

In ICA, when the values of $\boldsymbol{\lambda}$ are updated, the linear and quadratic functions of the residual $\boldsymbol{r}(\boldsymbol{x})$ in the ICA function (4) are respectively $\mu^k \boldsymbol{r}(\boldsymbol{x}^k)^T \boldsymbol{r}(\boldsymbol{x})$ and $(2\mu^k)^{-1} \boldsymbol{r}(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x})$. Thus, since the values of μ^k are soon significantly less than unity, the linear term becomes relatively negligible. In this way the ICA function reduces to the quadratic penalty function (3) and the later behaviour of ICA is akin to that of a simple quadratic penalty method.

3.2 Preliminary experiments

The effectiveness of ICA is assessed via experiments with `Clp` (Version 1.16.10), using a set of 30 representative LP test problems in Table 1. This is the set used by Huangfu and Hall in [10], with QAP15 replacing DCP2 because QAP problems are of particular interest and DCP2 is not a public test problem, and NUG15 replacing NUG12 for consistency with the choice of QAP problems used by Mittelmann [12]. The three problems NUG15, QAP12 and QAP15 are linearizations of quadratic assignment problems, where NUG15 and QAP15 differ only by row and column permutations. The experiments are carried out on an Intel i7-6700T processor rated at 2.80GHz with 16GB of available memory. In all cases the `Clp` presolve routine is run first, and is included in the total solution times.

To assess the effectiveness of ICA in speeding up the `Clp` primal simplex solver over all the test problems, total solution times were first recorded for `Clp` with the `-primals` option. This forces `Clp` to use the primal simplex solver but makes no use of ICA. To compare these with total solution time when `Clp` uses the primal simplex solver following ICA, it was necessary to edit the source code so that `Clp` is forced to use ICA and the primal simplex solver. Otherwise, `Clp` ran in its default state. The relative total solution times are given in the columns in Table 1 headed "Speed-up". The geometric mean speed-up is 1.9, demonstrating clearly the general value of ICA for the `Clp` primal simplex solver. Although ICA is of little value (speed-up below 1.25) for seven of the 30 problems, for only two of these problems does it lead to a small slow-down. However, for ten of the 30 problems the speed-up is at least 2.5, a huge improvement. The columns headed "Idiot (%)" give the percentage of the total solution time accounted for by ICA, the mean value being 6.2%. For five problems the percentage is ten or more, and this achieves a handsome speed-up in three cases. However, it does include TRUSS, for which ICA takes up 17% of an overall solution time that is 20% more than with the vanilla primal simplex solver. For only this problem can ICA be considered a significant and unwise investment. Of the ten problems where ICA results in a speed-up of at least 2.5, for only three does it account for at least ten percent of the total solution time. Indeed, for five of these problems ICA is no more than one percent of the total solution time.

This remarkably cheap way to improve the performance of the primal simplex solver is not always of value to `Clp` because, when it is run without command line options (other than the model file name), it decides whether to use its primal or dual simplex solver. When the former is used, `Clp` uses problem characteristics to decide whether to use ICA and, if used, to set parameter values for the algorithm. Default `Clp` chooses the primal simplex solver (and always performs ICA) for just the ten LP problems whose name is given in bold type. For half of these problems there is a speed-up of at least 2.5, so ICA contributes significantly to the ultimate performance of `Clp`. However, for five problems (CRE-B, PDS-100, STORM-125, STORM_1000 and STP3D), ICA yields a primal simplex speed-up of at least 2.5 but, when free to choose, `Clp` uses its dual simplex solver. In each case the dual simplex solver is at least as fast as using the primal simplex solver following ICA, the geometric mean superiority being a factor of 4.0, so the choice to use the dual simplex solver is justified.

Further evidence of the importance of ICA to the performance of `Clp` is given in Table 2, which gives the solution times from the Mittelmann benchmarks [12] for the three major commercial simplex solvers and `Clp` when applied to five notable problem instances. When solving LINF_520C, `Clp` is vastly faster than the three commercial solvers. For the three QAP linearizations (NUG15, QAP12 and QAP15), `Clp` is very much faster than `Cplex`. Finally, for SELF, `Clp` is significantly faster than the commercial solvers.

To asses the limiting behaviour of ICA as a means of finding a point that is both feasible and optimal, `Clp` was run with the `-idiot 200` option using the modified code that forces ICA to be used on all problems. The results are given in Table 3, where the columns headed "Residual" contain the final values of $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$. The columns headed "Objective" contain values of $(f - f^*)/\max\{1, |f^*|\}$ as a measure of how relatively close

Table 2: Solution times for Cplex-12.8.0, Gurobi-7.5.0, Xpress-8.4.0 and Clp-1.16.10 on five notable problem instances from the Mittelmann benchmarks (29/12/17) [12]

Model	Cplex	Gurobi	Xpress	Clp
LINF_520C	495	1057	255	35
NUG15	338	14	7	14
QAP12	26	1	1	5
QAP15	365	14	6	13
SELF	18	12	15	5

Table 3: Residual and relative objective error following ICA in Clp

Model	Residual	Objective	Model	Residual	Objective
CRE-B	1.3×10^{-9}	6.1×10^{-2}	PDS-40	7.0×10^{-8}	3.0×10^{-2}
DANO3MIP	6.1×10^{-10}	2.0×10^{-2}	PDS-80	2.2×10^{-7}	3.4×10^{-1}
DBIC1	3.8×10^{-1}	8.5×10^{-2}	PILOT87	2.1×10^{0}	6.8×10^{-1}
DFL001	1.1×10^{-9}	3.7×10^{-3}	QAP12	3.6×10^{-10}	1.7×10^{-1}
FOME12	6.4×10^{-9}	4.3×10^{-3}	QAP15	2.1×10^{-10}	2.8×10^{-3}
FOME13	1.2×10^{-8}	5.2×10^{-3}	SELF	5.7×10^{-5}	2.4×10^{-3}
KEN-18	5.4×10^{-8}	7.1×10^{-2}	SGPF5Y6	4.0×10^{-10}	2.1×10^{-1}
L30	1.1×10^{-9}	3.9×10^{0}	STAT96V4	3.0×10^{-3}	1.0×10^{0}
LINF_520C	1.1×10^{-1}	9.1×10^{-3}	STORM_1000	5.9×10^{-6}	5.9×10^{-2}
LP22	1.1×10^{-9}	1.3×10^{-3}	STORM-125	1.4×10^{0}	1.2×10^{-1}
MAROS-R7	4.0×10^{-9}	2.3×10^{-5}	STP3D	7.0×10^{-5}	2.7×10^{-2}
MOD2	3.9×10^{0}	2.1×10^{-1}	TRUSS	7.1×10^{-1}	3.2×10^{-1}
NS1688926	2.5×10^{-9}	4.8×10^{5}	WATSON_1	7.7×10^{-6}	8.7×10^{-1}
NUG15	2.1×10^{-10}	3.7×10^{-4}	WATSON_2	1.4×10^{-10}	9.7×10^{-1}
PDS-100	7.6×10^{-10}	3.7×10^{-4}	WORLD	4.3×10^{0}	5.5×10^{-1}

the final value of f is to the known optimal value f^*, referred to below as the objective error. This measure of optimality is clearly of no practical value because f^* is not known. However, it is instructive empirically, and motivates later theoretical analysis. The geometric mean of the residuals is 1.2×10^{-6} and the geometric mean of the objective error measures is 6.1×10^{-2}.

For 17 of the 30 problems in Table 3, the norm of the final residual is less than 10^{-7}. Since this is the default primal feasibility tolerance for the Clp simplex solver, ICA can be considered to have obtained an acceptably feasible point. Among these problems, the objective error ranges between 4.8×10^{5} for NS1688926 and 2.3×10^{-5} for MAROS-R7, with only eight problems having a value less than 10^{-2}. Thus, even if ICA yields a feasible point, it may be far from being optimal. A single quality measure for the point returned by ICA is convenient, and this is provided by the product of the residual and objective error, conveniently referred to as the "solution error". As illustrated by the distribution of the objective errors and residual in Figure 1, it is unsurprising that there

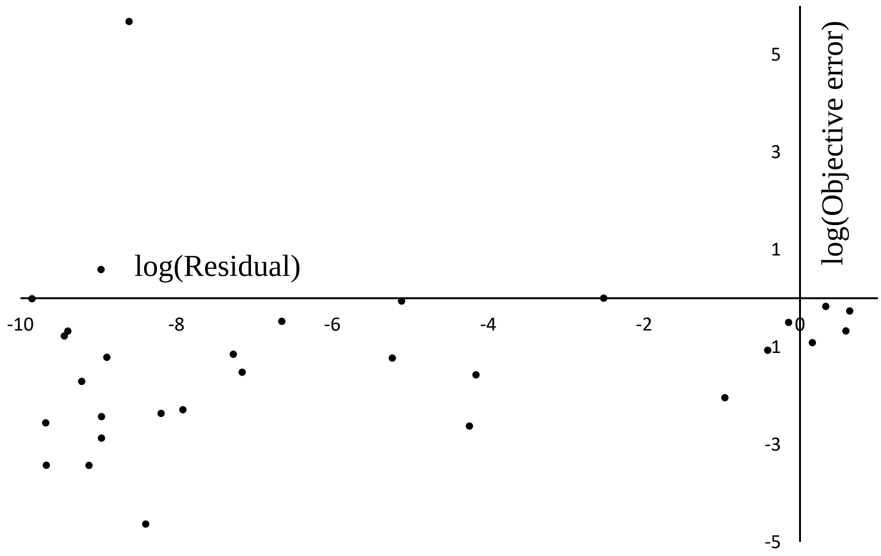

Figure 1: Distribution of residual and objective errors

are no problems for which a low value of this product corresponds to an accurate optimal objective function value but large residual.

The observations resulting from the experiments above yield three questions that merit further study. First, since ICA yields a near-optimal solution for some problems, to what extent does it possess theoretical optimality and convergence properties? Second, since ICA performs particularly well for some problems and badly for others, which problem features might characterize this behaviour? Third, for any problem class where ICA appears to perform well, might this be valuable other than in the context of crash-starting the primal simplex method? These questions are addressed in the remainder of this paper.

3.3 Analysis

In analysing ICA, the initial focus is the function (5). Fully expanded, this is the quadratic function

$$h(\boldsymbol{x}) = \frac{1}{2\mu}\boldsymbol{x}^T A^T A\boldsymbol{x} + (\boldsymbol{c}^T + \boldsymbol{\lambda}^T A - \frac{1}{\mu}\boldsymbol{b}^T A)\boldsymbol{x} - \boldsymbol{\lambda}^T\boldsymbol{b} + \frac{1}{2\mu}\boldsymbol{b}^T\boldsymbol{b}.$$

Although convexity of the function follows from the Hessian matrix A^TA being positive semi-definite, the Hessian has rank $m < n$. However, the possibility of unboundedness of $h(\boldsymbol{x})$ on $\boldsymbol{x} \geq \boldsymbol{0}$ can be discounted as follows. First, observe that unboundedness could only occur in non-negative directions of zero curvature, so they must satisfy $A\boldsymbol{d} = \boldsymbol{0}$. Hence $h(\boldsymbol{x} + \alpha\boldsymbol{d}) = h(\boldsymbol{x}) + \alpha\boldsymbol{c}^T\boldsymbol{d}$, which, if unbounded below for increasing α, implies unboundedness of the LP along the ray $\boldsymbol{x}+\alpha\boldsymbol{d}$ from any point $\boldsymbol{x}\geq\boldsymbol{0}$ satisfying $A\boldsymbol{x}=\boldsymbol{b}$. Thus, as long as the LP is neither infeasible nor unbounded, $h(\boldsymbol{x})$ is bounded below on $\boldsymbol{x}\geq\boldsymbol{0}$.

For some problems, the size of the residual and objective measures in Table 3 indicate that ICA has found a point that is close to being optimal. It is therefore of interest to know whether ICA possesses theoretical optimality and convergence properties. With approximate minimization of the ICA function (5), it is not conducive to detailed mathematical analysis. However, Theorem 1 shows that if the ICA function is minimized exactly and an optimal solution to the LP exists, every limit point of the sequence $\{\boldsymbol{x}^k\}$ is a solution to the problem.

3.3.1 Notes

- During each iteration, at most one of the parameters μ and $\boldsymbol{\lambda}$ is updated: in `Clp`, μ is updated once every few (e.g. 3 or 6) iterations. How often μ is updated does not affect the validity of the proof as long as $\{\mu^k\}\to 0$ as $k\to\infty$ and $\boldsymbol{\lambda}$ is updated at least once every W iterations for some constant $W\geq 1$.

- In the statement of Algorithm 2 it is said that ω is larger than 1. This is not required for the proof, which would still hold in the case of non-monotonicity of $\{\mu^k\}$ as long as $\{\mu^k\}\to 0$ as $k\to\infty$.

Theorem 1. *Suppose that $\boldsymbol{x}^k$ is the exact global minimizer of $h^k(\boldsymbol{x})$ for each $k=1,2,\ldots$ and that $\{\mu^k\}\to 0$ as $k\to\infty$. Then every limit point of the sequence $\{\boldsymbol{x}^k\}$ is a solution to problem (1).*

Proof. Let $\bar{\boldsymbol{x}}$ be a solution of (1) so that, for all feasible $\boldsymbol{x}$, $\boldsymbol{c}^T\bar{\boldsymbol{x}}\leq\boldsymbol{c}^T\boldsymbol{x}$. For each k, $\boldsymbol{x}^k$ is the exact global minimizer for

$$\begin{array}{ll}\min\limits_{\boldsymbol{x}} & h^k(\boldsymbol{x})=\boldsymbol{c}^T\boldsymbol{x}+{\boldsymbol{\lambda}^k}^T\boldsymbol{r}(\boldsymbol{x})+\dfrac{1}{2\mu^k}\boldsymbol{r}(\boldsymbol{x})^T\boldsymbol{r}(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{x}\geq\boldsymbol{0},\end{array}\tag{6}$$

and, since $\bar{\boldsymbol{x}}$ is feasible for (1), it is also feasible for (6). Thus, since $h^k(\boldsymbol{x}^k)\leq h^k(\bar{\boldsymbol{x}})$ for each k, it follows that

$$\boldsymbol{c}^T\boldsymbol{x}^k+{\boldsymbol{\lambda}^k}^T\boldsymbol{r}(\boldsymbol{x}^k)+\frac{1}{2\mu^k}\boldsymbol{r}(\boldsymbol{x}^k)^T\boldsymbol{r}(\boldsymbol{x}^k)\leq\boldsymbol{c}^T\bar{\boldsymbol{x}}+{\boldsymbol{\lambda}^k}^T\boldsymbol{r}(\bar{\boldsymbol{x}})+\frac{1}{2\mu^k}\boldsymbol{r}(\bar{\boldsymbol{x}})^T\boldsymbol{r}(\bar{\boldsymbol{x}}).\tag{7}$$

Since $\bar{\boldsymbol{x}}$ is a solution of (1), $\boldsymbol{r}(\bar{\boldsymbol{x}}) = 0$ and (7) simplifies to

$$\begin{aligned} \boldsymbol{c}^T\boldsymbol{x}^k + {\boldsymbol{\lambda}^k}^T\boldsymbol{r}(\boldsymbol{x}^k) + \frac{1}{2\mu^k}\boldsymbol{r}(\boldsymbol{x}^k)^T\boldsymbol{r}(\boldsymbol{x}^k) &\le \boldsymbol{c}^T\bar{\boldsymbol{x}} \\ \Longrightarrow \qquad \frac{1}{2\mu^k}\boldsymbol{r}(\boldsymbol{x}^k)^T\boldsymbol{r}(\boldsymbol{x}^k) &\le \boldsymbol{c}^T\bar{\boldsymbol{x}} - \boldsymbol{c}^T\boldsymbol{x}^k - {\boldsymbol{\lambda}^k}^T\boldsymbol{r}(\boldsymbol{x}^k) \\ \Longrightarrow \qquad \boldsymbol{r}(\boldsymbol{x}^k)^T\boldsymbol{r}(\boldsymbol{x}^k) &\le 2\mu^k\big(\boldsymbol{c}^T\bar{\boldsymbol{x}} - \boldsymbol{c}^T\boldsymbol{x}^k - {\boldsymbol{\lambda}^k}^T\boldsymbol{r}(\boldsymbol{x}^k)\big). \end{aligned} \tag{8}$$

At the end of the previous iteration of the loop in Algorithm 2, one of the two parameters μ and $\boldsymbol{\lambda}$ was updated. If during the previous iteration the update was of $\boldsymbol{\lambda}$, then

$$\boldsymbol{\lambda}^k = \mu^{k-1}\boldsymbol{r}(\boldsymbol{x}^{k-1}).$$

Alternatively, during the previous iteration, μ was updated and $\boldsymbol{\lambda}$ remained unchanged, so $\boldsymbol{\lambda}^k = \boldsymbol{\lambda}^{k-1}$. Consider iterations $k-W, \dots, k-1$ of the loop and let p be the index of the latest iteration when $\boldsymbol{\lambda}$ was changed. Then for some p satisfying $k - W \le p < k$,

$$\boldsymbol{\lambda}^k = \mu^p\boldsymbol{r}(\boldsymbol{x}^p).$$

Suppose $\boldsymbol{x}^*$ is a limit point of $\{\boldsymbol{x}^k\}$, so that there is an infinite subsequence $\mathcal{K}$ such that

$$\lim_{k\in\mathcal{K}} \boldsymbol{x}^k = \boldsymbol{x}^*.$$

Taking the limit in inequality (8) gives

$$\lim_{k\in\mathcal{K}} \boldsymbol{r}(\boldsymbol{x}^k)^T\boldsymbol{r}(\boldsymbol{x}^k) \le \lim_{k\in\mathcal{K}} 2\mu^k\big(\boldsymbol{c}^T\bar{\boldsymbol{x}} - \boldsymbol{c}^T\boldsymbol{x}^k - {\boldsymbol{\lambda}^k}^T\boldsymbol{r}(\boldsymbol{x}^k)\big). \tag{9}$$

For all $k > W$ there is an index p with $k - W \le p < k$ and $\boldsymbol{\lambda}^k = \mu^p\boldsymbol{r}(\boldsymbol{x}^p)$, so the value of $\boldsymbol{\lambda}^k$ can be substituted in (9) to give

$$\begin{aligned} \lim_{k\in\mathcal{K}} \boldsymbol{r}(\boldsymbol{x}^k)^T\boldsymbol{r}(\boldsymbol{x}^k) &\le \lim_{k\in\mathcal{K}} 2\mu^k\big(\boldsymbol{c}^T\bar{\boldsymbol{x}} - \boldsymbol{c}^T\boldsymbol{x}^k - {\boldsymbol{\lambda}^k}^T\boldsymbol{r}(\boldsymbol{x}^k)\big) \\ \Longrightarrow \qquad \boldsymbol{r}(\boldsymbol{x}^*)^T\boldsymbol{r}(\boldsymbol{x}^*) &= \lim_{k\in\mathcal{K}} 2\mu^k\big(\boldsymbol{c}^T\bar{\boldsymbol{x}} - \boldsymbol{c}^T\boldsymbol{x}^k - \mu^p\boldsymbol{r}(\boldsymbol{x}^p)^T\boldsymbol{r}(\boldsymbol{x}^k)\big) \\ \Longrightarrow \qquad \|\boldsymbol{r}(\boldsymbol{x}^*)\|^2 &= \lim_{k\in\mathcal{K}} 2\mu^k(\boldsymbol{c}^T\bar{\boldsymbol{x}} - \boldsymbol{c}^T\boldsymbol{x}^k) - \lim_{k\in\mathcal{K}} 2\mu^k\mu^p\boldsymbol{r}(\boldsymbol{x}^p)^T\boldsymbol{r}(\boldsymbol{x}^k) = 0, \end{aligned}$$

since $\{\mu^k\} \to 0$ for $k \in \mathcal{K}$, so $A\boldsymbol{x}^* = \boldsymbol{b}$. For each $\boldsymbol{x}^k$, $\boldsymbol{x}^k \ge \boldsymbol{0}$, so after taking the limit, $\boldsymbol{x}^* \ge \boldsymbol{0}$. Thus, $\boldsymbol{x}^*$ is feasible for (1). To show optimality of $\boldsymbol{r}(\boldsymbol{x}^*)$, from (7)

$$\begin{aligned} \boldsymbol{c}^T\boldsymbol{x}^k + {\boldsymbol{\lambda}^k}^T\boldsymbol{r}(\boldsymbol{x}^k) + \frac{1}{2\mu^k}\boldsymbol{r}(\boldsymbol{x}^k)^T\boldsymbol{r}(\boldsymbol{x}^k) &\le \boldsymbol{c}^T\bar{\boldsymbol{x}} \\ \Longrightarrow \qquad \lim_{k\in\mathcal{K}}\big(\boldsymbol{c}^T\boldsymbol{x}^k + {\boldsymbol{\lambda}^k}^T\boldsymbol{r}(\boldsymbol{x}^k) + \frac{1}{2\mu^k}\boldsymbol{r}(\boldsymbol{x}^k)^T\boldsymbol{r}(\boldsymbol{x}^k)\big) &\le \lim_{k\in\mathcal{K}} \boldsymbol{c}^T\bar{\boldsymbol{x}} \\ \Longrightarrow \qquad \boldsymbol{c}^T\boldsymbol{x}^* + \lim_{k\in\mathcal{K}} {\boldsymbol{\lambda}^k}^T\boldsymbol{r}(\boldsymbol{x}^k) + \lim_{k\in\mathcal{K}} \frac{1}{2\mu^k}\boldsymbol{r}(\boldsymbol{x}^k)^T\boldsymbol{r}(\boldsymbol{x}^k) &\le \boldsymbol{c}^T\bar{\boldsymbol{x}}. \end{aligned} \tag{10}$$

For all $k > W$ there is an index p with $k - W \le p < k$ and $\boldsymbol{\lambda}^k = \mu^p \boldsymbol{r}(\boldsymbol{x}^p)$ such that

$$\lim_{k\in\mathcal{K}} {\boldsymbol{\lambda}^k}^T \boldsymbol{r}(\boldsymbol{x}^k) = \lim_{k\in\mathcal{K}} \mu^p \boldsymbol{r}(\boldsymbol{x}^p)^T \boldsymbol{r}(\boldsymbol{x}^k) = 0,$$

since $\{\mu^k\} \to 0$ for $k = 1, 2, \ldots$ and $p \to \infty$ as $k \to \infty$. This value can be substituted in (10) to give

$$\boldsymbol{c}^T \boldsymbol{x}^* + \lim_{k\in\mathcal{K}} \frac{1}{2\mu^k} \boldsymbol{r}(\boldsymbol{x}^k)^T \boldsymbol{r}(\boldsymbol{x}^k) \le \boldsymbol{c}^T \bar{\boldsymbol{x}}.$$

For each k, $\mu^k > 0$ and $\boldsymbol{r}(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}) \ge 0$ for each $\boldsymbol{x}$, so that

$$\frac{1}{2\mu^k} \boldsymbol{r}(\boldsymbol{x}^k)^T \boldsymbol{r}(\boldsymbol{x}^k) \ge 0 \quad \forall k \implies \boldsymbol{c}^T \boldsymbol{x}^* \le \boldsymbol{c}^T \boldsymbol{x}^* + \lim_{k\in\mathcal{K}} \frac{1}{2\mu^k} \boldsymbol{r}(\boldsymbol{x}^k)^T \boldsymbol{r}(\boldsymbol{x}^k) \le \boldsymbol{c}^T \bar{\boldsymbol{x}}.$$

Consequently, $\boldsymbol{x}^*$ is feasible for (1) and has an objective value less than or equal to the optimal value $\boldsymbol{c}^T \bar{\boldsymbol{x}}$, so $\boldsymbol{x}^*$ is a solution of (1). □

4 Fast approximate solution of LP problems

Although Theorem 1 establishes an important "best case" result for the behaviour of ICA, the results in Table 3 show that this is far from being representative of its practical performance. For some problems ICA yields a near-optimal point; for others it terminates at a point that is far from being feasible. Which problem characteristics might explain this behaviour and, if it is seen to perform well for a whole class of problems, to what extent is this of further value?

4.1 Problem characteristics affecting the performance of the Idiot crash

There is a clear relation between the condition number of the matrix A and the solution error of the point returned by ICA. Of the problems in Table 3, all but STORM_1000 are sufficiently small for the condition of A (after the `Clp` presolve) to be computed with the resources available to the authors. These values are plotted against the solution error in Figure 2, where the solution error is the product of the residual and (relative) objective error introduced in Section 3.2. Figure 2 clearly shows that the problems solved accurately have low condition number. Notable amongst these are the QAPs which, with the exception of MAROS-R7, have very much the smallest condition numbers of the 29 problems in Table 3 for which condition numbers could be computed.

Nocedal and Wright [13, p.512] observe that "there has been a resurgence of interest in penalty methods, in part because of their ability to handle degenerate problems". However, analysis of optimal basic solutions of the problems in Table 3 showed no meaningful correlation between their primal or dual degeneracy and accuracy of the point returned by ICA.

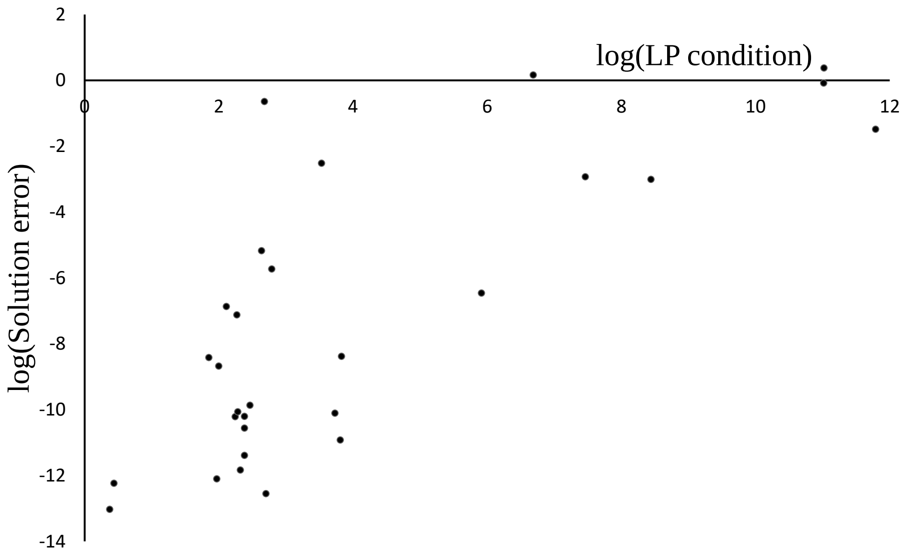

Figure 2: Solution error and LP condition

4.2 The Idiot crash on QAPs

Since ICA yields a near-optimal point for the three QAPs in Table 3, it is of interest to know the extent to which this behaviour is typical of the whole class of such problems, and its practical value. Both of these issues are explored in this section.

Quadratic assignment problems

The quadratic assignment problem (QAP) is a combinatorial optimization problem, being a special case of the facility location problem. It concerns a set of facilities and a set of locations. For each pair of locations there is a distance, and for each pair of facilities there is a weight or flow specified, for instance the number of items transported between the two facilities. The problem is to assign all facilities to different locations so that the sum of the distances multiplied by the corresponding flows is minimized. QAPs are well known for being very difficult to solve, even for small instances. They are NP-hard and the travelling salesman problem can be seen as a special case. Often, rather than the quadratic problem itself, an equivalent linearization is solved. A comprehensive survey of QAP problems and their solution is given by Loiola et al. [11].

The test problems NUG15, QAP12 and QAP15 referred to above are examples of the Adams and Johnson linearization [1]. Although there are many specialized techniques for solving QAP problems, and alternative linearizations, the popular Adams and Johnson linearization is known to be hard to solve using the simplex method or interior point methods [16]. Table 4 gives various performance measures for ICA when applied to

Table 4: Performance of the Idiot crash on QAP linearizations

Model	Rows	Columns	Optimum	Residual	Objective	Error	Time
NUG05	210	225	50.00	9.4×10^{-9}	50.01	1.5×10^{-4}	0.04
NUG06	372	486	86.00	7.8×10^{-9}	86.01	1.2×10^{-4}	0.11
NUG07	602	931	148.00	7.9×10^{-9}	148.64	4.3×10^{-3}	0.25
NUG08	912	1613	203.50	7.0×10^{-9}	204.41	4.5×10^{-3}	0.47
NUG12	3192	8856	522.89	8.8×10^{-9}	523.86	1.8×10^{-3}	2.58
NUG15	6330	22275	1041.00	8.9×10^{-9}	1041.38	3.7×10^{-4}	5.13
NUG20	15240	72600	2182.00	7.5×10^{-9}	2183.03	4.7×10^{-4}	14.94
NUG30	52260	379350	4805.00	1.1×10^{-8}	4811.41	1.3×10^{-3}	82.28

the Nugent [14] problems, using the default iteration limit of `Clp`. The first of these is the value of the residual $\|\boldsymbol{Ax}-\boldsymbol{b}\|_2$ at the point obtained by ICA, which is clearly feasible to within the `Clp` simplex tolerance. The objective function value and relative error are also given, and the latter is well within 1%. Finally, the time for ICA is given. Whilst this is growing, ICA clearly obtains a near-optimal solution for QAP instances NUG20 and NUG30, which cannot be solved with commercial simplex or interior point implementations on the machine used for ICA experiments because of excessive time or memory requirements.

There is currently no practical measure of the point obtained by ICA that gives any guarantee it can be taken as a near-optimal solution of the problem. The result of Theorem 1 cannot be used because the major iteration minimization is approximate, and the major iterations are terminated rather than being performed to the limit. Clearly the measure of objective error in Table 4 requires knowledge of the optimal objective function value. What can be guaranteed, however, is that since the point returned is feasible, the corresponding objective value is an upper bound on the optimal objective function value. With the aim of identifying an interval containing the optimal objective function value, ICA was applied to the dual of the linearization. Although it obtained points that were feasible for the dual problems to within the `Clp` simplex tolerance, the objective values were far from being optimal, so the lower bounds thus obtained were too weak to be of value.

5 Conclusions

Forrest's aim in developing ICA for LP problems was to determine a point that, when used to obtain a starting basis for the primal revised simplex method, results in a significant reduction in the time required to solve the problem. This paper has distilled the essence of ICA and presented it in algorithmic form for the first time. Practical experiments have demonstrated that, for some large-scale LP test problems, Forrest's aim is achieved. For LP problems when ICA is not advantageous, this is identified without meaningful detriment to the performance of `Clp`. For the best case in which ICA subproblems are solved exactly, Theorem 1 shows that every limit point of the sequence of ICA iterations is a solution of the corresponding LP problem. It is observed empirically that, typically, the lower the condition of the constraint matrix A, the closer the point obtained by ICA is to being an optimal solution of the LP problem. For linearizations of quadratic assignment problems, it has been demonstrated that ICA consistently yields near-optimal solutions, achieving this in minutes for instances that are intractable on the same machine using commercial LP solvers. Thus, in addition to achieving Forrest's initial aim, ICA is seen as being useful in its own right as a fast solver for amenable LP problems.

Acknowledgements: The Authors would like to thank the Editor-in-Chief for his patience throughout the review process, and one particular Reviewer for his/her painstaking work on revisions of the manuscript of this paper.

References

[1] W. P. Adams and T. A. Johnson. Improved linear programming-based lower bounds for the quadratic assignment problem. *DIMACS series in discrete mathematics and theoretical computer science*, 16:43–75, 1994.

[2] COIN-OR. Clp. `https://projects.coin-or.org/Clp`, 2014. Accessed: 16/02/2018.

[3] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A). *Springer Series in Computational Mathematics*, 17, 1992.

[4] Y. G. Evtushenko, A. I. Golikov, and N. Mollaverdy. Augmented Lagrangian method for large-scale linear programming problems. *Optimization Methods and Software*, 20(4–5):515–524, 2005.

[5] J. J. H. Forrest. [cbc] coinstructuredmodel crash. `https://list.coin-or.org/pipermail/cbc/2014-March/001297.html`, 2014. Accessed: 16/02/2018.

[6] J. J. H. Forrest. Idiot.hpp. `https://projects.coin-or.org/Clp/browser/trunk/Clp/src/Idiot.hpp?rev=2144`, 2014. Accessed: 16/02/2018.

[7] J. J. H. Forrest. GAMS-CBC. `https://www.gams.com/latest/docs/S_CBC.html`, 2018. Accessed: 16/02/2018.

[8] O. Güler. Augmented Lagrangian algorithms for linear programming. *Journal of Optimization Theory and Applications*, 75(3):445–470, 1992.

[9] M. R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969.

[10] Q. Huangfu and J.A.J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018.

[11] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657–690, 2007.

[12] H. D. Mittelmann. Benchmarks for optimization software. `http://plato.la.asu.edu/bench.html`, 2018. Accessed: 16/02/2018.

[13] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2 edition, 2006.

[14] C. E. Nugent, T. E. Vollmann, and J. Ruml. An experimental comparison of techniques for the assignment of facilities to locations. *Operations Research*, 16(1):150–173, 1968.

[15] M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298, London, 1969. Academic Press.

[16] M. G. C. Resende, K. G. Ramakrishnan, and Z. Drezner. Computing lower bounds for the quadratic assignment problem with an interior point algorithm for linear programming. *Operations Research*, 43(5):781–791, 1995.

[17] R. T. Rockafellar. Augmented Lagrange multiplier functions and duality in nonconvex programming. *SIAM Journal on Control*, 12(2):268–285, 1974.