



THE UNIVERSITY *of* EDINBURGH

Edinburgh Research Explorer

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*, pp. 488-501. <https://doi.org/10.1080/10556788.2019.1604702>

Digital Object Identifier (DOI):

[10.1080/10556788.2019.1604702](https://doi.org/10.1080/10556788.2019.1604702)

Link:

[Link to publication record in Edinburgh Research Explorer](#)

Document Version:

Early version, also known as pre-print

Published In:

Optimization Methods and Software

General rights

Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy

The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.



A 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

This paper provides the first meaningful documentation and analysis of an established technique which aims to obtain an approximate solution to linear programming problems prior to applying 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. There is also a discussion of 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 when solving discrete and nonlinear optimization problems. Finding an approximate solution particularly rapidly is valuable as a “crash start” to an exact solution method. There are also applications where it is preferable to trade off solution accuracy for a significant increase in speed.

The “Idiot” crash within the open source LP solver `Clp` [2] aims to find an approximate solution of an LP problem prior to applying the primal revised simplex method. In essence, the crash replaces the minimization of the linear objective subject to linear constraints by the 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 which 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

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

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{c} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$ 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 applying the primal revised simplex method. Beyond its definition as source code [5], a 2014 reference by Forrest to having given “a bad talk on it years ago” [4], as well as a few sentences of comments in the source code, documentation for the mixed-integer programming solver `Cbc` [6], and a public email [4], the Idiot crash lacks documentation or analysis. Forrest’s comments stress the unsophisticated nature of the crash and only hint at its usefulness as a means of possibly obtaining an approximate solution of an LP problem prior to applying the primal simplex algorithm. However, for several test problems used in the Mittelmann benchmarks [11], `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, the Idiot crash 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 which, other than satisfying the bounds $\mathbf{x} \geq \mathbf{0}$, has no guaranteed properties. In particular, it satisfies no known bound on the residual $\|A\mathbf{x} - \mathbf{b}\|_2$ 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 via the Idiot crash.

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 [4], a more general quadratic function of constraint violations is minimized. This includes a linear term, making the Idiot 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

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

the quadratic penalty method minimizes

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

for an increasing sequence of positive values of μ . If \mathbf{x}^k is the global minimizer of $\phi(\mathbf{x}, \mu^k)$ and $\mu^k \downarrow 0$ then Nocedal and Wright [12] show that every limit point \mathbf{x}^* of the sequence $\{\mathbf{x}^k\}$ is a global solution of (2). The subproblem of minimizing $\phi(\mathbf{x}, \mu^k)$ is known to be increasingly ill-conditioned as smaller values of μ^k are used [12] and this is one motivation for the use of the augmented Lagrangian method.

The augmented Lagrangian method

The augmented Lagrangian method, outlined in Algorithm 1, was originally presented as an approach to solving nonlinear programming problems. It was first proposed by Hestenes in his survey of multiplier and gradient methods [8] and then fully interpreted and analysed, first by Powell [14] and then by Rockafellar [16]. The augmented Lagrangian function (4) is in a sense a combination of the Lagrangian function and the quadratic penalty function [12]. It is the quadratic penalty function with an explicit estimate of the Lagrange multipliers $\boldsymbol{\lambda}$.

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

Although originally intended for nonlinear programming problems, the augmented Lagrangian method has also been applied to linear programming problems [3, 7]. However, neither article assesses its performance on large scale practical LP problems.

3 The Idiot crash

This section presents the Idiot crash algorithm, followed by some practical and mathematical analysis of its behaviour. Experiments assess the extent to which the Idiot crash accelerates the time required to solve representative LP test problems using the primal

Algorithm 1 The augmented Lagrangian algorithm

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

revised 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 which has an optimal solution.

3.1 The Idiot crash algorithm

The Idiot crash algorithm minimizes the function

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

subject to the bounds $\mathbf{x} \geq \mathbf{0}$ for sequences of values of parameters $\boldsymbol{\lambda}$ and μ . The minimization is not performed exactly, but approximately, by minimizing with respect to each component of \mathbf{x} in turn. The general structure of the algorithm is set out in Algorithm 2. The number of iterations is determined heuristically according to the size of the LP and progress of the algorithm. Unless the Idiot crash is abandoned after up to 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 the Idiot crash implemented in `C1p` has several additional features. At the start of the crash, the approximate component-wise minimization is performed twice for each component of \mathbf{x} . If a 10% decrease in primal infeasibility is not observed in about 30 iterations, it is considered that the Idiot crash would not be beneficial and the simplex algorithm is started from the origin. If a 10% reduction of the primal infeasibility is observed, then the mechanism for approximate minimization is adjusted. During each subsequent iteration, the function $h(\mathbf{x})$ is minimized for each component 105 times. There is no indication why this particular value was chosen. However, one of the features is the possibility to decrease this number, so to minimize for each component fewer than 105 times. From the 50th minimization onwards a check is performed after the function

Algorithm 2 The Idiot penalty algorithm

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

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

If a criterion is satisfied (see 3.3.1) update μ :

$$\mu^{k+1} = \mu^k / \omega, \text{ for some factor } \omega > 1$$

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k$$

Else update $\boldsymbol{\lambda}$:

$$\mu^{k+1} = \mu^k$$

$$\boldsymbol{\lambda}^{k+1} = \mu^k \mathbf{r}(\mathbf{x}^k)$$

End

is minimized 10 times for each component. 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, one of 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 during it. 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 \mathbf{x} at the end of each iteration. These features are omitted from this paper since experiments showed that they have little or no 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 it is not. The dominant computational cost corresponds to a matrix-vector product with the matrix A for each set of minimizations over the components of \mathbf{x} .

Relation to augmented Lagrangian and quadratic penalty function methods

In form, the augmented Lagrangian function (4) and Idiot 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 the Idiot crash, new values of $\boldsymbol{\lambda}$ are given by $\boldsymbol{\lambda}^{k+1} = \mu^k \mathbf{r}(\mathbf{x}^k)$. Since μ is reduced to around machine precision and the aim is to reduce $\mathbf{r}(\mathbf{x})$ to zero, the components of $\boldsymbol{\lambda}$ become small. Contrast this with the values of $\boldsymbol{\lambda}$ in the augmented Lagrangian method, as set out in Algorithm 1. These are updated by the value $\mu^k \mathbf{r}(\mathbf{x}^k)$ and converge to the (generally non-zero) Lagrange multipliers for the equations.

In the Idiot algorithm, when the values of $\boldsymbol{\lambda}$ are updated, the linear and quadratic functions of the residual $\mathbf{r}(\mathbf{x})$ in the Idiot function (4) are respectively $\mu^k \mathbf{r}(\mathbf{x}^k)^T \mathbf{r}(\mathbf{x})$ and $(2\mu^k)^{-1} \mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x})$. Thus, since the values of μ^k are soon significantly less than unity,

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

Table 1: Test problems, the speed-up of the primal simplex solver when the Idiot crash is used, and the percentage of solution time accounted for by the Idiot crash

the linear term becomes relatively negligible. In this way the Idiot objective function reduces to the quadratic penalty function (3) and the later behaviour of the Idiot crash is akin to that of a simple quadratic penalty method.

3.2 Preliminary experiments

The effectiveness of the Idiot crash 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 [9], with `QAP15` replacing `DCP2` due to `QAP` problems being of particular interest and the latter not being a public test problem, and `NUG15` replacing `NUG12` for consistency with the choice of `QAP` problems used by Mittelman [11]. The three problems `NUG15`, `QAP12` and `QAP15` are linearizations of quadratic assignment problems, where `NUG15` and `QAP15` differ only via row and column permutations. The experiments in this paper are carried out on a 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 the Idiot crash in speeding up the `Clp` primal simplex solver over all the test problems, total solution times were first recorded when running `Clp` with the `-primals` option. This forces `Clp` to use the primal simplex solver but makes no use of the Idiot crash. To compare these with total solution time when `Clp` uses the primal simplex solver following the Idiot crash, it was necessary to edit the source code so that `Clp` is forced to use the Idiot crash and primal simplex solver. However, otherwise, it ran as 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 the Idiot crash for the `C1p` primal simplex solver. Although the Idiot crash is of little or no 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 as a consequence of using the Idiot crash. The columns headed "Idiot (%)" give the percentage of the total solution time accounted for by the Idiot crash, 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 the Idiot crash takes up 17% of an overall solution time which is 20% more than when using the vanilla primal simplex solver. For only this problem can the Idiot crash be considered a significant and unwise investment. Of the ten problems where the Idiot crash 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 the Idiot crash 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 `C1p` since, 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, `C1p` uses problem characteristics to decide whether to use the Idiot crash and, if used, to set parameter values for the algorithm. Default `C1p` chooses the primal simplex solver (and always performs the Idiot crash) 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 the Idiot crash contributes significantly to the ultimate performance of `C1p`. However, for five problems (`CRE-B`, `PDS-100`, `STORM-125`, `STORM_1000` and `STP3D`), the Idiot crash yields a primal simplex speed-up of at least 2.5 but, when free to choose, `C1p` uses its dual simplex solver. In each case the dual simplex solver is at least as fast as using the primal simplex solver following the Idiot crash, 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 the Idiot crash to the performance of `C1p` is given in Table 2, which gives the solution times from the Mittelmann benchmarks [11] for the three major commercial simplex solvers and `C1p` when applied to five notable problem instances. When solving `LINF_520C`, `C1p` is vastly faster than the three commercial solvers. For the three QAP linearisations (`NUG15`, `QAP12` and `QAP15`), `C1p` is very much faster than `Cplex`. Finally, for `SELF`, `C1p` is significantly faster than the commercial solvers.

To assess the limiting behaviour of the Idiot crash as a means of finding a point which is both feasible and optimal, `C1p` was run with the `-idiot 200` option using the modified code which forces the Idiot crash to be used on all problems. The results are given in Table 3, where the columns headed "Residual" contains the final values of $\|A\mathbf{x} - \mathbf{b}\|_2$. The columns headed "Objective" contains values of $(f - f^*) / \max\{1, |f^*|\}$ as a measure of how relatively close 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, since 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

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 2: Performance of Cplex-12.8.0, Gurobi-7.5.0, Xpress-8.4.0 and Clp-1.16.10 on five notable problem instances from the Mittelman benchmarks (29/12/17) [11]

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}

Table 3: Residual and objective error following the Idiot crash in Clp

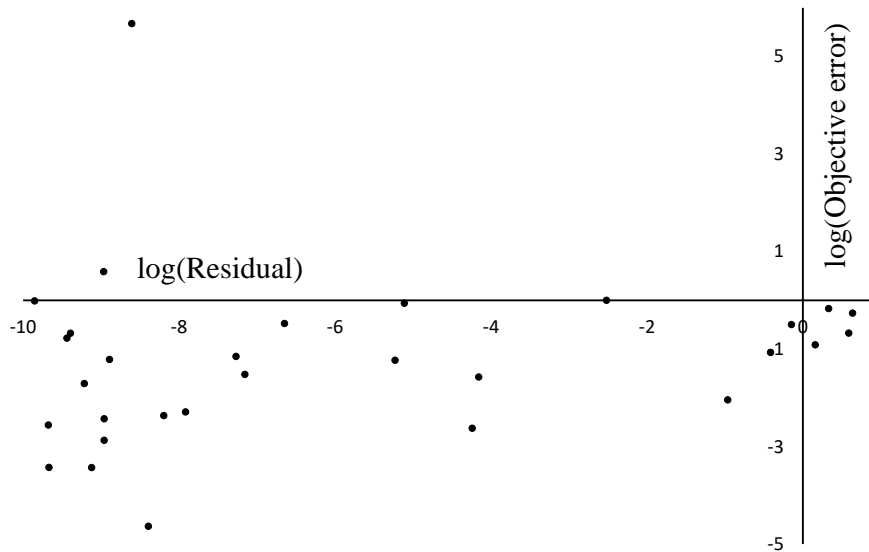


Figure 1: Distribution of residual and objective errors

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 `C1p` simplex solver, the Idiot crash can be considered to have obtained an acceptably feasible point. Amongst 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 the Idiot crash yields a feasible point, it may be far from being optimal. A single quality measure for the point returned by the Idiot crash is convenient, and this is provided by the product of the residual and objective error. As illustrated by the distribution of the objective errors and residual in Figure 1, it is unsurprising that there 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 which merit further study. Firstly, since the Idiot crash yields a near-optimal solution for some problems, to what extent does the Idiot crash possess theoretical optimality and convergence properties? Secondly, since the Idiot crash performs particularly well for some problems and badly for others, what problem features might characterise this behaviour? Thirdly, for any problem class where the Idiot crash appears to perform well, might this be valuable? These questions are addressed in the remainder of this paper.

3.3 Analysis

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

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

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

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

3.3.1 Notes

- During each iteration of the loop at most one of the two 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 $\{\mu^k\} \rightarrow 0$ as $k \rightarrow \infty$ and $\boldsymbol{\lambda}$ is updated at least once every W iterations for some constant W .
- In the statement of Algorithm 2 it is said that the constant 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 $\{\mu^k\} \rightarrow 0$ as $k \rightarrow \infty$.

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

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

$$\begin{aligned} \min_{\mathbf{x}} \quad & h^k(\mathbf{x}) = \mathbf{c}^T \mathbf{x} + \boldsymbol{\lambda}^k{}^T \mathbf{r}(\mathbf{x}) + \frac{1}{2\mu^k} \mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{6}$$

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

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

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

$$\begin{aligned} \mathbf{c}^T \mathbf{x}^k + \boldsymbol{\lambda}^{kT} \mathbf{r}(\mathbf{x}^k) + \frac{1}{2\mu^k} \mathbf{r}(\mathbf{x}^k)^T \mathbf{r}(\mathbf{x}^k) &\leq \mathbf{c}^T \bar{\mathbf{x}} \\ \implies \frac{1}{2\mu^k} \mathbf{r}(\mathbf{x}^k)^T \mathbf{r}(\mathbf{x}^k) &\leq \mathbf{c}^T \bar{\mathbf{x}} - \mathbf{c}^T \mathbf{x}^k - \boldsymbol{\lambda}^{kT} \mathbf{r}(\mathbf{x}^k) \\ \implies \mathbf{r}(\mathbf{x}^k)^T \mathbf{r}(\mathbf{x}^k) &\leq 2\mu^k (\mathbf{c}^T \bar{\mathbf{x}} - \mathbf{c}^T \mathbf{x}^k - \boldsymbol{\lambda}^{kT} \mathbf{r}(\mathbf{x}^k)). \end{aligned} \quad (8)$$

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

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

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 the value of $\boldsymbol{\lambda}$ was changed. Then for some p satisfying $k-W \leq p < k$,

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

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

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

Taking the limit in inequality (8),

$$\lim_{k \in \mathcal{K}} \mathbf{r}(\mathbf{x}^k)^T \mathbf{r}(\mathbf{x}^k) \leq \lim_{k \in \mathcal{K}} 2\mu^k (\mathbf{c}^T \bar{\mathbf{x}} - \mathbf{c}^T \mathbf{x}^k - \boldsymbol{\lambda}^{kT} \mathbf{r}(\mathbf{x}^k)). \quad (9)$$

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

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

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

$$\begin{aligned}
& \mathbf{c}^T \mathbf{x}^k + \boldsymbol{\lambda}^{kT} \mathbf{r}(\mathbf{x}^k) + \frac{1}{2\mu^k} \mathbf{r}(\mathbf{x}^k)^T \mathbf{r}(\mathbf{x}^k) \leq \mathbf{c}^T \bar{\mathbf{x}} \\
\implies & \lim_{k \in \mathcal{K}} (\mathbf{c}^T \mathbf{x}^k + \boldsymbol{\lambda}^{kT} \mathbf{r}(\mathbf{x}^k) + \frac{1}{2\mu^k} \mathbf{r}(\mathbf{x}^k)^T \mathbf{r}(\mathbf{x}^k)) \leq \lim_{k \in \mathcal{K}} \mathbf{c}^T \bar{\mathbf{x}} \\
\implies & \mathbf{c}^T \mathbf{x}^* + \lim_{k \in \mathcal{K}} \boldsymbol{\lambda}^{kT} \mathbf{r}(\mathbf{x}^k) + \lim_{k \in \mathcal{K}} \frac{1}{2\mu^k} \mathbf{r}(\mathbf{x}^k)^T \mathbf{r}(\mathbf{x}^k) \leq \mathbf{c}^T \bar{\mathbf{x}} \tag{10}
\end{aligned}$$

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

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

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

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

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

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

Consequently, \mathbf{x}^* is feasible for (1) and has an objective value less than or equal to the optimal value $\mathbf{c}^T \bar{\mathbf{x}}$ so \mathbf{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 the Idiot crash, the results in Table 3 show that this is far from being representative of its practical performance. For some problems it yields a near-optimal point; for others it terminates at a point which is far from being feasible. What 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 the Idiot crash. 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, which clearly shows that the problems solved accurately

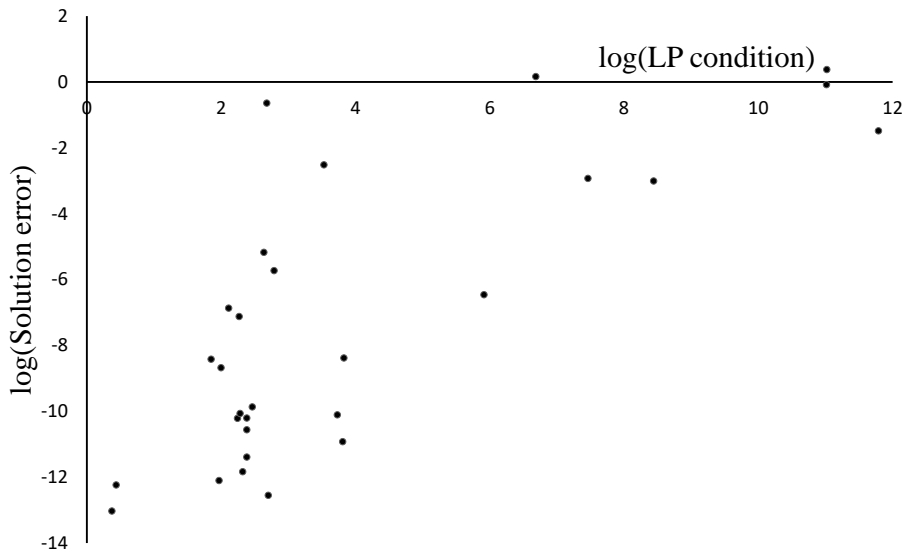


Figure 2: Solution error and LP condition

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 [12, 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 the Idiot crash.

4.2 The Idiot crash on QAPs

Since the Idiot crash 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

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

Table 4: Performance of the Idiot crash on QAP linearizations

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 solve the quadratic problem an equivalent linearization is solved. A comprehensive survey of QAP problems and their solution is given by Loiola *et al.* [10].

The test problems NUG15, QAP12 and QAP15 referred to above are examples of the Adams and Johnson linearization [1]. Although there are many specialised 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 [15]. Table 4 gives various performance measures for the Idiot crash when applied to the Nugent [13] problems, using the default iteration limit of `Clp`. The first of these is the value of the residual $\|A\mathbf{x} - \mathbf{b}\|_2$ at the point obtained by the Idiot crash, which clearly feasible to within the `Clp` simplex tolerance. The objective function value and relative error are also given, and the latter is within 1%. Finally, the time for Idiot is given. Whilst this is growing, Idiot 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 the Idiot experiments due to excessive time or memory requirements.

There is currently no practical measure of the point obtained by the Idiot crash which gives any guarantee that it can be taken as a near-optimal solution of the problem. The result of Theorem 1 cannot be used since 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, the Idiot crash was applied to the dual of the linearization. Although it obtained points which 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 the Idiot crash for LP problems was to determine a point which, 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 the Idiot crash 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 the Idiot crash is not advantageous, this is identified without meaningful detriment to the performance of Clp. For the best case in which the Idiot sub-problems are solved exactly, Theorem 1 shows that every limit point of the sequence of Idiot 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 the Idiot crash is to being an optimal solution of the LP problem. For linearizations of quadratic assignment problems, it has been demonstrated that the Idiot crash consistently yields near-optimal solutions, achieving this in minutes for instances which are intractable on the same machine using commercial LP solvers. Thus, in addition to achieving Forrest's initial aim, the Idiot crash is seen as being useful in its own right as a fast solver for amenable LP problems.

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] 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.
- [4] J. Forrest. [cbc] coinstructuredmodel crash. <https://list.coin-or.org/pipermail/cbc/2014-March/001297.html>, 2014. Accessed: 16/02/2018.
- [5] J. Forrest. Idiot.hpp. <https://projects.coin-or.org/Clp/browser/trunk/Clp/src/Idiot.hpp?rev=2144>, 2014. Accessed: 16/02/2018.
- [6] J. Forrest. GAMS-CBC. https://www.gams.com/latest/docs/S_CBC.html, 2018. Accessed: 16/02/2018.
- [7] O. Güler. Augmented lagrangian algorithms for linear programming. *Journal of optimization theory and applications*, 75(3):445–470, 1992.
- [8] M. R. Hestenes. Survey paper multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969.

- [9] Q. Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018.
- [10] 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.
- [11] H. D. Mittelmann. Benchmarks for optimization software. <http://plato.la.asu.edu/bench.html>, 2018. Accessed: 16/02/2018.
- [12] J. Nocedal and S. Wright. *Numerical optimization*. Springer, 2 edition, 2006.
- [13] 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.
- [14] M. J. D. Powell. *A method for nonlinear constraints in minimization problems*, pages 283–298. Academic Press, London, 1969.
- [15] 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.
- [16] R. T. Rockafellar. Augmented lagrange multiplier functions and duality in nonconvex programming. *SIAM Journal on Control*, 12(2):268–285, 1974.