Working Papers

03/2012

Two greedy algorithms for a binary quadratically constrained linear program in survey data editing

David Salgado, Ignacio Arbués and María Elisa Esteban

First draft: June 2012

This draft: June 2012

Two greedy algorithms for a binary quadratically constrained linear program in survey data editing

Abstract

We propose a binary quadratically constrained linear program as an approach to selective editing. In a practice-oriented framework and allowing for some overediting whilst strictly fulfilling accuracy restrictions, we propose two greedy algorithms to find feasible suboptimal solutions. Their running times are quartic and cubic, respectively, in the number of sampling units and linear in the number of restrictions. We present computational evidence from several hundreds of instances randomly generated.

Keywords

Combinatorial optimization, quadratic constraint, linear program, greedy algorithm, selective editing.

Authors and Affiliations

David Salgado, Ignacio Arbués and María Elisa Esteban

D. G. of Methodology, Quality and Information and Communications Technology

Spain National Statistics Institute

Two greedy algorithms for a binary quadratically constrained linear program in survey data editing

D. Salgado[1], I. Arbués[1], M.E Esteban[1]

[1]D.G. Metodología, Calidad y Tecnologías de la Información y de las Comunicaciones, Instituto Nacional de Estadística, Paseo de la Castellana, 183, 28071 Madrid (Spain)

June 22, 2012

Abstract

We propose a binary quadratically constrained linear program as an approach to selective editing in the survey production process. In a practice-oriented framework and allowing for some overediting whilst strictly fulfilling estimates accuracy restrictions, we propose two greedy algorithms to find feasible suboptimal solutions. Their running times are quartic and cubic, respectively, in the number of sampling units to edit and linear in the number of restrictions in both cases. We present computational evidence from several hundreds of instances randomly generated.

Keywords: Combinatorial optimization; quadratic constraint; linear program; greedy algorithm; selective editing.

MSC Classification: 90C27, 90C20, 90C59, 90C90.

1 Introduction

Data editing is a critical stage in the production process of sample survey estimates (de Waal, 2009; de Waal et al., 2011). Data editing can be defined as the "*procedure(s) designed and used for detecting erroneous and/or questionable survey data [...] with the goal of correcting [...] as much of the erroneous data (not necessarily all of the questioned data) as possible, usually prior to data imputation and summary procedures*" (FCSM, 1990). Under this umbrella term, data editing gathers a bunch of different techniques to detect errors in survey data (see e.g. de Waal et al. (2011); UNECE (1994, 1997)). Regarding mathematical programming, the most relevant technique is automated data editing, which focuses upon the so-called *error localization problem*. This problem arises as follows. Survey estimates are constructed from sample data collected in questionnaires containing the reported values of the surveyed variables of interest. Subject matter experts formulate consistency checks, known as

edit rules (or simply as *edits*), to guarantee the quality of the collected data. Usually these fail one or several edits and the expert must decide which variable values contain an error and thus must be treated (generally by producing a synthetic value with so-called imputation techniques or else by recollecting data). Since the seminal work by Fellegi and Holt Fellegi and Holt (1976), this problem has been formulated as a mixed-integer linear minimization program in different ways (Bruni, 2004, 2005; Bruni et al., 2001; Chen, 1998; Garfinkel et al., 1986, 1988; McKeown, 1984; Ragsdale and McKeown, 1996; de Waal, 2003; de Waal and Coutinho, 2005; de Waal and Quere, 2003; Winkler, 1997, 1998; Winkler and Chen, 2002), where the goal is to change in each questionnaire as fewest values as possible and the feasibility region is determined by the set of formulated edits.

Automatic data editing brings desirable quality elements into the survey production process, such as statistical defensibility, rationality of the process, increased capacities, etc. (see e.g. Pierzchala (1990)). These features should be extended to other editing techniques. In particular, we have focused upon selective editing, i.e. that editing modality aimed at detecting influential errors in survey data (see de Waal et al. (2011) and multiple references therein). Under our conviction that the preceding desirable properties stem out from the use of mathematical programming, efforts have been dedicated to formulate selective editing as an optimization problem. A first direction (Arbués et al., 2009, 2010) points at stochastic optimization. A second approach (Salgado, 2011) resorts to combinatorial optimization. A detailed discussion of both approaches, their derivation from a generic common optimization problem and their implications for the survey production process will be undertaken elsewhere. In the present paper, we undergo the resolution of the combinatorial version by proposing two greedy algorithms, analyzing their running times, their precision and presenting computational evidence.

The paper is organized as follows. In section 2 we show very briefly how the problem arises when demanding an optimal reduction of the survey editing tasks whilst controlling the accuracy of the estimation and express the motivation for finding a fast, although possibly inexact, algorithm to solve the problem. In section 3 we present a first greedy algorithm which takes advantage of both the structure of the problem and the implications for the survey production process. In section 4, using a similar approach, we propose a faster, although less precise, algorithm. In section 5 we make precision considerations. In section 6 we present computational evidence of the preceding analysis using randomly generated instances. We close with concluding remarks in section 7.

2 The problem: origin and motivation

The general mathematical framework where editing takes place is the problem of estimation in finite populations, commonly known as survey sampling (Cochran, 1977; Hansen et al., 1966; Särndal et al., 1992). Given a set $U = \{1, \ldots, N\}$ of N identifiable

so-called population units with numeric variables $y_k^{(q)}$ ($q = 1, \ldots, Q$) associated to each element k ($k = 1, \ldots, N$), population quantities $F_U^{(m)} = f_m(\mathbf{y}_1, \ldots, \mathbf{y}_N)$ are to be estimated by selecting a probabilistic sample $s \subset U$ according to a sampling design $p(\cdot)$, where $p(\cdot)$ denotes a probability measure over the set $\mathcal{S} = \mathcal{Q}(U)$ of all possible samples s. Without a substantial loss of practical generality, let us assume that the population quantities to be estimated are the population totals $Y_U^{(q)} = \sum_{k \in U} y_k^{(q)}$. By and large, these are estimated by using so-called *sampling weights* ω_{ks} attached to each population unit k for the selected sample s in the expression of a homogeneous linear estimator of the form $\hat{Y}_U^{(q)} = \sum_{k \in s} \omega_{ks} y_k^{(q)}$ (see e.g. Särndal et al. (1992) for details on how to construct the sampling weights). Since the sampling scheme is probabilistic, it is mandatory to express the accuracy of the estimators, usually through the mean squared error due to sampling variability: $\mathrm{MSE}_p\left(\hat{Y}_U^{(q)}\right) = \mathbb{E}_p\left[\left(\hat{Y}_U^{(q)} - Y_U^{(q)}\right)^2\right]$. The smaller $\mathrm{MSE}_p\left(\hat{Y}_U^{(q)}\right)$, the more precise the estimation.

In the survey production process variable values $y_k^{(q)}$ for units k in the sample s are measured through different data collection modes, regretfully all subjected to errors (see e.g. Groves (1989)). Selective editing focuses upon so-called influential measurement errors. That is, measurement errors $y_k^{(obs,q)} \neq y_k^{(0,q)}$, where $y_k^{(obs,q)}$ denotes the collected or observed value of variable q for unit k and $y_k^{(0,q)}$ stands for the true value of variable p for unit k, which have a great impact on estimates $\left|\hat{Y}_U^{(obs,q)} - \hat{Y}_U^{(0,q)}\right| \gg 1$, where $\hat{Y}_U^{(obs,q)} = \sum_{k \in s} \omega_{ks} y_k^{(obs,q)}$ and $\hat{Y}_U^{(0,q)} = \sum_{k \in s} \omega_{ks} y_k^{(0,q)}$.

To deal with measurement errors it is customary to use additive model errors, so that $y_k^{(obs,q)} = y_k^{(0,q)} + \epsilon_k^{(q)}$, where $\epsilon_k^{(q)}$ denotes the measurement error for variable p and unit k. As a working hypothesis, this error is usually conceived as a random variable, hence also $y^{(obs,q)}$ and conveniently $y_k^{(0,q)}$, with a joint probability distribution denoted by m. The combinatorial optimization problem setting starts from introducing binary variables $r_k \in B$, which flag those units which must be edited ($r_k = 0$) and those which do not need editing at this stage ($r_k = 1$). This assumption rests upon the idea that editing tasks drive us from the observed values $y_k^{(obs,q)}$ for all variables of a given unit k to their true values $y_k^{(0,q)}$, so that we can naturally define the edited value of each variable p and unit k as $y_k^{(ed,q)}(\boldsymbol{r}) = y_k^{(0,q)} + r_k \epsilon_k^{(q)}$. Thus, population quantities estimates are calculated with the estimators $\hat{Y}_U^{(ed,q)}(\boldsymbol{r}) = \sum_{k \in s} \omega_{ks} y_k^{(ed,q)}(\boldsymbol{r})$.

As guiding principles in selective editing we pursue (i) to reduce the amount of editing resources to be used and (ii) to keep estimation accuracy within acceptable levels. Principle (i) drives us to the objective pseudo-Boolean function $z(\boldsymbol{r}) = \sum_{k \in s} r_k = \boldsymbol{e}^t \boldsymbol{r}$, where $\boldsymbol{e}$ denotes the vector of ones. Principle (ii) allows us to construct the feasibility region of our problem. This is done by imposing $\mathrm{MSE}_{pm}\left(\hat{Y}_U^{(ed,q)}(\boldsymbol{r})\right) \leq m_q^2$, where m_q^2 are conveniently chosen nonnegative upper bounds (to be discussed also

elsewhere). But this mean squared error admits the following decomposition (see Salgado (2011) for details):

$$\begin{aligned}\mathbf{MSE}_{pm}\left(\hat{Y}_U^{(ed,q)}(\boldsymbol{r})\right) &= \mathbf{MSE}_p\left(\hat{Y}_U^{(0,q)}\right) + \mathbb{E}_{pm}\left[\left(\hat{Y}_U^{(ed,q)}(\boldsymbol{r}) - \hat{Y}_U^{(0,q)}\right)^2\right] \\ &+ 2\cdot\mathbb{E}_{pm}\left[\left(\hat{Y}_U^{(0,q)} - Y_U^{(q)}\right)\cdot\left(\hat{Y}_U^{(ed,q)}(\boldsymbol{r}) - \hat{Y}_U^{(0,q)}\right)\right].\end{aligned}$$

Now, the first term of the rhs is the mean squared error due only to sampling variability, so it is not affected by editing tasks, and, on the other hand, under general working hypotheses the estimators $\hat{Y}_U^{(0,q)} - Y_U^{(q)}$ and $\hat{Y}_U^{(ed,q)}(\boldsymbol{r}) - \hat{Y}_U^{(0,q)}$ can be considered uncorrelated, so that the cross term can be discarded. Thus we will focus upon $\mathbb{E}_{pm}\left[\left(\hat{Y}_U^{(ed,q)}(\boldsymbol{r}) - \hat{Y}_U^{(0,q)}\right)^2\right]$ by writing

$$\mathbb{E}_{pm}\left[\left(\hat{Y}_U^{(ed,q)}(\boldsymbol{r}) - \hat{Y}_U^{(0,q)}\right)^2\right] = \mathbb{E}_p\left[\mathbb{E}_m\left[\left(\hat{Y}_U^{(ed,q)}(\boldsymbol{r}) - \hat{Y}_U^{(0,q)}\right)^2\Big|s\right]\right].$$

We will set the accuracy by bounding

$$\mathbb{E}_m\left[\left(\hat{Y}_U^{(ed,q)}(\boldsymbol{r}) - \hat{Y}_U^{(0,q)}\right)^2\Big|s\right] \leq m_q^2, \tag{1}$$

where the upper bounds m_q^2 are chosen accordingly. It is immediate to rewrite inequality (1) in an algebraic fashion by defining matrices $M^{(q)}$, with entries $M_{kl}^{(q)} = \omega_{ks}\omega_{ls}\mathbb{E}_m\left[\epsilon_k^{(q)}\epsilon_l^{(q)}\Big|s\right]$. Thus restrictions are expressed as $\boldsymbol{r}^t M^{(q)}\boldsymbol{r} \leq m_q^2$.

If we denote by n the sample size, i.e. $n = |s|$, the binary quadratically constrained linear program (BQCLP hereafter) can be finally expressed as:

$$\begin{aligned}\max\quad & \boldsymbol{e}^t\boldsymbol{r} \\ \text{s.t.}\quad & \boldsymbol{r}^t M^{(q)}\boldsymbol{r} \leq m_q^2,\ q = 1,\ldots,Q, \\ & \boldsymbol{r} \in B^n.\end{aligned} \tag{2}$$

Matrices $M^{(q)}$ are constructed after statistically modelling the measurement error, which is beyond the scope of this work. They will be positive semidefinite matrices. A further remark about the survey production process will drive us to a slightly generalized BQCLP. Editing tasks in a statistical operation are subjected to the conditions of its field work organization. In this sense, a prioritization of units for their editing is required, which we propose to find by concatenating succesive solutions to problem (2) with different bounds m_q^2 after imposing further restrictions of some of the components of $\boldsymbol{r}$, i.e. fixing $r_k = 1$ for $k \in I_1^*$ and $r_k = 0$ for $k \in I_0^*$ for each succesive resolution of problem (2). Details will be discussed elsewhere; nonetheless, problem (2) is straightforwardly generalized to

$$
\begin{aligned}
\max \quad & \boldsymbol{e}^t \boldsymbol{r} \\
\text{s.t.} \quad & \boldsymbol{r}^t M^{(q)} \boldsymbol{r} \leq m_q^2,\ q = 1, \ldots, Q, \\
& r_k = 1, k \in I_1^*, \\
& r_k = 0, k \in I_0^*, \\
& \boldsymbol{r} \in B^n.
\end{aligned}
\tag{3a}
$$

By later convenience we reformulate problem (3a) in terms of an index set variable I_1 defined by $I_1 = \{k \in I : r_k = 1\}$, where $I = \{1, \ldots, n\}$. Thus we will reformulate problem (3a) as

$$
\begin{aligned}
\max \quad & |I_1| \\
\text{s.t.} \quad & \sum_{k \in I_1} \sum_{l \in I_1} M_{kl}^{(q)} \leq m_q^2, \quad q = 1, \ldots, Q, \\
& I_1 \supset I_1^*, \\
& I - I_1 \supset I_0^*, \\
& I_1 \in \mathcal{P}(I).
\end{aligned}
\tag{3b}
$$

Finally, let us point out as practical computational requirements that problem (3b) is to be applied to maximally disaggregated publication cells or the smallest population domains U_d such that $U = \bigcup_{d=1}^{D} U_d$, whose respective sample sizes $n_d = |s_d|$ typically range up to very few thousands[1]. Units prioritization requires concatenating around n_d BQCLPs for each domain U_d. This means that computation speed is a concern. On the other hand, a precision loss in the case of suboptimal feasible solutions implies that a number of units greater than the optimal one is actually flagged for editing, whereas accuracy constraints are all fulfilled. Provided that the loss of efficiency by flagging too many units is not too great, this establishes an acceptable practical framework to search for a solution to the BQCLP.

3 A first algorithm

Within the general practice-oriented framework depicted in the preceding section we will propose a first algorithm to find a suboptimal feasible solution to the general BQCLP (3b). We define the following infeasibility function.

Definition. *Let $M^{(q)}$, m_q^2, I_1^* and I_0^* define an instance of a BQCLP. We define the associated infeasibility function h by*

$$
\begin{aligned}
h: \quad & \mathcal{P}(I) \to \mathbb{R}^+ \\
& I_1 \to h(I_1) = \sum_{q=1}^{Q} \left(\sum_{k \in I_1} \sum_{l \in I_1} M_{kl}^{(q)} - m_q^2 \right)^+,
\end{aligned}
$$

[1]This remark refers to official statistics publication requirements at Spanish National Statistical Institute as of the present paper's date of publication.

where we denote $(x)^+ = \max\{x, 0\}$.

Notice that $h(I_1) = 0$ if, and only if, I_1 is feasible. Also, note that computation of $h(I_1)$ takes an asymptotic time $O\left(n_1^2 \cdot Q\right)$, where $n_1 = |I_1|$. We propose the following greedy algorithm:

Algorithm 1. *Let* $M^{(q)}$, m_q^2, I_1^* *and* I_0^* *define an instance of a BQCLP. Define an algorithm with the following steps:*

1. $I_1 := I - I_0^*$.
2. IF $h(I_1) = 0$ OR $I_1 = I_1^*$, STOP.
3. $\mathcal{I}(I_1) := \{I \in \mathcal{P}(I) : I = I_1 - \{i\}, i \in I_1 - I_1^*\}$.
4. $I_1 := \operatorname{argmin}(h(I) : I \in \mathcal{I}(I_1))$. *In case of multiple* I_1 *choose one at random.* RETURN TO STEP 2.

Starting from the initial solution with the a priori greatest cardinality, this algorithm drops out iteratively one index searching for feasibility by minimizing locally the infeasibility function. Notice that if $I_1 = I_1^*$ is not feasible ($h(I_1^*) > 0$), the algorithm may stop resulting in a nonfeasible solution. On the one hand, regretfully this does not mean that the feasibility region is empty, since the algorithm is heuristic: a possible solution $I_1 = I_1^* \cup \{i^*\}$, with $i^* \in I - (I_0^* \cup I_1^*)$ is not ruled out. An elementary example is given by matrix $M = \left(\begin{smallmatrix} 5 & -2 \\ -2 & 1 \end{smallmatrix}\right)$, bound $m^2 = 4$ and fixed index set $I_1^* = \{1\}$. It is clear that $h(I_1^*) > 0$ while $h(\{1, 2\}) = 0$. On the other hand, fortunately enough, fixing I_1^* such that $h(I_1^*) > 0$ will not be a practical case, since it implies not to edit a set of units incurring in a loss of precision above the established bounds.

Running time in a worst-case analysis is straightforwardly computed. Since steps 2 to 4 are run $n - n_1^* - n_0^*$ times in the worst case, where $n_0^* = |I_0^*|$ and $n_1^* = |I_1^*|$, and the longer step in terms of elementary operations, step 4, takes an asymptotic time $O\left((n_1 - n_1^*)n_1^2 P\right)$, the running time $T(n, Q)$ is given by $T(n, Q) = O\left((n - n_1^* - n_0^*)^4 \cdot Q\right)$.

4 A second algorithm

With the purpose of speeding up the algorithm, let us carry out the following modification. Let us consider two succesive solutions I_1^* and $I_1 = I_1^* \cup \{i^*\}$ (in algorithm 1, I_1 appears before I_1^*). Instead of choosing I_1^* as that index set minimizing the infeasibility function h, let us focus on maximizing the difference $h(I_1) - h(I_1^*)$. Define $\mathcal{Q}^+ = \left\{q \in \{1, \ldots, Q\} : \sum_{k \in I_1} \sum_{l \in I_1} M_{kl}^{(q)} - m_q^2 > 0\right\}$ and $\mathcal{Q}^0 = \{1, \ldots, Q\} - \mathcal{Q}^+$. Then write

$$h(I_1) - h(I_1^*) \;=\; \sum_{q=1}^{Q} \left(\sum_{k \in I_1^*} \sum_{l \in I_1^*} M_{kl}^{(q)} - m_q^2 + 2 \sum_{k \in I_1} M_{ki^*}^{(q)} - M_{i^*i^*}^{(q)} \right)^+$$

$$- \sum_{q=1}^{Q} \left(\sum_{k \in I_1^*} \sum_{l \in I_1^*} M_{kl}^{(q)} - m_q^2 \right)^+$$

In a heuristic fashion we will choose i^* as

$$i^* = \text{argmax}_{i \in I_1 - I_1^*} \left(\sum_{q \in \mathcal{Q}^+} \left(2 \sum_{k \in I_1} M_{ki}^{(q)} - M_{ii}^{(q)} \right) \right).$$

This enables us to propose a modified algorithm:

Algorithm 2. *Let $M^{(q)}$, m_q^2, I_1^* and I_0^* define an instance of a BQCLP. Define an algorithm with the following steps:*

1. *Set $I_1 := I - I_0^*$ and $\mathcal{Q}^+ := \left\{ q \in \{1, \ldots, Q\} : \sum_{k \in I_1} \sum_{l \in I_1} M_{kl}^{(q)} - m_q^2 > 0 \right\}$.*
2. *If $\mathcal{Q}^+ = \emptyset$,* STOP.
3. *Set $i^* := \text{argmax}_{i \in I_1 - I_1^*} \left\{ \sum_{q \in \mathcal{Q}^+} \left(2 \sum_{k \in I_1} M_{ki}^{(q)} - M_{ii}^{(q)} \right) \right\}$. In case of multiple i^*, choose one at random.*
4. *Set $I_1 := I_1 - \{i^*\}$ and $\mathcal{Q}^+ := \left\{ q \in \{1, \ldots, Q\} : \sum_{k \in I_1} \sum_{l \in I_1} M_{kl}^{(q)} - m_q^2 > 0 \right\}$.* RETURN TO STEP 2.

The idea behind the algorithm is again to search for feasibility by minimizing the infeasibility function, now in a loose way. In this version precision is sacrificed for speed, as the following analysis shows. In the worst case, $n - n_1^* - n_0^*$ iterations are again needed, but step 4 takes now an asymptotic running time $O\left((n_1 - n_1^*)^2 P\right)$, hence the global running time $T(n, Q)$ of algorithm 2 is given by $T(n, Q) = O\left((n - n_1^* - n_0^*)^3 \cdot Q\right)$.

5 Precision considerations

It is immediate to realize that the solutions $I_1^{(1)}$ and $I_1^{(2)}$ found with the preceding algorithms yield lower bounds $z_2^* \leq z_1^* \leq z^*$ for the optimal unknown value z^* of the objective function. To make precision considerations we will compute upper bounds $z_U \geq z^*$ from the semidefinite relaxation of the BQCLP (see e.g. Luo et al. (2010); Poljak et al. (1995)).

To arrive at this SDP relaxation, firstly let us come back to the formulation (3a) and substitute restrictions $r_k = 1$ if $k \in I_1^*$ and $r_k = 0$ if $k \in I_0^*$. The problem is reformulated as

$$\begin{aligned}
\text{max} \quad & n_1^* + \sum_{k \notin I_1^* \cup I_0^*} r_k \\
\text{s.t.} \quad & \sum_{k \notin I_1^* \cup I_0^*} \sum_{l \notin I_1^* \cup I_0^*} M_{kl}^{(q)} r_k r_l + 2 \sum_{k \notin I_1^* \cup I_0^*} \sum_{l \in I_1^*} M_{kl}^{(q)} r_k + \sum_{k \in I_1^*} \sum_{l \in I_1^*} M_{kl}^{(q)} \leq m_q^2, \; q = 1, \ldots, Q, \\
& r_k \in B, \; k \in I.
\end{aligned}$$

This can be recast in almost the same form as problem (2) by defining a new index set $\bar{I} = I - (I_1^* \cup I_0^*)$, with cardinality $\bar{n} = n - n_1^* - n_0^*$, new matrices $\bar{M}^{(q)}$ of size $\bar{n} \times \bar{n}$ with entries

$$\begin{aligned} \bar{M}_{kk}^{(q)} &= M_{kk}^{(q)} + 2 \sum_{l \in I_1^*} M_{kl}^{(q)}, \\ \bar{M}_{kl}^{(q)} &= M_{kl}^{(q)}, \; k \neq l, \end{aligned}$$

and new bounds $\bar{m}_p^2 = m_q^2 - \sum_{k \in I_1^*} \sum_{l \in I_1^*} M_{kl}^{(q)}$. The problem is now expressed as

$$\begin{array}{ll} \max & n_1^* + \boldsymbol{e}^t \boldsymbol{r} \\ \text{s.t.} & \boldsymbol{r}^t \bar{M}^{(q)} \boldsymbol{r} \leq \bar{m}_p^2, \; q = 1, \ldots, Q, \\ & \boldsymbol{r} \in B^{\bar{n}}. \end{array} \tag{4}$$

Now change $(0,1)$ variables r_k into $(-1,1)$ variables $\bar{r}_k$ and lift up the problem to the symmetric matrix space $\mathcal{S}_{\bar{n}+1}$ by defining $\bar{R} = \begin{pmatrix} 1 & \bar{\boldsymbol{r}}^t \\ \bar{\boldsymbol{r}} & \bar{\boldsymbol{r}}\bar{\boldsymbol{r}}^t \end{pmatrix}$. The SDP reformulation of problem (4) is given by

$$\begin{array}{ll} \max & \text{tr}\left(\bar{R} \cdot Q_0\right) \\ \text{s.t.} & \text{tr}\left(\bar{R} \cdot Q_p\right) \leq \bar{m}_p^2, \; q = 1, \ldots, Q, \\ & \text{diag}\left(\bar{R}\right) = \boldsymbol{e}, \\ & \text{rank}\left(\bar{R}\right) = 1, \\ & \bar{R} \succeq 0, \end{array}$$

where $Q_0 = \frac{1}{4} \begin{pmatrix} 2\bar{n} + 4n_1^* & \boldsymbol{e}^t \\ \boldsymbol{e} & \mathbb{O} \end{pmatrix}$, $Q_p = \frac{1}{4} \begin{pmatrix} \boldsymbol{e}^t \bar{M}^{(q)} \boldsymbol{e} & \boldsymbol{e}^t \bar{M}^{(q)} \\ \bar{M}^{(q)} \boldsymbol{e} & \bar{M}^{(q)} \end{pmatrix}$ and $\succeq$ denotes the Löwner partial order. As usual, the SDP relaxation comes from dropping out the nonconvex rank restriction:

$$\begin{array}{ll} \max & \text{tr}\left(\bar{R} \cdot Q_0\right) \\ \text{s.t.} & \text{tr}\left(\bar{R} \cdot Q_p\right) \leq \bar{m}_p^2, \; q = 1, \ldots, Q, \\ & \text{diag}\left(\bar{R}\right) = \boldsymbol{e}, \\ & \bar{R} \succeq 0. \end{array} \tag{5}$$

The optimal solution to problem (5) will provide the upper bound z_U for problem (3a). To solve it we use the primal-dual interior point algorithm in Helmberg et al. (1996), implemented in the C library CSDP for semidefinite programming (Borchers, 1999). In general, the resolution of (5) yields a real optimal value z_{SDP}^*, from which we fix an integer upper bound z_U by $z_U = \lfloor z_{SDP}^* \rfloor$. Needless to say, should $z_1 = z_U$ or $z_2 = z_U$, we would obtain, respectively, an optimal solution to the BQCLP.

6 Running times and precision loss: computational evidence

To test the preceding algorithms we have applied them to several hundreds of randomly generated instances. Firstly, since the generalized BQCLP (3a) can be cast in the original form (2), we have set $I_0^* = I_1^* = \emptyset$ for our simulations. Secondly, instances have been generated building matrices $M^{(q)}$ as close to real error modelling matrices as possible. Thus, for each instance size n each matrix $M^{(q)}$ has been generated as follows:

1. Construct a matrix A_q of size $n \times r$ with random entries independently distributed according to a normal probability distribution with random mean $\text{Unif}_c(0,1)$, where $\text{Unif}_c(0,1)$ stands for the absolutely continuous probability distribution in the interval $(0,1)$, and variance 100, i.e. $[A_q]_{ij} \simeq N(\text{Unif}_c(0,1), \sigma^2 = 100)$, and r is a discrete uniform random variable $r \simeq \text{Unif}_d[0,n]$.
2. Compute $B_p = A_q A_q^t$.
3. Redefine $B_p := B_p + |\lambda_{min}(B_p)|\mathbb{I}_n$.

Bounds have been generated independently as uniform random variables $m_q^2 \simeq \text{Unif}_c\left[0, \sum_{k\in I}\sum_{l\in I} M_{kl}^{(q)}\right]$.

We have implemented both algorithms in R language and have used the R interface in package Rcsdp (Corrada, 2010) to the CSDP library. Codes of the functions implementing algorithms 1 and 2 are included in appendix A.

We have generated $n/10$ instances for each size n from $n = 50$ to $n = 1500$ in intervals of 50 units with $Q = 1$. In figure 1 we represent the maximum, minimum and mean values of parameter r for the $n/10$ values of each instance size n. For algorithm 1 we have used instances up to size $n = 500$ and for algorithm 2, up to $n = 1500$.

Regarding algorithm 1, we have measured with the built-in function `system.time` the time needed to solve each instance using a 2.00GB RAM 3.00GHz CPU clock Pentium 4 PC. In figure 2 we represent the maximum, minimum, mean and median values of the $n/10$ measured CPU times for each instance size n with $Q = 1$. To check the asymptotic running time $O(n^\alpha)$ we have adjusted a regression line $\log(\max t_{CPU}(n))$ vs. $\log(n)$ obtaining an exponent value $\alpha = 4.0 \pm 0.1$, with a 95% confidence (figure 3).

We have proceeded along similar lines for algorithm 2, but with instance sizes ranging from $n = 50$ up to $n = 1500$ in intervals of 50 units. CPU times and the respective adjusted regression line are represented in figures 4 and 5. The exponent value is now given by $\alpha = 3.1 \pm 0.1$, with a 95% confidence.

To compare the speed of both algorithms we have generated again $n/10$ instances for each size ranging from $n = 50$ to $n = 500$ in 50-units intervals. We have solved

each instance with both algorithms and measured their CPU time difference $\Delta T(n) = T_1(n) - T_2(n)$. In figure 6 we represent maximum, minimum, median and mean values of CPU time differences $\Delta T(n)$ for each instance size n.

Regarding the precision, we have solved each instance with both algorithms and have computed the difference $\Delta z_i^* = z_U - z_i^* = z_U - \sum_{k \in I} r_k^{(i)}$, $i = 1, 2$, between the SDP-relaxed floor value $z_U = \lfloor z_{SDP}^* \rfloor$ and the corresponding suboptimal feasible solution z_i^*. We represent maximum, minimum, median and $90-, 95-, 99-$quantile values of Δz_i^* for each instance size n in figures 7 and 8. Notice that the value z_i^* is indeed optimal if $\Delta z_i^* = 0$. Table 1 shows the number of solutions with $\Delta z_i^* = 0$ found for each instance size. Note that this does not necessarily exhaust the number of optimal solutions.

Instance size	**50**	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**
Total Instances	5	10	15	20	25	30	35	40	45	50
Algorithm 1	2	8	12	12	12	19	18	20	23	22
Algorithm 2	1	6	7	8	6	12	12	11	13	17

Table 1: Number of solutions with $\Delta z_i^* = 0$ for algorithms 1 and 2, respectively.

On the other hand, to study the behaviour of both algorithms with respect to the number Q of restrictions, we have fixed the size $n = 200$ and have computed CPU times for $n/10 = 20$ instances with an increasing number of restrictions given by $Q = 5, 10, 15, \ldots, 75$ for algorithm 1 and $Q = 5, 10, 15, \ldots, 100$ for algorithm 2. Results are represented in figures 9 and 10. To check for the running time $O(Q^\alpha)$, we have respectively adjusted a regression line $\log(\max t_{CPU}(n))$ vs. $\log Q$. Computational results, depicted in figures 11 and 12, are consonant with the theoretical analysis.

7 Concluding remarks

We propose to formulate the selective editing technique in survey data editing as a mathematical programming problem aiming at optimizing resources whilst keeping estimation accuracy under control. Assuming some overediting (excess in the number of units to edit), but demanding the strict fulfillment of all accuracy restrictions, we give two greedy algorithms to find a feasible suboptimal solution. From practical considerations, running time is a concern. In this sense, the heuristics is justified: both algorithms run in times $O(n^4 \cdot Q)$ and $O(n^3 \cdot Q)$, respectively, where n denotes the number of sampling units and Q the number of population quantities to estimate. Regarding precision, the amount of overediting is not substantial and exact optimal solutions are found in a non-neglectable number of cases. These results allows us to conclude that, within this view of selective editing, the task is reduced to construct the matrices $M^{(q)}$ by statistically modelling the measurement error, since the above

algorithms can provide solutions to the combinatorial optimization problem which are both fast and precise enough for their use in the survey production process.

A Source codes

We include the source codes for the implementation in R functions of algorithms 1 and 2.

R code A.1: Source code for algorithm 1.

```
GreedySolution <-
function(matrices,bounds,fixed.1=integer(0),fixed.0=integer(0)){
### CASE any(is.na(matrices))==T ###
if (any(is.na(matrices))) stop("Missing values in array of matrices.")
### DATA READING ###
Q <- length(bounds)
n <- dim(matrices)[1]
if (dim(matrices)[3]!=Q) stop("Numbers of matrices and bounds differ.")
### CASE n=1 ###
if (n==1) {if (any(matrices>bounds)) return(numeric(n)) else return(rep(1,times=n))}
### FUNCTION h.restrictions  ###
h.restrictions <- function(I.1){
    mat <- array(matrices[I.1,I.1,],c(length(I.1),length(I.1),Q))
    h.restrictions.vec <- -bounds+apply(mat,3,sum)
    h.restrictions.vec[h.restrictions.vec<0] <- 0
    return(sum(h.restrictions.vec))}
### LOOP DOWN UNTIL FEASIBILITY ###
I.1 <- 1:n
if (length(fixed.0)>0) I.1 <- I.1[-fixed.0]
if (length(fixed.1)==0) {
    while (h.restrictions(I.1)>0){
            I.1.next <- matrix(numeric(length(I.1)*(length(I.1)-1)),nrow=length(I.1))
            for (i in 1:nrow(I.1.next)) I.1.next[i,] <- I.1[-i]
            h.restrictions.next <- apply(I.1.next,1,h.restrictions)
            i.min <- which.min(h.restrictions.next)
            if (length(i.min)>1) i.min <- i.min[sample(1:length(i.min),1)]
            I.1 <- I.1.next[i.min,]}}
if (length(fixed.1)>0) {
    while (h.restrictions(I.1)>0 & length(setdiff(I.1,fixed.1))>0){
            I.1.next <- matrix(numeric(length(setdiff(I.1,fixed.1))*(length(I.1)-1)),nrow=length(setdiff(I.1,fixed.1)))
            for (i in 1:nrow(I.1.next)) I.1.next[i,] <- union(fixed.1,setdiff(I.1,fixed.1)[-i])
            h.restrictions.next <- apply(I.1.next,1,h.restrictions)
            i.min <- which.min(h.restrictions.next)
            if (length(i.min)>1) i.min <- i.min[sample(1:length(i.min),1)]
            I.1 <- I.1.next[i.min,]}}
Solution <- numeric(n)
Solution[I.1] <- 1
Feasible <- (h.restrictions(I.1)==0)
Suboptimal.Val <- sum(Solution)
Output <- list(Solution=Solution,SuboptVal=Suboptimal.Val,Feasible=Feasible)
return(Output)
}
```

R code A.2: Source code for algorithm 2.

```
FastGreedySolution <-
function(matrices,bounds,fixed.1=integer(0),fixed.0=integer(0)){
### CASE any(is.na(matrices)) ###
if (any(is.na(matrices))) stop("Missing values in array of matrices.")
### DATA READING ###
Q <- length(bounds)
n <- dim(matrices)[1]
if (dim(matrices)[3]!=Q) stop("Numbers of matrices and bounds differ.")
### CASE n=1 ###
if (n==1) {if (any(matrices>bounds)) return(numeric(n)) else return(rep(1,times=n))}
### FUNCTION h.vec  ###
h.vec <- function(I.1){
    mat <- array(matrices[I.1,I.1,],c(length(I.1),length(I.1),Q))
    h.restrictions.vec <- -bounds+apply(mat,3,sum)
    return(h.restrictions.vec)}
### FUNCTION Delta  ###
Delta <- function(I.1){
```

```
        aux <- matrix(rep(0,times=n^2),ncol=n)
        for (p in Q.pos){aux <- aux+matrices[,,p]}
        dismin <- 2*colSums(aux,na.rm=T)-diag(aux)
        dismin[c(I.0,fixed.1)] <- NA
        return(dismin)
                    }
### LOOPS DOWN UNTIL FEASIBILITY  ###
Solutions <- matrix(numeric(n*Q),ncol=n)
I.1 <- 1:n
if (length(fixed.0)>0) I.1 <- I.1[-fixed.0]
Q.pos <- which(h.vec(I.1)>0)
I.0 <- setdiff(1:n,I.1)
while (length(Q.pos)!=0){
    i.max <- which(Delta(I.1)==max(Delta(I.1),na.rm=T))
    if (length(i.max)>0) i.max <- i.max[sample(1:length(i.max),1)]
    I.1 <- setdiff(I.1,i.max)
    I.0 <- setdiff(1:n,I.1)
    Q.pos <- which(h.vec(I.1)>0)}
Solution <- numeric(n)
Solution[I.1] <- 1
Suboptimal.Val <- sum(Solution)
h.fin <- h.vec(I.1)
h.fin[h.fin<0] <- 0
Feasible <- (sum(h.fin)==0)
Output <- list(Solution=Solution,SuboptVal=Suboptimal.Val,Feasible=Feasible)
return(Output)
}
```

References

I. Arbués, M. González, and P. Revilla, 2009. La depuración selectiva como un problema de optimización estocástica. *Boletín de Estadística e Investigación Operativa* **25**, 32–41.

I. Arbués, M. González, and P. Revilla, 2010. A class of stochastic optimization problems with application to selective data editing. *Optimization* **61**, 265–286 (2012).

B. Borchers, 1999. Csdp, a c library for semidefinite programming. *Optimization Methods and Software* **11**, 613–623. URL `https://projects.coin-or.org/Csdp/`.

R. Bruni, 2004. Discrete models for data imputation. *Discrete Applied Mathematics* **144**, 59–69.

R. Bruni, 2005. Error correction for massive data sets. *Optimization Methods and Software* **20**, 295–314.

R. Bruni, A. Reale, and R. Torelli, 2001. Optimization techniques for edit validation and data imputation. *Proceedings of Statistics Canada Symposium 2001 Achieving Data Quality in a Statistical Agency: a Methodological Perspective*.

B.-C. Chen, 1998. Set covering algorithms in edit generation. *U.S. Bureau of the Census Research Report Series (Statistics 1998-06)*.

W.G. Cochran, 1977. *Sampling Techniques*. Wiley, New York, 3rd edition.

Héctor Corrada Bravo, 2010. *Rcsdp: R interface to the CSDP semidefinite programming library*. URL `http://CRAN.R-project.org/package=Rcsdp`. R package version 0.1-41.

T. de Waal, 2003. Solving the error localization problem by means of vertex generation. *J. Official Stat.* **29**, 71–79.

T. de Waal, 2009. *Sample Surveys: Design, Methods and Applications*, chapter Statistical data editing, pages 187–214. North Holland.

T. de Waal and W. Coutinho, 2005. Automatic editing for business surveys: an assessment of selected algorithms. *International Statistical Review* **73**, 73–102.

T. de Waal and R. Quere, 2003. A fast and simple algorithm for automatic editing of mixed data. *Journal of Official Statistics* **19**, 383–402.

T. de Waal, J. Pannekoek, and S. Scholtus, 2011. *Handbook of statistical data editing and imputation*. Wiley.

I.P. Fellegi and D. Holt, 1976. A systematic approach to automatic edit and imputation. *J. Amer. Stat. Assoc.* **71**, 17–35.

R.S. Garfinkel, A.S. Kunnathur, and G.E. Liepins, 1986. Optimal imputation of erroneous data: categorical data, general edits. *Operations Research* **34**, 744–751.

R.S. Garfinkel, A.S. Kunnathur, and G.E. Liepins, 1988. Error localization for erroneous data: continuous data, linear constraints. *SIAM Journal of Scientific and Statistical Computations* **9**, 922–930.

R.M. Groves, 1989. *Survey errors and survey costs*. Wiley, New York.

M.H. Hansen, W.N. Hurwitz, and W.G. Madow, 1966. *Sample survey: methods and theory*. Wiley, New York, 7th edition.

C. Helmberg, F. Rendl, R.J. Vanderbei, and H. Wolkowicz, 1996. An interior point method for semidefinite programming. *SIAM Journal on Optimization* **6**, 342–361.

Z.-Q. Luo, W.-K. Ma, A.M.-C. So, Y. Ye, and S. Zhang, 2010. Semidefinite relaxation of quadratic optimization problems. *IEEE Signal Processing Magazine* **27**, 20–34.

P.G. McKeown, 1984. A mathematical programming approach to editing of continuous survey data. *SIAM Journal of Scientific and Statistical Computations* **5**, 784–797.

M. Pierzchala, 1990. A review of the state of the art in automated data editing and imputation. *J. Official Stat.* **6**, 355–377.

S. Poljak, F. Rendl, and H. Wolkowicz, 1995. A recipe for semidefinite relaxation for (0,1)-quadratic programming. *Journal of Global Optimization* **7**, 51–73.

C.T. Ragsdale and P.G. McKeown, 1986. On solving the continuous data editing problem. *Computers Ops Res* **23**, 263–273.

D. Salgado, 2011. Exploiting auxiliary information: selective editing as a combinatorial optimization problem. *INE Spain Working Paper 04/11*.

C.-E. Särndal, B. Swensson, and J.H. Wretman, 1992. *Model assisted survey sampling*. Springer, New York.

UNECE (editor), 1994. *Statistical data editing: methods and techniques*, volume 1, United Nations, New York.

UNECE (editor), 1997. *Statistical data editing: methods and techniques*, volume 2, United Nations, New York.

U.S. Federal Committee on Statistical Methodology, 1990. Data Editing in Federal Statistical Agencies. Technical report, Statistical Policy Office: U.S. Office of Management and Budget, Washington, D.C.

W.E. Winkler, 1997. Editing discrete data. *U.S. Bureau of the Census Research Report Series (Statistics 1997-04).*.

W.E. Winkler, 1998. Set-covering and editing discrete data. *U.S. Bureau of the Census Research Report Series (Statistics 1998-01)*.

W.E. Winkler and B.-C. Chen, 2002. Extending the fellegi-holt model of statistical data editing. *U.S. Bureau of the Census Research Report Series (Statistics 2002-02)*.

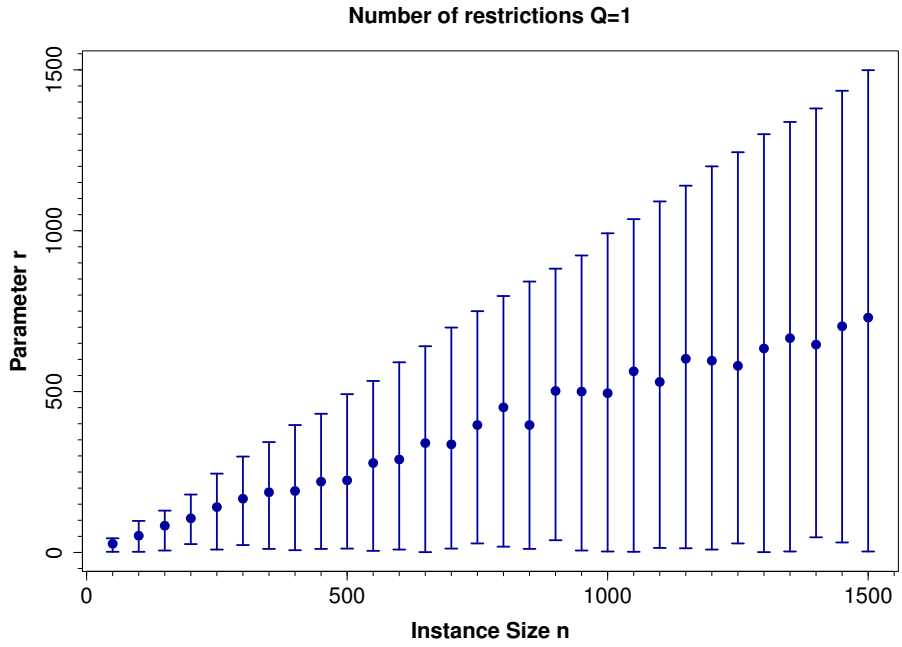

Figure 1: Maximum, minimum and mean values of parameter r for each instance size n.

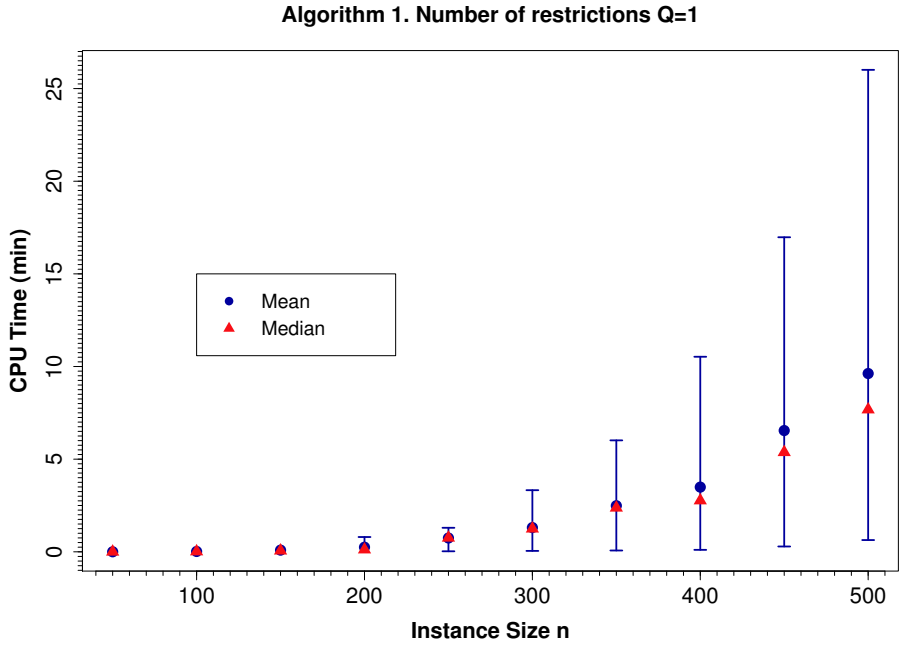

Figure 2: Maximum, minimum, mean and median values of CPU Times (in minutes) for algorithm 1 for each instance size n.

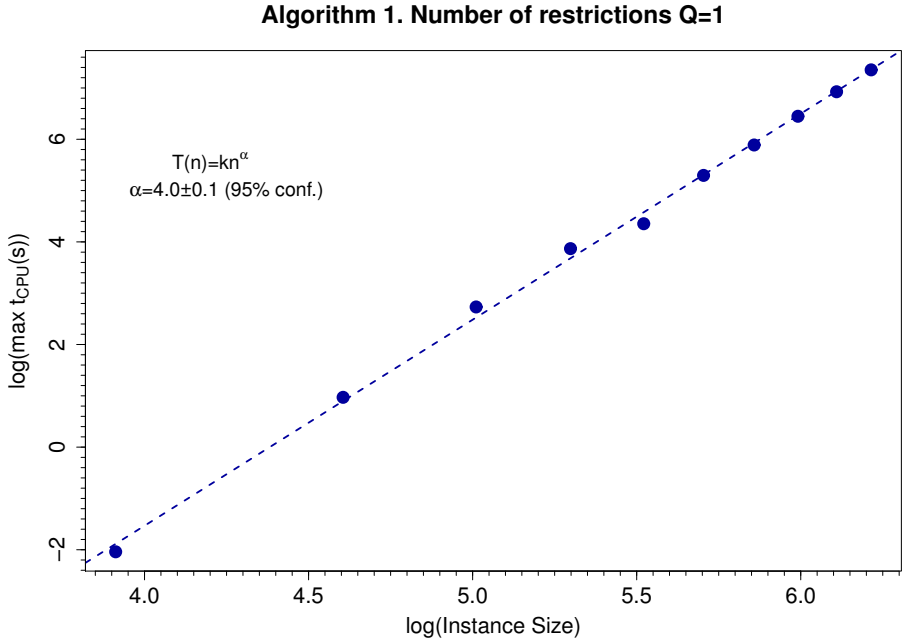

Figure 3: Regression line to estimate α in $T(n) = kn^{\alpha}$ for algorithm 1.

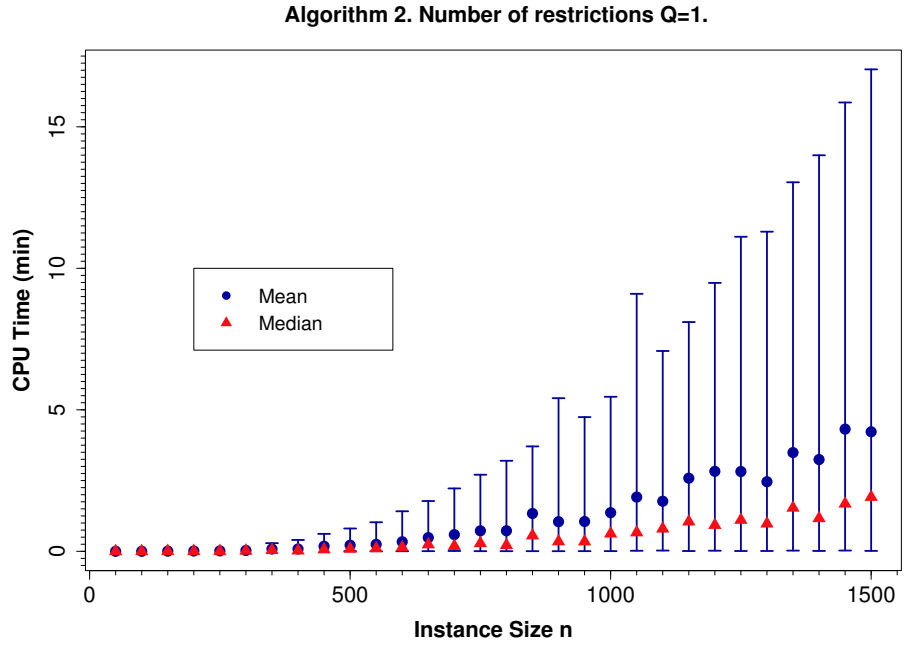

Figure 4: Maximum, minimum, mean and median values of CPU Times (in minutes) for algorithm 2 for each instance size n.

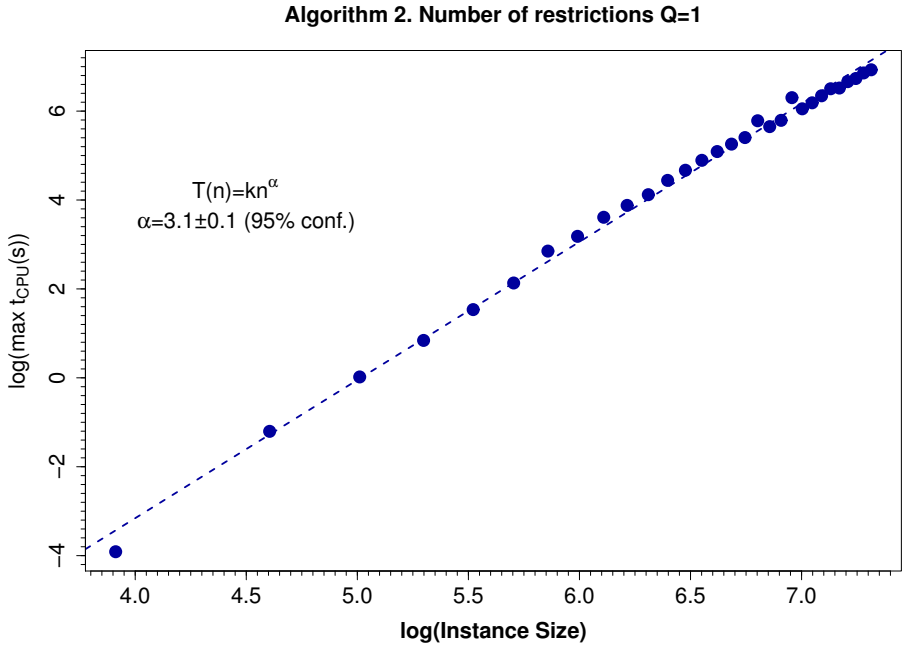

Figure 5: Regression line to estimate α in $T(n) = kn^{\alpha}$ for algorithm 2.

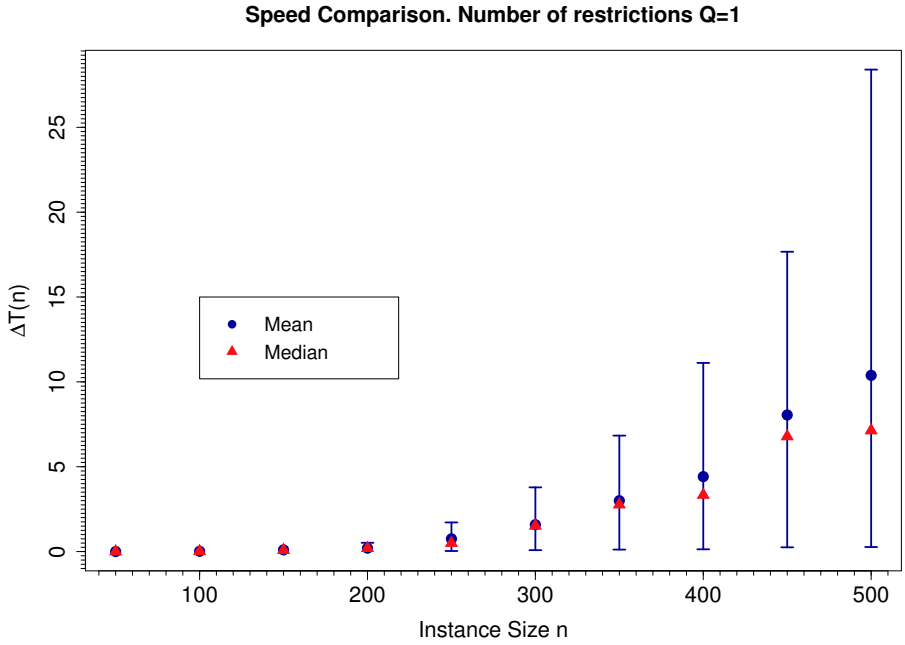

Figure 6: Comparison of CPU Time speeds for algorithms 1 and 2.

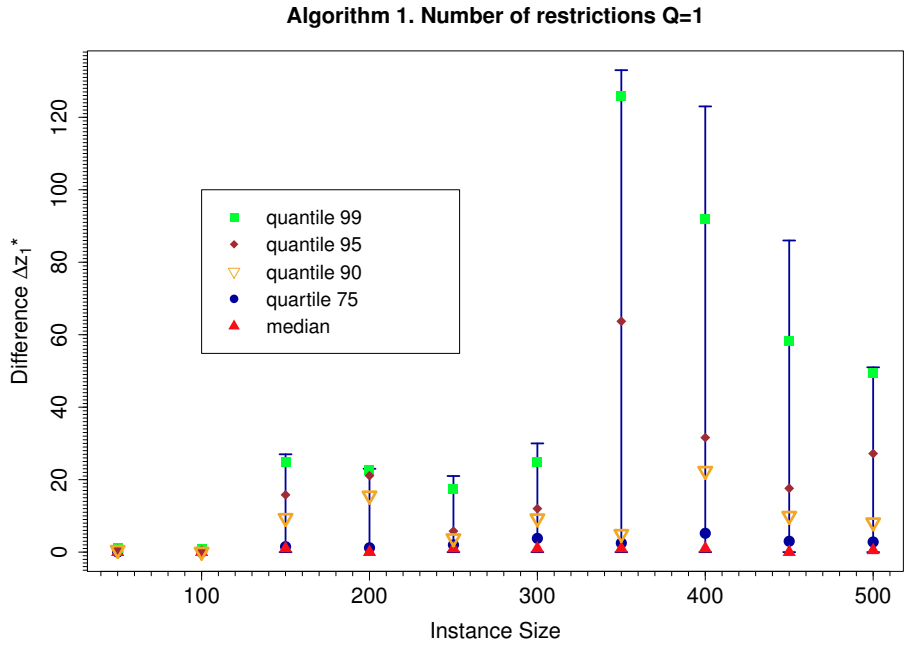

Figure 7: Maximum, minimum, median and $90, 95, 99-$quartile values of Δz_i^* for algorithm 1 for each instance size n.

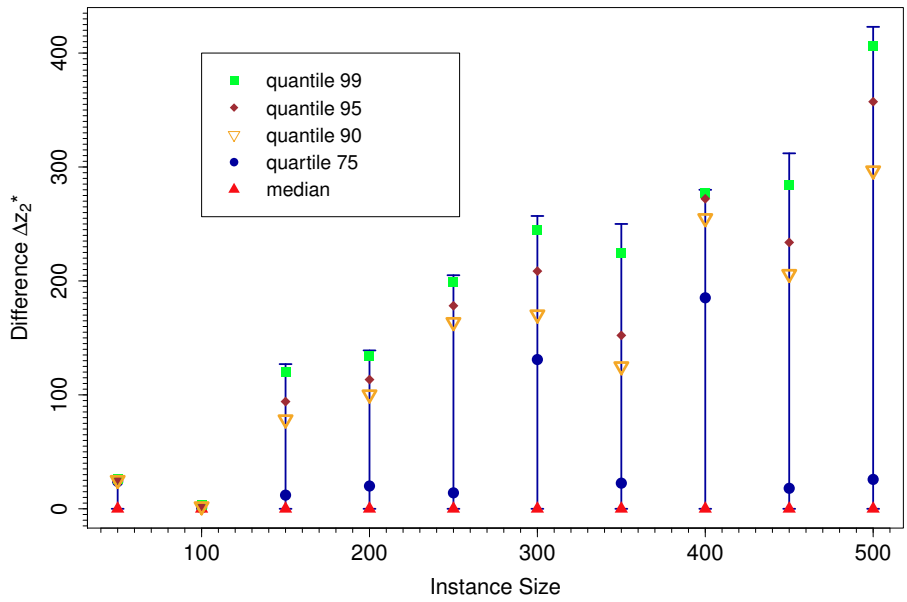

Figure 8: Maximum, minimum, median and $90, 95, 99-$quartile values of Δz_i^* for algorithm 2 for each instance size n.

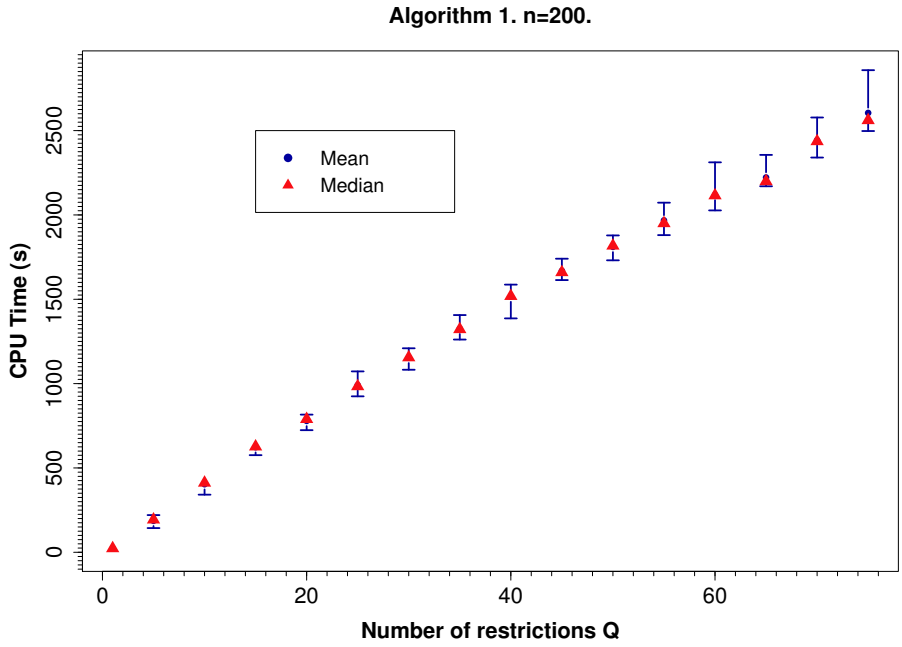

Figure 9: Maximum, minimum, mean and median values of CPU Times (in seconds) for algorithm 1 for each number of restrictions Q.

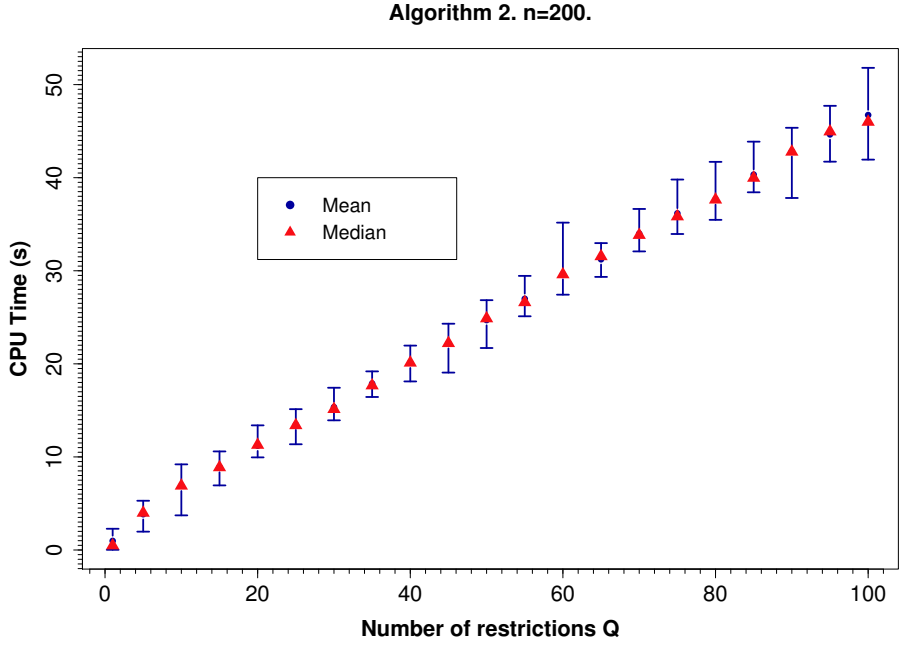

Figure 10: Maximum, minimum, mean and median values of CPU Times (in seconds) for algorithm 2 for each number of restrictions Q.

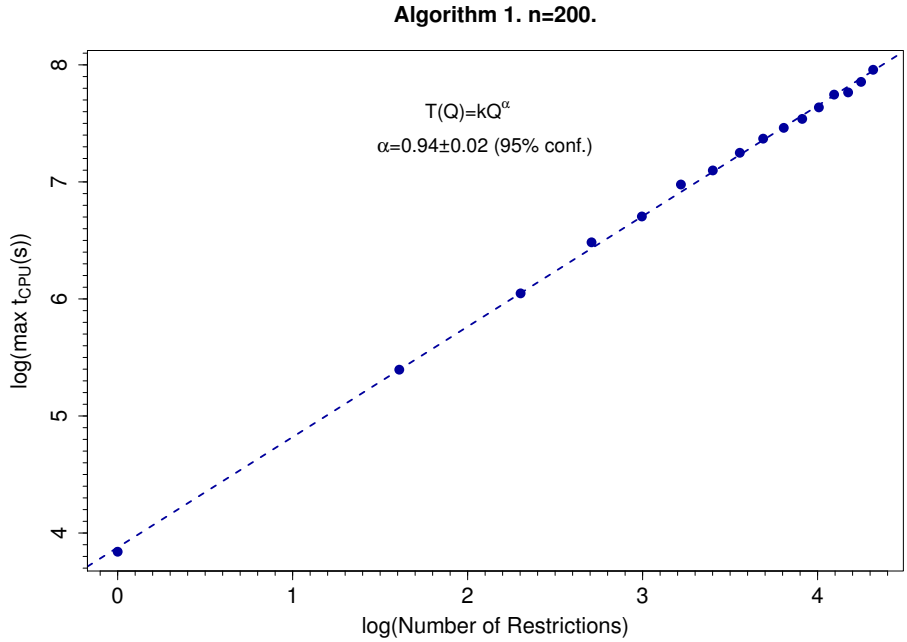

Figure 11: Regression line to estimate α in $T(Q) = O(Q^{\alpha})$ for algorithm 1.

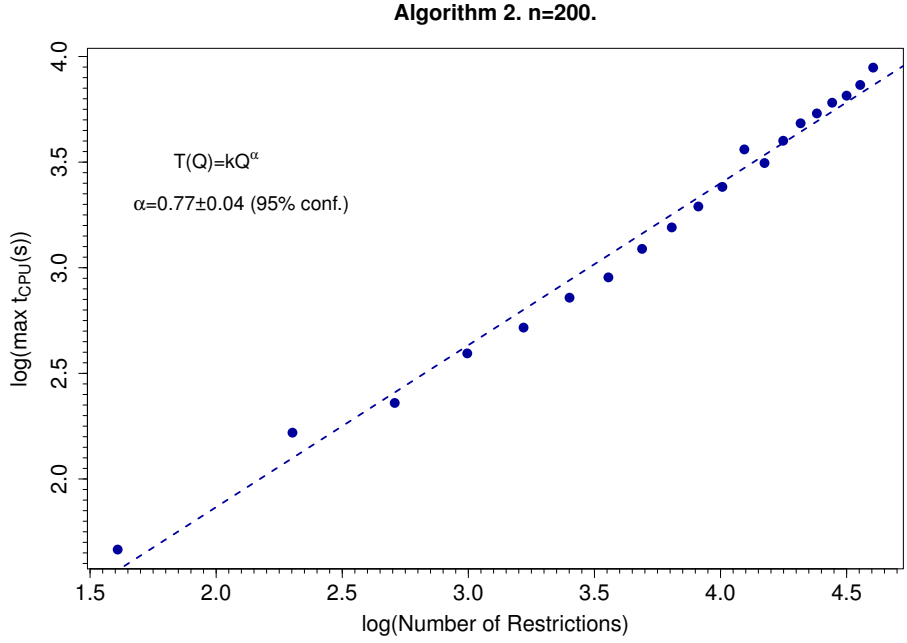

Figure 12: Regression line to estimate α in $T(Q) = O(Q^{\alpha})$ for algorithm 2.