Quadratic interior-point methods in statistical disclosure control

Jordi Castro
Dept. of Statistics and Operations Research
Universitat Politècnica de Catalunya
Pau Gargallo 5, 08028 Barcelona (Catalonia, Spain)
`jcastro@eio.upc.es`

Technical Report DR 2003-10
May 2003

Quadratic interior-point methods in statistical disclosure control

Jordi Castro [*]
Dept. of Statistics and Operations Research
Universitat Politècnica de Catalunya
Pau Gargallo 5, 08028 Barcelona (Spain)
jcastro@eio.upc.es

Abstract

The safe dissemination of statistical tabular data is one of the main concerns of National Statistical Institutes (NSIs). Although each cell of the tables is made up of the aggregated information of several individuals, the statistical secret can be violated. NSIs must guarantee that no individual information can be derived from the released tables. One widely used type of methods to reduce the disclosure risk is based on the perturbation of the cell values. We consider a new controlled perturbation method which, given a set of tables to be protected, finds the closest safe ones—thus reducing the information loss while preserving confidentiality. This approach means solving a quadratic optimization problem with a much larger number of variables than constraints. Real instances can provide problems with millions of variables. We show that interior-point methods are an effective choice for that model, and, also, that specialized algorithms which exploit the problem structure can be faster than state-of-the art general solvers. Computational results are presented for instances of up to 1000000 variables.

Key words:Interior-point methods – Quadratic Programming – Large-scale programming – Statistical confidentiality – Controlled perturbation methods

AMS subject classification 90C06 – 90C20 – 90C51 – 90C90

1 Introduction

In the current Information Society, National Statistical Institutes (NSIs) play a fundamental role, routinely releasing large volumes of data for their further exploitation. The released data are usually classified as aggregated or disaggregated. Disaggregated data (a.k.a. microdata or microfiles) correspond to files of records, each record providing the values for a set of variables of an individual. Aggregated data (a.k.a. tabular data) are obtained from microdata crossing two or more variables, which results in sets of tables with a likely large number of cells. In this paper we focus on tabular data protection (see, e.g., [20, 23] for a comprehensive introduction to this field). Although each cell shows aggregated

[*]Partially supported by the EU IST-2000-25069 CASC project.

		z_1	z_2	
⋮	...	...	...	...
51–55	...	38000€	40000€	...
56–60	...	39000€	42000€	...
⋮	...	...	...	...

(a)

		z_1	z_2	
⋮	...	...	...	...
51–55	...	20	1 or 2	...
56–60	...	30	35	...
⋮	...	...	...	...

(b)

Figure 1: Example of disclosure in tabular data. (a) Average salary per age and zip code. (b) Number of individuals per age and zip code

data for several individuals, there is a risk of disclosing individual data. This is clearly shown in the example of Fig. 1. The table (a) of that Figure gives the average salary for age interval and zip code, while table (b) shows the number of individuals for the same variables. If there was only one individual in zip code z_2 and age interval 51–55, then any external attacker would know the salary of this single person is 40000 €. For two individuals, any of them could deduce the salary of the other, becoming an internal attacker. Usually, cells showing information about few individuals are considered sensitive, although other rules can be used in practice. See, for instance, [10, 22] for a recent discussion about sensitivity rules.

In the example of Figure 1 we should protect table (a), a single two-dimensional table. This can be considered the simplest case. However, in practice we must deal with more complex situations. The full range can be classified as:

- Single two-dimensional, three-dimensional, and, in general, multidimensional tables. Those tables can be individually protected.

- Hierarchical tables, i.e., sets of tables with variables that have a hierarchical relation (e.g., zip code and city). In that case, the total or marginal cells for some table can be the internal cells for the others. They have to be protected together, to avoid the disclosure of sensitive data.

- Linked tables. It is a generalization of the previous situation, where several tables are made from the same microdata, thus sharing information or cells, either hierarchical or not. Again, they have to be protected together.

Eventually, we could consider the whole set of linked tables that can be produced from some microfiles (e.g., a population census). Clearly, the number of cells involved in that case could be of several millions. All the above situations can both refer to frequency tables (i.e., cell values are integer and are usually associated to the number of individuals in that cell) or magnitude tables (i.e., cell values are real, and, for instance, can show the mean for some other variable of all the individuals in that cell).

The methods for protection of tabular data can be classified in perturbative (they change the cell values) or nonperturbative (no change is performed). The most widely used nonpertutative method is *cell suppression*, where some *secondary* cells are removed to avoid the disclosure of some sensitive *primary* cells (which are removed as well). That results in a difficult combinatorial optimization problem, which finds the pattern of secondary suppressions that makes the table safe with a minimum number of cells or information loss. Some heuristics [3, 8, 16] and exact methods [11, 12] have been suggested for the cell suppression problem.

Among the perturbative methods, one of the techniques that received more attention due to its simplicity was *rounding*. This method rounds cell values to a multiple of a fixed integer rounding base. *Controlled rounding* is a variant where the additivity of the table is preserved (i.e., rounded marginal values are the sum of the corresponding slice of rounded cells) [2]. However, preserving the (integer) additivity is not easy in a multidimensional table or a set of linked tables. Moreover, in practice it can be necessary to maintain the original marginal values, instead of rounding them.

To avoid the above lacks of rounding, we suggest a new perturbation method that finds the minimum-L_2-distance (or closest) tables to those to be protected, preserving marginal values, as well as any set of additional linear constraints. Finding the minimum-L_2-distance tables means we try to minimize the information loss when delivering the perturbed table. In this work we focus on tables of magnitudes (i.e., cell values $\in \mathbb{R}$). For tables of frequencies (integer cell values) this procedure could still be applied followed by some heuristic post-process. The main drawback of this approach is the solution of a very large quadratic optimization problem. We'll show that interior-point methods can solve this problem very efficiently. Recently, and independently of this work, a similar idea was suggested in [9], using the L_1 distance for the perturbed table. In practice, however, the optimization problem obtained with the L_2 distance can be solved more efficiently, as shown in [6].

The structure of the document is as follows. In Section 2 we introduce the *minimum-L_2-distance* perturbation method. In Section 3 we show some computational experience using a general state-of-the-art interior-point solver. From those results we conclude it is worth using a specialized interior-point algorithm that exploits the problem structure. This is the subject of Section 4. Finally in Section 5 we present some computational results in the solution of three-dimensional tables of up to 1000000 cells using a specialized interior-point algorithm.

2 The *minimum-L_2-distance* perturbation method

Any instance problem, either with one table or a number of (linked or hierarchical) tables, can be represented by the following elements:

- A set of cells $a_i, i = 1, \ldots, n$, that satisfy some linear relations $Aa = b$ (a being the vector of a_i's). For instance, for a two-dimensional table of $r+1$ rows and $c+1$ columns (last row and column correspond to marginal values) we have

$$\sum_{i=1}^{r} a_{ij} = a_{(r+1)j} \qquad j = 1 \ldots c \tag{1}$$

$$\sum_{j=1}^{c} a_{ij} = a_{i(c+1)} \qquad i = 1 \ldots r. \tag{2}$$

For a three-dimensional table with $l+1$ levels (levels correspond to third dimension, last level is marginal), the relations are

$$\begin{aligned} \sum_{i=1}^{r} a_{ijk} &= \quad a_{(r+1)jk} \qquad j=1\ldots c, \quad k=1\ldots l && (3)\\ \sum_{j=1}^{c} a_{ijk} &= \quad a_{i(c+1)k} \qquad i=1\ldots r, \quad k=1\ldots l && (4)\\ \sum_{k=1}^{l} a_{ijk} &= \quad a_{ij(l+1)} \qquad i=1\ldots r, \quad j=1\ldots c. && (5)\end{aligned}$$

In practice most tables have positive cell values, and bounds $a \geq 0$ have to be considered.

- A lower and upper bound for each cell $i = 1,\ldots,n$, respectively $\underline{a}_i$ and $\overline{a}_i$, which are considered to be known by any attacker. If no previous knowledge is assumed for cell i, we would simply set $\underline{a}_i = 0$ ($\underline{a}_i = -\infty$ if bounds $a \geq 0$ were not assumed) and $\overline{a}_i = +\infty$.
- A set $\mathcal{P} = \{i_1, i_2, \ldots, i_p\}$ of indices of confidential cells.
- A lower and upper protection level for each confidential cell $i = 1,\ldots,i_p$, respectively lpl_i and upl_i, such that the released value should be greater or equal than $a_i + upl_i$ or less or equal than $a_i - lpl_i$. Modelling these "or" constraints would need an extra binary variable for each confidential cell, resulting in a combinatorial optimization problem. To avoid it, we'll assume the user (e.g., the NSI) fixes in advance the sense of the protection for each confidential cell.

The *minimum-L_2-distance* method finds (using the L_2 distance) the closest set of *perturbed* values x_i to a_i, $i = 1,\ldots,n$, such that the tables relations, lower and upper cell bounds, and sensitive cells protection levels are satisfied. The optimization problem can be written as

$$\begin{array}{rl} \min\limits_{x} & ||x-a||_2^2 \\ \text{subject to} & Ax = b \\ & \underline{a}_i \leq x_i \leq \overline{a}_i \quad i = 1\ldots n \\ & x_i \leq a_i - lpl_i \text{ or } x_i \geq a_i + upl_i \quad i \in \mathcal{P}, \end{array} \qquad (6)$$

x being the vector of the cell values $x_i, i = 1\ldots n$ of the perturbed table.

Problem (6) can be applied to any kind of table, since it does not constraint the structure of the cell relations $Ax = b$. Any other set of linear conditions can also be added to (6). For instance, we could impose that, in the perturbed table, values x_i related to some non-confidential cells must be close enough to the original values a_i, e.g., $(1-\alpha)a_i \leq x_i \leq (1+\beta)a_i$ for some small α and β. For cells corresponding to national or regional totals, or for cells with a zero value, $\alpha = \beta = 0$ can be a good choice (i.e., we don't perturb the original cell value). This is the usual practice in those situations. We may also want to affect the distance by any positive semidefinite diagonal metric matrix $W = \text{diag}(w_1, \dots, w_n)$ (e.g., $w_i = 1/a_i$). In the computational results of Sections 3 and 5 we used $w_i = 1$. The more general model can be written as:

$$\begin{aligned} \min_x \quad & (x-a)^T W (x-a) && (7) \\ \text{subject to} \quad & Ax = b && (8) \\ & c \le Tx \le d && (9) \\ & l \le x \le u, && (10) \end{aligned}$$

(9) being any set of additional linear equality or inequality constraints, and l and u in (10) the final lower and upper bounds of the perturbed cell values.

3 Computational experience using a general interior-point solver

(7–10) is a large (possibly with millions of variables), separable convex quadratic problem. It is known that the computational cost of quadratic separable problems is the same than for linear ones, if solved through an interior-point algorithm [25]. Therefore, in principle, that seems to be the best choice.

To confirm the good behaviour of interior-point methods in this problem, we performed some preliminary limited computational experience with some small two-dimensional and three-dimensional tables. These instances, and those used below in this Section and in Section 5, were obtained with two different generators that have been used in the literature. The first generator follows the description of [16]. Cell values are randomly obtained from an integer uniform distribution $[1 \dots 1000]$ with probability 0.8 and are 0 with probability 0.2. The second one is similar to the first generator of [11]. Cell values are randomly obtained from integer uniform distributions $[1 \dots 4]$ for confidential cells and $\{0\} \cup [5 \dots 500]$ for the remaining entries. Cells to be protected are randomly chosen from the internal (i.e., nonmarginal) cells in both generators. We extended the original generators for three-dimensional tables using the same distributions. They can be obtained from the author on request.

We generated three small two-dimensional and three-dimensional tables, whose sizes are showed in Table 1. Columns r, c, l and $|\mathcal{P}|$ give, respectively, the number of rows, columns, levels and number of sensitive cells for each instance. The first three rows correspond to two-dimensional tables, and thus column l is empty. Columns n and m show, respectively, the number of variables (number of cells) and constraints of the resulting quadratic optimization problem. Note that, for two-dimensional problems and from (1,2), $n = rc$ and $m = r + c$, while for three-dimensional instances and from (3–5), $n = rcl$ and $m = (r + c)l + rc$. As for the rest of instances of the paper, we considered that the lower and upper bounds known for each cell are $0.9a_i$ and $1.1a_i$, respectively, and the value of the sensitive cells was set to $x_i = 1.1a_i$ in the perturbed table. We solved each problem with four solvers: the interior-point barrier algorithm of Cplex 8.0 [15], the dual and primal simplex algorithms for quadratic problems of Cplex 8.0 (see, e.g, [24] for a description of this simplex variant), and Minos 5.5 [18, 19]—one of the most efficient reduced-gradient type solvers. The CPU time in seconds required by each solver is shown, respectively, in columns "Barrier", "Dual", "Primal"and "Minos". The executions were carried on a Compaq Evo N610c

Table 1: Dimensions of some small instances and results with four solvers

						Cplex 8.0			
r	c	l	$\|\mathcal{P}\|$	n	m	Barrier	Dual	Primal	Minos
25	25	—	10	625	50	0.01	0.02	0.08	1.63
50	50	—	20	2500	100	0.03	0.09	1.02	11.14
100	100	—	20	10000	200	0.21	0.66	22.50	>700*
10	10	10	20	1000	300	0.05	0.19	0.66	3.15
15	15	15	20	3375	675	0.29	2.92	16.3	164.71
25	25	25	20	15625	1875	4	160	868	>4600*

* Execution was stopped

Table 2: Dimensions of some large instances and results with the barrier Cplex 8.0 solver

r	c	l	$\|\mathcal{P}\|$	n	m	CPU
1000	500	—	10000	500000	1500	47.1
1000	750	—	10000	750000	1750	72.9
1000	1000	—	10000	1000000	2000	136.0
100	100	25	10000	250000	15000	198.5
100	100	50	10000	500000	20000	896.7
100	100	100	10000	1000000	30000	Not enough memory

notebook, with a Pentium Mobile 4 at 1.8 GHz and 512 MB of RAM. The problem (7–10) was implemented using the AMPL modelling language [13]. Looking at Table 1, it is clear that the interior-point solver is the best option, mainly when the size of the instances increases.

Unfortunately, real problems are much larger than those used for Table 1. And for large instances even general interior-point solvers can be computationally expensive. This is clearly shown in Table 2, which reports the CPU time in seconds (column "CPU") required by the interior-point barrier algorithm of Cplex 8.0 for three two-dimensional and three-dimensional large instances. The meaning of the other columns is the same that in Table 1. The largest three-dimensional problem, involving one million of cells, could not be solved with the available memory. The two-dimensional problem with the same number of variables did not have such limitation. In fact, two-dimensional problems could be solved more efficiently. This is mainly due to the lesser number of constraints they involve. It can be concluded that general interior-point solvers are too expensive, both in memory and execution time, when applied to large instances. Next Section shows that using a specialized interior-point solver which exploits the problem structure is instrumental.

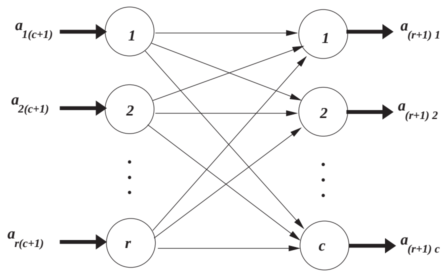

Figure 2: Network representation of a $(r + 1) \times (c + 1)$ table

4 Exploiting the problem structure

4.1 Two-dimensional tables

The linear relations (1,2) of a $(r + 1) \times (c + 1)$ table can be modeled as the network of Figure 2. Arcs are associated to cells and nodes to equations. Injections correspond to marginal row $r + 1$ and marginal column $c + 1$. Thus (7,8,10)—assuming no extra constraints (9) are considered—is a large convex separable quadratic minimum-cost network flows problem. Some effective specialized interior-point methods have been devised for linear network problems. They solve the normal equations at each iteration of the interior-point method (discussed below) by a preconditioned conjugate gradient. The most effective preconditioner is the "maximum spanning tree preconditioner" [21], and its variants [17]. As far as we know, it has not been applied to quadratic network flows problems. In that case, the preconditioner would probably not preserve its good properties, mainly when we are are close to the optimal solution: in a quadratic problem the optimizer is no longer located in a vertex, and thus the maximum spanning tree does not correspond to any optimal basis. However, in the first and intermediate iterations of the interior-point method, it could be more efficient than using a Cholesky factorization, as a general solver does. Therefore, in principle, there is room for improving the execution times showed in Table 2. This is still part of the further work to be done.

4.2 Three-dimensional tables

As we observed in Table 2, three-dimensional tables are computationally more challenging than two-dimensional ones with a similar number of cells. As shown in [5], the relations (3–5) of a three-dimensional table are equivalent to those of a multicommodity network flows problem with equality mutual capacity constraints (see, e.g., Chapter 17 of [1] for an introduction to multicommodity problems). Indeed, the structure of the constraints (8) defined by (3–5) is

$$\begin{bmatrix} N & & & \\ & N & & \\ & & \ddots & \\ & & & N \\ \mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} \end{bmatrix} \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^l \end{bmatrix} = \begin{bmatrix} b^1 \\ b^2 \\ \vdots \\ b^l \\ a^{l+1} \end{bmatrix}, \tag{11}$$

$N \in \mathbb{R}^{(r+c)\times(rc)}$ being the network linear relations of the two-dimensional table associated to each level (depicted in Figure 2), $x^k \in \mathbb{R}^{rc}, k = 1\dots l$, the cells (flows) of level k, $b^k \in \mathbb{R}^{r+c}, k = 1\dots l$, the row and column marginals (injections) of level k, and $a^{l+1} \in \mathbb{R}^{rc}$ the level marginal values (mutual arc capacities). Constraints involving the network matrix N correspond to (3–4), while (5) are the linking constraints. If marginal values (i.e., the right-hand-side term of (11)) are considered also variables (for instance, with a large penalization for deviations from the original values) the structure of (8) would be an extension of the standard multicommodity network flows problem:

$$\begin{bmatrix} N & & & -\mathbb{1} & & & \\ & \ddots & & & \ddots & & \\ & & N & & & -\mathbb{1} & \\ \mathbb{1} & \cdots & \mathbb{1} & & & & -\mathbb{1} \end{bmatrix} \begin{bmatrix} x^1 \\ \vdots \\ x^l \\ b^1 \\ \vdots \\ b^l \\ x^{l+1} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}. \qquad (12)$$

The optimization problem (7,8,10) to be solved for a three-dimensional table is thus a convex separable quadratic multicommodity network flows problem, which can be written in standard form as

$$\begin{array}{ll} \min\limits_{x} & \sum\limits_{k=1}^{l} \left((c^k)^T x^k + (x^k)^T Q^k x^k\right) \\ \text{subject to} & Ax = b \\ & 0 \le x^k \le u^k, k = 1\dots l, \end{array} \qquad (13)$$

where $Ax = b$ is either (11) or (12) (in the latter case, marginal cells are also variables, and then the two l's in (13) should be replaced by $l+1$, and the variables b^k of (12) should be considered included in the associated x^k of (13)).

We extended the specialized interior-point method of [4] (initially developed for linear multicommodity problems) for the solution of (13). As far as we know, this is the only specialized method for general quadratic multicommodity flows, unlike the linear case, where there are several available algorithms (see, e.g, Chapter 17 of [1]). In fact, the method developed in [4] is not restricted to multicommodity problems. It can also be used for a wide range of block diagonal structured problems. In particular, it solves the extended formulation (12). Next, we justify the above assertions. Most details about the original specialized interior-point algorithm for linear multicommodity problems are omitted; they can be found in [4].

The most expensive computation of a primal-dual interior-point method is the solution of the normal equations

$$(A\Theta A^T)\Delta y = \bar{b} \qquad (14)$$

at each iteration. A is the constraints matrix of (13), Δy is the direction for the dual variables, and Θ is a positive definite matrix, which can be partitioned in l (or $l+1$ if formulation (12) is considered) blocks, one for each commodity. The expression of each block is

$$\Theta^k = ((\Theta^k_{lin})^{-1} + Q^k)^{-1}, \qquad (15)$$

Θ^k_{lin} being the positive definite diagonal matrix of the linear problem, and Q^k the matrix of quadratic costs of commodity k in (13). In our context, Q^k is diagonal; therefore Θ^k is also diagonal and easily computable.

Exploiting the structures of A in (11) and Θ, we obtain

$$A\Theta A^T = \left[\begin{array}{cc} B & C \\ C^T & D \end{array}\right] = \left[\begin{array}{ccc|c} N\Theta^1 N^T & \cdots & 0 & N\Theta^1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & N\Theta^l N^T & N\Theta^l \\ \hline \Theta^1 N^T & \cdots & \Theta^l N^T & \sum_{k=1}^{l} \Theta^k \end{array}\right]. \tag{16}$$

If, instead, the extended formulation (12) is considered we have

$$A\Theta A^T = \left[\begin{array}{cc} B & C \\ C^T & D \end{array}\right] = \left[\begin{array}{ccc|c} N\Theta^1_i N^T + \Theta^1_m & \cdots & 0 & N\Theta^1_i \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & N\Theta^l_i N^T + \Theta^l_m & N\Theta^l_i \\ \hline \Theta^1_i N^T & \cdots & \Theta^l_i N^T & \sum_{k=1}^{l+1} \Theta^k_i \end{array}\right], \tag{17}$$

where the Θ^k_i and Θ^k_m matrices correspond, respectively, to the internal and marginal cells of the table.

Either using (16) or (17), and appropriately partitioning Δy and $\bar{b}$, we can write (14) as

$$\left[\begin{array}{cc} B & C \\ C^T & D \end{array}\right] \left[\begin{array}{c} \Delta y_1 \\ \Delta y_2 \end{array}\right] = \left[\begin{array}{c} \bar{b}_1 \\ \bar{b}_2 \end{array}\right]. \tag{18}$$

By block multiplication, we can reduce (18) to

$$(D - C^T B^{-1} C)\Delta y_2 = (\bar{b}_2 - C^T B^{-1} \bar{b}_1) \tag{19}$$

$$B\Delta y_1 = (\bar{b}_1 - C\Delta y_2). \tag{20}$$

Following [4], (20) is solved by performing a Cholesky factorization of each diagonal block of B, while system with matrix

$$H = D - C^T B^{-1} C, \tag{21}$$

the Schur complement of (18), is solved by a preconditioned conjugate gradient method. A good preconditioner is instrumental for the performance of the method. The preconditioner of [4] was developed for linear multicommodity problems. However, it can be applied to any problem where the following result holds:

Proposition 1 *If D and $D + C^T BC$ in (18) are positive definite then the inverse of (21) can be computed as*

$$H^{-1} = \left(\sum_{i=0}^{\infty} (D^{-1}(C^T BC))^i \right) D^{-1}. \tag{22}$$

Proof. See pp. 860–861 of [4].

Both (16) and (17) satisfy the premises of Proposition 1. Their D^{-1} matrices can be easily computed, since

$$D = D_{lin} + \sum_{k=1}^{l} (Q^k)^{-1},$$

D_{lin} being the diagonal positive definite matrix for the linear problem without the quadratic term. The preconditioner for those problems, as for the linear ones, is thus obtained by truncating the infinite power series (22) at some term h . In practice $h = 0$ or $h = 1$ are good choices. Note that for $h = 0$ the preconditioner is equal to $\hat{H}^{-1} = D^{-1}$, which, since Q^k are diagonal, is also diagonal. This is instrumental in the overall performance of the algorithm. All the computational results of this work were obtained with $h = 0$. The goodness of the preconditioner depends of the spectral radius of $D^{-1}(C^T BC))$, and, in practice, it was observed to work better for quadratic than for linear problems [7]. A thorough study of the behaviour of the spectral radius for quadratic problems is part of the further work to be done.

5 Computational experience using a specialized interior-point algorithm

We implemented the specialized interior-point method described in the last Section in a code named QIPM. It is a quadratic extension of the package IPM, originally developed in [4] for linear multicommodity problems. IPM can be found in `http://www-eio.upc.es/~jcastro`. QIPM can be obtained from the author on request. We compared the performance of QIPM against Cplex 8.0 using a set of 162 three-dimensional instances, obtained with the two generators described in Section 3. The 81 instances for each generator were produced considering all the combinations for $r, c, l \in \{25, 50, 100\}$ and $|\mathcal{P}| \in \{1000, 5000, 10000\}$ (r, c, l and $|\mathcal{P}|$ with the same meaning that in Tables 1 and 2). The lower and upper bounds, and the values of the sensitive cells were computed as in Section 3.

Table 3 shows the results obtained with Cplex 8.0 and QIPM for some of the largest instances. Column "g" gives the generator used. Columns r, c, l, $|\mathcal{P}|$, n and m have the same meaning that in previous tables. Columns "it." and "CPU" show the number of interior-point iterations and the execution times for both codes. The execution environment was described in Section 3. The largest instances could not be solved with Cplex 8.0 due to a lack of memory. Clearly, the specialized interior-point method is about two orders of magnitude faster than the general solver, although the number of iterations is similar. Also, the execution times of QIPM smoothly increase with the size of the problem, and almost proportionally to the number of variables. It is worth noting that QIPM uses standard Cholesky factorization routines, whereas Cplex 8.0 includes a highly tuned and optimized factorization code. Then, in principle, there is still room for improving the QIPM performance.

The results obtained for all the 162 instances are summarized in Figures 3–5. The holes observed in those figures correspond to infeasible problems due to the tight bounds considered. Figure 3 shows the ratio between the CPU times of

Table 3: Dimensions and results with Cplex 8.0 and QIPM for some of the largest instances

							Cplex 8.0		QIPM	
g	r	c	l	$\|\mathcal{P}\|$	n	m	it.	CPU	it.	CPU
1	100	50	100	1000	500000	20000	8	892.9	9	6.4
2	100	50	100	1000	500000	20000	7	923.5	7	5.5
1	100	50	100	5000	500000	20000	8	1284.4	9	6.5
2	100	50	100	5000	500000	20000	7	909.2	7	5.4
1	100	50	100	10000	500000	20000	8	884.7	9	6.4
2	100	50	100	10000	500000	20000	7	912.7	7	5.4
1	100	100	25	1000	250000	15000	8	185.1	9	4.1
2	100	100	25	1000	250000	15000	8	206.2	7	3.4
1	100	100	25	5000	250000	15000	9	205.2	10	4.4
2	100	100	25	5000	250000	15000	8	205.1	7	3.5
1	100	100	25	10000	250000	15000	8	179.2	11	4.9
2	100	100	25	10000	250000	15000	8	198.5	7	3.4
1	100	100	50	1000	500000	20000	8	875.4	9	7.8
2	100	100	50	1000	500000	20000	7	899.3	7	6.7
1	100	100	50	5000	500000	20000	8	792.3	9	7.7
2	100	100	50	5000	500000	20000	7	909.6	7	6.5
1	100	100	50	10000	500000	20000	8	866.3	9	7.8
2	100	100	50	10000	500000	20000	7	896.7	7	6.6
1	100	100	100	1000	1000000	30000		*	8	14.1
2	100	100	100	1000	1000000	30000		*	7	13.0
1	100	100	100	5000	1000000	30000		*	9	15.6
2	100	100	100	5000	1000000	30000		*	7	13.4
1	100	100	100	10000	1000000	30000		*	9	15.5
2	100	100	100	10000	1000000	30000		*	7	12.8

* Not enough memory

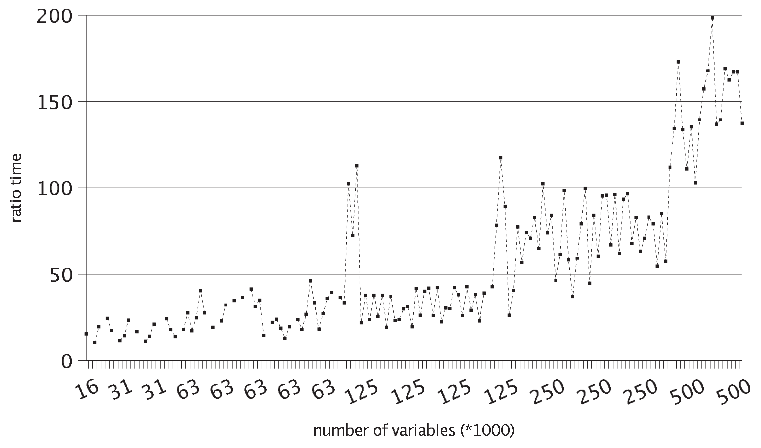

Figure 3: Ratio of the Cplex 8.0 and QIPM CPU times

Cplex 8.0 and QIPM with respect to the number of variables of the problem. As shown by the Figure, the ratios increase with the problem size, which makes QIPM a very efficient tool for large instances. For instances with more than 250000 variables, QIPM was at least 50 times faster than Cplex 8.0. For the largest instances it was about 175 times faster.

Figure 4 shows the objective function relative error—in absolute value—of the QIPM solution, $\left|(f^*_{\text{Cplex}} - f^*_{\text{QIPM}})/(1 + f^*_{\text{Cplex}})\right|$, assuming Cplex 8.0 provides the exact one. The default optimality tolerance used in [4] for linear problems was 10^{-6}. Although for quadratic problems this tolerance could likely be decreased, since the preconditioner works better than in the linear case, we preserved that default value. This explains why most of the relative errors are around 10^{-6} in Figure 4. Although there is no a clear tendency, the largest relative errors were obtained in some of the largest instances.

Finally, Figure 5 shows the difference between the number of interior-point iterations of QIPM and Cplex 8.0, with respect to the problem size. All the differences are in the range -1,…,3. QIPM, at most, saved one iteration, while Cplex 8.0 saved two in several instances. The values are quite uniformly distributed and independent of the number of variables of the instances.

6 Conclusions

The results obtained with the minimum-L_2-distance method show that, first, it can be a promising tool for the protection of statistical tabular data; and second, that specialized interior-point methods can solve large instances in few seconds, more efficiently than general solvers. However, this work can be improved in several ways. First, the minimum-L_2-distance method can be adjusted to fit the real needs of NSIs, which would likely mean the inclusion of additional constraints or terms in the objective function. Second, we have to consider the general situation, that involves multidimensional, hierarchical and linked tables. That means the development of specialized interior-point solvers for the

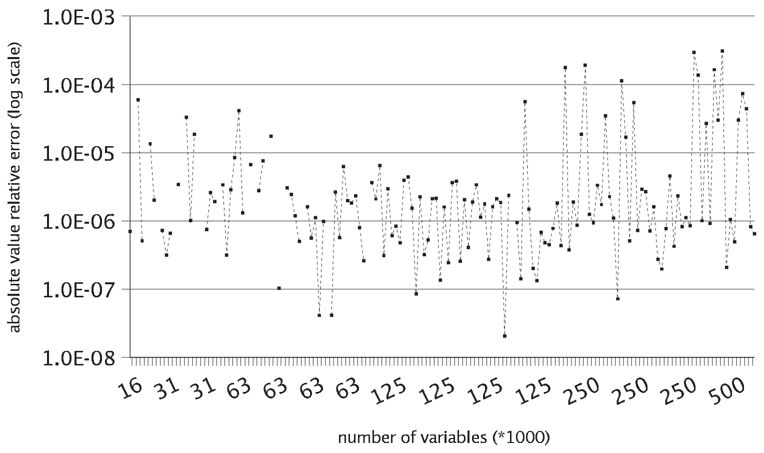

Figure 4: Objective function relative error of the QIPM solution, in absolute value

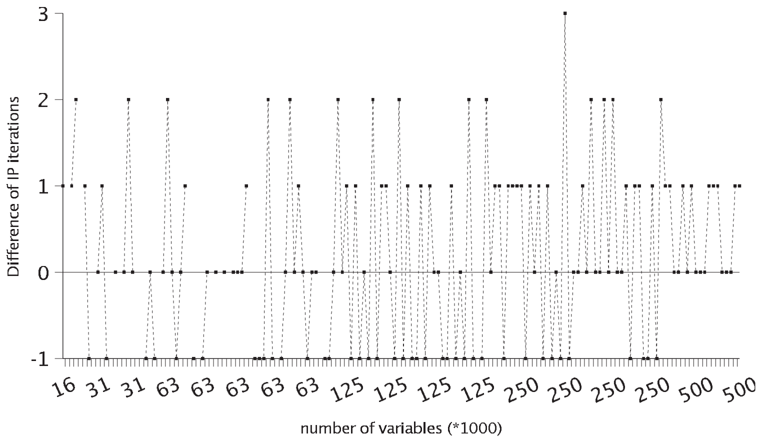

Figure 5: Difference between the number of interior-point iterations performed by QIPM and Cplex 8.0

resulting structured problems [14]. Extending the method for frequency tables, through some type of heuristic post-process, is another of the future tasks to be done.

References

[1] R.K Ahuja, T.L Magnanti and J.B. Orlin, *Network Flows* (Prentice Hall, Upper Saddle River 1993).

[2] M. Bacharach, Matrix rounding problems, Management Science **9**, (1966) 732–742.

[3] F.D. Carvalho, N.P. Dellaert and M.D. Osório, Statistical disclosure in two-dimensional tables: general tables, J. of the American Statistical Association **89,** (1994) 1547–1557.

[4] J. Castro, A specialized interior-point algorithm for multicommodity network flows, SIAM J. on Optimization **10(3),** (2000) 852–877.

[5] J. Castro, Network flows heuristics for complementary cell suppression: an empirical evaluation and extensions, in *Inference Control in Statistical Databases. Lecture Notes in Computer Science, vol. 2316*, J. Domingo-Ferrer (Ed.), (Springer, Berlin 2002) 59–73.

[6] J. Castro, Minimum-distance controlled perturbation methods for tabular data protection, working paper, Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya (2003).

[7] J. Castro, Solving quadratic multicommodity problems through an interior-point algorithm, in *System Modelling and Optimization*, E.W. Sachs (Ed.), (Kluwer, in press). Avalaible from `http://www-eio.upc.es/~jcastro` as Research Report DR2001-14, Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya (2001).

[8] L.H. Cox, Network models for complementary cell suppression, J. of the American Statistical Association **90,** (1995) 1453–1462.

[9] R.A. Dandekar and L.H. Cox, Synthetic tabular data: an alternative to complementary cell suppression, manuscript, Energy Information Administration, U.S. Dept. of Energy, (2002). Available from the first author on request (`Ramesh.Dandekar@eia.doe.gov`).

[10] J. Domingo-Ferrer and V. Torra, A critique of the sensitivity rules usually employed for statistical table protection, Int. J. of Uncertainty Fuzziness and Knowledge-based Systems **10(5)**, (2002) 545–556.

[11] M. Fischetti and J.J. Salazar, Models and algorithms for the 2-dimensional cell suppression problem in statistical disclosure control, Mathematical Programming **84,** (1999) 283–312.

[12] M. Fischetti and J.J. Salazar, Models and algorithms for optimizing cell suppression in tabular data with linear constraints, J. of the American Statistical Association **95,** (2000) 916–928.

[13] R. Fourer, D.M. Gay and B.W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming* (Boyd & Fraser, Danvers 1993).

[14] J. Gondzio and R. Sarkissian, Parallel interior point solver for structured linear programs, Mathematical Programming (to appear). Avalaible from `http://www.maths.ed.ac.uk/~gondzio` as Technical Report MS-00-025, Dept. of Mathematics and Statistics, University of Edinburgh (2002).

[15] ILOG CPLEX, *ILOG CPLEX 8.0 Reference Manual Library* (ILOG, Gentilly 2002).

[16] J.P. Kelly, B.L. Golden and A.A. Assad, Cell Suppression: disclosure protection for sensitive tabular data, Networks **22,** (1992) 28–55.

[17] S. Mehrotra and J. Wang, Conjugate gradient based implementation of interior point methods for network flow problems, in *AMS Summer Conference Proceedings*, L. Adams and J. Nazareth (Eds.), (SIAM, Philadelphia 1996) 124–142.

[18] B.A.Murtagh and M.A. Saunders, Large-scale linearly constrained optimization, Mathematical Programming **14,** (1978) 41–72.

[19] B.A. Murtagh and M.A. Saunders, MINOS 5.0. User's guide, Dept. of Operations Research, Stanford University, (1983).

[20] A. Oganian, *Security and Information Loss in Statistical Database Protection* (PhD thesis, Dept. of Applied Mathematics 4, Universitat Politècnica de Catalunya 2002).

[21] M.G.C. Resende and G. Veiga, An implementation of the dual affine scaling algorithm for minimum-cost flow on bipartite uncapacitated networks, SIAM J. on Optimization **3,** (1993) 516–537.

[22] D.A. Robertson and R. Ethier, Cell suppression: experience and theory, in *Inference Control in Statistical Databases. Lecture Notes in Computer Science, vol. 2316*, J. Domingo-Ferrer (Ed.), (Springer, Berlin 2002) 8–20.

[23] L. Willenborg and T. de Waal, *Elements of Statistical Disclosure Control. Lecture Notes in Statistics, vol. 155* (Springer, New York 2000).

[24] P. Wolfe, The simplex method for quadratic programming, Econometrica **27,** (1959) 382–398.

[25] S.J. Wright, *Primal-Dual Interior-Point Methods* (SIAM, Philadelphia 1997).