Automatic Structure Detection in Constraints of Tabular Data

Jordi Castro[1,⋆,⋆⋆] and Daniel Baena[2]

[1] Department of Statistics and Operations Research,
Universitat Politècnica de Catalunya,
Jordi Girona 1–3, 08034 Barcelona, Catalonia
jordi.castro@upc.edu
http://www-eio.upc.es/~jcastro
[2] Institut d'Estadística de Catalunya,
Via Laietana 58, 08003 Barcelona, Catalonia
dbaena@idescat.net

Abstract. Methods for the protection of statistical tabular data—as controlled tabular adjustment, cell suppression, or controlled rounding—need to solve several linear programming subproblems. For large multi-dimensional linked and hierarchical tables, such subproblems turn out to be computationally challenging. One of the techniques used to reduce the solution time of mathematical programming problems is to exploit the constraints structure using some specialized algorithm. Two of the most usual structures are block-angular matrices with either linking rows (primal block-angular structure) or linking columns (dual block-angular structure). Although constraints associated to tabular data have intrinsically a lot of structure, current software for tabular data protection neither detail nor exploit it, and simply provide a single matrix, or at most a set of smallest submatrices. We provide in this work an efficient tool for the automatic detection of primal or dual block-angular structure in constraints matrices. We test it on some of the complex CSPLIB instances, showing that when the number of linking rows or columns is small, the computational savings are significant.

Keywords: statistical disclosure control, cell suppression, controlled tabular adjustment, linear constraints, multilevel matrix ordering algorithms.

1 Introduction

From an algorithmic point of view, one of the main differences between disclosure control techniques for microdata and tabular data is that the latter must deal with many linear constraints, associated to total and subtotal cells. Current methods for tabular data protection, such as, e.g., cell suppression [4,9,14], controlled tabular adjustment [6,12] and controlled rounding [10,20], deal with

⋆ Supported by the Spanish MCyT Project TIC2003-00997.
⋆⋆ Corresponding author.

those linear additivity constraints through mathematical programming technology. Unfortunately, the resulting optimization problems turn out to be computationally expensive, even for continuous variables. For instance, the simplex algorithm, which is the preferred option for many linear programming problems [2], has shown to be inefficient compared to polynomial-time interior-point algorithms [21] when dealing with tabular data constraints [6,20].

One of the most used techniques in mathematical programming for reducing the computational cost of a problem is to exploit its structure, either through decomposition or partitioned basis factorization. Two of the most relevant structures are primal block-angular

$$A = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_k \\ L_1 & L_2 & \dots & L_k \end{bmatrix}, \tag{1}$$

where k is the number of diagonal blocks, $A \in \mathbb{R}^{m \times n}, A_i \in \mathbb{R}^{m_i \times n_i}, L_i \in \mathbb{R}^{l \times n_i}, i = 1, \dots, k$, and dual block-angular

$$A = \begin{bmatrix} A_1 & & & L_1 \\ & A_2 & & L_2 \\ & & \ddots & \vdots \\ & & A_k & L_k \end{bmatrix}, \tag{2}$$

k, A, A_i being as before, and $L_i \in \mathbb{R}^{m_i \times l}$. Structures (1) and (2) appear in problems with l linking constraints and l linking variables, respectively, and have been extensively studied in the literature [3, Chapter 12]. Classical decomposition procedures, based on the simplex method, are Dantzig-Wolfe for primal block-angular structures [13], and Benders for dual block-angular ones [1]. Specialized interior-point approaches for structured problems have also been recently suggested [8,15]; these are promising approaches for tabular data protection, since, as noted above, interior-point methods outperform simplex implementations in this class of problems. It is worth noting that homogeneous sizes for diagonal blocks A_i benefit the performance of any decomposition approach, mainly if some sort of parallelism is going to be applied.

Unfortunately, current methods and software for tabular data protection do not exploit constraints structure in general tables. Structure has only been exploited for two-dimensional tables with at most one hierarchical variable, whose constraints are modeled as a network [7,9], and for three-dimensional tables, that provide multicommodity network models [5]. For general tables, state-of-the-art protection software, as τ-Argus [17], provide a single constraints matrix, or at most a set of linked submatrices, without detailing each matrix structure. Indeed, the constraints structure is particular to each kind of table, and it is not clear that fully exploiting such structure would be worthy for an optimization algorithm. In addition, writing software for the detection of the particular structure of any table would be a cumbersome task. The purpose of this work is to overcome such difficulties, i.e., to develop and test a tool for automatic detection of structures (1) and (2) in constraints matrices derived from tabular data. The tool developed is based on the multilevel matrix ordering algorithm for unsymmetric matrices of [16]. Unlike other highly recognized and efficient algorithms, such as that implemented in METIS [18], the former is tailored for unsymmetric matrices, which is the case for tabular data constraints. The procedure is applied to a set of standard tabular data instances, being its behaviour instance dependent: for most instances and $k = 2$ a small linking block (i.e., l/m (l/n) for primal (dual) block-angular structures is less than 0.2), whereas for others the relative size of the linking block can be up to 0.4 (for $k > 2$ these relative sizes of the linking block grow). When the relative size of the linking block is small it makes sense to apply a decomposition approach, and we provide preliminary computational results comparing the performance of a linear programming solver depending on whether structure is exploited.

The paper is organized as follows. Section 2 outlines the multilevel matrix reordering algorithm and gives details about its implementation. Section 3 reports results using this algorithm on a standard set of tabular data instances. Finally, Section 4 analyzes the computational savings due to using the reordered matrix in a mathematical programming solver.

2 The Matrix Reordering Algorithm

Given any matrix, obtaining the optimal reordering that transforms the matrix into either the primal block-angular structure (1) or the dual one (2) is a difficult combinatorial optimization problem (in this context, "optimal" means "with the smallest linking block"). Several heuristics have been provided in the past for this problem. We have chosen the recent multilevel approach of [16], which is among the most efficient ones for unsymmetric matrices, and it is implemented in commercial libraries (e.g., in routine HSL_MC66L of the HSL archive, formerly the Harwell Subroutine Library). A comprehensive description of such algorithm is out of the scope of this work, and it will just be outlined; details can be found in [16] and references therein.

We first define some concepts to be used later in the overview of the algorithm. Given a sparse matrix $A \in \mathbb{R}^{m \times n}$, the *net* of column j is the number of rows associated to such column, i.e, $\{i \in \{1, \dots, m\} : a_{ij} \neq 0\}$. A *row partition* is a partition of the set of rows $\{1, \dots, n\}$, i.e., R_1, R_2 such that $R_1 \cap R_2 = \emptyset$ and $R_1 \cup R_2 = \{1, \dots, m\}$. A net is cut by a row partition if there are rows of the net in both R_1 and R_2. The *net-cut* of a row partition is the number of nets cut by this row partition. Note that in a block-diagonal matrix, i.e, without neither linking constraints nor linking columns, the net-cut is zero. Therefore this is the value to be reduced by any heuristic in order to obtain close-to-block-angular structures. The *gain* is the decrement of the net-cut obtained after moving a row from R_1 to R_2 or vice-versa; the gain is negative if the net-cut is increased.

The *edge-weight* of rows (i_1, i_2) is the number of columns shared by these rows, i.e, the cardinality of $\{j \in \{1, \ldots, n\} : a_{i_1,j} \neq 0 \text{ and } a_{i_2,j} \neq 0\}$.

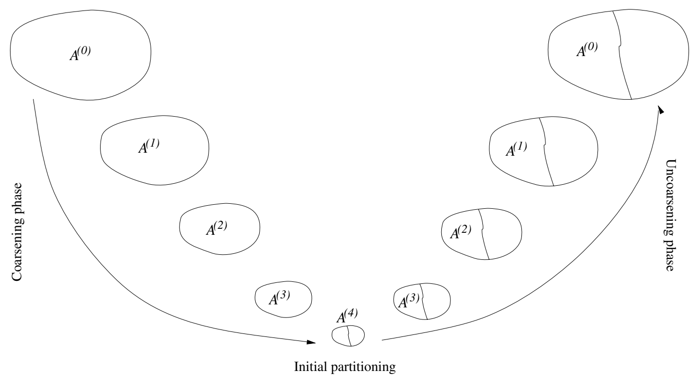

Fig. 1. The three stages of the multilevel ordering algorithm: coarsening, partitioning, uncoarsening; example with 4 levels

The multilevel ordering algorithm of [16] is made of the three following stages, which are shown in Figure 1:

Coarsening phase. Matrix $A = A^{(0)}$ is successively transformed in a sequence of smaller matrices $A^{(1)}, A^{(2)}, \ldots, A^{(r)}$, r being the deepest level, such that the number of rows is reduced at each transformation, i.e., $m = m^{(0)} > m^{(1)} > m^{(2)} > \ldots > m^{(r)}$. The procedure successively collapses the "closest rows" in a single one using the notion of edge-weight defined above. In our particular implementation we used the heavy-edge matching criterion of [18] (see that reference for details).

Partitioning phase. It is based on the classical Kernighan-Lin (KL) heuristic [19] for partitioning graphs. Graphs are known to be associated to symmetric sparse matrices, i.e., $a_{ij} \neq 0$ and $a_{ji} \neq 0$ if there is an edge joining nodes i and j in the graph. The KL heuristic can be extended for unsymmetric matrices if we use the notion of *net-cut* and *gain*, instead of the original one of *edge-cut* (i.e., number of edges in a graph cut by a node partition) of the KL algorithm. In short, the KL algorithm is an iterative procedure that starting from an initial row partition $R_1^{(0)}, R_2^{(0)}$, performs two nested loops. The inner iterations successively look for the row with the largest gain. This row is obtained from the set $R_1^{(0)}$ or $R_2^{(0)}$ with the maximum cardinality, in an attempt to guarantee similar dimensions for diagonal blocks. This row is moved to the other subset of rows, subsets $R_1^{(i)}, R_2^{(i)}$ are updated, and the row is locked. This is performed until all rows have been locked. This ends the inner iterations. The outer iterations repeat the above sequence of inner iterations, unlocking all rows at the beginning, until there is no improvement in the overall net-cut. The procedure records the best partitioning up to now obtained, which is returned as the solution.

If one needs more than two diagonal blocks (i.e., $k > 2$ in (1) or (2)), the KL algorithm can be recursively applied to any resulting submatrix defined by R_1 or R_2, thus eventually obtaining a row partition $R_1, R_2, \ldots, R_k$.

Uncoarsening phase. During this phase the partitioning of $A^{(r)}$ is projected to the original matrix through matrices $A^{(r-1)}$, $A^{(r-2)}$, ..., $A^{(1)}$, $A^{(0)}$. Two rows i_1 and i_2 collapsed in a single one in A^h belonging to $R_p, 1 \leq p \leq k$, will appear as two different rows of R_p in A^{h-1} . Optionally, any new matrix A^h can be refined with the KL algorithm.

The KL algorithm itself can be applied to the matrix A (i.e., as if we had a single level algorithm, or equivalently with a coarsening phase with $r = 0$). However, as it will be shown in Section 3, it is computationally expensive, because of the large number of rows m of A, and can provide unsatisfactory partitionings. On the other hand, applied in combination with a multilevel approach (i.e, coarsening and uncoarsening phases) it is extremely efficient, and the partitioning is significantly improved. Note that the coarsening phase collapses rows with the largest edge-weight, and such rows are expected to be in the same subset R_p in a solution, which is exactly what the uncoarsening phase does.

The multilevel algorithm above described has been implemented in C, in a library named *UMOA* (unconstrained matrix ordering algorithm), which is roughly made of 3100 lines of code. The package can be freely obtained from the authors. It has been optimized using efficient data structures for the coarsening and uncoarsening phases (i.e., AVL trees), as well as for the computation and updating of gains and net-cuts in the KL algorithm. Among the several features of the package, we mention that: (i) when $k > 2$ the package allows full control to the user about how to recursively obtain the additional subsets R_p from R_1 and R_2; (ii) it can be used to obtain both primal and dual block-angular structures (1) and (2). (We note that the reordered matrix in primal block-angular form is not equivalent to the transpose of the reordered matrix in dual block-angular form.)

3 Reordering Tabular Data Instances

We applied the multilevel reordering algorithm of Section 2 to a subset of the CSPLIB test suite, a set of instances for tabular data protection (Fischetti and Salazar 2001), plus to additional large instances ("five20b", and "five20c"). CSPLIB can be freely obtained from `http://webpages.ull.es/users/casc/-#CSPlib:`. CSPLIB contains both low-dimensional artificially generated problems, and real-world highly structured ones. Some of the complex instances were contributed by National Statistical Agencies—as, e.g., Centraal Bureau voor de Statistiek (Netherlands), Energy Information Administration of the Department of Energy (U.S.), Office for National Statistics (United Kingdom) and Statistisches Bumdesant (Germany).

Table 1 shows the features of the instances considered. The small CSPLIB instances were omitted. Column "Name" shows the instance identifier. Columns "n", "m" and "N. coef" provide, respectively, the number of columns (cells),

Table 1. Dimensions of the largest CSPLIB instances

Name	n	m	N.coef
bts4	36570	36310	136912
five20b	34552	52983	208335
five20c	34501	58825	231345
hier13	2020	3313	11929
hier13x13x13a	2197	3549	11661
hier13x7x7d	637	525	2401
hier16	3564	5484	19996
hier16x16x16a	4096	5376	21504
jjtabeltest	3025	1650	7590
nine12	10399	11362	52624
nine5d	10733	17295	58135
ninenew	6546	7340	32920
targus	162	63	360
toy3dsarah	2890	1649	9690
two5in6	5681	9629	34310

rows (additivity constraints) and nonzero coefficients of the constraints matrix A to be reordered.

Tables 2, 3 and 4 show, respectively, the results obtained reordering the matrices in dual block-angular form for $k = 2$, in primal block-angular form for $k = 2$, and in dual-block angular form for $k = 4$ and 8. Last two columns of Tables 2 and 3 provide information for the single level algorithm (i.e., KL algorithm was applied to the whole matrix). Columns "m_1", "n_1", "m_2" and "n_2" of tables 2 and 3 provide the number of rows m_i and columns n_i of the two diagonal blocks. Columns "$100{\cdot}l/n$" ("$100{\cdot}l/m$") of Tables 2 and 4 (Table 3) give the relative size of the linking columns (constraints) block. Columns "CPU" of the three tables report the seconds of CPU time required to compute the reordering. The runs were carried on a standard PC running Linux with an AMD Athlon 1600+ at 1.4GHz and 320 MB of RAM. Therefore, the reordered matrix can be efficiently obtained without the need of sophisticated computational resources.

From Tables 2–4 it can be concluded that:

– The multilevel approach is instrumental in reordering tabular data constraints. It is not only one order of magnitude faster that the single level approach, but also provides much better reorderings (i.e., the linking block becomes much narrower). In particular, all the matrices could be reordered in few seconds on a desktop computer.
– The sizes of the diagonal blocks are similar, which may be a benefit for an optimization solver. This is also instrumental if parallel computations want to be exploited.
– The size of the linking block is instance dependent (from 7.4% in instance "bts4" of Table 2 to 59.8% in instance "hier16" of Table 3). Thus, in principle, not all the reordered matrices are appropriate for a specialized solver for structured problems.

Table 2. Results for dual block-angular ordering with $k = 2$

	Multilevel						Single level	
	A_1		A_2					
Name	m_1	n_1	m_2	n_2	$100 \cdot l/n$	CPU	$100 \cdot l/n$	CPU
bts4	17838	16898	18472	16937	7.4	2	21.4	53
five20b	25196	14343	27787	15586	13.3	6	48.4	597
five20c	28820	14034	30005	15093	15.5	7	54.3	416
hier13	1656	583	1657	563	43.2	0	45.0	0
hier13x13x13a	1890	774	1659	526	40.8	0	46.0	0
hier13x13x7d	792	419	651	218	46.1	0	37.8	0
hier16	2486	740	2998	1069	49.2	1	55.2	1
hier16x16x16a	2614	1074	2762	1138	45.9	0	45.2	0
jjtabeltest	849	1269	801	1383	12.3	0	15.1	0
ninenew	3007	2115	4333	3085	20.5	0	40.5	1
nine5d	9007	4568	8288	4303	17.3	1	47.1	9
nine12	5880	4249	5482	4098	19.7	3	35.5	3
targus	19	26	44	92	27.1	0	21.6	0
toy3dsarah	741	1118	908	1389	13.2	0	24.7	0
two5in6	5344	1749	4285	1502	42.7	0	56.7	3

Table 3. Results for primal block-angular ordering with $k = 2$

	Multilevel						Single level	
	A_1		A_2					
Name	m_1	n_1	m_2	n_2	$100 \cdot l/m$	CPU	$100 \cdot l/m$	CPU
bts4	16307	18389	16137	18181	10.5	2	21.7	37
five20b	26006	17735	22576	16817	12.7	4	57.9	50
five20c	26693	17286	26218	17215	17.1	5	60.3	41
hier13	1375	1091	1076	929	42.6	0	53.7	0
hier13x13x13a	1320	1075	1428	1122	36.4	0	51.6	0
hier13x13x7d	483	562	510	621	38.0	0	36.8	0
hier16	1674	1732	1676	1832	59.8	0	60.4	0
hier16x16x16a	1989	2118	1676	1978	41.7	0	44.6	0
jjtabeltest	737	1457	686	1568	7.5	0	16.8	0
ninenew	3181	3239	2721	3307	21.9	1	46.8	1
nine5d	7111	5383	7975	5350	20.5	1	40.5	2
nine12	4846	5206	4666	5193	17.7	1	45.6	2
targus	22	82	22	80	11.7	0	19.1	0
toy3dsarah	534	1446	529	1444	20.2	0	20.2	0
two5in6	3630	2816	3817	2865	38.4	0	37.1	0

– In general, the size of the linking block increases with k, the number of diagonal blocks. Although the more diagonal blocks, the more "decomposable" becomes the solution of linear systems of equations involving that matrix (see Section 4), the overall solution procedure can become very inefficient due to a large linking block. This tradeoff is usually optimized by small

Table 4. Results for dual block-angular ordering with $k = 4$ and $k = 8$

	$k = 4$		$k = 8$	
Name	$100 \cdot l/n$	CPU	$100 \cdot l/n$	CPU
bts4	15.4	3	22.9	3
five20b	34.0	7	43.8	9
five20c	26.8	9	44.9	11
hier13	66.9	0	82.8	0
hier13x13x13a	64.1	0	80.8	0
hier13x13x7d	71.7	0	88.6	0
hier16	73.2	1	85.4	0
hier16x16x16a	69.4	0	83.7	0
jjtabeltest	24.6	0	36.9	0
ninenew	34.1	0	54.3	0
nine5d	44.3	1	70.6	1
nine12	31.0	1	47.6	1
targus	53.7	0	54.3	0
toy3dsarah	45.6	0	68.9	0
two5in6	73.9	0	87.2	0

values of k (e.g., $k = 2$, 3 or 4), unless the information about the data allows specific larger ones.

– Primal and dual block-angular structures provide linking blocks of different size. The best (primal or dual) ordering is instance dependent, and it seems not to be a clear trend.

The above conclusions are consistent with those obtained in [16] for other types of matrices (for the dual block-angular structure, the only one considered in [16]).

Figures 2–5 of Appendix A show the original matrix, 2-blocks dual, 2-blocks primal, and 4-blocks dual reorderings for the four largest instances tested.

4 Using the Reordered Matrices

The numerical kernel of any optimization algorithm is to deal with linear systems of equations derived from the constraints matrix. If the block bordered structure of the constraints matrix is exploited, significant savings can be obtained. This is valid for both simplex and interior-point methods, which have been extensively used for cell-suppression, controlled tabular adjustment, and controlled rounding. We will focus on the use of primal block-angular matrices in interior-point methods (which have shown to be the most efficient option for tabular data), and will test them for the controlled tabular adjustment problem.

The main computational burden of an interior-point method [21] is to solve systems of equation with matrix $A\Theta A^T$, where Θ is a diagonal positive definite matrix. For our purposes, and without loss of generality, we will assume $\Theta = I$, thus, the system to be solved is

$$(AA^T)\Delta y = g. \tag{3}$$

This system is named the "normal equations" and it is usually solved by a sparse Cholesky factorization. If A has the structure of (1), we can recast the matrix of system (3) as

$$AA^T = \left[\begin{array}{ccc|c} A_1A_1^T & & & A_1L_1^T \\ & \ddots & & \vdots \\ & & A_kA_k^T & A_kL_k^T \\ \hline L_1A_1^T & \dots & L_kA_k^T & \sum_{i=1}^k L_iL_i^T \end{array}\right] = \begin{bmatrix} B & C \\ C^T & D \end{bmatrix}, \tag{4}$$

B, C and D being the blocks of AA^T. Appropriately partitioning g and Δy in (3), the normal equations can be written as

$$\begin{bmatrix} B & C \\ C^T & D \end{bmatrix} \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}. \tag{5}$$

By eliminating Δy_1 from the first group of equations of (5), we obtain

$$(D - C^TB^{-1}C)\Delta y_2 = (g_2 - C^TB^{-1}g_1) \tag{6}$$
$$B\Delta y_1 = (g_1 - C\Delta y_2). \tag{7}$$

Therefore we have reduced the solution of system (3) to the solution of systems with matrix B (which is made of k smallest subsystems $A_iA_i^T, i = 1, \dots, k$) and with one system with matrix $(D - C^TB^{-1}C)$, which is named the "Schur complement" of (3). In general, if the solution of systems with $A_iA_i^T, i = 1, \dots, k$ are not too expensive, and the sparsity (the number of nonzero elements) of the Schur complement is not degraded (i.e, decreased too much), we can obtain significant computational savings by solving (6)–(7) instead of (3). Moreover, there are efficient procedures for (6) based on iterative linear solvers [8].

The above procedure, using a preconditioned conjugate gradient (i.e., iterative solver) for (6) with a specialized preconditioner (see [5,8] for details), has been implemented in an interior-point package for primal block-angular problems [8]. Such procedure can be used for the efficient solution of controlled tabular adjustment (CTA) problems, once the tabular data constraints have been previously reordered as shown in Section 3. Table 5 reports preliminary computational results with an early implementation of CTA based on the specialized interior-point algorithm of [8]. For each of the instances of Table 1—but the two largest ones, that failed with the iterative solver—we solved systems (6)–(7) from some interior-point iterations with a sparse Cholesky factorization (the standard procedure used by general interior-point solvers, such as CPLEX) and with the specialized procedure of [8]. Column "Ratio time" of Table 5 show the ratio between both solution times, i.e., how many times faster is the specialized procedure compared to the standard one. We note that the problems were not

Table 5. Ratio time for the solution of CTA by an interior-point method without and with exploitation of structure

Name	Ratio time
bts4	1.5
hier13	12.7
hier13x13x13a	11.6
hier13x13x7d	3.9
hier16	43.5
hier16x16x16a	43.2
jjtabeltest	0.7
ninenew	7.5
nine5d	2.8
nine12	5.1
targus	1.0
toy3dsarah	6.0
two5in6	10.9

solved up to optimality with the approach of [8], since that procedure has still to be tuned for problems like CTA (which has, for instance, equality linking constraints instead of the inequality ones considered in [8]). However those figures are a good indicator of the expected overall performance in the solution of CTA by exploiting the constraints structure in an interior-point method.

5 Conclusions

The structure detection tool used in this work for constraints of tabular data can provide significant computational savings for CTA, and, in general, for any tabular data protection method that has to solve a sequence of linear programming subproblems. Many additional tasks have still to be done. Among them, the main one is to tune the specialized approach of [8] for fully exploiting the reordered matrix, and for obtaining an optimal solution to CTA in a fraction of the time needed by a general solver.

References

1. Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Computational Management Science **2** (2005) 3–19
2. Bixby, R.E.: Solving real-world linear programs: a decade and more of progress. Operations Research **50** (2002) 3–15
3. Bradley, S.P, Hax, A.C., Magnanti, T.L.: Applied Mathematical Programming. Addison-Wesley, Reading (1977).
4. Castro, J.: Network flows heuristics for complementary cell suppression: an empirical evaluation and extensions. Lect. Notes in Comp. Sci. **2316** (2002) 59–73. Volume Inference Control in Statistical Databases, ed. J. Domingo-Ferrer, Springer, Berlin

5. Castro, J.: Quadratic interior-point methods in statistical disclosure control. Computational Management Science **2** (2005) 107–121
6. Castro, J.: Minimum-distance controlled perturbation methods for large-scale tabular data protection. European Journal of Operational Research **171** (2006) 39–52
7. Castro, J.: A shortest paths heuristic for statistical disclosure control in positive tables. To appear in INFORMS Journal on Computing. Available as Research Report DR 2004/10 Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya, 2004
8. Castro, J.: An interior-point approach for primal block-angular problems. To appear in Computational Optimization and Applications (2007). Available as Research Report DR 2005/20 Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya, 2005
9. Cox, L.H.: Network models for complementary cell suppression. J. Am. Stat. Assoc. **90** (1995) 1453–1462
10. Cox, L. H., George, J. A.: Controlled rounding for tables with subtotals. Annals of Operations Research **20** (1989) 141–157
11. Dandekar, R.A.: personal communication (2005)
12. Dandekar, R.A., Cox, L.H.: Synthetic tabular Data: an alternative to complementary cell suppression, manuscript, Energy Information Administration, U.S.
13. Dantzig, G.B., Wolfe, P.: Decomposition principle for linear programs. Operations Research **8** (1960) 101–111
14. Fischetti, M., Salazar, J.J.: Solving the cell suppression problem on tabular data with linear constraints. Management Science **47** (2001) 1008–1026
15. Gondzio, J., Sarkissian, R.: Parallel interior point solver for structured linear programs. Mathematical Programming **96** (2003) 561–584
16. Hu, Y.F., Maguire, K.C.F., Blake, R.J.: A multilevel unsymmetric matrix ordering algorithm for parallel process simulation. Computers and Chemical Engineering **23** (2000) 1631–1647
17. Hundepool, A.: The CASC project. Lect. Notes in Comp. Sci. **2316** (2002) 172–180. Volume Inference Control in Statistical Databases, ed. J. Domingo-Ferrer, Springer, Berlin
18. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on scientific Computing **20** (1999) 359–392.
19. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Systems Technical Journal **49** (1970) 291–308
20. Salazar, J.J.: Controlled rounding and cell perturbation: statistical disclosure limitation methods for tabular data. Mathematical Programming **105** (2006) 583–603
21. Wright, S.J.: Primal-Dual Interior-Point Methods. SIAM, Philadelphia (1997).

A Sparsity Pattern of Original and Reordered Matrices

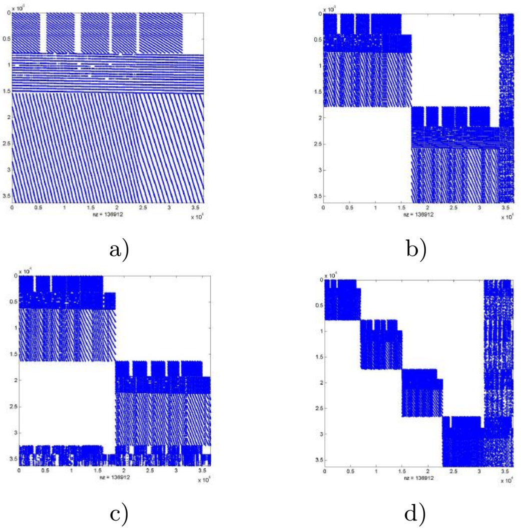

Fig. 2. Results for instance bts4: a) original matrix; b) reordered matrix in dual 2-blocks-angular form; c) reordered matrix in primal 2-blocks-angular form; d) reordered matrix in dual 4-blocks-angular form

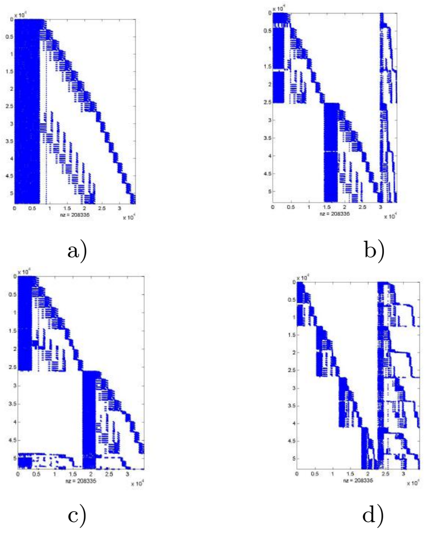

Fig. 3. Results for instance five20b: a) original matrix; b) reordered matrix in dual 2-blocks-angular form; c) reordered matrix in primal 2-blocks-angular form; d) reordered matrix in dual 4-blocks-angular form

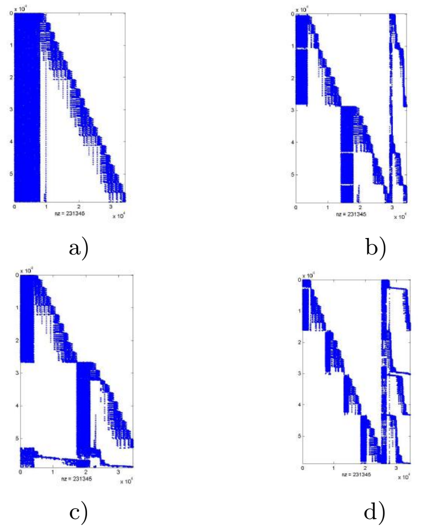

Fig. 4. Results for instance five20c: a) original matrix; b) reordered matrix in dual 2-blocks-angular form; c) reordered matrix in primal 2-blocks-angular form; d) reordered matrix in dual 4-blocks-angular form

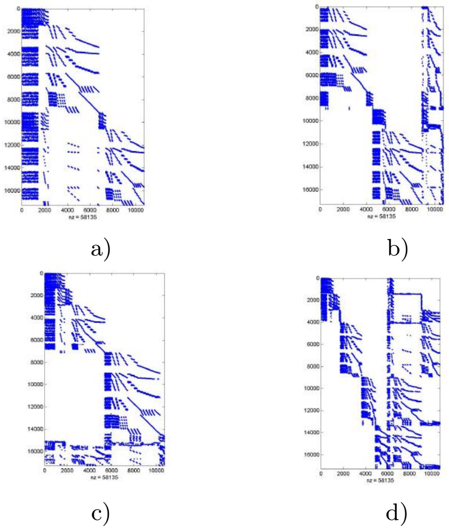

Fig. 5. Results for instance nine5d: a) original matrix; b) reordered matrix in dual 2-blocks-angular form; c) reordered matrix in primal 2-blocks-angular form; d) reordered matrix in dual 4-blocks-angular form