Network Flows Heuristics for Complementary Cell Suppression: An Empirical Evaluation and Extensions*

Jordi Castro**

Statistics and Operations Research Dept., Universitat Politècnica de Catalunya
Pau Gargallo 5, 08028 Barcelona (Spain)
jcastro@eio.upc.es
http://www-eio.upc.es/ jcastro

Abstract. Several network flows heuristics have been suggested in the past for the solution of the complementary suppression problem. However, a limited computational experience using them is reported in the literature, and, moreover, they were only appropriate for two-dimensional tables. The purpose of this paper is twofold. First, we perform an empirical comparison of two network flows heuristics. They are improved versions of already existing approaches. Second, we show that extensions of network flows methods (i.e., multicommodity network flows and network flows with side constraints) can model three-dimensional, hierarchical and linked tables. Exploiting this network structure can improve the performance of any solution method solely based on linear programming formulations.
Keywords: Complementary cell suppression problem, linear programming, network optimization, network flows with side constraints, multicommodity network flows.

1 Introduction

Cell suppression is a widely used technique by statistical agencies to avoid the disclosure of confidential tabular data. Given a list of primary cells to be protected, the objective of the cell suppression problem (CSP) is to find a set of complementary cells that have to be additionally suppressed. This pattern of suppressions is found under some criteria as, e.g., minimum number of suppressions, or minimum value suppressed.

CSP was shown to be NP-hard in [19]. This motivated that most of the former approaches focused on heuristic methods for approximate solutions. Methods based on graph theory were suggested for two-dimensional tables in [4], and extended for three dimensions in [8]; they were designed for general tables. The hypercube method, currently under consideration by the Federal Statistical Office of Germany, was also based in geometric considerations of the problem [14].

* Work supported by the IST-2000-25069 CASC project.
** Author supported by CICYT Project TAP99-1075-C02-02.

J. Domingo-Ferrer (Ed.): Inference Control in Statistical Databases, LNCS 2316, pp. 59–73, 2002.

A different kind of techniques was obtained by using linear programming (LP), in particular, network optimization. This paper is devoted to this kind of methods. We will perform an empirical evaluation of two network flows heuristics. It will also be shown how three-dimensional, linked and hierarchical tables can be modeled as multicommodity network flows and network flows with side constraints [1]. This allows the use of specialized LP codes that exploit the network structure of the problem [5,6,13].

There is a fairly extensive literature on network flows methods for CSP. In [19] an algorithm for sliding protection was suggested, but it was only applied to small scale two-dimensional tables. In [7] several alternative network methods were reviewed, some of them successfully applied in practice in U.S. [18] and Canada [21] (although in this latter case a pure LP formulation was considered). However, these heuristics were only appropriate for two-dimensional tables, since they relied on a minimum cost network flows solver. Multi-dimensional, linked and hierarchical tables had to be split into several two-dimensional tables, which forced some kind of backtracking procedure. This inconvenient could be removed by using general LP solvers. However, limitations of past LP technology resulted in inefficient implementations [18]. As noted in [3] this drawback has been overcome by current LP solvers, and some steps have been performed for including them in CSP production codes [20]. Exploiting that three-dimensional, linked and hierarchical tables can be modeled through multicommodity networks and networks with side constraints opens the possibility of using a new range of solvers for CSP.

Recently, an exact procedure based on state-of-the-art mixed integer linear programming (MILP) techniques (i.e., branch-and-cut and Bender's decomposition) was proposed in [9,10]. This method has been able to solve large non-trivial CSP instances very efficiently. As stated in [10], the exact algorithm developed includes an initial LP-based heuristic phase, which is similar to those suggested in the past using network flows codes. Therefore, the exact procedure can also take profit of recent improvements in heuristic methods. Moreover, the Bender's decomposition subproblems could also benefit of the underlying network with side constraints structure of the constraints matrix.

This paper is organized as follows. Section 2 shows the exact formulation and a linear relaxation of CSP. The two heuristics considered in this work are based on the exact and relaxed formulations, respectively. These heuristics are outlined in Section 3 and compared in Section 4. Section 5 shows the extension of network flows models for three-dimensional, linked and hierarchical tables. Finally, Section 6 presents some preliminary computational results with a network flows heuristic for three-dimensional tables.

2 Formulation of CSP

Given a (usually) positive table (i.e., a set of cells $a_i \geq 0, i = 1 \dots \overline{n}$, satisfying some linear relations $Aa = b$), a set $\mathcal{P}$ of $|\mathcal{P}|$ primary cells to be protected, and upper and lower protection levels U_i and L_i for each primary cell $i = 1 \dots |\mathcal{P}|$,

the purpose of CSP is to find a set $\mathcal{C}$ of additional complementary cells whose suppression guarantees that, for each $p \in \mathcal{P}$,

$$\underline{a_p} \leq a_p - L_p \quad \text{and} \quad \overline{a_p} \geq a_p + U_p, \tag{1}$$

$\underline{a_p}$ and $\overline{a_p}$ being defined as

$$\underline{a_p} = \min_{x_i, i=1\ldots\overline{n}} x_p \quad \text{s.t.} \quad \begin{array}{l} Ax = b \\ x_i \geq 0 \;\; i \in \mathcal{P} \cup \mathcal{C} \\ x_i = a_i \;\; i \notin \mathcal{P} \cup \mathcal{C} \end{array} \quad \text{and} \quad \overline{a_p} = \max_{x_i, i=1\ldots\overline{n}} x_p \quad \text{s.t.} \quad \begin{array}{l} Ax = b \\ x_i \geq 0 \;\; i \in \mathcal{P} \cup \mathcal{C} \\ x_i = a_i \;\; i \notin \mathcal{P} \cup \mathcal{C}. \end{array} \tag{2}$$

$\underline{a_p}$ and $\overline{a_p}$ in (2) are the lowest and greatest possible values that can be deduced for each primary cell from the published table, once the entries in $\mathcal{P} \cup \mathcal{C}$ have been suppressed. Imposing (1), the desired level of protection is guaranteed. CSP can thus be formulated as an optimization problem of minimizing some function that measures the cost of suppressing additional cells subject to that conditions (1) and (2) are satisfied for each primary cell.

CSP was first formulated in [19] as a large MILP problem. For each entry a_i a binary variable $y_i, i = 1 \ldots \overline{n}$ is considered. y_i is set to 1 if the cell is suppressed, otherwise is 0. For each primary cell $p \in \mathcal{P}$, two auxiliary vectors $x^{l,p} \in \mathbb{R}^{\overline{n}}$ and $x^{u,p} \in \mathbb{R}^{\overline{n}}$ are introduced to impose, respectively, the lower and upper protection requirements of (1) and (2). These vectors represent cell deviations (positive or negative) from the original a_i values. The resulting model is

$$\begin{array}{ll} \min & \displaystyle\sum_{i=1}^{\overline{n}} a_i y_i \\ \text{s.t.} & \\ & \left.\begin{array}{rrll} & Ax^{l,p} & = 0 & \\ -a_i y_i \leq & x_i^{l,p} & \leq M y_i & i = 1 \ldots \overline{n} \\ & x_p^{l,p} & = -L_p & \\ & & & \\ & Ax^{u,p} & = 0 & \\ -a_i y_i \leq & x_i^{u,p} & \leq M y_i & i = 1 \ldots \overline{n} \\ & x_p^{u,p} & = U_p & \end{array}\right\} \quad \text{for each } p \in \mathcal{P} \\ & \\ & y_i \in \{0, 1\} \end{array} \tag{3}$$

Inequality constraints impose the bounds of $x_i^{l,p}$ and $x_i^{u,p}$ when $y_i = 1$ (M being a large value), and prevent deviations in nonsuppressed cells (i.e., $y_i = 0$). Clearly, the constraints of (3) guarantee that the solutions of the linear programs (2) will satisfy (1). A similar formulation was used in [10].

The first heuristic of Section 3 was inspired by the above formulation, since it attempts to find a "good" (i.e., close to the optimum) feasible point for (3) considering the combinatorial nature of the objective function. However, most network flows heuristics consider a linear objective function. They can thus be

seen as derived from a linear relaxation of (3). This linear relaxation can be obtained by replacing the binary variables y_i in (3) by two variables z_i^l and z_i^u, $i = 1 \dots \overline{n}$, that represent the minimum deviation required in a cell to guarantee, respectively, the lower and upper protection levels of the primary cells. This linear relaxation model can be stated as

$$
\begin{array}{ll}
\min & \displaystyle\sum_{i=1}^{\overline{n}} a_i (z_i^l + z_i^u) \\
\text{s.t.} & \\
& \left.\begin{array}{rrll}
& Ax^{l,p} & = 0 & \\
-z_i^l \leq & x_i^{l,p} & \leq z_i^u & i = 1 \dots \overline{n} \\
& x_p^{l,p} & = -L_p & \\
& & & \\
& Ax^{u,p} & = 0 & \\
-z_i^l \leq & x_i^{u,p} & \leq z_i^u & i = 1 \dots \overline{n} \\
& x_p^{u,p} & = U_p &
\end{array}\right\} \quad \text{for each } p \in \mathcal{P} \\
& \\
& z^l, z^u \geq 0
\end{array}
\tag{4}
$$

The term "partial cell suppression problem" was coined in [11] for (4).

As noted in [10], (3) and (4) give rise to very large MILP and LP problems even for tables of moderate sizes and number of primary cells. However, their constraints matrices are highly structured. For instance, Figure 1 shows the structure of the constraints matrix for problem (4). This dual-block structure is similar to that of stochastic programming problems [2]. Such structure can be exploited through decomposition schemes, as done in [10] using Bender's method. As noted in [2], alternative decomposition approaches based on interior-point methods [16] could be attempted, although, in principle, and from the computational experience reported in [10], they don't look like a promising choice for CSP.

The second level of structure in (3) and (4) comes from the table linear relations matrix A. This structure has only been exploited up to now for two-dimensional tables, which can be modeled as a bipartite network. The two heuristics described in the next section belong to this category. In Section 5 it will be shown how more complicated structures can also be exploited through extended network flows formulations.

3 Network Flows Heuristics

Network flows heuristics attempt to find approximate solutions to (3) or (4). In fact, they can only guarantee a (hopefully good) feasible solution, i.e., a pattern of complementary suppressions $\mathcal{C}$ satisfying the lower and upper protection levels. Before outlining the two approaches considered in this work, we present the framework for CSP network flows heuristics. In this section we will mainly focus on two-dimensional tables.

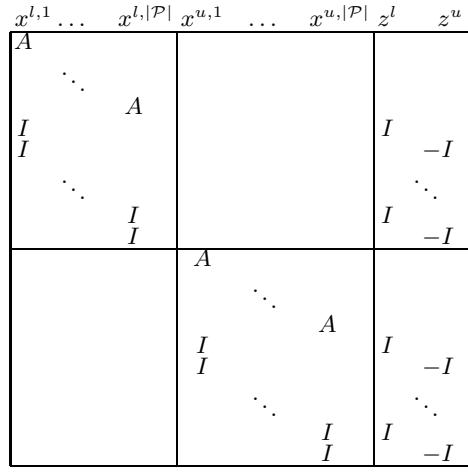

Fig. 1. Constraints matrix structure for problem (4)

3.1 General Framework

Heuristics for two-dimensional CSPs exploit that the linear relations of a $(m+1) \times (n+1)$ table defined by system $Ax = b$ can be modeled as the network of Figure 2. Arcs are associated to cells and nodes to equations; row $m+1$ and column $n+1$ correspond to marginals. For each cell two variables x_i^+ and x_i^- are defined, denoting respectively a positive or negative deviation from the cell value a_i. Clearly, the feasible deviations must satisfy

$$A(x^+ + x^-) = 0, \tag{5}$$

which can be modeled as a nonoriented version of the network of Figure 2.

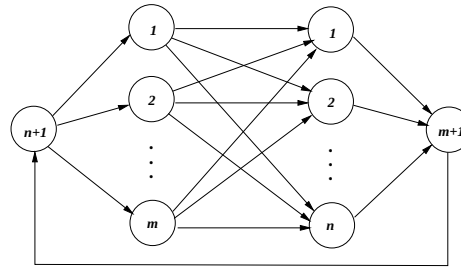

Fig. 2. Network representation of a $(m+1) \times (n+1)$ table

CSP heuristics perform one iteration for each primary cell $p \in \mathcal{P}$, as shown in Figure 3. At each iteration the protection levels of p are satisfied through the

solution of one or two successions of network flows problems. Some heuristics (e.g., that of Section 3.3) only need to solve a single network flows problem for each succession, while others (e.g., that of Section 3.2) can require a few ones to guarantee the protection levels. This is stated at line 5 of Figure 3. Two (successions of) network flows problems are solved when we allow different patterns of complementary suppressions for the upper and lower protection levels of p. This is in principle the best option, since it is closer to the original formulations of CSP (3) and (4) (where different vectors $x^{l,p}$ and $x^{l,p}$ were also considered). On the other hand, one (succession of) network flows problem(s) is solved when we look for a pattern of suppressions that satisfies both the upper and lower protection levels. This was the choice in [18]. Although the maximum number of network flows problems to be solved halves, this heuristic can not be considered a good approximation to (3) and (4), and therefore, in theory, it should not be able to provide good solutions. As in [19] and the heuristic method of [10], we considered the two-networks-flows (successions) approach, as shown at line 3 of Figure 3. The structure of the network flows problems solved at line 6 of the algorithm (i.e., injections, costs, and lower and upper capacities) depends on the particular heuristic being used.

```
Algorithm CSP Heuristic Framework(Table,P, U, L)
1   C = ∅; CLP_i = 0, CUP_i = 0, i ∈ P;
2   for_each p ∈ P do
3       for_each type of protection level X ∈ {U, L} do
4           if CXP_p < X_p (X = L or X = U) then
5               repeat /* some heuristics only require one repeat iteration */
6                   Solve network problem with flows x+ and x−;
7                   Obtain set T = {i : x_i^+ + x_i^− > 0} of positive flows;
8                   C := C ∪ T \ P;
9                   Update current protection levels CXP (X = L or X = U);
10              until protection level X_p is achieved (X = L or X = U);
11          end_if
12      end_for_each
13  end_for_each
End_algorithm
```

Fig. 3. General framework of heuristic procedures

After the solution of the network flows problem, additional suppressions are obtained, which are added to the current set $\mathcal{C}$ of complementary cells (lines 7 and 8 of Figure 3). The new suppressions can also satisfy the protection levels of the following primary cells. To avoid the solution of unnecessary network flows problems, we maintain two vectors CLP_i and CUP_i, $i \in \mathcal{P}$, with respectively the current lower and upper protection achieved for all primary cell. Primary

cell p is thus not treated if its protection levels are satisfied (line 4). This is a significant improvement of the methods described in [7,19]. The particular updating of CLP and CUP at line 9 of the algorithm is heuristic dependent. It is noteworthy that the algorithm of Figure 3 is also valid for three-dimensional, linked, and hierarchical tables. We only need to substitute the optimization problem at line 6 of the algorithm.

3.2 First Heuristic

The first heuristic is derived from that presented in [7]. This heuristic uses a variant of the objective function of (3), i.e., a suppressed cell has a fixed cost. To this end, only 0-1 values are allowed for the variables (arcs) x^+ and x^- that flow through the nonoriented network of Figure 2. Due to the unimodularity of the bases of network simplex methods [1], this is guaranteed if we impose bounds $0 \leq x^+ \leq 1$ and $0 \leq x^- \leq 1$. The objective function of the network problem $\sum_{i=1}^{\overline{n}} c_i(x_i^+ + x_i^-)$ (c_i being discussed later) is then a good approximation of that of (3).

We force a flow of 1 through arc x_p^+, p being the current primary cell selected at line 2 of Figure 3. This can be done either imposing a lower bound of 1 or using a high negative cost for x_p^+. The upper bound of x_p^- is set to 0 to avoid a trivial cycle. After solving the network flows problem, a cycle $\mathcal{T}$ of cells with 1-flows will be obtained and added to the set of complementary suppressions according to lines 7–8 of Figure 3. If the value $\gamma = \min\{a_i : i \in \mathcal{T}\}$ is greater than X_p ($X = L$ or $X = U$, following the notation of Figure 3), then the (lower or upper) protection level of p is guaranteed. If $\gamma < X_p$, we have only partially protected cell p; we then set $X_p := X_p - \gamma$ and repeat the procedure, finding additional cycles until the protection level is satisfied. This iterative procedure corresponds to lines 5–10 of Figure 3. In this heuristic only a vector of current protection level is maintained (i.e., $CLP_i = CUP_i$ in Figure 3), since the γ value can be used for both the upper and lower protection of the cells in the cycle.

The behavior of the heuristic is governed by the costs c_i of variables x_i^+ and x_i^- associated to cells a_i. Costs are chosen to force the selection of, first, cells $\in \mathcal{P} \cup \mathcal{C}$ and $a_i \geq X_p$, second, cells $\notin \mathcal{P} \cup \mathcal{C}$ and $a_i \geq X_p$, third, cells $\in \mathcal{P} \cup \mathcal{C}$ and $a_i < X_p$, and, finally, cells $\notin \mathcal{P} \cup \mathcal{C}$ and $a_i < X_p$ ($X = L$ or $X = U$), in an attempt to balance the number of new complementary suppressions and network flows problems to be solved. Clearly, for each of the above four categories, cells with the lowest a_i values are preferred.

The above network problem can also be formulated as a shortest-path between the row and column nodes of primary cell p in the nonoriented network of Figure 2, which relates this heuristic with the method described in [4] for general tables.

3.3 Second Heuristic

The second approach is based on [19] and is similar to the heuristics used by the U.S. Census Bureau [18] and Statistics Canada [21], and the heuristic of

[10]. Unlike the original method, that solved a single network flows problem [19], two separate problems are considered for the lower and upper protection of each primary cell (line 4 of Figure 3). For both problems, we set bounds $x_i^+ \geq 0$ and $a_i \geq x_i^- \geq 0$. For lower protection we force a flow L_p through arc x_p^-, while a flow U_p is sent through arc x_p^+ in the upper protection problem. Unlike the heuristic of Section 3.2, only one network flows problem needs to be solved for each protection level (lines 5 and 10 of Figure 3 are no longer required). The objective function $\sum_{i=1}^{\overline{n}} c_i(x_i^+ + x_i^-)$, $c_i = 0$ if $i \in \mathcal{P} \cup \mathcal{C}$ and $c_i = a_i$ otherwise, is minimized subject to (5) for each protection level. This objective is related with that of (4).

As in the first heuristic, after the solution of each network flows problem we obtain the cycle $\mathcal{T}$ and update the complementary suppressions (lines 7 and 8 of Figure 3). Defining $\gamma = \min\{a_i : i \in \mathcal{T}\}$, the current protection vectors are updated at line 9 of Figure 3 as $CLP_i := \max\{CLP_i, \gamma, x_i^- - x_i^+\}$ after the solution of the lower protection problem, and $CUP_i := \max\{CUP_i, \gamma, x_i^+ - x_i^-\}$ after solving the upper protection problem, for all $i \in \mathcal{T}$.

The lower-bounding procedure described in [19] was also applied to obtain an initial set of complementary suppressions. The clean-up post-process suggested in [19] to remove unnecessary complementary suppressions was not performed since it is computationally very inefficient.

4 Computational Comparison

The heuristics of Sections 3.2 and 3.3 have been implemented with the AMPL modeling language [12], which allows the quick development of algorithm prototypes. The network flows problems were solved with the Cplex 6.5 network simplex code [17]. Two generators for two-dimensional positive tables were developed. The first generator follows the description of [19]. Cell values are randomly obtained from an integer uniform distribution [1,1000] with probability 0.8 and are 0 with probability 0.2. The second one is similar to generator 1 of [9]. Cell values are randomly obtained from integer uniform distributions [1,4] for primary cells and $\{0\} \cup [5, 500]$ for the remaining entries. Primary cells are randomly chosen from the internal cells in both generators.

We produced 48 instances with each generator. Each instance is defined by three parameters (m, n, p), which denote the number of rows, columns and primary suppressions, respectively. The 48 instances were obtained considering all the combinations for $m \in \{50, 100, 150\}$ and $n, p \in \{50, 100, 150, 200\}$. In all the cases the lower and upper protection levels were a 15% of the cell value. This symmetric range protection slightly benefits the heuristic of Section 3.2, because one single network flows problem is enough for both the upper and lower protection requirements.

The results obtained are shown in Figures 4–11. The heuristics of Sections 3.2 and 3.3 are denoted as "first" and "second" heuristic, respectively. Executions were carried out on a Sun Ultra2 200MHz workstation (approximately equivalent in performance to a 350 Mhz Pentium PC). The horizontal axes of all the figures

refer to the instance number, $(50, 50, 50)$ being the first, $(50, 50, 100)$ the second and so on. The several groups of four points with positive slope correspond to the same table size for different numbers of primary suppressions.

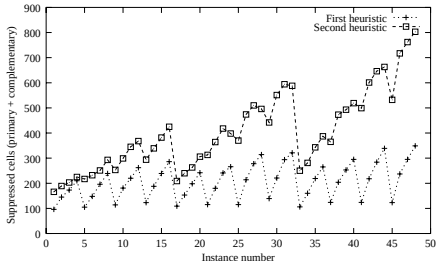

Fig. 4. Number of suppressed cells for instances of generator 1

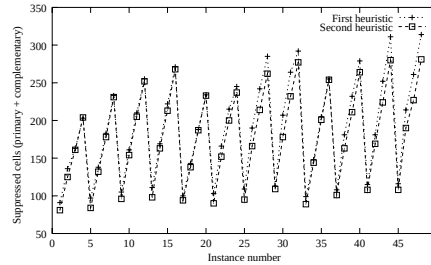

Fig. 5. Number of suppressed cells for instances of generator 2

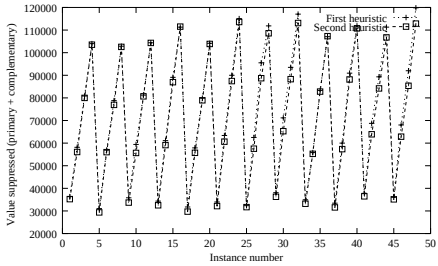

Fig. 6. Total value suppressed for instances of generator 1

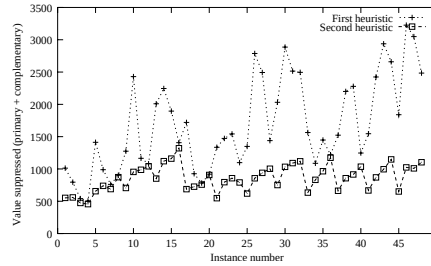

Fig. 7. Total value suppressed for instances of generator 2

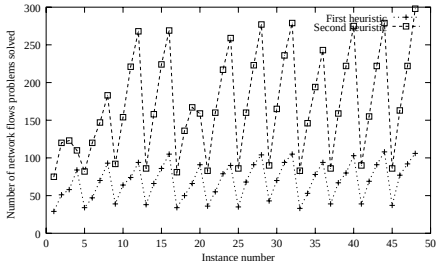

Fig. 8. Network flows problems solved for instances of generator 1

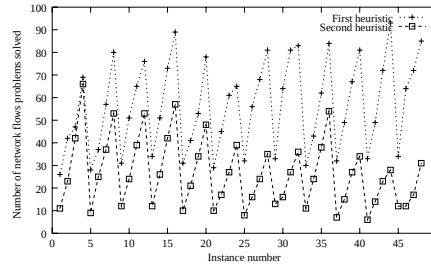

Fig. 9. Network flows problems solved for instances of generator 2

Clearly, the behavior of the heuristics depends on the set of instances. The first heuristic is more efficient for instances obtained with generator 1, since it suppresses less cells, solves less network flows problems, and is faster than the

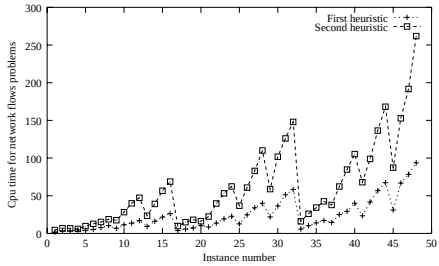

Fig. 10. CPU time of network flows problems for instances of generator 1

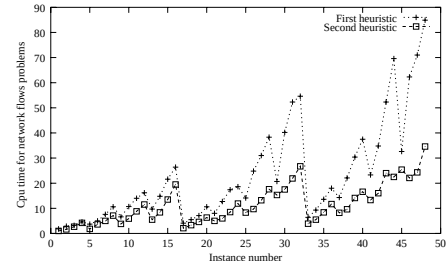

Fig. 11. CPU time of network flows problems for instances of generator 2

second heuristic (Figures 4, 8 and 10 respectively). However, the total value suppressed is similar, as shown in Figure 6. On the other hand, the second heuristic provides slightly better results for the second set of instances, mainly for the total value suppressed, number of network flows problems solved, and execution time (Figures 7, 9 and 11 respectively). Therefore, the choice of a heuristic should consider the particular structure of the tables to be protected.

It must be noted that, in theory, execution times could be improved for the first heuristic if we used some specialized shortest-path algorithm, instead of a minimum cost network solver. Unfortunately, this heuristic can not be easily extended to three-dimensional, linked and hierarchical tables. As shown in next Section, these tables can be modeled through multicommodity flows and flows with side constraints, which do not preserve the integrality property of minimum cost network flows models.

5 Extensions of Network Flows Models

In previous works on CSP, the structure of the table linear relations matrix A was only exploited for two-dimensional tables, which, as shown in Section 3, were modeled as a bipartite network. For more complicated structures, A was considered as the constraints matrix of a general LP problem [10,18]. However extensions of network models (i.e., multicommodity flows, and flows with side constraints) can still be applied for three-dimensional, linked and hierarchical tables.

Multicommodity flows are a generalization of network flows problems. In these models l commodities have to be routed through the same underlying network. The set of feasible multicommodity flows is

$$\mathcal{F}_1 = \{(x^1, \ldots, x^l) : Nx^k = b^k, l^k \leq x^k \leq u^k, k = 1 \ldots l, \sum_{k=1}^{l} x^k \leq u\}, \quad (6)$$

where x^k is the flows vector for each commodity $k = 1 \ldots l$, N is the node-arc incidence network matrix, b^k are the node injections at the network for

each commodity, u^k and l^k are respectively the individual lower and upper arc capacities for each commodity, and u is the mutual arc capacity for all the commodities. This kind of models have been extensively applied in distribution, routing, logistic and telecommunications problems [1].

In networks with side constraints the flows of some arcs must satisfy additional linear relations. Mutual capacity constraints of multicommodity problems are a particular case of side constraints. The set of feasible flows is thus defined as

$$\mathcal{F}_2 = \{x : Nx = b, \underline{b} \leq Tx \leq \overline{b}, l \leq x \leq u\}, \tag{7}$$

x being the flows vector, N the node-arc incidence network matrix, b the injections at the nodes of the network, u and l the lower and upper capacities of the arcs, and T the side constraints matrix.

Although the minimization of a linear cost function subject to constraints (6) or (7) can be solved with a general algorithm for LP, several specialized methods that exploit the partial network structure have been developed. Among them we find simplex-based [6], Lagrangian relaxation [13], interior-point [5] and approximation methods [15]. Since multicommodity flows and flows with side constraints can be used to model three-dimensional, linked and hierarchical tables, as shown in next subsections, these algorithms can also be applied to CSP. Moreover, unlike most efficient LP solvers, some of these specialized algorithms are freely available for noncommercial purposes (e.g., [5,6]).

5.1 Three-Dimensional Tables

The linear relations $Aa = b$ of the cell values a_i of a $(m+1) \times (n+1) \times (l+1)$ three-dimensional table can be stated as

$$\sum_{i=1}^{m} a_{ijk} = a_{(m+1)jk} \quad j = 1 \ldots n, \quad k = 1 \ldots l \tag{8}$$

$$\sum_{j=1}^{n} a_{ijk} = a_{i(n+1)k} \quad i = 1 \ldots m, \quad k = 1 \ldots l \tag{9}$$

$$\sum_{k=1}^{l} a_{ijk} = a_{ij(l+1)} \quad i = 1 \ldots m, \quad j = 1 \ldots n. \tag{10}$$

Cells $a_{(m+1)jk}$, $a_{i(n+1)k}$ and $a_{ij(l+1)}$ form, respectively, the row-marginal $n \times l$ table, the column-marginal $m \times l$ table, and the level-marginal $m \times n$ table.

Clearly, putting together in (8) and (9) equations related to the same level k, (8)–(10) can be written as

$$Na^k = b^k, \quad k = 1 \ldots l \tag{11}$$

$$\sum_{k=1}^{l} a^k = a^{l+1}, \tag{12}$$

N being the network linear relations of the two-dimensional table associated to each level (depicted in Figure 2), a^k the $m \times n$ cells (flows) of level k, b^k the row and column marginals of level k, and a^{l+1} the level marginal values. From (6), it is clear that (11) and (12) define a set of feasible multicommodity flows, by choosing appropriate upper and lower bounds u^k and l^k (e.g., $u^k = \infty$ and $l_k = 0$), and for the particular case of having equality mutual capacity constraints. Therefore, the heuristic of Figure 3 can be used for three-dimensional tables replacing line 6 by the solution of a multicommodity network flows problem.

5.2 Linked and Hierarchical Tables

Linked tables can be defined as tables that share some common cells or, more generally, whose entries are linked by some linear relation. Hierarchical tables (tables with subtotals) can be considered a particular case of linked tables, in which the common cells correspond to subtotal entries. Standard network flows models are only useful for hierarchical tables in one dimension, as shown in [18]. We focus on the general linked tables model.

Given a linked table made of t two or three-dimensional tables, and a set of four-dimensional elements with the information of the common cells,

$$\mathcal{E} = \{(u, r, v, s) : \text{cell } u \text{ in table } r \text{ must be equal to cell } v \text{ in table } s\},$$

the overall table relations can be written as

$$A^i a^i = b^i, \quad i = 1 \dots t \tag{13}$$

$$a^r_u - a^s_v = 0, \quad (u, r, v, s) \in \mathcal{E}. \tag{14}$$

$A^i a^i = b^i$ denote the network or multicommodity network equations, depending of the dimension of table i, while (14) impose the same value for the common cells. Clearly, (14) and the mutual capacity equations (12) of all the three-dimensional tables form a set of linear side constraints, that, together with the remaining network equations, match the model defined in (7). Therefore, linked (and hierarchical) tables can be modeled as a network with side constraints.

6 Preliminary Computational Results

We implemented an extension for three-dimensional tables of the heuristic described in Section 3.3, including a generalization of the lower-bounding procedure introduced in [19]. It was coded in AMPL [12]. Since no specialized multicommodity solver is currently linked to AMPL, we used the network+dual option of the Cplex 6.5 package [17]. This option finds first a feasible point for the network constraints using a specialized network primal simplex algorithm; this point is then used as a warm start for the dual simplex. The two generators of Section 4 were extended for three-dimensional tables. Table 1 reports the dimensions and results of the instances. Column "gen" shows the generator used. Column "$m \times n \times l$" gives the size of the table. Column "p" is the number of primary

suppressions. Columns "c.s." and "v.s." show respectively the total number of suppressed cells and of value suppressed (including primary suppressions). Column "nnf" is the number of multicommodity network flows problems solved, while column "CPU_nf" gives the overall CPU time spent in their solution. For all the instances the upper and lower protection levels were a 15% of the cell value. The execution environment was the same that for Section 4. The figures of the results columns (last four) are significantly greater than those obtained for two-dimensional tables with a similar number of cells and primary suppressions. This weakness of the heuristic is due to the complex cell interrelations of three-dimensional tables.

Table 1. Dimensions and results for 3D tables (using network+dual solver)

gen.	$m \times n \times l$	p	c.s.	v.s.	nnf	CPU_nf
1	$10 \times 10 \times 10$	50	254	67082	76	18.7
1	$10 \times 10 \times 10$	100	234	75317	110	31.3
1	$10 \times 10 \times 20$	50	317	77439	81	31.3
1	$10 \times 10 \times 20$	100	362	94765	127	82.9
1	$10 \times 20 \times 10$	50	298	79644	69	29.2
1	$10 \times 20 \times 10$	100	397	97964	115	102.6
1	$10 \times 20 \times 20$	50	473	97753	86	127.1
1	$10 \times 20 \times 20$	100	526	118745	144	224.2
2	$10 \times 10 \times 10$	50	170	10099	43	7.8
2	$10 \times 10 \times 10$	100	191	6194	38	9.1
2	$10 \times 10 \times 20$	50	222	14458	65	16.6
2	$10 \times 10 \times 20$	100	275	12027	43	24.6
2	$10 \times 20 \times 10$	50	222	14192	63	21.7
2	$10 \times 20 \times 10$	100	310	16035	89	40.0
2	$10 \times 20 \times 20$	50	296	19252	75	54.5
2	$10 \times 20 \times 20$	100	398	19841	103	86.9

We also solved the set of instances using the dual simplex option of Cplex 6.5, which does not exploit the network structure of the problem. The results are shown in Table 2. Clearly, the execution times drastically reduced in most instances. This is a surprising result, specially because the network+dual Cplex option is a highly regarded algorithm for multicommodity flows and flows with side constraints. A possible explanation is that the problem solved by the network primal simplex method is very degenerate (i.e., many basic variables at bounds) which means a large number of unproductive iterations. Additional experiments with larger instances and alternative network flow solvers have to be performed before concluding that the dual simplex method is the most efficient approach for three-dimensional tables.

Table 2. Dimensions and results for 3D tables (using dual solver)

gen.	$m \times n \times l$	p	c.s.	v.s.	nnf	CPU_nf
1	$10 \times 10 \times 10$	50	227	60961	56	7.6
1	$10 \times 10 \times 10$	100	259	78641	69	10.1
1	$10 \times 10 \times 20$	50	307	74239	74	19.0
1	$10 \times 10 \times 20$	100	384	97304	79	26.8
1	$10 \times 20 \times 10$	50	292	74327	60	15.5
1	$10 \times 20 \times 10$	100	446	103971	82	33.4
1	$10 \times 20 \times 20$	50	480	95978	73	64.1
1	$10 \times 20 \times 20$	100	608	124817	104	96.4
2	$10 \times 10 \times 10$	50	170	10099	38	4.6
2	$10 \times 10 \times 10$	100	190	6115	31	4.0
2	$10 \times 10 \times 20$	50	222	14458	65	15.5
2	$10 \times 10 \times 20$	100	261	10889	24	7.5
2	$10 \times 20 \times 10$	50	222	14192	63	14.7
2	$10 \times 20 \times 10$	100	306	15656	81	19.7
2	$10 \times 20 \times 20$	50	296	19252	75	50.5
2	$10 \times 20 \times 20$	100	390	19558	93	50.4

7 Conclusions

From our computational experience, it can be concluded that the efficiency of the two heuristics evaluated for two-dimensional tables depends on the particular structure of the instances to be solved. We also reported some preliminary results with a network flows heuristic for three-dimensional tables. Among the future tasks to be done we find a comprehensive evaluation of multicommodity models for three-dimensional tables, including larger instances and alternative specialized solvers, and the computational study of a heuristic for linked and hierarchical tables based on network flows with side constraints.

References

1. Ahuja, R.K, Magnanti, T.L., Orlin, J.B.: Network Flows. Prentice Hall (1993)
2. Birge, J.R., Louveaux, F.: Introduction to Stochastic Programming. Springer (1997)
3. Bixby, R.E., Fenelon, M., Gu, Z., Rothberg, E., Wunderling, R.: MIP: Theory and practice—Closing the gap. In System Modelling and Optimization. Methods, Theory and Applications, eds. M.J.D. Powell and S. Scholtes. Kluwer (2000) 19–49
4. Carvalho, F.D., Dellaert, N.P., Osório, M.D.: Statistical disclosure in two-dimensional tables: general tables. J. Am. Stat. Assoc. **89** (1994) 1547–1557
5. Castro, J.: A specialized interior-point algorithm for multicommodity network flows. SIAM J. on Opt. **10** (2000) 852–877
6. Castro, J., Nabona, N. An implementation of linear and nonlinear multicommodity network flows. European Journal of Operational Research **92** (1996) 37–53
7. Cox, L.H.: Network models for complementary cell suppression. J. Am. Stat. Assoc. **90** (1995) 1453–1462

8. Dellaert, N.P., Luijten, W.A.: Statistical disclosure in general three-dimensional tables. Statistica Neerlandica **53** (1999) 197–221
9. Fischetti, M., Salazar, J.J.: Models and algorithms for the 2-dimensional cell suppression problem in statistical disclosure control. Math. Prog. **84** (1999) 283–312
10. Fischetti, M., Salazar, J.J.: Models and algorithms for optimizing cell suppression in tabular data with linear constraints. J. Am. Stat. Assoc. **95** (2000) 916–928
11. Fischetti, M., Salazar, J.J.: Partial cell suppression: a new methodology for statistical disclosure control. Working paper, Department of Statistics, Operations Research and Computing, University of La Laguna (1998)
12. Fourer, R, Gay, D.M., Kernighan, B.W.: AMPL: A Modeling Language for Mathematical Programming. Duxbury Press (1993)
13. Frangioni, A., Gallo, G: A bundle type dual-ascent approach to linear multicommodity min cost flow problems. INFORMS J. on Comp. **11** (1999) 370–393
14. Giessing, S.: New tools for cell-suppression in τ-Argus: one piece of the CASC project work draft. Joint ECE/Eurostat Work Session on Statistical Data Confidentiality, Skopje (2001).
15. Goldberg, A.V., Oldham, J.D., Plotkin, S., Stein, C.: An implementation of a combinatorial approximation algorithm for minimum-cost multicommodity flow. In Lecture Notes in Computer Sciences. Proceedings of the 6th International Integer Programming and Combinatorial Optimization Conference, eds. R.E. Bixby, E.A. Boyd and R.Z. Ríos-Mercado. Springer (1998)
16. Gondzio, J, Sarkissian, R.: Parallel interior-point solver for structured linear programs. Technical Report MS-00-025, Department of Mathematics and Statistics, The University of Edinburgh (2000)
17. ILOG CPLEX: ILOG CPLEX 6.5 Reference Manual Library. ILOG (1999)
18. Jewett, R.: Disclosure analysis for the 1992 Economic Census. Manuscript, Economic Programming Division, Bureau of the Census (1993)
19. Kelly, J.P., Golden, B.L, Assad, A.A.: Cell Suppression: disclosure protection for sensitive tabular data. Networks **22** (1992) 28–55
20. Massell, P.B.: Cell suppression and audit programs used for economic magnitude data. SRD Research Report Series RR2001/01 (2001).
21. Robertson, D.: Improving Statistic's Canada cell suppression software (CONFID). Proceedings of Compstat 2000.