Models and Algorithms for the 2-Dimensional Cell Suppression Problem in Statistical Disclosure Control

Matteo Fischetti
DEI - University of Padova - Italy

Juan José Salazar
DEIOC - University of La Laguna - Spain

Abstract

We study the problem of protecting sensitive data in a statistical two-dimensional table, when the non-sensitive table entries are made public along with the row and column totals. In particular, we address the $\mathcal{NP}$-hard problem known in the literature as the (secondary) cell suppression problem. We introduce a new integer linear programming model and describe several new families of additional inequalities used to strengthen the linear relaxation of the model. Exact and heuristic separation procedures are also proposed and embedded within a branch-and-cut algorithm for the exact solution of the problem. The algorithm makes use of an efficient heuristic procedure to find near-optimal solutions. We report the exact solution of instances involving up to 250,000 cells and 10,000 sensitive cells, i.e., more than 3 orders of magnitude larger than those solved by previous techniques from the literature.

Keywords: Statistical Disclosure Control, Cell Suppression, Branch-and-Cut.

1 Introduction

A statistical agency collects data to be processed and published. Raw material is information obtained from individual respondents. Usually, this data is obtained under a pledge of confidentiality: statistical agencies have the responsibility of not releasing any data or data summaries from which individual respondent information can be revealed (*sensitive data*). On the other hand, statistical agencies aim at publishing as much information as possible. This results in a trade-off between privacy rights and information loss, an issue of primary importance in practice. We refer the interested reader to Willenborg and Waal [18] for an in-depth analysis of statistical disclosure control methodologies.

Age	A	B	C	Total
under 45	7	10	8	25
45–64	12	9	5	26
65–74	7	3	3	13
over 74	5	0	**2**	7
Total	31	22	18	71

(a) Original table

Age	A	B	C	Total
under 45	*	10	*	25
45–64	12	9	5	26
65–74	7	3	3	13
over 74	*	0	*	7
Total	31	22	18	71

(b) Published table

Figure 1: Millionaires by age and place of residence.

Starting in 1996, the European Union supported through EUROSTAT (the European statistical office) a 3-year ESPRIT research project aimed at developing and testing new methodologies within statistical disclosure control. The project, coordinated by Dr. Leon Willenborg from CBS (Central Bureau of Statistics, Netherlands), involves several research groups from both academia and national statistical offices. We participate in the project for the definition of mathematical models and solution algorithms for protecting sensitive information in tabular data.

Cell suppression is one of the most-widely applied techniques for disclosure avoidance. In this work we study the problem of applying this technique to protect sensitive information in a two-dimensional table of statistics, in which the non-sensitive data is published along with the row and column totals. We will introduce the problem with the help of a simple example, taken from Geurts [9]. Figure 1(a) exhibits a table giving the number of millionaires, classified by age and place of residence. Assuming that in a small village like C all inhabitants know that only two villagers are older than 74, each inhabitant can conclude that these two people are indeed millionaires. To protect the table against statistical disclosure, cell (4,3) is considered a *sensitive cell* by the statistical office which is going to publish the table, and its value 2 is substituted by '*' for 'missing' (*primary suppression*). But that is not enough: by using the marginal totals, an *attacker* interested in the disclosure of the sensitive cell can easily recompute its missing value. Then other table entries must be also suppressed, e.g., cells (1,1), (1,3), and (4,1); see Figure 1(b). With this choice, the attacker cannot disclose the nominal value of cell (4,3) exactly, although he can still compute a range for the values of this cell which are consistent with the published entries. Indeed, the minimum value LB_{43} for cell (4,3) can be computed by solving a linear program in which the values f_{ij} for the missing cells (i, j) are treated as unknowns, namely

$$LB_{43} := \min f_{43}$$

subject to

$$\begin{array}{rrrrl} f_{11} & & +f_{13} & & = 15 \\ & f_{41} & & +f_{43} & = 7 \\ f_{11} & +f_{41} & & & = 12 \\ & & f_{13} & +f_{43} & = 10 \\ f_{11}, & f_{41}, & f_{13}, & f_{43} & \geq 0. \end{array}$$

Notice that the right-hand-side values are known to the attacker, as they can be obtained as the difference between the marginal and the published values in a row/column.

Analogously, the maximum value UB_{43} for cell (4,3) can be computed by solving the linear program $UB_{43} := \max f_{43}$ subject to the same constraints as before. In the example, $LB_{43} = 0$ and $UB_{43} = 7$, i.e., the sensitive information is "protected" within the *protection interval* [0,7]. If this interval is considered sufficiently wide by the statistical office, then the sensitive cell $(4, 3)$ is called *protected*, otherwise new suppressions are needed.

A more detailed description of the problem is as follows. Given a set of sensitive cells PS (*primary suppressions*) along with the required protection levels, the statistical office aims at finding a set of secondary suppressions protecting all the sensitive cells against the attacker, and such that the loss of information associated with the suppressed entries is minimized. More specifically, let $[a_{ij} : i \in \{0, \ldots, NR\}, j \in \{0, \ldots, NC\}]$ be the given 2-dimensional table to be protected, having $NR + 1$ rows and $NC + 1$ columns. Zero indices correspond to marginal totals, which satisfy:

$$a_{i0} = \sum_{j=1}^{NC} a_{ij}, \qquad i = 1, \ldots, NR, \tag{1}$$

$$a_{0j} = \sum_{i=1}^{NR} a_{ij}, \qquad j = 1, \ldots, NC, \tag{2}$$

$$a_{00} = \sum_{i=1}^{NR} a_{i0} \qquad (= \sum_{j=1}^{NC} a_{0j}). \tag{3}$$

Each entry (i, j) of the table, including marginals, has an associated *weight* $w_{ij} \geq 0$ giving a measure of the loss of information incurred if the entry is suppressed. Moreover, let $(i_1, j_1), \ldots, (i_p, j_p)$ be the $p := |PS|$ sensitive (possibly marginal) cells in PS. Each sensitive cell (i_k, j_k) is considered *protected* when the associated protection interval $[LB_{i_k j_k}, UB_{i_k j_k}]$ computed by a possible attacker contains the closed segment $[a_{i_k j_k} - LPL_k, a_{i_k j_k} + UPL_k]$, where the non-negative *lower* and *upper protection levels* LPL_k and UPL_k are provided by the statistical office. Without loss of generality, for any k we assume that LPL_k and UPL_k are not both equal to 0. The problem then calls for a minimum-weight set of cells whose suppression protects all sensitive cells. This combinatorial optimization problem is known as the (secondary) *Cell Suppression Problem* (CSP), and belongs to the class of the strongly $\mathcal{NP}$-hard problems; see, e.g., Kelly, Golden and Assad [14], Geurts [9], Kao [12].

Previous works on CSP mainly concentrate on heuristics; see, e.g., Cox [3, 4], Kelly, Golden and Assad [14], and Carvalho, Dellaert and Osório [6]. Kelly [13] proposed a mixed-integer linear programming formulation involving a huge number of variables and constraints (for instance, the formulation involves more than 20,000,000 variables and 30,000,000 constraints for a 100x100 table with 5% sensitive entries). Geurts [9] refined this model, and reports computational experiences on small-size instances, the largest instance solved to optimality being a 20x6 table with 17 sensitive cells (the computing time is not reported; for smaller instances, the code required several thousand CPU seconds on a SUN Sparc 1+ workstation). Gusfield [11] gave a polynomial algorithm for a special case of the problem.

In this paper we propose a new integer linear programming model and a branch-and-cut algorithm for its exact solution. In Section 2 we introduce the new model, in which the protection level for each sensitive cell is ensured if a certain flow circulation is achieved in an appropriated network, a condition imposed by introducing capacity constraints. Although the model contains an exponential number of capacity constraints, its linear programming relaxation can still be solved efficiently by using a run-time constraint generation technique. Additional constraints are proposed in Section 3, which enhance the linear relaxation of the model considerably. For each class of constraints, efficient separation procedures are described in Section 4 which detect, at run time, the constraints to be generated and appended to the current problem formulation. In that section we also outline the overall branch-and-cut algorithm, give reduction criteria to reduce the size of the instance to be solved, and discuss a heuristic procedure to find near-optimal CSP solutions. Computational results on several classes of both real-world and random test instances are given in Section 5, showing that the new approach outperforms previous methods proposed in literature. We report the exact solution, on a personal computer, of instances corresponding to tables with up to 250,000 entries and about 10,000 sensitive cells, by far the largest ever tried. Finally, Section 6 draws some conclusions and addresses future directions of work.

2 A new integer linear programming model

The system of linear equations (1)–(3) can be represented in a natural way by a complete bipartite *network* $D = (V, A)$; see, e.g., Cox [3, 4] and Kelly, Golden and Assad [14]. The network has a node r_i for each row $i = 0, \ldots, NR$, and a node c_j for each column $j = 0, \ldots, NC$. We have an arc for each entry of the matrix, namely:

- an arc (r_i, c_j) associated with each entry a_{ij} with $i \neq 0$ and $j \neq 0$,
- an arc (c_0, r_i) for each marginal entry a_{i0} with $i \neq 0$,
- an arc (c_j, r_0) for each marginal entry a_{0j} with $j \neq 0$,

- the arc (r_0, c_0) corresponding to the grand total a_{00}.

To simplify notation, we do not distinguish between a cell (i, j) and the associated arc in A. In particular, we often denote by i and j, respectively, the starting and ending node of the arc associated with cell (i, j).

On the network representation, the table values a_{ij} can be viewed as *flows* associated with the arcs, equations (1)–(3) stipulating that the total flow entering any node must be equal to the total flow leaving the same node. In this view, table $[a_{ij}]$ corresponds to a *flow circulation* on the network.

For the sake of generality we next assume that the attacker knows a range of feasible values for each table entry a_{ij}, say $[a_{ij} - lb_{ij}, a_{ij} + ub_{ij}]$. (This range is typically equal to $[0, 0]$ for all cells with $a_{ij} = 0$, and to $[0, +\infty]$ for other cells.) In network terms, this implies that each arc $(i, j) \in A$ has given lower and upper bounds, namely $a_{ij} - lb_{ij}$ and $a_{ij} + ub_{ij}$, on the flow that can traverse it.

We say that a set $[f_{ij} : (i, j) \in A]$ defines a *feasible* flow circulation on D whenever (i) $a_{ij} - lb_{ij} \leq f_{ij} \leq a_{ij} + ub_{ij}$ for all $(i, j) \in A$, and (ii) $[f_{ij}]$ satisfies equations (1)–(3).

Clearly the original table $[a_{ij}]$, among others, defines a feasible flow circulation on D, that we call the *nominal flow circulation*.

We next aim at characterizing the feasible tables which are consistent with the nominal one when a given set SUP of its entries is suppressed. In view of the above-described correspondence, this amounts to studying the feasible flow circulations on D that can be obtained by changing, with respect to $[a_{ij}]$, the flow on a *given* subset SUP of arcs. Notice that SUP may contain arcs corresponding to marginal totals.

All the arcs $(i, j) \notin SUP$ correspond to published entries, hence they have to retain their nominal value a_{ij}. The flows on the arcs $(i, j) \in SUP$, instead, are allowed to change with respect to the nominal value, provided that their new values are again within the feasible range associated with lower/upper bounds and satisfy equations (1)–(3). As customary in network-flow theory, flow variations can be expressed in term of an *incremental network*, say $D(SUP) = (V, A(SUP))$, obtained from D as follows:

- remove all the arcs $(i, j) \in A \setminus SUP$;
- replace each arc $(i, j) \in SUP$ by two arcs, namely:
  - a *forward* arc (i, j) with capacity $q_{ij} := ub_{ij}$,
  - a *reverse* arc (j, i) with capacity $q_{ji} := lb_{ij}$.

Each forward arc of $D(SUP)$ corresponds to a flow *increase* of an arc of D, whereas each reverse arc corresponds to a *decrease* of the flow value. More precisely, any flow circulation $[y_{ij}]$ on the incremental network corresponds to the flow circulation $[f_{ij}]$ on the

original network, where $f_{ij} := a_{ij}$ for each $(i,j) \in A \setminus SUP$, and $f_{ij} := a_{ij} + (y_{ij} - y_{ji})$ for each $(i,j) \in SUP$.

In order to compute the protection interval for any given sensitive cell (i_k, j_k) one has to compute the maximum increase Δ_k^+ and decrease Δ_k^- for entry $a_{i_k j_k}$ in a table which is consistent with the published entries. In network terms, Δ_k^+ is the maximum increase that the flow on arc $(i_k, j_k) \in SUP$ can assume in a feasible flow circulation of D, and corresponds to the flow circulation $[y_{ij}]$ on $D(SUP)$ that maximizes $y_{i_k j_k} - y_{j_k i_k}$. Hence Δ_k^+ can be computed by first removing from $D(SUP)$ the two arcs associated with (i_k, j_k), thus obtaining the network $D(SUP \setminus \{(i_k, j_k)\})$, and then by finding the maximum flow from j_k to i_k in $D(SUP \setminus \{(i_k, j_k)\})$. Analogously, the maximum flow decrease Δ_k^- associated with (i_k, j_k) is computed as the value of the max-flow in $D(SUP \setminus \{(i_k, j_k)\})$ from i_k to j_k.

A basic result in network-flow theory is that the maximum flow that one can send from a given source node s to a given destination node t is equal to the capacity of the minimum (s,t)-cut, i.e., of the minimum cut $(S, V \setminus S)$ such that $s \in S$ and $t \notin S$. Hence Δ_k^+ (respectively, Δ_k^-) coincides with the capacity of the minimum (j_k, i_k)-cut (respectively, minimum (i_k, j_k)-cut) in the incremental network $D(SUP \setminus \{(i_k, j_k)\})$. It then follows that SUP is a legitimate set of suppressions if and only if, for each primary suppression (i_k, j_k), the capacity of any (j_k, i_k)-cut in $D(SUP \setminus \{(i_k, j_k)\})$ has a capacity at least equal to the upper protection level UPL_k, whereas any (i_k, j_k)-cut has a capacity at least equal to the lower protection level LPL_k.

The above considerations allow one to formulate CSP as follows. Recall that $D(A)$ is the incremental network associated with $SUP = A$ (all cells are potentially suppressed). For each arc (i,j) of $D(A)$, let $\phi(i,j)$ denote the arc of the original network D that originated it, i.e., $\phi(i,j) := (i,j)$ if (i,j) is a forward arc in the incremental network, and $\phi(i,j) := (j,i)$ otherwise. For all arcs (i,j) of the original network D, let the decision variable x_{ij} assume value 1 if entry (i,j) is suppressed, value 0 otherwise. The model reads:

$$\min \sum_{(i,j) \in A} w_{ij} x_{ij} \tag{4}$$

subject to

$$x_{ij} \in \{0,1\}, \qquad \text{for all } (i,j) \in A, \tag{5}$$

and for each sensitive cell $(i_k, j_k) \in PS$:

$$x_{i_k j_k} = 1, \tag{6}$$

$$\sum_{(i,j) \in \delta^+(S) \setminus \{(j_k, i_k)\}} q_{ij} x_{\phi(i,j)} \geq UPL_k, \qquad \text{for all } S \subset V : j_k \in S, i_k \notin S, \tag{7}$$

$$\sum_{(i,j) \in \delta^+(S) \setminus \{(i_k, j_k)\}} q_{ij} x_{\phi(i,j)} \geq LPL_k, \qquad \text{for all } S \subset V : i_k \in S, j_k \notin S, \tag{8}$$

where $\delta^+(S)$ denotes the cut containing the arcs of $D(A)$ leaving a given $S \subset V$, and arc capacities q_{ij} have been defined previously. Constraints (7) impose that the capacity of any (j_k, i_k)-cut $\delta^+(S)$ is not less than UPL_k, hence they enforce the upper protection level requirements. Constraints (8) play a similar role, and enforce the lower protection level requirements. Clearly, constraint (7) and (8) associated with $UPL_k = 0$ or $LPL_k = 0$, respectively, can be removed from the model (see the reduction procedure given in Section 4.1).

As an illustration, suppose that (1,1) is a sensitive cell with upper protection level L (say), and assume the attacker knows that all table entries are contained in range $[0, M]$, where M is a very large positive value. In order to protect cell (1,1) we then have to suppress the marginal total on column 1, or else additional entries in column 1 whose nominal values add up to, at least, L. This property is indeed expressed in our model by the capacity constraint (7) associated with the set $S = \{c_1\}$, namely $\sum_{i=2}^{NR} a_{i1}x_{i1} + (M - a_{01})x_{01} \geq L$. An analogous inequality can be written for preserving the marginal in row 1, i.e., for $S = \{r_1\}$, as well as for exponentially many other possible choices of S.

Inequalities (7) can be improved upon considerably by replacing, for all given (i_k, j_k), each coefficient q_{ij} by $q'_{ij} := \min\{q_{ij}, UPL_k\}$. The validity of the resulting inequality follows trivially from the fact that the $q_{ij}x_{ij} \geq 0$ and $x_{ij} \in \{0, 1\}$ for all (i, j). Analogously, each member of (8) can be improved by replacing, for all given (i_k, j_k), each coefficient q_{ij} by $q''_{ij} := \min\{q_{ij}, LPL_k\}$. These simple improvements are very effective in practice.

Although the model contains an exponential number of constraints, its linear programming relaxation is polynomially solvable since the separation problem for constraints (7) and (8) —even in their improved form— can be solved efficiently through max-flow computations; see Section 4 for details.

3 Additional inequalities

Additional inequalities can be used for strengthening the Linear Programming (LP) relaxation of the previous model. We next introduce five main classes of additional inequalities, that we call *bridge*, *comb*, *cover*, *2-edge cover*, and *path* inequalities. Computational results show that these inequalities allow for a considerable speed-up in the problem resolution. Bridge and comb inequalities are based on the property of the optimal solutions of being associated with bridgeless subgraphs of the undirected counterpart of the network D defined in Section 2. Cover and 2-edge cover inequalities are based on the knapsack substructure associated with each capacity constraint. Finally, path and modified path inequalities combine the bridgeless and knapsack substructures.

As customary, given any directed/undirected graph $G' = (V', A')$ and a point $x^* \in \Re_+^{A'}$ we denote by $A^* := \{(i, j) \in A' : x^*_{ij} > 0\}$ the *support* of x^*, and by $G^* = (V', A^*)$ the

corresponding *support graph*. In addition, for any real function $z : W \to \mathbb{R}$ defined on a discrete domain W and for any $Q \subseteq W$ we write $z(Q)$ for $\sum_{t \in Q} z_t$.

3.1 Bridge inequalities

A well-known CSP property is that it always has an *optimal* solution in which no row/column has just one suppressed entry. For primary suppressions this follows from the required protection levels, whereas for secondary suppressions the property derives from the assumption $w_{ij} \geq 0$ for the objective function coefficients in (4).

The following more general property can however be derived; see Gusfield [11], Carvalho, Dellaert and Osório [6], and Kao [12]. Let $G = (V, E)$ be the undirected counterpart of the network $D = (V, A)$, obtained by disregarding arc directions. By construction, G is a complete bipartite graph whose edges are in one-to-one correspondence with the entries of the table to be protected. For any $S \subset V$, let $\delta(S) := \{[i, j] \in E : i \in S, j \notin S\}$ be the undirected *cut* associated with S, and let $E(S) := \{[i, j] \in E : i, j \in S\}$ contain the edges with both terminal nodes in S. As customary, for singleton sets $S = \{v\}$ we write $\delta(v)$ instead of $\delta(\{v\})$. Now take any optimal solution to CSP, and let SUP be the set of the edges of G associated with the suppressed cells. Without loss of generality we can assume that SUP is *minimal* with respect to set inclusion, in the sense that no proper subset of SUP corresponds to a feasible CSP solution. We claim that the subgraph $G_{SUP} = (V, SUP)$ of G induced by SUP is bridgeless, i.e., it does not contain any *bridge* (a bridge being a cut of cardinality 1). Indeed, suppose by contradiction that there exists $S^* \subset V$ such that $|\delta(S^*) \cap SUP| = 1$. As already observed in Section 2, feasible tables correspond to flow circulations in the directed network $D = (V, A)$. A well-known property of flow circulations is that the total amount of flow entering any node set W is equal to the total amount of flow leaving W. Writing this property for $W = S^*$, an attacker can then obtain the equation $\sum_{(i,j) \in \delta^+(S^*)} a_{ij} = \sum_{(i,j) \in \delta^+(V \setminus S^*)} a_{ij}$, in which all the terms a_{ij} correspond to published table entries, except the one associated with the unique edge e^* in the bridge $\delta(S^*) \cap SUP$. Therefore, the value a_{e^*} of this entry can be recomputed exactly from the published data. But then SUP contains a useless suppression e^*, impossible because of the minimality assumption on SUP.

In view of the above property, we address the following graph augmentation problem, that we call the *Minimum-Weight Bridgeless Subgraph Problem* (MW-BSP): Given an undirected graph $G = (V, E)$ with edge weights $w_e \geq 0$ and an edge subset PS, find a bridgeless subgraph $G_{SUP} = (V, SUP)$ of G with $PS \subseteq SUP$ and such that the total weight $\sum_{e \in SUP} w_e$ is minimized. Even in case of a complete bipartite graph G, this problem turns out to be $\mathcal{NP}$-hard in the strong sense; see Frederickson and JáJá [8], and Kao [12]. An integer LP model for MW-BSP reads:

$$\min \sum_{e \in E} w_e x_e \tag{9}$$

subject to

$$x(\delta(S) \setminus \{e\}) \geq x_e, \quad \text{for all } S \subset V, e \in \delta(S) \tag{10}$$

$$x_e = 1, \quad \text{for all } e \in PS, \tag{11}$$

$$0 \leq x_e \leq 1 \text{ integer}, \quad \text{for all } e \in E, \tag{12}$$

where $x_e = 1$ if $e \in SUP$; $x_e = 0$ otherwise. Constraints (10) are called *bridge inequalities*, and avoid to have both $x_e = 1$ and $x(\delta(S) \setminus \{e\}) = 0$, i.e., they forbid the occurrence of a bridge $\{e\} = \delta(S) \cap SUP$ in G_{SUP}.

By construction, problem MW-BSP is a *relaxation* of CSP, in the sense that every inclusion-minimal CSP solution corresponds to a feasible MW-BSP solution (but not vice-versa). Therefore all valid constraints for MW-BSP, including the bridge inequalities (10), are valid CSP cuts (assuming without lost of generality that one restricts to inclusion-minimal CSP solutions). Notice that problems MW-BSP and CSP coincide for *general tables*, i.e., for tables with $lb_{ij} = ub_{ij} = +\infty$ for all cell (i, j); see Carvalho, Dellaert and Osório [6].

For singleton sets $S = \{v\}$ the bridge inequalities become the following constraints addressed in Kelly, Golden and Assad [14]:

$$x(\delta(v) \setminus \{e\}) \geq x_e, \text{ for all } v \in V, e \in \delta(v), \tag{13}$$

which avoid the occurrence of pendant nodes (i.e., of nodes with degree 1) in G_{SUP}. In CSP terms, these inequalities forbid to have just one suppression in any row i (when $v = r_i$) and in any column j (when $v = c_j$). We call (13) the *fan inequalities*.

3.2 Comb inequalities

Bridge inequalities are not the only cuts one can derive from the MW-BSP relaxation introduced in the previous section. In fact, as MW-BSP is an $\mathcal{NP}$-hard problem one expects a very rich facial structure of the associated polytope. We next introduce a new class of valid inequalities for MW-BSP, which is related to the famous class of comb inequalities for the traveling salesman problem; see, e.g., Grötschel and Padberg [10].

Consider the simple fractional solution x^* of the LP relaxation of the MW-BSP model (9)–(12) drawn in Figure 2, where squares correspond to row nodes in G, and circles to column nodes. Let $T := \{t_1, t_2, t_3\}$ and $H := \{v_1, v_2, v_3, w\}$. Notice that x^* satisfies with equality the following valid MW-BSP inequalities:

$$x_{t_i} \leq 1, \quad \text{for } i = 1, 2, 3, \tag{14}$$

$$x_e \geq 0, \quad \text{for all } e \in \delta(H) \setminus T, \tag{15}$$

$$x(\delta(v_i) \setminus \{t_i\}) \geq x_{t_i}, \quad \text{for } i = 1, 2, 3, \tag{16}$$

$$x(\delta(w) \setminus \{\sigma\}) \geq x_\sigma \ . \tag{17}$$

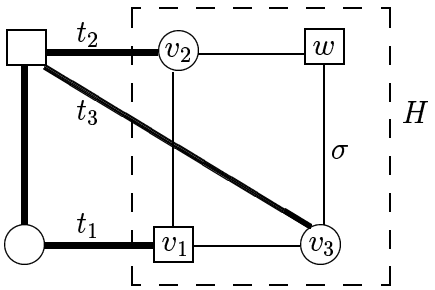


Figure 2: The support graph of a fractional solution x^*. Edges in bold and simple line correspond to variables with value 1 and 0.5, respectively.

We claim that no *integer* MW-BSP solution can satisfy with equality all the above inequalities. To prove the claim we observe that an integer point x satisfies with equality a given fan inequality $x(\delta(v) \setminus \{e\}) \geq x_e$ if and only if the 'apex' node v has degree 0 or 2 in the support of x. Hence the support of an integer x satisfying with equality the fan inequalities (16)–(17) would have even degree on all the nodes in H. Moreover, the tightness of conditions (14)–(15) would imply that the cut $\delta(H)$ contains exactly 3 edges of the support of x, impossible since no graph can have even degree on all the nodes of a shore of an odd cardinality cut.

The above discussion leads to the conclusion that a valid MW-BSP inequality which cuts off the fractional point of Figure 2 can be obtained by requiring a slack of 1 in at least one of inequalities (14)–(17). A general class of inequalities can be obtained as in the following theorem.

Theorem 3.1 *Let $S_1, S_2, \ldots, S_h$ be pairwise disjoint node subsets, and for $i = 1, \ldots, h$ let e_i be any edge in $\delta(S_i)$. Define $H := \cup_{i=1}^{h} S_i$, and let $T \subset \delta(H)$ with $|T|$ odd. Then the following* comb inequality

$$x(E(H) \setminus \bigcup_{i=1}^{h} E(S_i)) + x(\delta(H) \setminus T) \geq \sum_{i=1}^{h} x_{e_i} - \frac{|T|-1}{2} \tag{18}$$

is valid for MW-BSP (and hence for CSP).

Proof. A Chvátal-Gomory derivation of (18) is obtained by multiplying by 1/2 and adding up the following inequalities: $-x_e \geq -1$ for all $e \in T$; $x_e \geq 0$ for all $e \in \delta(H) \setminus T$; and $x(\delta(S_i)) - 2x_{e_i} \geq 0$ for all $i \in \{1, \ldots, h\}$. □

A relevant special case of comb inequalities arises when $|S_i| = 1$ for all i.

Corollary 3.1 *Let $H \subset V$ and $T \subset \delta(H)$ with $|T|$ odd, and for all $v \in H$ let e_v be any edge in $\delta(v)$. The following* matching inequality

$$x(E(H)) + x(\delta(H) \setminus T) \geq \sum_{v \in H} x_{e_v} - \frac{|T|-1}{2} \tag{19}$$

is valid for MW-BSP (and hence for CSP).

In the example of Figure 2, the fractional point x^* violates the matching inequality associated with the choice $e_{v_i} = t_i$ $(i = 1, 2, 3)$ and $e_w = \sigma$, as we have $x^*(E(H)) + x^*(\delta(H) \setminus T) = 2 + 0 < \sum_{v \in H} x^*_{e_v} - (|T| - 1)/2 = 3.5 - 1$.

3.3 Cover inequalities

Capacity cuts (7)–(8) have been derived without using the fact that x-variables must be 0-1 valued, a property that can be exploited to derive strong families of additional constraints which are useful to tighten the LP relaxation of our model. In particular, following the seminal work of Crowder, Johnson and Padberg [5] we can observe that each single inequality in (7)–(8) can be viewed as a knapsack constraint, hence it implies a number of "more combinatorial" restrictions. To be specific, let $\sum_{(i,j) \in A} \alpha_{ij} x_{ij} \geq \alpha_0$ represent any inequality in (7) and (8), in the strengthened form discussed at the end of Section 2. By means of the variable transformation $x_{ij} = 1 - \xi_{ij}$ we can bring the inequality to the form $\sum_{(i,j) \in A} \alpha_{ij} \xi_{ij} \leq \sum_{(i,j) \in A} \alpha_{ij} - \alpha_0$. Along with the 0-1 condition on the ξ-variables, this constraint defines a knapsack problem with item set A, item weights α_{ij}'s, and capacity $W := \sum_{(i,j) \in A} \alpha_{ij} - \alpha_0$. Hence any constraint that is valid for this knapsack problem is also valid for CSP. In particular, we have considered the well-known *cover inequalities*:

$$\sum_{(i,j) \in Q} \xi_{ij} \leq |Q| - 1, \text{ for all } Q \subset A : \sum_{(i,j) \in Q} \alpha_{ij} > W = \sum_{(i,j) \in A} \alpha_{ij} - \alpha_0, \tag{20}$$

which forbid the choice of all the items in any set Q whose global weight exceeds the knapsack capacity. In terms of the original variables, these inequalities read

$$\sum_{(i,j) \in Q} x_{ij} \geq 1, \text{ for all } Q \subset A : \sum_{(i,j) \in A \setminus Q} \alpha_{ij} < \alpha_0, \tag{21}$$

and require the suppression of at least one element of certain sets Q of cells.

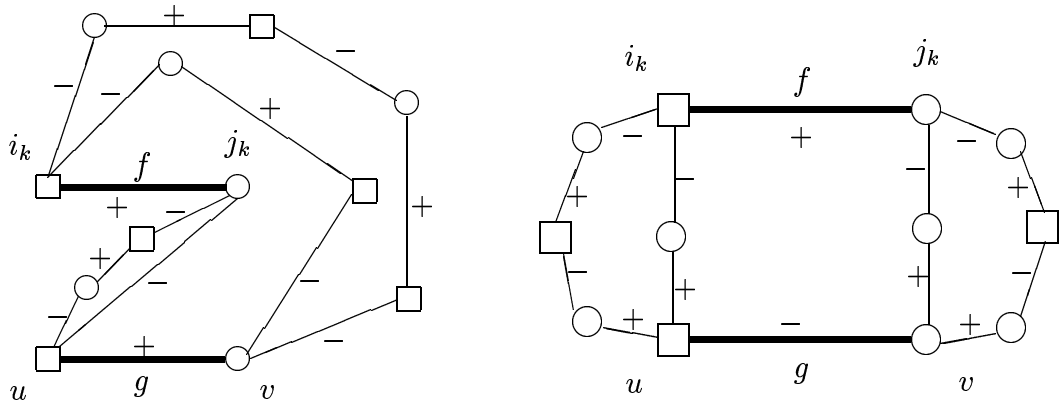
Cover inequalities (21) can easily be improved to their *lifted* form [16]:

$$\sum_{(i,j) \in EXT(Q)} x_{ij} \geq |EXT(Q)| - |Q| + 1, \text{ for all } Q \subset A : \sum_{(i,j) \in A \setminus Q} \alpha_{ij} < \alpha_0, \tag{22}$$

where $EXT(Q) := Q \cup \{(i,j) \in A : \alpha_{ij} \geq \max\{\alpha_{uv} : (u,v) \in Q\}\}$.

3.4 2-edge cover inequalities

According to our computational study, the following particular kind of cover inequalities plays an important role in tightening the LP relaxation of some hard CSP instances.



(a) same-sign compensation (b) opposite-sign compensation

Figure 3: The two possible cases when every cycle in SUP through f goes through edge g (recall that G is bipartite, hence it does not contain odd-cardinality cycles).

Consider the graph G defined in Subsection 3.1, and let $f = [i_k, j_k]$ be the edge of G corresponding to any given sensitive cell (i_k, j_k). Take any integer CSP solution x with support $SUP = \{[i,j] \in E : x_{ij} = 1\}$. We know that $f \in SUP$, hence because of the bridgeless property SUP must contain a cycle through f. Furthermore, in some cases we can find a convenient edge $g \in E \setminus \{f\}$ for which even $SUP \setminus \{g\}$ must contain a cycle through f, in the sense that the suppression of g does not provide, alone, a sufficient protection for f. In particular, this happens when $g = [u, v]$ corresponds to a table entry a_{uv} for which one of the following two easy-checkable conditions holds:

$$(u = i_k \text{ or } v = j_k) \text{ and } (UPL_k > lb_{uv} \text{ or } LPL_k > ub_{uv}) \tag{23}$$

$$(u \neq i_k \text{ and } v \neq j_k) \text{ and } (UPL_k > ub_{uv} \text{ or } LPL_k > lb_{uv}) \text{ and } (UPL_k > lb_{uv} \text{ or } LPL_k > ub_{uv}). \tag{24}$$

Indeed, suppose by contradiction that $SUP \setminus \{g\}$ does not contain a cycle through f. Then any variation of $a_{i_k j_k}$ must be compensated completely by a variation of a_{uv}. Depending on whether the variation of $a_{i_k j_k}$ is compensated by a variation of a_{uv} of the same or opposite sign, we have either case (a) or (b) in Figure 3. Notice that case (a) cannot arise when $u = i_k$ or $v = j_k$. In case (a), we must have $UPL_k \leq ub_{uv}$ and $LPL_k \leq lb_{uv}$, whereas in case (b) we must have $UPL_k \leq lb_{uv}$ and $LPL_k \leq ub_{uv}$. If none of the above conditions holds, as required in (23)–(24), then we can conclude that SUP must contain a cycle through f which does not contain g, as claimed. In this hypothesis we can then write the following valid *2-edge cover inequality*:

$$x(\delta(S) \setminus \{f, g\}) \geq 1, \text{ for each } S \subset V : \{f, g\} \subseteq \delta(S). \tag{25}$$

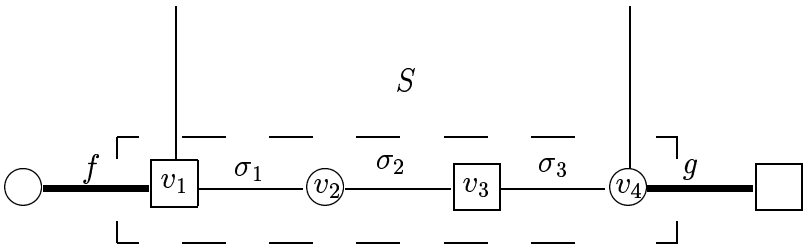


Figure 4: A part of the support graph of a fractional solution x^*. Edges in bold and simple line correspond to variables with value 1 and 0.5, respectively.

3.5 Path inequalities

The interaction between bridge and cover inequalities has interesting combinatorial consequences.

Consider the linear programming relaxation of the CSP model (4)–(8), and let x^* be the fractional solution whose support (viewed as a subgraph of G) is partially drawn in Figure 4. Assume that f corresponds to a sensitive cell, and g corresponds to a cell (u, v) satisfying condition (23) or (24). We have seen in the previous subsection that this implies that any feasible integer solution x must satisfy the 2-edge cover inequality $x(\delta(S) \setminus \{f, g\}) \geq 1$, where $S := \{v_1, v_2, v_3, v_4\}$. Now, the fractional point x^* satisfies with equality this constraint as well as the bound constraints $x_f \leq 1$ and $x_g \leq 1$. In addition, for each $v_i \in S$ there exists a fan inequality with apex in v_i which is tight for x^*, namely: $x(\delta(v_1)) \geq 2\,x_f$, $x(\delta(v_2)) \geq 2\,x_{\sigma_1}$, $x(\delta(v_3)) \geq 2\,x_{\sigma_2}$, and $x(\delta(v_4)) \geq 2\,x_g$. We claim that no feasible CSP integer solution x can satisfy with equality all these inequalities. Indeed, $x(\delta(S) \setminus \{f, g\}) = 1$ and $x_f = x_g = 1$ imply $x(\delta(S)) = 3$, whereas the tightness of the above fan inequalities with apex $v_1, \ldots, v_4$ implies that the degree of these vertices in the support graph of x is either 0 or 2. But this is impossible, as no graph can have even degree on all the nodes of a shore of an odd-cardinality cut.

The above considerations lead to the following result.

Theorem 3.2 *Let $S \subset V$ and $T \subset \delta(S)$, $|T| \geq 2$ and even, be such that the inequality $x(\delta(S) \setminus T) \geq 1$ is valid for CSP. For each $v \in S$, let e_v be any edge in $\delta(v)$. The* path inequality

$$x(E(S)) + x(\delta(S) \setminus T) \geq \sum_{v \in S} x_{e_v} - \frac{|T|}{2} + 1 \tag{26}$$

is then valid for CSP.

Proof. A Chvátal-Gomory derivation of (26) can be obtained by adding up the following valid inequalities multiplied by 1/2, and then rounding up the right-hand-side value: $x(\delta(S)) - x(T) \geq 1$; $x(\delta(v)) - 2\,x_{e_v} \geq 0$ for each $v \in S$; $-x_e \geq -1$ for each $e \in T$. □

Observe that, for every $v \in S$, the coefficient of x_{e_v} in a path inequality is equal to -1 when $e_v \in T$, or $e_v = e_w$ for some $w \in S \setminus \{v\}$. In the example of Figure 4, the path inequality associated with $T = \{f, g\}$ and $e_{v_1} = f$, $e_{v_2} = \sigma_1$, $e_{v_3} = \sigma_2$, $e_{v_4} = g$ is violated as $x^*(E(S)) + x^*(\delta(S) \setminus T) = 1.5 + 1 < \sum_{v \in S} x^*_{e_v} - |T|/2 + 1 = 3 - 1 + 1$.

The validity of a path inequality (26) follows from that of the cover inequality $x(\delta(S) \setminus T) \geq 1$. As already observed, a sufficient condition for the validity of this latter inequality when $|T| = 2$ is that $T = \{f, g\}$, where f corresponds to a sensitive cell (i_k, j_k), and g corresponds to a cell (u, v) satisfying condition (23) or (24).

A modified version of the path inequality can be found in Fischetti and Salazar [7].

4 A branch-and-cut algorithm

The model presented in the previous sections contains a huge number of constraints. We address its solution by using a *branch-and-cut* technique, as introduced in Padberg and Rinaldi [17]. We next outline the main parts of our implementation.

4.1 Problem reduction

The computing time spent in the solution of a given instance depends on its size, expressed in terms of both the number of decision variables involved, and the number of nonzero protection levels. We next outline simple criteria to reduce the size of a given CSP instance.

A (typically substantial) reduction on the number of nonzero protection levels is achieved in a *preprocessing* phase. It is based on the observation that primary suppressions alone may be sufficient to protect some of the sensitive cells. To be specific, let us consider the directed incremental network $D(PS)$ of Section 2, associated with the *original* set of sensitive cells PS (the one before the forthcoming reduction). In a preprocessing phase we check whether the upper protection level of any $(i_k, j_k) \in PS$ is already guaranteed in $D(PS)$, in the sense that there exists a feasible flow circulation f^U in $D(PS)$ such that $f^U_{i_k j_k} - f^U_{j_k i_k} \geq UPL_k$. In this case one can clearly set $UPL_k = 0$, thus avoiding to consider (i_k, j_k) explicitly. A similar reasoning applies to lower protection levels.

A naive implementation of the above idea may be excessively time consuming for large instances, in that it may require even more computing time than the whole branch-and-cut algorithm applied on the original instance. Hence a parametric implementation of the max-flow computations involved in preprocessing is needed. We suggest the following approach.

We initialize to "unlabeled" an array FLAG containing (at most) two entries for each $(i_k, j_k) \in PS$, one associated with UPL_k (if $UPL_k \neq 0$) and the other to LPL_k (if $LPL_k \neq 0$). We then consider, in sequence, the unlabeled entries of FLAG, according to decreasing

values of the associated UPL_k or LPL_k. Suppose the current entry is associated with a certain UPL_k (a similar reasoning applies when the entry is associated with LPL_k). We solve a max-flow problem to check the existence of a circulation f^U in $D(SUP)$ such that $f^U_{i_k j_k} - f^U_{j_k i_k} \geq UPL_k$, and change the current entry of FLAG to "protected" (if such a flow f^U exists) or to "sensitive" (otherwise). Notice that the max-flow computation can be stopped as soon as the current flow value $f^U_{i_k j_k} - f^U_{j_k i_k}$ attains the threshold UPL_k. Before proceeding with the next unlabeled entry, we use the information associated with the current flow f^U to possibly set to "protected" some other entries of FLAG, namely those associated with the arcs of $D(SUP)$ carrying a sufficient flow. In this way we avoid a (typically substantial) number of max-flow computations in the subsequent iterations.

We now address the reduction of variables in the LP programs to be solved within our branch-and-cut algorithm. Our approach is to fix to 0 or 1 some decision variables during the processing of the current node of the branch-decision tree. We use the classical criteria based on LP reduced costs; see, e.g., Padberg and Rinaldi [17]. According to our computational experience, these criteria allows one to fix a large percentage of the variables very early during the computation. Moreover, we have implemented a variable-pricing technique (see [17]) to speed-up the overall computation and to drastically reduce memory requirement when instances with more than 10,000 variables are considered. In order to save computing time, we skip the pricing procedure for 5 consecutive iterations whenever no negative reduced-cost variables are found.

4.2 Separation

Separation is a crucial phase of branch-and-cut algorithms. In our implementation, for each given LP solution x^* we first solve the separation problem for capacity constrains (7)–(8). If no violation is found and x^* is not integer, we proceed with the separation of bridge inequalities (10), cover inequalities (20), 2-edge cover inequalities (25), path inequalities (26), and comb inequalities (19). As a general rule, we try to generate several highly-violated cuts for each family, and interrupt the separation sequence when violations are found within the current family. In any case, no more than 100 cuts are added to the LP in a separation round. Based on our computational experience, in order to save computing time we decided to inhibit any separation procedure (except that for capacity cut constraints) failing to produce violated cuts after 5 consecutive calls. Moreover, for large-scale instances with $NR > 100$ or $NC > 100$ we do not check any 2-edge cover and path inequality for violation.

We next outline the separation procedure we have implemented for the various families of valid inequalities proposed in the previous sections.

4.2.1 Capacity constraints

We describe our exact separation scheme for (7), in their strengthened form; constraints (8) can be dealt with in a perfectly analogous way. For any given sensitive cell (i_k, j_k) with $UPL_k > 0$, a most violated constraint (7) is associated with a node set S in the network $D(A)$ such that $j_k \in S$, $i_k \notin S$, and $\sum_{(i,j)\in\delta^+(S)\setminus\{(j_k,i_k)\}} q'_{ij} x^*_{\phi(i,j)}$ is a minimum. This node set S can be located efficiently as follows. We define a modified network by removing from $D(A)$ the arc (j_k, i_k), and we assign all the remaining arcs (i, j) a modified capacity $q'_{ij} x^*_{\phi(i,j)}$. Notice that these capacities are well defined as x^* is given, and $q'_{ij} = \min\{q_{ij}, UPL_k\}$ can easily be computed for the current sensitive cell (i_k, j_k). With these definitions, $\sum_{(i,j)\in\delta^+(S)\setminus\{(j_k,i_k)\}} q'_{ij} x^*_{\phi(i,j)}$ can be interpreted as the capacity of cut $\delta^+(S)$ in the modified network, hence the minimum-capacity such cut can be located efficiently through a single max-flow computation.

The above scheme requires the solution of a max-flow problems for every nonzero protection level, which can be very time consuming for large-scale instances. Hence we have implemented a parametric max-flow computation for reducing the number of nonzero protection levels, which is in the spirit of the approach described in Subsection 4.1. In order to generate several highly violated cuts, for each max-flow computation we detect two minimum-capacity cuts, namely those associated with an inclusion minimal and maximal set S.

4.2.2 Bridge inequalities

Let SUP denote the support of the given fractional solution x^*. Clearly, no violated bridge inequality $x(\delta(S) \setminus \{e\}) \geq x_e$ can arise for $e \notin SUP$, nor for $e \in PS$ (as in this latter case the bridge inequality is dominated by a capacity constraint). For any given $e = [i, j] \in SUP \setminus PS$, in turn, a most violated such inequality corresponds to the node set $S \subset V$ with $e \in \delta(S)$ and such that $x^*(\delta(S) \setminus \{e\})$ is a minimum. Hence it can be found efficiently as follows. Consider the graph $G_{SUP} = (V, SUP)$, and associate each $f \in SUP$ with a positive capacity given by x^*_f. We temporarily remove edge e from G_{SUP}, and then define S as one of the two shores of a minimum-capacity cut $\delta(S)$ separating i and j in the resulting network. The set S can be identified efficiently by finding a maximum flow from i to j. Once S has been located, we check whether $x^*(\delta(S) \setminus \{e\}) < x^*_e$ (in which case a violated cut has been found) or $x^*(\delta(S) \setminus \{e\}) \geq x^*_e$ (in which case no violated bridge cut exists for the given edge e).

Also in this case, we generate two minimum capacity cuts for each max-flow problem, and use parametric techniques to reduce the computational effort.

4.2.3 Comb inequalities

We first describe an exact separation procedure for matching inequalities (19), which follows the general framework recently proposed by Caprara and Fischetti [2]. As shown in the proof of Theorem 3.1, matching inequalities admit a Chvátal-Gomory derivation obtained by combining fan inequalities (13) and bound constraints $x_e \leq 1$ after multiplication by 0 or 1/2, i.e., they are $\{0, 1/2\}$-*Chvátal-Gomory cuts* in the terminology used in [2] . Now let $M\,x \leq 0$ represent the inequality system (13), where each row of M is associated with a fan inequality of the type $x_e - x(\delta(v) \setminus \{e\}) \leq 0$. According to [2], the separation of $\{0, 1/2\}$-Chvátal-Gomory cuts only needs to consider the *parity support* $\overline{M}$ of M, where $\overline{M}$ is defined as the 0-1 matrix having the same dimensions as M, in which an entry is 1 if and only if the corresponding entry of M is odd.

Notice that $\overline{M}$ contains several repeated rows. Namely, for any $v \in V$ the rows of $\overline{M}$ associated with the fan inequalities $x_e - x(\delta(v) \setminus \{e\}) \leq 0$ for all $e \in \delta(v)$ are the same. In this situation it is possible to remove from $\overline{M}$ all duplicated rows by keeping, for each $v \in V$, only the fan inequality $x_{e_v} - x(\delta(v) \setminus \{e_v\}) \leq 0$ which maximizes $x^*_e - x^*(\delta(v) \setminus \{e\})$ for all $e \in \delta(v)$. Moreover, one can also remove the rows of $\overline{M}$ which correspond to a fan inequality with $x^*_{e_v} - x^*(\delta(v) \setminus \{e_v\}) \leq -1$, along with the columns associated with variables x_e having $x^*_e = 0$; see the reduction criteria in [2, Section 3.1]. After this reduction, it turns out that $\overline{M}$ coincides with the node-edge incidence matrix of a subgraph of G, hence it has exactly two nonzero entries in each column. In this situation, $\{0, 1/2\}$-Chvátal-Gomory separation can be solved efficiently, as it amounts to finding a minimum-weight odd cut in a certain labeled graph; see [2, Theorem 5].

As to the more general comb inequalities, we have implemented a simple heuristic based on the idea of shrinking certain node sets S with $x(E(S)) \geq |S| - 1$ to a single node, and then applying matching separation to the resulting graph.

4.2.4 Cover inequalities

According to Section 3.3, cover inequalities (21) are derived from a certain (given) capacity cut (7)–(8) of the form $\sum_{(i,j)\in A} \alpha_{ij} x_{ij} \geq \alpha_0$. In our implementation we heuristically restrict our attention to the capacity cuts which are present either in the current LP or in the constraint pool. Our separation procedure follows the standard approach (see Nemhauser and Wolsey [16]), and is described hereafter.

The separation problem for the cover cuts that can be derived from a *given* inequality $\sum_{(i,j)\in A} \alpha_{ij} x_{ij} \geq \alpha_0$ calls for a set $Q \subset A$ such that $\sum_{(i,j)\in Q} \alpha_{ij} > W := \sum_{(i,j)\in A} \alpha_{ij} - \alpha_0$, which minimizes the left-hand-side value $\sum_{(i,j)\in Q} x^*_{ij}$. This is an optimization problem in

itself, which can be formulated as the *0-1 knapsack problem*:

$$\min \left\{ \sum_{(i,j)\in A} x^*_{ij} z_{ij} \ : \ \sum_{(i,j)\in A} \alpha_{ij} z_{ij} \geq W + \epsilon, \ z \in \{0,1\}^A \right\} \tag{27}$$

where $z_{ij} = 1$ if and only if $(i,j) \in Q$, and ϵ is a sufficiently small positive value (e.g., $\epsilon = 1$ if all α_{ij}'s and α_0 are integer). Notice that x^*_{ij} and α_{ij} are *given* nonnegative values. Although $\mathcal{NP}$-hard, this problem can typically be solved in short computing time by means of specialized codes, see Martello and Toth [15]. Moreover, one can easily show that most of the decision variables z_{ij} can typically be fixed to 0 or 1 by a simple preprocessing step. Indeed, one can fix $z_{ij} = 0$ if $\alpha_{ij} = 0$ (as $x^*_{ij} \geq 0$) or $x^*_{ij} = 1$ (as we are only interested in sets Q leading to a violated cover inequality, i.e., such that $\sum_{(i,j)\in Q} x^*_{ij} < 1$), whereas one can fix $z_{ij} = 1$ whenever $\alpha_{ij} > 0$ and $x^*_{ij} = 0$ (as this choice does not penalize the objective function value).

For each capacity inequality in the current LP or in the pool, in turn, we first apply the above preprocessing step for fixing z-variables, and then solve to optimality the knapsack problem (27) through the algorithm MT1R described in Martello and Toth [15]. In this way we locate a set Q corresponding to a most violated cover inequality, for which we compute the associated "extension" $EXT(Q)$ and check for violation the lifted cover inequality (22).

4.2.5 2-edge cover inequalities

For 2-edge cover inequalities (25) we have implemented the following exact separation scheme. Let $SUP := \{[i,j] \in E : x^*_{ij} > 0\}$ be the support of the given point x^* to be separated. For $k = 1, \ldots, p$, in any sequence, we apply the following steps. Let f denote the edge in SUP corresponding to sensitive cell (i_k, j_k). For each $g = [u,v] \in SUP \setminus \{f\}$ which satisfies (23) or (24), we look for a cut $\delta(S)$ in G with minimum $x^*(\delta(S) \setminus \{f,g\})$. This cut can be located efficiently by solving (at most) two max-flow problems on the subgraph of G induced by $SUP \setminus \{f,g\}$, with $x^*_e > 0$ being interpreted as the capacity of each edge e.

The above separation scheme can be excessively time consuming, as for every f several edges g may exists which satisfy condition (23) and (24). A considerable speed-up can be obtained along the following lines. For each primary suppression $f = (i_k, j_k)$, we first apply the separation procedure described in Subsection 4.2.2 to find a violated bridge inequality of type $x(\delta(S) \setminus \{f\}) \geq 1$. If no violated such cut is found, then as a by-product of the max-flow computation we have found a flow, of value at least 1, between i_k and j_k in the subgraph induced by $SUP \setminus \{f\}$. Now let F be the support of this flow, containing all the edges $e \in SUP \setminus \{f\}$ carrying a nonzero flow. Clearly, the removal from $SUP \setminus \{f\}$ of any $g \notin F$ can not "break" this flow, hence $x(\delta(S) \setminus \{f,g\}) \geq 1$ certainly holds for every $g \notin F$ and for every $S \subset V$ such that $\{f,g\} \subseteq \delta(S)$. It then follows that, in the separation

of 2-edge cover inequalities for the given f, one only needs to consider the edges $g \in F$ which satisfy (23) or (24).

4.2.6 Path inequalities

We next give an efficient exact separation procedure for the path inequalities (26) with $|T| = 2$ and $T \cap PS \neq \emptyset$. Let $T = \{f, g\}$, where $f = [i_k, j_k] \in PS$ corresponds to a sensitive cell (i_k, j_k), and $g = [u, v]$ satisfies condition (23) or (24). According to Section 3.4, the 2-edge cover inequality $x(\delta(S) \setminus \{f, g\}) \geq 1$ holds for all $S \subset V$ such that $\{f, g\} \subseteq \delta(S)$, as required in Theorem 3.2. Among all these possible sets S, we want to find the one for which $x^*(E(S)) + x^*(\delta(S) \setminus \{f, g\}) - \sum_{v \in S} x^*_{e_v}$ is a minimum, which corresponds to a most violated path inequality (26) for the given set T. Notice that for each node v in (26) it is always convenient to choose e_v so that $x^*_{e_v}$ is as large as possible, hence we define $e_v :=$ arg max $\{x^*_e : e \in \delta(v)\}$ for each $v \in V$.

We need to bring (26) to an equivalent form which is more suitable for separation. To this end we define the auxiliary variables $z_v := x(\delta(v)) - 2x_{e_v}$ for all $v \in V$, and observe that

$$\sum_{v \in S} z_v = 2x(E(S)) + x(\delta(S)) - 2 \sum_{v \in S} x_{e_v},$$

hence (26) is equivalent to:

$$x(\delta(S)) + \sum_{v \in S} z_v \geq 2x(T) - |T| + 2 = 2(x_f + x_g). \tag{28}$$

Being f and g fixed, a most violated inequality (28) can be found as follows. As in separation of bridge inequalities, we consider the support graph of x^* as an undirected network G_{SUP} with edge capacities x^*_e. We have to consider (at most) four possible cases, depending on the pair of nodes in $W := \{i_k, j_k, u, v\}$ contained in S (notice that (28) changes if S is replaced by $V \setminus S$). For the case in which we impose, e.g., $i_k, u \in S$ and $j_k, v \in V \setminus S$, where $i_k \neq u$ and $j_k \neq v$, the best set S is found as follows. We add two new nodes s and t to network G_{SUP}, four new edges $[s, i_k]$, $[s, u]$, $[j_k, t]$, and $[v, t]$ with very large capacity, plus an edge $[w, t]$ for each $w \in V \setminus \{j_k, v\}$, whose capacity is set to z^*_w to take care of term $\sum_{w \in S} z_w$ in (28). Then we find, through a max-flow algorithm, the minimum-capacity cut $(\tilde{S}, V \setminus \tilde{S})$ with $s \in \tilde{S}$ and $t \in V \setminus \tilde{S}$ in the new network, and define $S := \tilde{S} \setminus \{s\}$. The other three cases are dealt with in a perfectly analogous way.

A final comment is in order. The proof of Theorem 3.2 implies that a violated path inequality is never associated with a set S containing a node v with $x^*(\delta(v)) - 2x^*_{e_v} \geq 1$, hence all such nodes can be removed from the separation network. This is also true for the nodes v with $x^*(\delta(v)) = 0$, which are isolated in the support graph associated with x^*. These considerations allow for a considerable speed-up of the whole separation procedure.

4.3 Heuristic

The convergence of the overall branch-and-cut scheme can be speeded up if a near-optimal CSP solution is found very early during computation. Therefore one is interested in a heuristic algorithm requiring short computing time, to be applied (possibly several times) at each node of the branch-decision tree.

A widely-used heuristic CSP procedure has been proposed by Kelly, Golden and Assad [14]. The procedure works in stages, in each of which one finds heuristically a set of new suppressions needed to guarantee the required upper and lower protection levels for a certain sensitive cell (i_k, j_k). To be more specific, we start by defining $SUP := PS$ as the current set of suppressions, and define $w_{ij} = 0$ for all $(i, j) \in SUP$. Then, iteratively, we consider all the sensitive cells (i_k, j_k) according to some heuristically-defined sequence. For each such (i_k, j_k), we consider the incremental network $D(A)$ defined in Section 2, in which each pair of forward/reverse arcs corresponds to a cell. The arcs of $D(A)$ have an associated *cost*, equal to w_{ij} for the pair of arcs corresponding to each cell $(i, j) \notin SUP$, and equal to 0 for all other arcs (i.e., for the arcs of $D(SUP)$). We then find a min-cost flow circulation f in $D(A)$ such that $f_{i_k j_k} - f_{j_k i_k} = UPL_k$, and insert in SUP all the cells $(i, j) \notin SUP$ whose associated (forward or reverse) arcs of $D(A)$ carry a non-zero flow in f. This guarantees the fulfillment of the upper protection level for (i_k, j_k) in the new set SUP of suppressions. Afterwards we set to 0 the cost of the arcs of $D(A)$ associated with the newly inserted cells of SUP, and apply a similar min-cost computation to extend SUP so as to guarantee the fulfillment of the lower protection level LPL_k for (i_k, j_k). Finally, when all sensitive cells (i_k, j_k) have been processed a refinement procedure is applied, which removes from SUP all the superfluous suppressions.

The above approach can be viewed as a greedy procedure that considers one sensitive cell at a time, and enforces the required protection levels without considering possible interactions with the remaining sensitive cells. Notice however that the choice of using a min-cost flow computation for selecting the new cells to be suppressed is not necessarily optimal even for the single cell (i_k, j_k) to be protected. In fact, a simple reduction for the 0-1 knapsack problem shows that CSP is $\mathcal{NP}$-hard even when $|PS| = 1$ and $LPL_k = 0$, i.e. when we have just one sensitive cell with upper (but not lower) protection level imposed.

We next describe a modification of the Kelly-Golden-Assad heuristic. It is based on the observation that the min-cost flow computation may give a rough approximation of the actual cost involved in the new suppressions, in that the cost w_{ij} of each new suppressed cell is weighed by the flow of the corresponding forward/reverse arc in the objective function of the min-cost flow problem. In other words, the min-cost flow problem "estimates" a cost of $w_{ij}(f_{ij} + f_{ji})$ for the suppression of a cell whereas the actual suppression cost of this cell is just w_{ij} and does not depend on the flow values.

In order to try to overcome the above drawback, we follow the idea of Carvalho, Dellaert

and Osório [6] of replacing each single max-flow computation by a series of min-cost path computations. To be specific, for each given sensitive cell (i_k, j_k) we check whether the current set SUP of suppressions guarantees already the required upper protection level. If not, we compute a shortest path P from i_k to j_k in $D(A)$, add to SUP the cells associated with the forward/reverse arcs in P, set to 0 the cost in $D(A)$ of the corresponding arc pairs, and repeat. A similar computation is applied for enforcing the lower protection level.

The modified approach has some advantages over the classical Kelly-Golden-Assad scheme. Indeed, shortest-path algorithms are typically much faster than min-cost flow algorithms, in particular for dense graphs such as $D(A)$. One could argue that the new approach needs extra computing time to check whether the current sensitive cell (i_k, j_k) is indeed protected in the current SUP. But this requires the solution of two max-flow problems on $D(SUP)$, which is typically much sparser than $D(A)$. The net result is that the new procedure typically runs much faster than the original one. Moreover, the solution quality of the solutions delivered by the new procedure is, on average, comparable or better than that of the original Kelly-Golden-Assad proposal, although a strict dominance cannot be claimed.

We apply our heuristic based on shortest-path computation at the very beginning of the branch-and-cut code, right after the preprocessing phase for reducing the number of nonzero protection levels. Moreover, we have implemented a modified version of the heuristic which exploits the information associated with the fractional optimal solution x^* of the LP problems solved during branch-and-cut execution. In this version, the cell weights w_{ij} are modified to $w'_{ij} := (1 - x^*_{ij})w_{ij}$ so as to encourage shortest paths to choose arcs associated with cells (i, j) having x^*_{ij} close to 1, which are likely to be part of the optimal solution. Moreover, the arcs in $D(A)$ associated with cells (i, j) with x_{ij} fixed to 0 (see Subsection 4.1) are removed from $D(A)$ so as to prevent their choice and to speed-up shortest path computations.

After some computational testing we decided to apply our modified heuristic at the end of each node of the branch-decision tree, just before branching.

4.4 Branching

We have implemented a best-bound first strategy. Branching variable x_{ij} is chosen according to the following procedure, akin to that introduced by Applegate, Bixby, Chvátal and Cook [1] in the context of the traveling salesman problem. We consider the 5 fractional variables with smallest $|x^*_{ij} - 0.5|$, breaking ties in favour of variables with large w_{ij}. For each such candidate, in turn, we solve the current LP relaxation amended by the additional constraint $x_{ij} = 0$ or $x_{ij} = 1$, thus obtaining lower bounds LB_0 and LB_1. Let UB be the value of the incumbent integer CSP solution. If $LB_0 \geq UB$ and $LB_1 \geq UB$, then the current node is fathomed. Otherwise, if $LB_0 \geq UB$ (or $LB_1 \geq UB$) we just fix $x_{ij} = 0$ (or

$x_{ij} = 1$) and re-enter the separation phase. If none of the above conditions hold for any of the 5 candidates, we branch by selecting the one producing the largest $LB_0 + LB_1$.

5 Computational results

The algorithm described in the previous section was implemented in ANSI C, compiled with Watcom C/C++ 10.6, and run on a PC Pentium/200 with 32 Mbyte RAM. As to the LP-solver, we used the commercial package CPLEX 3.0.

We illustrate the performances of our approach on both real-world and random test instances. Unless explicitly stated, we set $w_{ij} = a_{ij}$ if (i, j) is a candidate to be a secondary suppression ($w_{ij} = 10$ if $a_{ij} = 0$), and 0 otherwise.

The real-world instances we consider are the 49 problems of the SDC Library set up by Cor Hurkens and considered in Geurts [9], plus 9 new problems provided by four national statistical offices. For every sensitive cell of the first 49 instances, the upper and lower protection levels are equal to 10% of the nominal value, rounded up to the nearest integer; the protection levels in the remaining 9 instances are specific for each problem. The known range of feasible values for each suppressed cell is $[0, +\infty]$.

Three random data sets have been considered.

The first random data set has been suggested by Dr. Roberto Benedetti from ISTAT, the Italian statistical office, and simulates real-world frequency tables. Every internal cell (i, j) of the table has a random integer value a_{ij} in range [0,5[with probability 0.05, and in range [5,500[with probability 0.95. Cells with 0 nominal value cannot be suppressed, whereas all cells with nominal value in [1,5[are classified as sensitive. For every sensitive cell, the lower protection level is equal to the nominal value minus one, and the upper protection level is equal to the nominal value. We assume that the attacker knows that the statistical office never suppresses zero entries, hence the feasible range for suppressed cells is $[1, +\infty]$.

The second random data set is generated as in Kelly, Golden and Assad [14]. Here, every internal cell (i, j) has a random integer value a_{ij} in range [1,1000] with probability 0.8, and value 0 with probability 0.2. Zero entries cannot be suppressed. Internal and marginal cells with non-zero nominal value are classified as sensitive with probability 0.2 and 0.1, respectively. For every sensitive cell, the upper and lower protection levels are set to 15% of the nominal value, rounded up to the nearest integer. The known range of feasible values for each suppressed cell is $[0, +\infty]$.

The third random data set contains "general tables" generated as in Carvalho, Dellaert and Osório [6], in which the known range of feasible values for each suppressed cell is $[-\infty, +\infty]$. Every internal cell (i, j) has a random integer value a_{ij} in $[-100, 100]$ and a random integer weight w_{ij} in $[1, 10]$. All marginals must be published. Sensitive cells are

randomly generated according to two different criteria. For *sparse* instances, the number p of sensitive cells is randomly generated in $[1, \sqrt{d}]$, where $d := (NR + NC)/2$. For *dense* instances, instead, p is randomly generated in $[1 + d, 2d]$. Notice that protection levels are immaterial for general tables, hence we set the lower and upper protection levels to 0 and 1, respectively.

Tables 1–5 report computational results on the above instances, and give the following information:

name: name of the instance (if available);

NR: number of rows (without the marginal row);

NC: number of columns (without the marginal column);

p: number of sensitive cells (primary suppressions);

pl: number of nonzero protection levels after the problem reduction;

t_p: wall clock seconds spent for reducing the number of non-zero protection levels (pre-processing phase) ;

UB_h: percentage ratio (UB' − optimal solution value)/ (optimal solution value), where UB' is the upper bound value computed by our initial heuristic;

t_h: wall clock seconds spent by our initial heuristic;

UB_r: percentage ratio (UB − optimal solution value)/ (optimal solution value), where UB is the upper bound value computed by our heuristic at the end of the root node;

LB_r: percentage ratio (optimal solution value − LB)/ (optimal solution value), where LB is the lower bound value available at the end of the root node;

t_r: wall clock seconds spent at the root node of the branch-decision tree;

z^*: optimal solution value (real-world instances only);

s: number of secondary suppressions in the optimal solution found;

n: number of subproblems (nodes in the decision tree) elaborated;

t_t: wall clock seconds for the overall branch-and-cut algorithm;

ca: number of capacity constraints generated;

br: number of bridge constraints generated;

co: number of cover inequalities generated;

2e: number of 2-edge cover inequalities generated;

pa: number of path inequalities generated;

cm: number of comb inequalities generated.

Table 1 gives results on the CSP instances from the SDC library. Our approach solves each instance in less than one second on a PC Pentium/200. As a comparison, the code of Geurts [9] required 2445, 4244 and 3072 CPU seconds on a SUN Sparc 1+ computer for solving instances VB20, VB26 and VB31, respectively. Table 2 gives results for the 9 additional real-world instances provided by four national statistical offices. All of them are solved very easily by our code.

Results on the three random data sets are reported in Tables 3 to 5. Every row of these tables gives average results over 10 instances. We imposed a time-limit of 3,600 seconds for each instance, which has only attained for 45 (out of 550) problems of the Kelly-Golden-Assad generation. Statistics refer to solved instances. For the sake of brevity we do not report statistics for all the instances we solved, and refer the interested reader to Fischetti and Salazar [7] for more details.

According to Table 3, the instances of Benedetti's generation are quite easy to solve to proven optimality by using our branch-and-cut code. The initial preprocessing phase fixes to zero a very large percentage of the given protection levels, mainly for large-scale instances. Most of the cuts generated in the separation phase are capacity cut constraints or cover inequalities. A significant number of bridge, 2-edge, and path inequalities is however generated for small/medium size instances. Comb inequalities, instead, play no role for this class of problems. Instances with up to 250,000 entries are solved in about 10 minutes on a PC. Larger instances are very likely to need no secondary suppressions at all, hence have not been considered.

The instances of the Kelly-Golden-Assad generation addressed in Table 4 appear much harder to solve. Actually, in 45 out of 550 cases our code did not succeed in solving the problem to proven optimality within the 1-hour time limit. For these cases, however, the gap between the incumbent solution value and the best lower bound is very small (0.27%, on average). The problem difficulty is mainly due to the presence of marginal sensitive cells with abnormally large protection levels. The same problems without marginal sensitive cells can instead be solved very easily (e.g., 100×100 tables require just 5 seconds). On the whole, the performance of our code is still satisfactory, in that all tables up to 50×50 are solved to proven optimality within short computing time.

According to Table 5, the sparse problems of the Carvalho-Dellaert-Osório generation can be solved easily by our algorithm. The same holds for the dense instances of this class, as reported in [7]. For all the problems in our test-bed, the lower bound computed at the root node coincide with the optimal solution value. As expected, no cover inequality

is generated in that the problems in this class have no knapsack substructure. Bridge and comb inequalities, instead, play an important role in modeling the combinatorial structure of the associated bridgeless subgraph problem.

6 Conclusions

Cell suppression is a widely-used methodology in Statistical Disclosure Control. In this paper we have introduced a branch-and-cut algorithm based on a new integer linear programming model for the 2-dimensional cell suppression problem. Extensive computational results are presented, showing that the new algorithm outperforms those based on previous models. We report the exact solution, on a PC, of instances with up to 250,000 cells and about 10,000 sensitive cells, by far the largest instances solved in the literature.

Future work can be devoted to the study of new classes of inequalities to be embedded within the branch-and-cut scheme, as well as to the analysis of the polyhedral structure of the bridgeless subgraph problem.

The extension of the ideas herein presented to the case of multidimensional tables will be investigated in a future paper.

7 Acknowledgment

Work partially supported by the European Union - ESPRIT project 20462-SDC on Statistical Disclosure Control. We thank Alberto Caprara for his help in code testing.

References

[1] D. Applegate, R. Bixby, V. Chvátal, W. Cook, "Finding cuts in the TSP", working paper, DIMACS, 1994.

[2] A. Caprara, M. Fischetti, "0-1/2 Chvátal-Gomory Cuts", *Mathematical Programming*, 74 (1996) 211–235.

[3] L.H. Cox, "Suppression Methodology and Statistical Disclosure Control", *Journal of the American Statistical Association*, 75 (1980) 377–385.

[4] L.H. Cox, "Network Models for Complementary Cell Suppression", *Journal of the American Statistical Association*, 90 (1995) 1453–1462.

[5] H.P. Crowder, E.L. Johnson, and M.W. Padberg, "Solving Large-Scale Zero-One Linear Programming Problems", *Operations Research*, 31 (1983) 803–834.

[6] F.D. Carvalho, N.P. Dellaert, M.S. Osório, "Statistical Disclosure in Two-Dimensional Tables: General Tables", *Journal of the American Statistical Association*, 89 (1994) 1547–1557.

[7] M. Fischetti, J.J. Salazar, "Models and Algorithms for the 2-Dimensional Cell Suppression Problem in Statistical Disclosure Control", Technical Report OR/97/5, DEI, University of Padova, May 1997.

[8] G.N. Frederickson, J. JáJá, "Approximation Algorithms for several Graph Augmentation Problems", *SIAM Journal on Computing*, 10 (1981) 270–283.

[9] J. Geurts, "Heuristics for Cell Suppression in Tables", working paper, Netherlands Central Bureau of Statistics, Voorburg, 1992.

[10] M. Grötschel, M.W. Padberg, "Polyhedral theory", in E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (Editors), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, 1985.

[11] D. Gusfield, "A Graph Theoretic Approach to Statistical Data Security", *SIAM Journal on Computing* 17 (1988) 552–571.

[12] M.Y. Kao, "Data Security Equals Graph Connectivity", *SIAM Journal on Discrete Mathematics*, 9 (1996) 87–100.

[13] J.P. Kelly, "Confidentiality Protection in Two and Three-Dimensional Tables", Ph.D dissertation, University of Maryland, College Park, Maryland, 1990.

[14] J.P. Kelly, B.L. Golden, and A.A. Assad, "Cell Suppression: Disclosure Protection for Sensitive Tabular Data", *Networks*, 22 (1992) 397–417.

[15] S. Martello, P. Toth, "Knapsack Problems: Algorithms and Computer Implementations" *John Wiley & Sons*, Chichester, 1990.

[16] G. Nemhauser, and L. Wolsey, "Integer and Combinatorial Optimization", *John Wiley & Sons*, Chichester, 1988.

[17] M. Padberg, G. Rinaldi, "A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems", *SIAM Reviews*, 33 (1991) 60–100.

[18] L.C.R.J. Willenborg, T. de Waal, "Statistical Disclosure Control in Practice", *Lecture Notes in Statistics 111*, *Springer*, New York, 1996.

name	NR	NC	p	pl	t_p	UB_h	t_h	UB_r	LB_r	t_r	z^*	s	n	t_t	ca	br	co	2e	pa	cm
DEMO1	3	4	9	2	0.0	0.00	0.0	0.00	0.00	0.2	5	1	1	0.18	2	0	0	0	0	0
DEMO2	6	7	11	7	0.0	0.00	0.0	0.00	0.00	0.1	218	4	1	0.15	16	4	2	2	3	0
DEMO3	4	4	4	8	0.0	35.00	0.0	0.00	0.00	0.2	320	5	1	0.24	25	8	13	0	17	0
DEMO4	6	6	5	10	0.0	0.00	0.0	0.00	0.00	0.1	88	4	1	0.15	26	2	6	3	3	0
DEMO5	5	4	4	7	0.0	15.79	0.0	0.00	0.00	0.1	19	5	1	0.16	14	0	0	0	8	0
DEMO6	4	4	4	8	0.0	45.45	0.0	0.00	0.00	0.1	33	4	1	0.15	8	0	0	0	0	0
DEMO7	8	6	14	16	0.0	6.12	0.0	0.00	0.00	0.3	27289	8	1	0.35	34	4	36	2	3	0
DEMO8	19	6	6	12	0.0	20.00	0.0	0.00	0.00	0.2	3000	5	1	0.22	24	6	3	0	9	0
DEMO9	3	3	3	6	0.0	16.77	0.0	0.00	0.00	0.1	161	3	1	0.13	6	0	0	0	0	0
DEMO10	5	3	4	8	0.0	0.00	0.0	0.00	0.00	0.1	6	2	1	0.16	8	0	0	0	0	0
FREQ1	19	6	22	18	0.0	0.00	0.0	0.00	0.00	0.1	20	6	1	0.16	18	0	0	0	0	0
FREQ2	19	6	40	12	0.0	0.00	0.0	0.00	0.00	0.1	23	6	1	0.13	12	0	0	0	0	0
MOPS2C	6	5	4	8	0.0	0.00	0.0	0.00	0.00	0.1	644	5	1	0.16	10	2	5	0	2	0
MOPS70	17	6	11	22	0.0	1.52	0.0	0.00	0.00	0.3	3749	5	1	0.31	60	14	30	0	4	0
MOPS80	17	6	32	17	0.0	0.00	0.0	0.00	0.00	0.2	9435	8	1	0.24	32	2	10	2	2	0
VB01	5	10	32	10	0.0	0.00	0.0	0.00	0.00	0.2	208245	6	1	0.24	24	0	11	0	0	0
VB02	2	10	9	3	0.0	0.00	0.0	0.00	0.00	0.1	12169	2	1	0.13	4	0	0	0	0	0
VB03	13	6	24	16	0.0	3.72	0.0	0.00	0.00	0.2	7334	9	1	0.26	28	4	10	1	1	0
VB04	20	6	17	34	0.0	3.12	0.0	0.00	0.00	0.2	32	11	1	0.18	34	0	0	0	0	0
VB05	17	5	9	18	0.0	0.11	0.0	0.00	0.00	0.2	1741	9	1	0.18	30	0	0	0	0	1
VB06	16	6	6	2	0.0	0.00	0.0	0.00	0.00	0.1	488	1	1	0.13	2	0	2	0	6	0
VB07	18	5	9	8	0.0	4.64	0.0	0.00	0.00	0.5	7956	4	1	0.53	70	74	34	2	7	0
VB09	13	6	14	24	0.0	5.84	0.0	0.00	0.00	0.4	7829	10	1	0.39	62	20	19	3	3	0
VB10	6	5	2	4	0.0	0.00	0.0	0.00	0.00	0.1	10	2	1	0.13	4	0	0	0	0	0
VB11	3	4	2	4	0.0	0.00	0.0	0.00	0.00	0.1	32	2	1	0.13	8	0	0	0	0	0
VB12	9	5	9	10	0.0	0.00	0.0	0.00	0.00	0.1	10	3	1	0.13	10	0	0	0	0	0
VB13	3	10	6	12	0.0	0.00	0.0	0.00	0.00	0.1	19	4	1	0.16	12	0	0	0	0	0
VB14	6	8	16	11	0.0	0.56	0.0	0.00	0.00	0.2	61096	5	1	0.18	22	0	4	0	0	0
VB15	12	5	10	20	0.0	0.00	0.0	0.00	0.00	0.1	61815	7	1	0.16	22	0	0	0	0	0
VB16	4	12	12	13	0.0	0.00	0.0	0.00	0.00	0.1	712879	4	1	0.15	16	0	0	0	0	0
VB17	12	3	7	14	0.0	0.00	0.0	0.00	0.00	0.1	63740	5	1	0.16	14	0	0	0	0	0
VB18	8	4	5	10	0.0	4.98	0.0	0.00	0.00	0.2	5057	5	1	0.18	14	4	3	0	0	0
VB19	13	6	24	10	0.0	0.00	0.0	0.00	0.00	0.1	36	5	1	0.15	10	0	0	0	0	0
VB20	6	6	1	2	0.0	0.00	0.0	0.00	0.00	0.1	41	3	1	0.13	6	0	0	0	0	0
VB21	3	5	4	8	0.0	0.00	0.0	0.00	0.00	0.1	5	2	1	0.13	8	0	0	0	0	0
VB22	4	4	4	0	0.0	0.00	0.0	0.00	0.00	0.0	0	0	0	0.01	0	0	0	0	0	0
VB23	3	3	2	4	0.0	0.00	0.0	0.00	0.00	0.1	3364	5	1	0.13	10	6	2	3	3	0
VB24	15	4	6	12	0.0	0.00	0.0	0.00	0.00	0.1	1247	5	1	0.15	18	0	6	0	2	0
VB25	2	10	4	8	0.0	0.00	0.0	0.00	0.00	0.1	8791	2	1	0.13	8	0	0	0	0	0
VB26	5	4	7	6	0.0	0.00	0.0	0.00	0.00	0.1	12119	4	1	0.15	18	4	6	0	0	0
VB27	5	3	1	2	0.0	0.00	0.0	0.00	0.00	0.1	6105	5	1	0.13	8	0	0	0	0	0
VB28	5	10	4	8	0.0	0.00	0.0	0.00	0.00	0.3	818	10	1	0.31	58	26	44	0	0	0
VB29	14	5	11	22	0.0	5.56	0.0	0.00	0.00	0.1	18	5	1	0.16	22	0	0	0	0	0
VB30	5	6	3	6	0.0	22.22	0.0	0.00	0.00	0.1	9	3	1	0.13	12	0	0	0	0	0
VB31	7	12	16	24	0.0	14.29	0.0	0.00	0.00	0.1	14	5	1	0.15	24	0	0	0	0	0
VB32	11	6	9	18	0.0	0.00	0.0	0.00	0.00	0.2	11895	8	1	0.18	26	0	2	0	1	0
VB33	7	6	8	5	0.0	0.00	0.0	0.00	0.00	0.1	1291	1	1	0.13	6	0	0	0	0	0
VB34	4	7	6	12	0.0	1.16	0.0	0.00	0.00	0.2	6903	6	1	0.18	24	6	8	1	1	0
VB35	4	4	1	2	0.0	0.00	0.0	0.00	0.00	0.1	11	3	1	0.16	6	0	0	0	0	0

Table 1: Statistics on instances from the SDC library.

NR	NC	p	pl	t_p	UB_h	t_h	UB_r	LB_r	t_r	z^*	s	n	t_t	ca	br	co	2e	pa	cm
7	7	5	10	0.0	0.00	0.0	0.00	0.00	0.2	136	6	1	0.18	32	10	6	1	4	0
7	7	6	9	0.0	0.00	0.0	0.00	0.00	0.1	197	9	1	0.15	30	2	0	0	0	0
12	6	5	10	0.0	0.00	0.0	0.00	0.75	0.2	97719132	6	2	0.37	40	12	10	3	11	0
13	8	10	10	0.0	0.60	0.0	0.00	0.00	0.3	5834714	7	1	0.29	36	18	15	2	3	0
41	8	13	18	0.0	14.29	0.0	0.00	0.00	0.3	175	11	1	0.33	146	22	0	0	0	0
41	11	17	28	0.0	0.00	0.0	0.00	0.00	0.2	99	13	1	0.24	42	0	6	0	1	0
182	60	2467	2	22.2	0.00	0.2	0.00	0.00	22.7	403	2	1	22.70	4	0	4	–	–	0
358	45	4923	54	171.9	0.00	0.8	0.00	0.00	174.8	256	27	1	174.84	54	0	0	–	–	0
358	45	4923	65	173.1	0.00	20.3	0.00	0.00	199.8	327	38	1	199.82	76	0	22	–	–	0

Table 2: Statistics on real-world instances provided by four statistical offices.

NR	NC	p	pl	t_p	UB_h	t_h	UB_r	LB_r	t_r	s	n	t_t	ca	br	co	2e	pa	cm
20	10	8.2	14.5	0.0	6.01	0.0	1.68	2.57	0.5	11.0	1.5	0.56	67.0	21.5	8.8	1.3	11.8	0.0
20	20	15.6	26.3	0.0	5.01	0.1	2.61	1.55	0.8	17.3	2.5	1.30	121.2	30.0	15.6	3.6	18.3	0.0
40	10	15.6	26.4	0.0	3.91	0.0	0.44	0.13	0.6	15.8	1.2	0.64	71.8	14.0	8.0	1.6	5.7	0.0
40	20	32.5	53.1	0.0	4.39	0.1	0.64	1.27	1.5	26.8	3.0	2.23	154.6	22.5	23.8	5.1	12.6	0.0
40	30	46.3	74.4	0.0	6.66	0.4	2.51	1.05	2.7	32.2	3.8	4.99	222.1	25.8	31.3	9.1	21.6	0.0
40	40	60.6	95.1	0.0	6.51	0.8	1.58	0.85	4.2	38.4	2.5	5.82	243.6	18.4	37.0	10.2	25.8	0.0
60	10	24.4	42.4	0.0	1.33	0.1	0.26	0.18	0.7	22.0	1.1	0.78	77.5	10.7	12.3	2.3	3.6	0.0
60	20	46.3	76.1	0.0	3.42	0.4	1.03	0.42	2.0	34.0	2.5	2.85	149.0	12.8	24.2	6.7	10.7	0.0
60	30	70.3	105.8	0.0	3.88	1.0	0.86	0.43	4.3	41.8	1.7	5.22	202.0	12.4	33.3	10.2	14.3	0.0
60	40	93.5	118.3	0.1	5.60	1.8	0.75	0.45	6.3	46.8	2.1	8.36	251.6	15.0	39.5	11.6	17.9	0.0
60	50	117.5	134.9	0.1	5.94	2.8	0.64	0.40	8.7	49.8	3.9	15.24	279.0	12.6	58.0	13.7	15.2	0.0
60	60	140.3	133.0	0.2	5.83	3.7	1.28	0.88	10.4	53.7	5.0	18.31	273.2	10.4	48.1	14.8	21.9	0.0
80	10	32.5	56.3	0.0	0.68	0.1	0.00	0.34	0.8	27.5	1.1	0.91	80.8	3.2	9.4	3.2	3.7	0.0
80	20	60.6	96.8	0.0	1.22	0.6	0.25	0.10	2.7	43.2	1.2	3.10	156.8	5.2	21.8	6.6	8.3	0.0
80	30	93.5	118.3	0.1	1.89	1.6	0.16	0.26	6.0	51.3	1.9	7.94	212.4	7.9	37.9	11.1	12.5	0.0
80	40	125.2	134.2	0.1	3.02	3.1	0.07	0.25	8.5	58.5	5.8	20.70	247.0	6.6	39.4	12.0	14.8	0.0
80	50	154.8	141.3	0.2	4.14	4.6	1.10	0.39	11.3	61.7	3.2	16.27	267.4	7.4	56.8	15.1	15.4	0.0
80	60	186.0	146.7	0.2	5.00	6.1	1.15	0.21	14.9	62.3	1.7	16.78	275.8	6.6	59.8	15.3	15.8	0.0
80	70	217.7	141.4	0.3	7.05	7.8	1.14	0.31	18.5	63.7	1.4	21.32	273.4	6.0	69.0	17.9	12.9	0.0
80	80	248.2	142.4	0.4	6.58	9.8	0.36	0.35	20.4	63.9	1.9	27.45	273.8	6.4	76.2	12.3	10.7	0.0
100	10	39.5	68.5	0.0	0.41	0.2	0.00	0.00	1.0	33.4	1.0	1.08	92.0	0.8	9.9	3.2	3.4	0.0
100	20	78.7	118.2	0.1	0.40	1.0	0.00	0.07	3.6	54.6	1.1	4.09	170.6	2.4	18.4	7.9	8.6	0.0
100	30	117.5	133.2	0.1	0.78	2.5	0.00	0.00	6.2	63.2	1.0	6.25	210.6	3.0	31.0	12.2	12.7	0.0
100	40	154.8	145.2	0.2	2.01	4.8	0.11	0.14	10.4	71.5	1.4	11.83	233.4	3.2	40.1	13.2	14.6	0.0
100	50	193.5	151.8	0.3	2.35	6.7	0.16	0.18	15.0	75.3	1.6	17.58	266.8	5.0	60.5	17.0	13.4	0.0
100	60	233.3	151.4	0.4	3.32	9.0	0.23	0.08	19.6	73.1	1.2	20.81	270.4	4.0	65.3	15.6	14.3	0.0
100	70	274.9	145.0	0.5	4.91	10.9	0.13	0.22	22.8	72.4	1.7	28.35	266.4	3.2	71.4	15.2	11.2	0.0
100	80	315.7	140.2	0.6	5.87	13.2	0.19	0.08	25.5	70.0	1.2	27.36	254.1	3.2	82.1	12.0	7.2	0.0
100	90	352.9	139.3	0.7	5.80	15.3	0.00	0.00	28.6	68.7	1.0	28.60	261.1	1.8	89.4	8.2	5.4	0.0
100	100	393.7	130.0	0.8	6.39	15.8	0.00	0.00	30.5	66.7	1.0	30.53	234.2	1.0	79.3	13.7	7.1	0.0
200	50	393.7	223.3	1.1	0.03	35.5	0.00	0.00	45.4	147.9	1.0	45.35	352.6	0.0	82.0	–	–	0.0
200	100	782.5	144.1	2.8	0.78	40.2	0.00	0.00	71.1	102.8	1.0	71.13	262.2	0.2	93.7	–	–	0.0
200	150	1185.7	78.0	5.6	3.48	23.8	0.00	0.00	52.8	56.0	1.0	52.85	147.6	0.0	93.1	–	–	0.0
200	200	1584.7	48.2	9.2	5.32	14.0	0.00	0.00	36.4	33.7	1.0	36.43	94.4	0.0	72.2	–	–	0.0
300	50	587.2	321.4	2.3	0.00	114.2	0.00	0.00	144.0	222.3	1.0	144.03	512.2	0.0	120.7	–	–	0.0
300	100	1185.7	199.9	6.4	0.02	110.0	0.00	0.00	160.5	153.5	1.0	160.49	361.8	0.0	120.8	–	–	0.0
300	150	1784.1	94.1	11.7	0.23	48.5	0.00	0.00	112.2	77.7	1.0	112.23	179.8	0.0	93.7	–	–	0.0
300	200	2383.1	42.5	20.3	0.67	18.3	0.00	0.00	52.5	35.5	1.0	52.49	83.4	0.0	66.5	–	–	0.0
300	250	2986.7	18.0	31.1	0.68	7.3	0.00	0.00	46.5	15.2	1.0	46.51	36.4	0.0	30.5	–	–	0.0
300	300	3587.6	9.3	44.2	0.15	4.0	0.00	0.00	54.4	8.3	1.0	54.39	18.4	0.0	17.2	–	–	0.0
400	50	782.5	424.7	4.1	0.00	262.4	0.00	0.00	317.9	295.0	1.0	317.97	675.4	0.0	157.0	–	–	0.0
400	100	1584.7	266.8	11.2	0.00	256.3	0.00	0.00	376.6	206.6	1.0	376.64	486.4	1.2	165.9	–	–	0.0
400	150	2383.1	121.3	22.0	0.00	105.6	0.00	0.00	267.6	101.6	1.0	267.65	233.6	0.0	122.5	–	–	0.0
400	200	3186.9	51.9	36.1	0.06	33.8	0.00	0.00	96.1	44.8	1.0	96.12	102.0	0.0	77.2	–	–	0.0
400	250	3986.8	17.1	54.2	0.00	9.1	0.00	0.00	72.7	14.8	1.0	72.69	34.2	0.0	28.6	–	–	0.0
400	300	4798.1	6.3	82.9	0.00	4.0	0.00	0.00	93.7	5.7	1.0	93.76	12.6	0.0	12.0	–	–	0.0
400	350	5607.3	2.0	117.2	3.85	2.1	0.00	0.00	124.2	1.7	0.8	124.24	4.0	0.0	3.6	–	–	0.0
400	400	6403.1	0.9	165.4	0.00	1.5	0.00	0.00	170.2	0.9	0.5	170.25	1.8	0.0	1.2	–	–	0.0
500	50	984.4	532.5	6.4	0.00	499.2	0.00	0.00	591.5	369.4	1.0	591.50	849.2	0.0	199.8	–	–	0.0
500	100	1984.4	332.3	17.9	0.00	505.2	0.00	0.00	633.0	258.6	1.0	633.01	606.6	0.0	208.5	–	–	0.0
500	150	2986.7	150.3	34.0	0.00	193.2	0.00	0.00	357.6	125.8	1.0	357.59	290.4	0.0	145.4	–	–	0.0
500	200	3986.8	64.0	55.8	0.00	59.6	0.00	0.00	176.3	55.1	1.0	176.33	125.4	0.0	92.3	–	–	0.0
500	250	5003.7	21.7	92.1	0.00	16.0	0.00	0.00	120.7	18.8	1.0	120.69	43.4	0.0	36.4	–	–	0.0
500	300	6003.9	6.4	137.3	0.00	5.5	0.00	0.00	151.1	5.8	1.0	151.10	12.6	0.0	11.4	–	–	0.0
500	350	6994.1	2.2	200.6	0.00	3.1	0.00	0.00	209.6	2.0	0.8	209.68	4.4	0.0	4.2	–	–	0.0
500	400	7994.6	0.8	279.1	0.00	0.9	0.00	0.00	283.7	0.8	0.4	283.73	1.6	0.0	1.2	–	–	0.0
500	450	8981.5	0.5	361.8	0.00	0.9	0.00	0.00	366.7	0.5	0.4	366.72	1.0	0.0	1.0	–	–	0.0
500	500	9989.2	0.3	456.2	0.00	1.1	0.00	0.00	460.5	0.3	0.3	460.56	0.6	0.0	0.6	–	–	0.0

Table 3: Statistics on Benedetti's data set.

NR	NC	p	pl	t_p	UB_h	t_h	UB_r	LB_r	t_r	s	n	t_t	ca	br	co	2e	pa	cm	#
10	10	18.2	23.2	0.0	3.55	0.0	1.00	0.35	0.6	8.3	1.5	0.70	46.2	7.0	26.5	0.8	2.2	0.0	10
20	10	36.6	20.3	0.0	2.94	0.0	0.73	0.40	0.6	9.1	1.3	0.72	34.5	1.8	29.2	1.2	1.8	0.0	10
20	20	68.1	15.2	0.0	9.40	0.1	3.75	0.96	1.2	9.6	3.9	1.99	27.4	2.1	82.0	0.8	1.9	0.0	10
30	10	52.9	24.9	0.0	1.64	0.1	0.35	0.64	0.9	12.5	1.3	1.04	38.6	2.6	48.6	1.0	1.0	0.0	10
30	20	101.6	11.4	0.0	3.64	0.1	0.60	0.31	1.1	8.3	10.3	4.59	16.8	0.2	51.7	0.6	0.8	0.0	10
30	30	149.2	6.2	0.0	15.85	0.1	2.24	0.43	2.2	6.3	35.6	10.93	8.1	0.0	81.8	0.8	0.8	0.0	10
40	10	69.3	30.8	0.0	1.15	0.2	0.92	0.10	1.5	18.0	1.3	1.60	47.6	0.0	44.9	2.0	2.0	0.0	10
40	20	134.0	12.9	0.0	3.22	0.2	2.07	0.56	1.7	11.1	32.6	13.09	21.7	0.4	127.2	1.1	1.2	0.0	10
40	30	198.6	5.8	0.1	16.40	0.1	5.00	1.08	1.9	5.7	68.2	23.04	7.4	0.0	57.9	0.9	0.9	0.0	10
40	40	265.2	3.0	0.2	5.79	0.1	2.28	0.32	2.1	6.4	189.8	75.20	3.4	0.0	78.0	0.0	0.0	0.0	10
50	10	87.1	35.3	0.0	1.20	0.3	0.83	0.35	1.9	20.9	2.8	2.59	53.6	2.0	65.4	2.0	2.4	0.0	10
50	20	167.0	14.5	0.1	8.75	0.2	5.95	1.12	2.1	12.2	14.8	7.62	24.7	0.2	99.6	1.2	1.3	0.0	10
50	30	248.8	5.8	0.1	11.04	0.1	3.43	0.44	2.3	5.4	24.6	10.29	6.9	0.0	57.1	0.6	0.6	0.0	10
50	40	329.4	4.6	0.2	13.72	0.3	9.62	2.01	5.3	8.8	189.3	117.56	5.0	0.0	121.3	0.0	0.0	0.0	10
50	50	411.4	3.6	0.3	16.86	0.3	5.83	0.68	5.1	7.7	36.4	32.61	3.6	0.0	106.9	0.1	0.1	0.0	10
60	10	104.1	42.0	0.1	1.44	0.5	0.18	0.14	2.2	25.3	1.7	2.72	63.2	0.4	52.8	2.6	2.6	0.0	10
60	20	199.8	17.0	0.1	9.01	0.4	2.06	0.12	4.1	13.5	22.7	13.85	29.2	0.0	160.8	1.7	1.9	0.0	10
60	30	296.0	8.2	0.2	17.63	0.2	6.01	1.38	4.2	9.7	109.6	53.30	9.5	0.0	81.7	0.8	0.8	0.0	10
60	40	398.9	2.9	0.3	6.45	0.2	4.37	0.02	3.8	7.3	198.6	141.58	2.8	0.0	98.4	0.0	0.0	0.0	9
60	50	490.3	2.6	0.5	9.11	0.3	5.43	1.51	6.7	7.4	402.6	337.27	2.6	0.0	123.9	0.0	0.0	0.0	9
60	60	591.5	2.8	0.7	16.86	0.4	9.51	1.71	12.1	7.6	68.0	111.03	2.8	0.0	107.5	0.0	0.0	0.0	8
60	60	591.5	2.8	0.6	16.86	0.4	9.51	1.71	12.0	7.6	68.0	110.94	2.8	0.0	107.5	0.0	0.0	0.0	8
70	10	120.8	49.5	0.0	0.23	0.8	0.11	0.04	2.7	28.5	2.7	3.99	73.6	0.0	47.1	3.0	3.0	0.0	10
70	20	235.6	21.2	0.2	3.00	0.6	1.34	0.31	5.2	18.1	169.7	92.27	33.9	0.0	127.9	2.2	2.4	0.0	10
70	30	344.4	8.4	0.3	3.08	0.3	1.29	0.14	5.2	8.4	395.7	342.62	8.8	0.0	97.7	0.8	0.8	0.0	9
70	40	463.6	4.3	0.4	13.60	0.3	5.11	0.52	5.7	8.6	93.3	77.54	4.2	0.0	99.3	0.0	0.0	0.0	9
70	50	568.9	3.3	0.6	15.31	0.4	2.60	0.08	8.1	8.6	268.3	240.47	3.2	0.0	136.9	0.0	0.0	0.0	9
70	60	685.6	3.5	0.8	15.38	0.6	8.05	1.34	12.8	8.6	438.6	540.89	3.5	0.0	146.7	0.0	0.0	0.0	10
70	70	793.7	3.6	1.1	14.82	0.7	1.96	0.13	19.4	10.8	202.0	292.89	3.6	0.0	159.2	0.0	0.0	0.0	9
80	10	136.8	55.0	0.1	1.36	1.2	0.48	0.30	3.8	32.3	2.1	4.83	82.6	0.0	60.6	3.3	3.3	0.0	10
80	20	266.5	22.4	0.2	4.18	0.8	3.65	0.57	4.8	20.0	107.6	78.42	36.0	0.0	141.2	2.6	2.7	0.0	10
80	30	399.8	8.9	0.4	11.25	0.5	5.60	1.54	8.9	10.4	385.2	343.85	11.9	0.0	127.9	1.0	1.0	0.0	9
80	40	532.9	4.9	0.6	11.41	0.4	7.35	0.79	6.9	10.2	189.9	191.97	4.8	0.0	101.4	0.1	0.1	0.0	8
80	50	645.6	3.6	0.8	17.26	0.5	6.74	0.61	7.9	9.2	61.6	131.03	3.4	0.0	130.9	0.0	0.0	0.0	8
80	60	779.2	4.1	1.0	16.10	0.8	5.32	2.52	20.0	10.4	204.1	320.65	4.0	0.0	150.1	0.0	0.0	0.0	9
80	70	904.2	2.9	1.4	23.95	0.6	6.76	0.76	15.0	7.0	177.2	247.77	2.9	0.0	110.0	0.0	0.0	0.0	10
80	80	1033.4	3.0	1.8	14.94	0.8	9.76	0.05	25.6	8.9	79.1	357.64	3.0	0.0	164.8	0.0	0.0	0.0	9
90	10	152.5	63.3	0.1	1.22	1.7	0.34	0.10	5.4	36.8	3.2	6.93	94.2	1.0	67.5	4.1	4.1	0.0	10
90	20	297.7	27.2	0.3	3.15	1.2	2.38	0.45	7.2	20.7	71.2	76.87	39.3	0.6	146.5	2.1	2.3	0.0	10
90	30	444.2	9.0	0.5	6.53	0.4	5.40	0.36	7.3	10.8	462.9	336.80	14.4	0.0	165.6	1.0	1.0	0.0	9
90	40	596.6	4.6	0.7	4.06	0.5	2.95	0.17	8.7	12.1	64.6	97.80	4.5	0.0	176.8	0.1	0.1	0.0	8
90	50	733.4	3.7	1.0	10.40	0.5	2.36	0.10	21.9	8.6	334.2	384.64	4.1	0.0	129.1	0.0	0.0	0.0	9
90	60	880.0	3.0	1.3	8.89	0.5	5.22	0.13	17.8	8.3	77.8	123.18	3.0	0.0	121.2	0.0	0.0	0.0	6
90	70	1013.8	2.9	1.7	12.33	0.7	6.51	2.03	21.0	6.8	110.8	434.01	3.0	0.0	132.1	0.0	0.0	0.0	9
90	80	1171.7	4.2	2.1	25.46	1.6	11.55	1.47	31.9	9.1	87.2	534.63	4.2	0.0	255.1	0.0	0.0	0.0	9
90	90	1310.1	2.5	2.9	12.80	1.1	5.47	0.56	39.5	7.6	142.9	600.70	2.5	0.0	185.6	0.0	0.0	0.0	8
100	10	171.0	69.1	0.2	0.70	2.5	0.38	0.27	7.0	41.2	2.5	9.01	103.2	0.0	79.6	5.0	5.0	0.0	10
100	20	332.4	29.6	0.3	1.26	1.6	1.16	0.29	8.4	25.7	110.5	104.31	50.8	0.0	195.2	3.4	3.6	0.0	10
100	30	495.6	10.9	0.6	12.86	0.6	3.61	0.45	10.2	12.6	79.6	79.68	13.0	0.0	147.1	1.4	1.4	0.0	8
100	40	655.2	5.3	0.9	6.05	0.5	2.50	0.91	11.1	11.0	486.9	484.35	5.1	0.0	130.7	0.1	0.1	0.0	9
100	50	814.9	3.3	1.2	23.05	0.5	3.88	0.00	10.4	7.6	241.3	298.69	3.1	0.0	131.7	0.0	0.0	0.0	7
100	60	967.2	3.0	1.7	8.91	0.8	4.35	0.86	19.7	8.5	349.2	595.09	3.0	0.0	100.0	0.0	0.0	0.0	8
100	70	1131.7	3.2	2.3	12.54	1.0	6.69	0.05	28.0	9.7	161.7	501.18	3.1	0.0	113.8	0.0	0.0	0.0	10
100	80	1298.3	3.3	2.7	19.33	1.4	3.66	0.15	40.5	12.3	164.9	610.56	3.3	0.0	187.3	0.0	0.0	0.0	7
100	90	1463.3	2.3	3.6	7.54	0.9	4.47	0.10	38.4	7.9	37.7	353.63	2.3	0.0	175.0	0.0	0.0	0.0	7
100	100	1619.2	3.7	4.1	13.52	1.9	8.87	0.56	82.7	9.1	143.4	866.63	3.7	0.0	220.0	0.0	0.0	0.0	9

Column # gives the number of instances successfully solved within the 3,600 second time-limit.

Table 4: Statistics on Kelly-Golden-Assad data set.

NR	NC	p	pl	t_p	UB_h	t_h	UB_r	LB_r	t_r	s	n	t_t	ca	br	co	2e	pa	cm
20	10	2.8	2.8	0.0	20.34	0.0	0.00	0.00	0.3	4.4	1.0	0.42	22.0	5.6	0.0	0.0	0.0	0.3
20	20	3.0	3.0	0.0	18.62	0.0	0.00	0.00	0.4	5.2	1.0	0.46	24.4	4.2	0.0	0.0	0.0	0.0
40	10	3.0	3.0	0.0	15.95	0.0	0.00	0.00	0.4	4.8	1.0	0.50	27.6	9.8	0.0	0.0	0.0	0.7
40	20	2.6	2.6	0.0	14.73	0.0	0.00	0.00	0.5	5.8	1.0	0.57	29.6	16.0	0.0	0.0	0.0	0.3
40	30	2.6	2.6	0.0	26.57	0.0	0.00	0.00	0.5	5.4	1.0	0.57	26.2	8.8	0.0	0.0	0.0	0.0
40	40	4.8	4.8	0.0	32.53	0.0	0.00	0.00	0.8	7.8	1.0	0.79	58.4	23.2	0.0	0.0	0.0	0.0
60	10	2.6	2.6	0.0	6.42	0.0	0.00	0.00	0.5	5.2	1.0	0.57	28.2	11.4	0.0	0.0	0.0	0.9
60	20	4.8	4.8	0.0	28.53	0.0	0.00	0.00	0.8	7.1	1.0	0.89	60.4	31.0	0.0	0.0	0.0	1.4
60	30	4.8	4.8	0.0	29.91	0.0	0.00	0.00	8.2	7.6	1.0	8.26	80.2	51.6	0.0	0.0	0.0	12.0
60	40	4.0	4.0	0.0	31.08	0.0	0.00	0.00	0.9	6.6	1.0	0.91	48.4	10.6	0.0	0.0	0.0	0.0
60	50	4.0	4.0	0.0	30.99	0.1	0.00	0.00	1.9	7.0	1.0	1.96	56.4	26.2	0.0	0.0	0.0	4.7
60	60	4.0	4.0	0.0	23.51	0.1	0.00	0.00	1.1	7.2	1.0	1.11	54.0	15.4	0.0	0.0	0.0	0.0
60	60	4.0	4.0	0.0	23.51	0.1	0.00	0.00	1.0	7.2	1.0	1.10	54.0	15.4	0.0	0.0	0.0	0.0
80	10	4.8	4.8	0.0	20.17	0.0	0.00	0.00	1.2	6.8	1.0	1.26	56.6	32.8	0.0	0.0	0.0	6.8
80	20	4.0	4.0	0.0	23.70	0.0	0.00	0.00	0.7	6.0	1.0	0.72	45.7	14.2	0.0	0.0	0.0	0.1
80	30	4.0	4.0	0.0	27.58	0.0	0.00	0.00	0.8	6.2	1.0	0.88	46.8	13.2	0.0	0.0	0.0	0.0
80	40	4.0	4.0	0.0	26.37	0.0	0.00	0.00	9.0	7.3	1.0	9.09	77.2	43.6	0.0	0.0	0.0	8.9
80	50	4.1	4.1	0.0	37.30	0.1	0.00	0.00	2.4	6.7	1.0	2.43	74.8	20.4	0.0	0.0	0.0	3.3
80	60	4.1	4.1	0.0	27.63	0.1	0.00	0.00	2.0	6.9	1.0	2.10	74.2	43.4	0.0	0.0	0.0	0.9
80	70	4.1	4.1	0.0	32.90	0.1	0.00	0.00	1.9	6.5	1.0	1.95	66.4	31.0	0.0	0.0	0.0	0.1
80	80	4.1	4.1	0.0	35.21	0.1	0.00	0.00	6.8	6.4	1.0	6.87	71.8	35.6	0.0	0.0	0.0	10.1
100	10	4.0	4.0	0.0	14.98	0.0	0.00	0.00	0.5	6.4	1.0	0.55	24.2	0.0	0.0	0.0	0.0	0.0
100	20	4.0	4.0	0.0	34.08	0.0	0.00	0.00	0.9	6.0	1.0	0.94	47.0	16.0	0.0	0.0	0.0	0.2
100	30	4.1	4.1	0.0	28.94	0.0	0.77	0.00	15.3	6.9	1.1	17.09	77.4	49.6	0.0	0.0	0.0	20.5
100	40	4.1	4.1	0.0	40.03	0.1	0.00	0.00	1.3	6.5	1.0	1.32	48.4	18.2	0.0	0.0	0.0	0.2
100	50	4.1	4.1	0.0	32.57	0.1	0.00	0.00	16.7	6.9	1.0	16.73	107.3	54.5	0.0	0.0	0.0	4.8
100	60	4.1	4.1	0.0	28.71	0.1	0.00	0.00	1.8	7.1	1.0	1.83	61.8	24.0	0.0	0.0	0.0	0.2
100	70	6.0	6.0	0.0	44.26	0.2	1.67	0.00	53.4	9.3	1.1	57.83	159.0	74.2	0.0	0.0	0.0	15.1
100	80	6.0	6.0	0.0	41.99	0.2	0.00	0.00	14.1	8.6	1.0	14.20	155.8	93.0	0.0	0.0	0.0	6.3
100	90	6.0	6.0	0.0	26.24	0.3	0.00	0.00	4.7	8.9	1.0	4.71	111.4	54.2	0.0	0.0	0.0	0.6
100	100	6.0	6.0	0.0	42.49	0.1	0.00	0.00	4.5	8.8	1.0	4.50	115.8	39.2	0.0	0.0	0.0	4.0
200	50	5.6	5.6	0.0	24.97	0.2	0.00	0.00	28.2	7.7	1.0	28.19	125.8	82.4	0.0	–	–	13.2
200	100	8.4	8.4	0.0	42.38	0.3	0.00	0.00	11.9	10.8	1.0	11.91	188.9	100.4	0.0	–	–	6.0
200	150	10.4	10.4	0.0	54.52	0.5	2.63	0.00	55.6	12.7	1.2	98.66	399.2	259.0	0.0	–	–	23.5
200	200	8.0	8.0	0.0	42.60	0.5	3.33	0.00	73.6	10.6	1.1	77.68	317.6	216.2	0.0	–	–	15.2
300	50	10.4	10.4	0.0	53.33	0.3	2.00	0.00	34.1	13.0	1.4	51.49	217.4	203.8	0.0	–	–	59.9
300	100	8.0	8.0	0.0	45.65	0.4	0.00	0.00	7.2	10.2	1.0	7.17	175.0	42.0	0.0	–	–	0.3
300	150	8.0	8.0	0.0	47.24	0.6	0.00	0.00	30.0	10.8	1.0	30.01	278.8	171.8	0.0	–	–	6.0
300	200	7.2	7.2	0.0	36.92	0.8	2.86	0.00	34.3	9.6	1.1	35.43	199.2	77.0	0.0	–	–	6.0
300	250	11.7	11.7	0.0	51.57	1.3	0.00	0.00	75.5	14.1	1.0	75.61	383.0	171.4	0.0	–	–	7.1
300	300	8.6	8.6	0.0	51.78	1.4	0.00	0.00	60.2	11.5	1.0	60.25	284.2	146.4	0.0	–	–	3.5
400	50	8.0	8.0	0.0	46.61	0.4	0.00	0.00	5.4	10.2	1.0	5.44	116.6	72.4	0.0	–	–	1.7
400	100	7.2	7.2	0.0	45.38	0.6	0.00	0.00	7.5	9.1	1.0	7.58	100.6	42.6	0.0	–	–	0.7
400	150	11.7	11.7	0.0	66.31	1.2	0.00	0.00	34.8	13.7	1.0	34.83	314.0	166.2	0.0	–	–	1.7
400	200	8.6	8.6	0.0	46.55	1.2	2.09	0.00	224.5	11.4	1.3	292.88	366.4	333.8	0.0	–	–	38.8
400	250	9.4	9.4	0.0	66.33	1.6	0.00	0.00	102.4	11.8	1.0	102.47	327.0	227.0	0.0	–	–	8.6
400	300	9.4	9.4	0.0	50.96	1.9	0.00	0.00	63.4	11.6	1.0	63.56	334.6	104.2	0.0	–	–	5.4
400	350	12.0	12.0	0.0	46.81	2.6	0.00	0.00	127.6	14.4	1.0	127.66	446.8	252.6	0.0	–	–	5.3
400	400	12.0	12.0	0.0	41.77	2.9	0.00	0.00	67.1	14.3	1.0	67.17	374.4	99.6	0.0	–	–	0.0
500	50	11.7	11.7	0.0	53.17	0.8	0.00	0.00	11.8	13.8	1.0	11.82	194.0	95.6	0.0	–	–	2.2
500	100	8.6	8.6	0.0	55.29	1.0	0.00	0.00	22.5	10.4	1.0	22.59	200.2	141.2	0.0	–	–	4.7
500	150	9.4	9.4	0.0	48.66	1.6	0.91	0.00	137.2	12.4	1.1	141.65	354.0	263.0	0.0	–	–	12.4
500	200	9.4	9.4	0.0	56.40	2.2	0.00	0.00	91.6	12.2	1.0	91.71	385.8	299.6	0.0	–	–	14.0
500	250	12.0	12.0	0.0	55.98	3.2	0.00	0.00	333.5	15.1	1.0	333.56	345.2	171.8	0.0	–	–	23.0
500	300	12.0	12.0	0.0	50.62	4.0	0.00	0.00	120.1	14.1	1.0	120.17	404.6	262.8	0.0	–	–	3.5
500	350	12.5	12.5	0.0	68.46	5.0	3.16	0.00	396.5	15.7	1.2	398.86	426.0	256.6	0.0	–	–	40.2
500	400	13.0	13.0	0.0	60.92	5.9	2.05	0.00	639.8	16.8	1.5	687.18	743.0	466.6	0.0	–	–	54.9
500	450	13.0	13.0	0.1	54.64	6.6	0.00	0.00	358.1	15.6	1.0	358.24	586.8	352.8	0.0	–	–	9.6
500	500	15.7	15.7	0.0	64.39	7.4	0.00	0.00	185.3	17.8	1.0	185.45	474.2	310.6	0.0	–	–	2.0

Table 5: Statistics on Carvalho-Dellaert-Osório data set: sparse instances.