Stabilized Benders methods for large-scale combinatorial optimization, with application to data privacy

Daniel Baena Jordi Castro Antonio Frangioni
Dept. of Stat. and O.R. Dip. di Informatica
Univ. Politècnica de Catalunya Univ. di Pisa
daniel.baena@upc.edu jordi.castro@upc.edu frangio@di.unipi.it

Research Report UPC-DEIO DR 2017-03
October 2017; updated December 2018

Stabilized Benders methods for large-scale combinatorial optimization, with application to data privacy

Daniel Baena[*] Jordi Castro[†] Antonio Frangioni[‡]

Abstract

The Cell Suppression Problem (CSP) is a very large Mixed-Integer Linear Problem arising in statistical disclosure control. However, CSP has the typical structure that allows application of the Benders decomposition, which is known to suffer from oscillation and slow convergence, compounded with the fact that the master problem is combinatorial. To overcome this drawback we present a stabilized Benders decomposition whose master is restricted to a neighborhood of successful candidates by local branching constraints, which are dynamically adjusted, and even dropped, during the iterations. Our experiments with synthetic and real-world instances with up to 24000 binary variables, 181M continuous variables and 367M constraints show that our approach is competitive with both the current state-of-the-art code for CSP, and the Benders implementation in CPLEX 12.7. In some instances, stabilized Benders provided a very good solution in less than one minute, while the other approaches found no feasible solution in one hour.

Keywords: Benders' decomposition; Mixed-Integer Linear Problems; stabilization; local branching; large-scale optimization; statistical tabular data protection; cell suppression problem

Mathematics Subject Classification: 90C06, 90C11, 90C90

1 Introduction

Nowadays, Mixed-Integer Linear Problems (MILP) are routinely used in real-world applications. However, for very large-scale and complex problems,

[*]Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya, Jordi Girona 1–3, 08034 Barcelona. `daniel.baena@upc.edu`

[†]Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya, Jordi Girona 1–3, 08034 Barcelona. `jordi.castro@upc.edu`

[‡]Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 3, 56127 Pisa `frangio@di.unipi.it`

current state-of-the-art solvers may not yield good solutions in a reasonable amount of time, and therefore specialized methods are required. One of these cases is that of the Cell Suppression Problem (CSP) from the statistical disclosure control field. CSP has relatively few (but still in the thousands) binary variables, but very many (millions) continuous ones. Furthermore, once the binary variables are fixed, the problem in the continuous ones decomposes into many independent subproblems. As such, CSP is the ideal candidate for *Benders decomposition*, a general procedure originally developed in [5] for problems whose variables can be partitioned into "complicating" and "easy" ones (being, respectively, the binary and continuous in the MILP setting). In short, Benders decomposition projects the problem onto the complicating variables, resulting in a *master problem* with convex nondifferentiable—actually, polyhedral—objective function and discrete variables. Local information about the function is obtained through (both optimality and feasibility) cuts which are generated by the solution of the *subproblem* in the easy variables. Most often, as in the CSP case, the subproblem actually decomposes in many smaller independent ones, which is clearly beneficial. For MILP, the master is often a binary optimization problem, and the subproblems are linear. This approach was extended in [24] to convex nonlinear optimization problems by the use of convex duality theory.

The interest in Benders decomposition is clear from the vast literature that has been developed around it (for instance, Google Scholar reports more than 19000 documents for "Benders algorithm"). This method has performed very well in many applications; to mention just a few: network design [4, 12], supply chain design [33], data privacy [11, 19, 20], very large facility location (with either linear [10] or quadratic costs [17]), stochastic optimization (where it is usually denoted as *L-shaped* method, see [6] and references therein), and unit commitment [30, 34]. However, it is well known that in some problems Benders method may exhibit poor performances due a large number of iterations. The main factors explaining such a bad behaviour are: (i) the cuts generated by the subproblem are not good (i.e., "deep") enough, and, as a result, the points computed by the master provide poor lower bounds; (ii) the master is a difficult binary problem, and it becomes tougher at each iteration due the new cuts added; (iii) the solutions of the master tend to wildly oscillate, from a good point (which can be close to optimality) to a much worse one. Some remedies have been suggested in the literature for the above drawbacks (a recent review on them can be found in [31]). For (i), methods producing deeper cuts were introduced as early as [29], and more recently in [22]. For (ii), as shown in [36] and [1] for, respectively, the continuous and integer case, master problems do not need to be solved at optimality to guarantee the convergence of Benders decomposition, thus saving some computational time at each iteration (even more: infeasible points for the master can also be used in the subproblem). Finally, for (iii) stabilization was achieved in [16] by the use of an in-out approach [4, 21];

and in [33] by the addition of trust-region constraints to the master. However, such trust-region was only used for the first iterations of the Benders algorithms, because, as stated in [31], otherwise convergence could not be ensured. In this work we consider a generalization of this approach, where the stabilization constraints are dynamically adjusted and dropped during all the Benders iterations, by the use of both local branching and *reverse* local branching constraints, which results in a convergent algorithm. This approach, developed for this work on CSP, was later inserted in the more general framework of [1] for Benders decomposition algorithms with possibly inexact solutions of the subproblems. However, that paper applied it to an entirely different problem (chance constrained stochastic optimization), and was mainly devoted to the theoretical analysis of the approaches. Hence, the results there were focused on providing very general conditions guaranteeing global convergence, with comparatively little effort in devising ones that are effective in practice. Instead, here we focus on the effectiveness of the approach, comparing it with state-of-the-art ones for the problem at hand. We mention in passing that alternative stabilization procedures have been developed for related (but different) column-generation approaches [3, 7, 23].

Local branching has previously been applied in other Benders decomposition approaches, such as the one of [32], but it was not used to stabilize the cutting-plane algorithm, which is instead the main aim of our proposal. In [32], the point provided by the master problem is used to start a local branching phase in the original formulation of the problem (not in the master), using a small radius for the local branching. The purpose of this local branching phase is to efficiently find alternative solutions around the point provided by the master problem. Benders subproblems are solved for all these points, thus obtaining a pool of new cuts for the Benders master problem. This strategy is expected to improve the lower and upper bounds provided by respectively the master problem and the subproblems, reducing the number of Benders iterations. In our approach local branching constraints are instead added to the Benders master problem (not to the original problem). And the motivation is not to increase the lower bound provided by the master, but to stabilize it around a region of "good points". Indeed, this way the lower bound computed by the master is no longer a global bound (unlike in [32]), but a local lower bound. This means that the overall Benders algorithm has to be substantially modified, whereas [32] uses a standard Benders approach with the only change that several subproblems are solved for a pool of points. In addition in our approach the radius of the local branching constraints added to the Benders master problem has to quickly increase for reasons of efficiency, unlike in [32], where a small radius has always to be used. Despite of the above major differences, it is worth to mention that both approaches have one common step: when the original problem with local branching constraints solved by [32] is infeasible, this information can be added to the Benders master to exclude part of the feasible region. We also apply a similar trick when the Benders master is infeasible due to the local branching constraints (we name the new added constraints *reverse* local branching constraints). As stated above, however, the radius used in our approach will in theory be greater than the one used in [32], so the *reverse* local branching constraint is expected to cut a larger portion of the feasible region, thus being more effective.

Local branching and Benders decomposition were also combined in [2], but, as stated by the authors, that approach is very different from the one of [32], and also ours. In the heuristic of [2] the objective function of the Benders master problem is replaced by the Hamming distance expression of the local branching constraint, and a certain improvement in the true objective function is imposed as a constraint. In addition, in that approach the radius of the local branching constraint plays no role, whereas is instrumental in our proposal.

Our main motivation for developing this stabilized Benders decomposition was the solution of the Cell Suppression Problem, which arises in the discipline of Statistical Disclosure Control (SDC). In short, SDC aims at avoiding that confidential information could be derived from data released by some entity while, at the same time, maintaining as much as possible the data utility (that is, modifying as less as possible the original data). More details about SDC can be found in the recent survey [9] and monographs [27, 35]. SDC is one of the main concerns of National Statistical Agencies, which must guarantee that no confidential individual information can be obtained from the released statistical outputs. The most widely applied SDC method for tabular data is probably *cell suppression*, to be described in detail in §4. This method, introduced in [28], can be formulated as a huge MILP problem, that easily reaches sizes of thousands of binary variables and millions of constraints and continuous variables. Although some fast heuristics have been developed [8], they are only valid for some classes of instances. The current state-of-the-art exact method is a cutting-plane algorithm which relies on Benders cuts (that is, a Benders decomposition algorithm), initially applied only to a particular type of tables [19], and later to general ones [20].

The outline of the paper is as follows. In §2 we recall the classical Benders algorithm. In §3 we present the stabilized Benders decomposition for general MILP problems. In §4 we apply and particularize stabilized Benders to the CSP. Then, in §5 the stabilized Benders approach is compared with the state-of-the-art code of [20], and also with the Benders implementation of CPLEX 12.7, showing that our stabilization technique provides better solutions with the same computational effort. Finally, some conclusions and some perspectives for further research are drawn in §6.

2 Benders decomposition

We consider Benders decomposition [5] applied to a MILP of the form

$$\min \left\{ d^\top y + c^\top x \ : \ Fy + Dx = b \ , \ y \in Y \ , \ x \geq 0 \right\} \qquad (P)$$

where $y \in \mathbb{R}^{nc}$ and $x \in \mathbb{R}^{ne}$ are, respectively, the complicating (binary/integer) and easy (continuous) variables, $d \in \mathbb{R}^{nc}$, $c \in \mathbb{R}^{ne}$, $F \in \mathbb{R}^{m \times nc}$ and $D \in \mathbb{R}^{m \times ne}$. The set Y entails the constraints that make the problem "hard"; in many cases, such as in ours, $Y = \{0,1\}^{nc}$. Benders approach is based on *reformulating* (P) as

$$\min_y \left\{ d^\top y + Q(y) \ : \ y \in Y \right\} \qquad (P')$$

where

$$Q(y) = \min_x \left\{ c^\top x \ : \ Dx = b - Fy \ , \ x \geq 0 \right\} \qquad (Q)$$

is the *value function* of the problem when the complicating y variables are considered as parameters in the right-hand side of the constraint. The function Q is convex and nondifferentiable—actually, polyhedral—and it is conveniently characterized via linear duality as

$$Q(y) = \max_u \left\{ u^\top (b - Fy) \ : \ D^\top u \leq c \ , \ u \in \mathbb{R}^m \right\} \ . \qquad (Q_D)$$

For a fixed y there are three possible outcomes:

1. (Q) is unbounded above, which immediately proves that (P) is; in the following we ignore this, implicitly assuming that (Q_D) is nonempty;
2. (Q) is infeasible, and therefore any LP solver determines an unbounded direction v of the dual polyhedron—such that $D^\top v \leq 0$, typically, an extreme ray—for which $v^\top (b - Fy) > 0$;
3. neither of the above, hence any LP solver would determine an optimal solution u—typically, an extreme point of the dual polyhedron—such that $Q(y) = u^\top (b - Fy)$.

Hence, we can (in principle) define the set $\mathcal{U}$ of extreme points u and the set $\mathcal{V}$ of extreme rays v of the dual polyhedron: introducing a single auxiliary variable θ, we can then reformulate (P) as

$$\begin{aligned} \min \ & d^\top y + \theta && && (1) \\ & \bar{u}^\top (b - Fy) \leq \theta && \bar{u} \in \mathcal{U} && (2) \\ & \bar{v}^\top (b - Fy) \leq 0 && \bar{v} \in \mathcal{V} && (3) \\ & y \in Y. && && (4) \end{aligned}$$

Problem (1)–(4) is impractical since $\mathcal{U}$ and $\mathcal{V}$ can be very large, but it is perfectly suitable for row generation. Indeed, one only has to define two

```
1: Initialize $\mathcal{I}$ and $\mathcal{J}$. Set the best upper bound $\rho = +\infty$.
2: Solve master problem obtaining $\hat{\theta}$ and $\hat{y}$.
3: Solve subproblem $(Q_D)$ using $y = \hat{y}$.
4: if $(Q_D)$ has finite optimal solution in vertex $\bar{u}$ then
5:     Update upper bound: $\rho \leftarrow \min\{\, \rho\, ,\, d^\top \hat{y} + \bar{u}^\top (b - F\hat{y})\, \}$.
6:     if $\hat{\theta} = \bar{u}^\top (b - Fy)$ then
7:         STOP. Optimal solution is $y^* = \hat{y}$ with $Q(y^*) = \hat{\theta}$ and total cost $\rho$.
8:     else
9:         constraint $\bar{u}^\top (b - Fy) \leq \theta$ is violated by $(\hat{\theta}, \hat{y})$: $\mathcal{I} \leftarrow \mathcal{I} \cup \{\bar{u}\}$
10:    end if
11: else
12:    $(Q_D)$ is unbounded along segment $\bar{u} + \lambda \bar{v}$, i.e.,
       constraint $\bar{v}^\top (b - Fy) \leq 0$ is violated by $(\hat{\theta}, \hat{y})$: $\mathcal{J} \leftarrow \mathcal{J} \cup \{\bar{v}\}$.
13:    Vertex may also be added: $\mathcal{I} \leftarrow \mathcal{I} \cup \{\bar{u}\}$.
14: end if
15: Go to step 2.
```

Figure 1: Benders decomposition algorithm

suitably small subsets $\mathcal{I} \subset \mathcal{U}$ and $\mathcal{J} \subset \mathcal{V}$ and define the relaxation of (1)–(4) where constraints (2) and (3) are only defined for $\bar{u} \in \mathcal{I}$ and $\bar{v} \in \mathcal{J}$, respectively. This is called the *Master Problem* (MP). The steps of Benders algorithm are summarized in Figure 1.

A simple initialization is to put $\mathcal{I} = \mathcal{J} = \emptyset$, in which case at the first iteration one can take $\hat{\theta} = -\infty$ and $\hat{y}$ any feasible point in Y. Convergence of Benders decomposition is always guaranteed in a finite number of iterations (at most $|\mathcal{U}| + |\mathcal{V}|$), but of course getting even close to such a bound would be disastrous. Unfortunately, in practice the number of required iterations may actually be excessive due to, among other causes, instability issues. To overcome this drawback, we describe in next Section a stabilized Benders decomposition approach.

3 Stabilizing Benders decomposition through local branching constraints

One of the main causes for the slow convergence of Benders decomposition is the generation of weak cuts as a result of obtaining "bad" points $\hat{y}$ out of the MP, an effect that is well-known in cutting-plane methods when Y is a convex set [26]. The idea behind the stabilized Benders decomposition is to search new solutions $\hat{y}$ in a neighbourhood of a good *stability center* point, such as the point where the best (estimated) function value so far has been obtained. The typical benefit is an expected reduction of the number of iterations, and therefore of the total computational time, because the iterates accrue information around the stability center, constructing an "accurate" model of the objective function, which then effectively drives the search towards even better iterates [3, 7, 23]. When Y is a combinatorial set, the cost of the MP can be significant, in that it is a combinatorial problem. Stabilizing by restricting the feasible region is then attractive, as a MP with a smaller feasible region may be easier to solve.

In particular when y are binary variables, the stabilization can be achieved by simply adding linear constraints to the MP which impose that the Hamming distance of the new iterate to the stability center $\bar{y}$—which is not necessarily feasible—to be at most $\kappa \geq 1$. That is, in order to define a classical *trust region* of radius κ (which can be either a constant or dynamically updated at certain iterations) around the stability center, one can use a classical *local branching constraint* [18] which limits the "switching" of binary variables to at most κ components:

$$\Delta(y, \bar{y}) := \textstyle\sum_{j\,:\,\bar{y}_j=1}(1 - y_j) + \sum_{j\,:\,\bar{y}_j=0} y_j \leq \kappa \ . \tag{5}$$

This sort of stabilization was applied in [33], but using a constant radius, and only for the very first iterations of Benders decomposition. The nice aspect of (5) is that its complement, defining the set of points that have distance *larger* than κ from $\bar{y}$, is also a linear constraint: $\Delta(y, \bar{y}) \geq \kappa + 1$ (called a *reverse local branching constraint*). Indeed, local branching and reverse or local branching constraints can be used as a branching criterion within an enumerative scheme. That is, in this setting excluding regions around $\bar{y}$ from the feasible region of the MP is as easy as setting the trust region, unlike in the convex case where the trust region is typically a convex constraint, so its complement would be a reverse convex constraint, making the MP much harder to solve. In fact, (5) is a special case of *no-good cuts* [13], which become much more complex when the original problem is not a binary one. In our case, excluding regions comes with comparatively little cost; therefore, we introduce a set $\mathcal{R}$ of pairs $(\bar{y}', \kappa')$ denoting regions excluded by reverse local branching constraints, so as to define the Stabilized Master Problem (SMP) as

$$\begin{aligned}
\min\ & d^\top y + \theta && && (1)\\
& (2)\ ,\ (3)\ ,\ (4) && && \\
& \Delta(y, \bar{y}) \leq \kappa && && (6)\\
& \Delta(y, \bar{y}') \geq \kappa' + 1 && (\bar{y}', \kappa') \in \mathcal{R} && (7)
\end{aligned}$$

With that, it is easy to define the *stabilized Benders decomposition* framework shown in Figure 2. Note that, if the feasible region of the master problem is significantly reduced, the local branching constraints should reasonably make it easier to solve. This is in particular true because local branching constraints seem to significantly increase the "relaxation grip" of the formulation, i.e., the number of variables that are integer in the solution of the continuous relaxation, as discussed, e.g., in [2].

In the initialization phase, it is convenient in our application to find an initial solution $(\tilde{x}, \tilde{y})$ of the original problem (P) via a primal heuristic. This can be used to initialize $\rho \leftarrow c^\top \tilde{x} + d^\top \tilde{y}$ (assuming $(\tilde{x}, \tilde{y})$ is feasible, otherwise $\rho = +\infty$), and the stability center $\bar{y} \leftarrow \tilde{y}$. Note that $\tilde{y}$ need not necessarily be feasible, hence this can always be done even if the heuristic fails. At each iteration we solve the SMP to obtain a new solution $\hat{y}$ and a lower bound $\hat{\theta} \leq Q(\hat{y})$; note, however, that this lower bound is only local to the trust region. If (Q_D) has finite optimal solution $\bar{u}$ and $\hat{\theta} = \bar{u}^\top (b - F\hat{y}) = Q(\hat{y})$ we know that there is no better solution in this trust region and we update the stability center to the current $\hat{y}$. A reverse local branching constraint for the previous stability center $\bar{y}$ is also added to avoid the algorithm could select it again, possibly getting stuck. Note that in stabilized methods it is possible to move the stability center as soon as we find a better function value, i.e., whenever ρ decreases (“enough”), but we use a more conservative rule and we move it only when we find the optimum within the current trust region, as this allows us to add the reverse local branching constraint at step 19.

Also, we found it computationally convenient to introduce a specific modification: each time local optimality is reached, we immediately drop all the stabilization constraints, i.e, we solve the ordinary un-stabilized MP rather than the SMP, save for the reverse local branching constraints. This gives us a valid global lower bound (step 15) outside of the regions excluded by $\mathcal{R}$, in which we know that no better solution lies: if this lower bound equals the upper bound, we have reached an optimal solution and we can stop (step 17). If, instead $\hat{\theta} < Q(\hat{y})$ then an optimality cut is added to the SMP. If (Q_D) is unbounded, i.e., $Q(\hat{y}) = \infty$, then a feasibility cut is added to master. An occurrence that is specific of our treatment, as it cannot happen in the non-stabilized approach (save if (P) itself is unfeasible) is that the SMP can be empty: feasibility cuts have proven all the solutions in the current trust region to be unfeasible. In this case, we have to expand the trust region (increase κ). By adding the corresponding reverse local branching constraint we ensure that the master problem will no longer consider the previous trust region; hopefully, this makes the SMP easier to solve by reducing its feasible region. If the trust region can no longer be expanded, then ρ is a global optimum: there is no feasible solution outside the regions excluded by the reverse local branching constraints, and these have been completely explored. Note that these regions may actually be empty themselves ($\rho = \infty$), which proves that (P) is.

It is important to remark that one of the most important steps of the algorithm is the updating of κ (line 8). Different rules can be devised depending on the particular problem at hand. In Section 5 we will detail the particular rule used for the cell suppression problem.

It is easy to prove that the algorithm of Figure 2 solves the problem in a

```
1.  Initialize $\mathcal{I}$ and $\mathcal{J}$, the best upper bound $\rho$, the stability center $\bar{y}$, $\kappa \geq 1$,
    and set $\mathcal{R} \leftarrow \emptyset$.
2.  Solve the Stabilized Master Problem.
3.  **if** the SMP is infeasible **then**
4.    **if** $\kappa \geq nc$ **then**
5.      STOP. $\rho$ is the optimal value of $(P)$.
6.    **end if**
7.    Add reverse local branching constraint $(\bar{y}, \kappa)$ to $\mathcal{R}$
8.    Choose a new $\kappa \in \{\kappa + 1, \ldots, nc\}$ for the trust region constraint.
9.  **else**
10.   Let $(\hat{\theta}, \hat{y})$ be the solution of the SMP.
11.   Solve subproblem $(Q_D)$ with $y = \hat{y}$.
12.   **if** $(Q_D)$ is feasible with optimal solution $\bar{u}$ **then**
13.     Update upper bound: $\rho \leftarrow \min\{\, \rho \,,\, d^\top \hat{y} + \bar{u}^\top (b - F\hat{y}) \,\}$
14.     **if** $\hat{\theta} = \bar{u}^\top (b - F\hat{y})$ **then**
15.       Solve the SMP without the trust region constraint (i.e., $\kappa \leftarrow nc$),
          let $\underline{\rho}$ be its optimal value (a valid global lower bound).
16.       **if** $\rho = \underline{\rho}$ **then**
17.         STOP. $\rho$ is the optimal value of $(P)$.
18.       **end if**
19.       Add reverse local branching constraint $(\bar{y}, \kappa)$ to $\mathcal{R}$.
20.       Change the stability center in the trust region constraint:$\bar{y} \leftarrow \hat{y}$.
21.       Optionally, reset $\kappa \geq 1$.
22.     **else**
23.       Constraint $\bar{u}^\top (b - Fy) \leq \theta$ is violated by $(\hat{\theta}, \hat{y})$: $\mathcal{I} \leftarrow \mathcal{I} \cup \{\bar{u}\}$
24.     **end if**
25.   **else**
26.     $(Q_D)$ is unbounded along segment $\bar{u} + \lambda \bar{v}$, i.e.,
        constraint $\bar{v}^\top (b - Fy) \leq 0$ is violated by $(\hat{\theta}, \hat{y})$: $\mathcal{J} \leftarrow \mathcal{J} \cup \{\bar{v}\}$.
27.     Vertex may also be added: $\mathcal{I} \leftarrow \mathcal{I} \cup \{\bar{u}\}$.
28.   **end if**
29. **end if**
30. GOTO line 2.
```

Figure 2: The stabilized Benders method through local branching constraints

finite number of iterations:

Theorem 1. *The stabilized Benders decomposition algorithm of Figure 2 solves (P) in a finite number of iterations.*

Proof. The algorithm only stops when it has determined an optimal solution, and it has proven it to be such. At each iteration of the algorithm of Figure 2 one of the following four actions is performed: (i) an optimality cut is added to the SMP; (ii) a feasibility cut is added to the SMP; (iii) the stability center is changed; (iv) the radius of the trust region is increased. The number of optimality and feasibility cuts is finite, and they are not repeated as in standard Benders decomposition. The number of stability centers is finite (since Y is a combinatorial bounded set), and they are not repeated because of the new optimality and feasibility cuts, and reverse local branching constraints added to the master. The different values taken by the radius of the trust region are also finite, and never repeated for a given stability center, since it is a monotonically increasing sequence bounded by nc (line 8 of the algorithm). (Even more: the values of κ are never repeated for all the stability centers if the optional reset of line 21 is not applied). Therefore, the algorithm will eventually stop with an optimal solution. $\square$

Of course, the algorithm can be easily extended to produce ε-optimal solutions for every fixed $\varepsilon > 0$, and a number of other practical improvements is also possible such as judicious removal of cuts from $\mathcal{I}$ and $\mathcal{J}$. The interested reader can refer to [1] for a comprehensive theoretical analysis of stabilized Benders decompositions algorithms.

4 Application to data privacy: the cell suppression problem

National statistical agencies (NSAs) work with two types of data: microdata and tabular data. Microdata files contain records of individuals or respondents (persons or enterprises) with attributes. For instance, a national census might collect attributes such as age, address, salary, etc. Tabular data is obtained by crossing two or more categorical variables of a microdata file. For each cell, the table may report either the number of individuals that fall into that cell (frequency tables) or information about another variable (magnitude tables). Tables contain summarized data from microdata files and are the most common form for disseminating information of NSAs. Although tabular data may be thought to be automatically anonymized since it reports aggregated information for several respondents, there is a disclosure risk of individual information. Figure 3 (from [8]) illustrates this situation with a simple case. The left table (a) reports the salary of individuals by age (row variable) and town (column variable), while table (b) provides the number

		t_1	t_2	
⋮	...	...	...	...
51–55	...	38000€	40000€	...
56–60	...	39000€	42000€	...
⋮	...	...	...	...

(a)

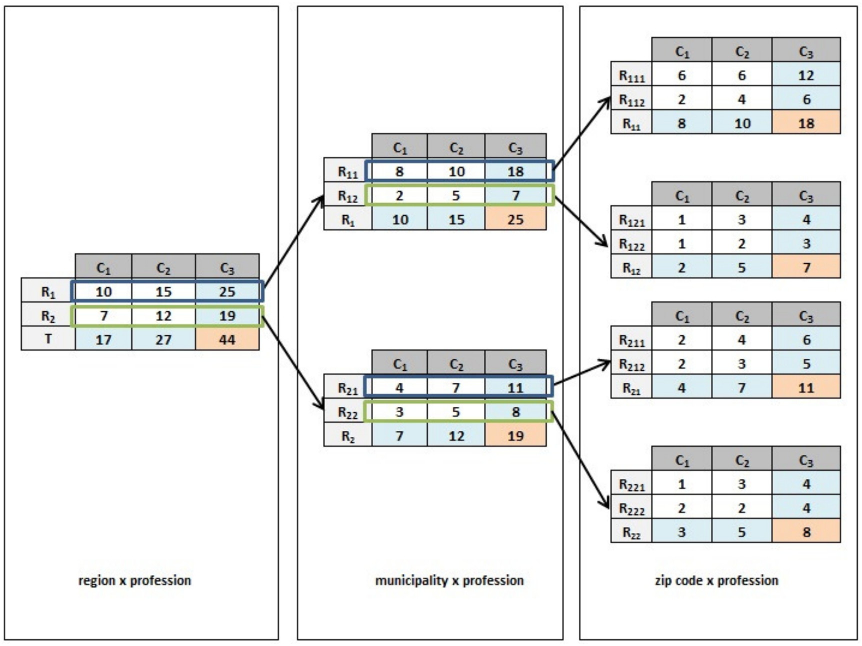
		t_1	t_2	
⋮	...	...	...	...
51–55	...	20	1 or 2	...
56–60	...	30	35	...
⋮	...	...	...	...

(b)

Figure 3: Example of disclosure in tabular data. (a) Salary per age and town. (b) Number of individuals per age and town. If there is only one individual in town t_2 and age interval 51–55, then any external attacker knows the salary of this single person is 40000€. For two individuals, any of them can deduce the salary of the other, becoming an internal attacker.

of individuals. If there was only one individual of age 51–55 in town t_2, then any external attacker would know the confidential salary of this person. For two or more individuals, any of them (or may be a coalition of several respondents) could either disclose the other's salary or compute a good estimation of the rest of respondents. Cells that require protection (such as that of the example) are named *sensitive*, *unsafe*, *primary* or *confidential* cells.

Tables can be classified according to different criteria [9]. For our purposes, the most important one is the table structure, since some protection methods can only be applied to particular table structures. According to their structure, tables may be classified as single k-dimensional, hierarchical or linked tables. A single k-dimensional table is obtained by crossing k categorical variables. For instance, the tables of Figure 3 are two-dimensional (2D) tables. A hierarchical table is a set of tables obtained by crossing some categorical variables, and some of them have a hierarchical structure; that is, some tables are subtables of other tables. Hierarchical tables are relevant for NSAs, since a significant percentage of the tables they release belong to this category. The simplest hierarchical table is known as *two-dimensional tables with one hierarchical variable*, or, shortly, 1H2D tables. These tables are obtained by crossing a particular categorical variable with a set of, say, h categorical variables that have a hierarchical relation; this results in a set of h two-dimensional tables with some common cells. For instance, Figure 4 (from [9]) illustrates a particular 1H2D table. The left subtable shows number of respondents for "region"×"profession"; the middle subtables is a "zoom in" of regions, providing the number of respondents in municipalities of each region; finally the right subtables details the ZIP codes of municipalities. A linked table is any set of tables obtained from the same microdata file. Note that, hierarchical and k-dimensional tables are particular cases of linked tables. Marginal cells of any table contain the total sum of a row or



	C_1	C_2	C_3
R_1	10	15	25
R_2	7	12	19
T	17	27	44

	C_1	C_2	C_3
R_{11}	8	10	18
R_{12}	2	5	7
R_1	10	15	25

	C_1	C_2	C_3
R_{21}	4	7	11
R_{22}	3	5	8
R_2	7	12	19

	C_1	C_2	C_3
R_{111}	6	6	12
R_{112}	2	4	6
R_{11}	8	10	18

	C_1	C_2	C_3
R_{121}	1	3	4
R_{122}	1	2	3
R_{12}	2	5	7

	C_1	C_2	C_3
R_{211}	2	4	6
R_{212}	2	3	5
R_{21}	4	7	11

	C_1	C_2	C_3
R_{221}	1	3	4
R_{222}	2	2	4
R_{22}	3	5	8

Figure 4: Example of 1H2D table made of different subtables: "region"×"profession", "municipality"×"profession" and "zip code"×"profession".

column.

The Cell Suppression Problem (CSP) [28, 19, 20, 8] is a statistical disclosure method based on removing the minimum amount of information (measured as a function of the number of cells or cell values) that makes the resulting table safe. Let us consider a table with a set $\mathcal{N}$ of n cells, of values $a = (a_i)_{i \in \mathcal{N}}$, that satisfy m linear relations $Aa = b$, $A \in \mathbb{R}^{m \times n}$, and lower and upper bounds $l \leq a \leq u$. Given a set $\mathcal{S} \subseteq \mathcal{N}$ of sensitive cells (that will be suppressed), CSP attempts to find a set $\mathcal{K} \subseteq \mathcal{N}$ of additional cells to be suppressed (named *complementary* cells) which guarantee that the minimum $\underline{a}_s$ and maximum $\overline{a}_s$ values that can be recomputed for each cell $s \in \mathcal{S}$ are out of a certain protection interval, that is,

$$\underline{a}_s \leq a_s - lpl_s \text{ and } \overline{a}_s \geq a_s + upl_s \qquad s \in \mathcal{S}, \tag{8}$$

where $lpl \in \mathbb{R}^{|\mathcal{S}|}$ and $upl \in \mathbb{R}^{|\mathcal{S}|}$ are the lower and upper protection levels of sensitive cells. After suppression, the minimum and maximum values of each sensitive cell $s \in \mathcal{S}$ can be obtained by the solution of the two following linear optimization problems:

$$\begin{array}{rlll} \underline{a}_s = \min\limits_x & x_s & & \\ \text{s. to} & Ax = b & & \\ & l_i \le x_i \le u_i & i \in \mathcal{S} \cup \mathcal{K} & \\ & x_i = a_i & i \notin \mathcal{S} \cup \mathcal{K} & \end{array} \qquad \begin{array}{rll} \overline{a}_s = \max\limits_x & x_s & \\ \text{s. to} & Ax = b & \\ & l_i \le x_i \le u_i & i \in \mathcal{S} \cup \mathcal{K} \\ & x_i = a_i & i \notin \mathcal{S} \cup \mathcal{K}\ . \end{array} \tag{9}$$

The monolithic model for CSP, originally formulated in [28], considers two sets of variables: (i) $y \in \{0,1\}^n$, such that y_i, $i \in \mathcal{N}$, is 1 if cell i has to be suppressed, 0 otherwise; (ii) for each primary cell $s \in \mathcal{S}$, two auxiliary vectors $x^{l,s} \in \mathbb{R}^n$ and $x^{u,s} \in \mathbb{R}^n$, which represent cell deviations (positive or negative) from the original a values. The resulting model is:

$$\begin{array}{llllll} \min\limits_{y,x^l,x^u} & \sum_{i\in\mathcal{N}} w_i y_i & & & & \\ \text{s. to} & & Ax^{l,s} & = 0 & & \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array}\right\} s \in \mathcal{S} \\ & (l_i - a_i) y_i \le & x_i^{l,s} & \le (u_i - a_i) y_i & i \in \mathcal{N} & \\ & & x_s^{l,s} & \le -lpl_s & & \\ & & Ax^{u,s} & = 0 & & \\ & (l_i - a_i) y_i \le & x_i^{u,s} & \le (u_i - a_i) y_i & i \in \mathcal{N} & \\ & & x_s^{u,s} & \ge upl_s & & \\ & y_i \in \{0,1\} & & & & i \in \mathcal{N}\ . \end{array} \tag{10}$$

The inequality constraints of (10) with both right- and left-hand sides impose bounds on $x_i^{l,s}$ and $x_i^{u,s}$ when $y_i = 1$, and prevent deviations in non-suppressed cells (i.e., $y_i = 0$). Clearly, the constraints of (10) guarantee that the solutions of the linear programs (9) will satisfy (8).

The formulation (10) of CSP is a very large MILP problem with n binary variables, $2n|\mathcal{S}|$ continuous variables and $2(m+2n)|\mathcal{S}|$ constraints. For instance, for a table of 4000 cells, 1000 sensitive cells, and 2500 linear relations, the formulation has 8000000 continuous variables, 4000 binary variables, and 21000000 constraints. Solving it with general-purpose MILP solvers, even if state-of-the-art, is impractical even for tables of moderate size. However, the structure of (10) is ideal for applying the (stabilized) Benders decomposition algorithms of the previous sections. It is worth noting that, unlike ours, which is based in the algorithm of Figure 2, the approach of [19, 20] embedded Benders cuts within a branch-and-cut algorithm.

The Stabilized Master Problem for CSP can be written as

$$\begin{array}{lll} \min_y & \sum_{i\in\mathcal{N}} w_i y_i & \\ \text{s. to} & y_s = 1 & s \in \mathcal{S} \\ & y_i \in \{0,1\} & i \in \mathcal{N} \\ & \bar{v}^\top y \ge \bar{\beta} & (\bar{v}, \bar{\beta}) \in \mathcal{J} \\ & (6)\ ,\ (7) & \end{array} \tag{11}$$

where $\bar{v}$ and $\bar{\beta}$ are the left and right hand sides of the feasibility cuts. In our case, for obvious reasons we call them *protection* or *infeasibility* cuts. Note that our problem only involves infeasibility cuts, since variables $x^{l,s}$ and $x^{u,s}$ of (10) do not appear in the objective function. We also remark that sensitive cells are always suppressed even for $\mathcal{J} = \emptyset$.

In order to guarantee that deviations $x^{l,s}$ and $x^{u,s}$ satisfy the first group of constraints of (10) and that, therefore, the suppression pattern y_i, $i \in \mathcal{N}$ is safe, we solve a Benders subproblem for each primary cell $s \in \mathcal{S}$. Since variables $x^{l,s}$ and $x^{u,s}$ have no cost in (10), the subproblems can be reduced to a feasibility problem. Dropping the index $s \in \mathcal{S}$ to simplify the notation, the subproblems for lower and upper protection of a sensitive cell are, respectively,

$$\begin{array}{rll} \min_x & 0 & \\ \text{s. to} & Ax = 0 & \\ & x_i \geq (l_i - a_i)y_i & i \in \mathcal{N} \\ & x_i \leq (u_i - a_i)y_i & i \in \mathcal{N} \\ & x_s \leq -lpl_s & \end{array} \tag{12}$$

$$\begin{array}{rll} \max_x & 0 & \\ \text{s. to} & Ax = 0 & \\ & x_i \geq (l_i - a_i)y_i & i \in \mathcal{N} \\ & x_i \leq (u_i - a_i)y_i & i \in \mathcal{N} \\ & x_s \geq upl_s \; . & \end{array} \tag{13}$$

Alternatively, (12) and (13) can be formulated as

$$\begin{array}{rlll} -lpl_s \geq \min_x & x_s & & \\ \text{s. to} & Ax = 0 & & [\,\lambda\,] \\ & x_i \geq (l_i - a_i)y_i & i \in \mathcal{N} & [\,\mu^l\,] \\ & x_i \leq (u_i - a_i)y_i & i \in \mathcal{N} & [\,\mu^u\,] \end{array} \tag{14}$$

$$\begin{array}{rlll} upl_s \leq \max_x & x_s & & \\ \text{s. to} & Ax = 0 & & [\,\lambda\,] \\ & x_i \geq (l_i - a_i)y_i & i \in \mathcal{N} & [\,\mu^l\,] \\ & x_i \leq (u_i - a_i)y_i & i \in \mathcal{N} & [\,\mu^u\,] \; , \end{array} \tag{15}$$

where for future reference we indicate the Lagrange multipliers (dual variables) $\lambda \in \mathbb{R}^m$, $\mu^l \in \mathbb{R}^n$ and $\mu^u \in \mathbb{R}^n$ of each group of constraints. Problems (14) and (15) clearly have always an optimal solution, as $x = 0$ (no deviation) is feasible, and they are not unbounded since $-\infty < l_s - a_s \leq x_s \leq u_s - a_s < \infty$. The linear dual of (14) is

$$\begin{array}{rl} \max_{\lambda,\mu} & \sum_{i \in \mathcal{N}} [\,(l_i - a_i)\mu_i^l - (u_i - a_i)\mu_i^u\,]y_i \\ \text{s. to} & A^\top \lambda + \mu^l - \mu^u = e_s \\ & \mu^l \geq 0 \;\; , \;\; \mu^u \geq 0 \; , \end{array} \tag{16}$$

where e_s is the s-th column of the identity matrix. The lower/upper protection level of primary cell s is satisfied if

$$\begin{aligned} -lpl_s &\geq \textstyle\sum_{i\in\mathcal{N}}[\ \ (l_i - a_i)\mu_i^l - (u_i - a_i)\mu_i^u\,]y_i && (17)\\ upl_s &\leq \textstyle\sum_{i\in\mathcal{N}}[-(l_i - a_i)\mu_i^l + (u_i - a_i)\mu_i^u\,]y_i\ . && (18)\end{aligned}$$

If (17) and (18) hold for all $s \in \mathcal{S}$, then the suppression pattern y guarantees lower and upper protection levels. If, for some $s \in \mathcal{S}$, one among (17) and (18) is not satisfied, then the corresponding cut is added to $\mathcal{J}$.

(Stabilized) Benders decomposition applied to CSP iteratively solves the SMP (11) in variables y and provides a suppression pattern. The protection is checked by solving $2|\mathcal{S}|$ subproblems, one lower and one upper per primary cell. If all ones are protected, then the suppression pattern is optimal; otherwise, one or more feasibility cuts are added to the SMP, which is solved again.

As shown by the next proposition, it is equivalent to either use (12)–(13) (the standard Benders subproblems) or (14)–(15) to compute the feasibility cuts:

Proposition 1. *The Benders feasibility cuts provided by subproblems* (12)–(13) *are equivalent to those obtained with* (14)–(15).

Proof. We only prove the result for the lower protection case, as the upper protection one is analogous. The linear dual of (12) has variables $\tilde{\lambda} \in \mathbb{R}^m$, $\tilde{\mu}^l \in \mathbb{R}^n$, $\tilde{\mu}^u \in \mathbb{R}^n$, and $\tilde{\mu}_s \in \mathbb{R}$, and boils down to

$$\begin{array}{rl} \max_{\tilde{\lambda},\tilde{\mu}} & lpl_s\tilde{\mu}_s + \sum_{i\in\mathcal{N}}[\,(l_i - a_i)\tilde{\mu}_i^l - (u_i - a_i)\tilde{\mu}_i^u\,]y_i \\ \text{s. to} & A^\top\tilde{\lambda} + \tilde{\mu}^l - \tilde{\mu}^u - e_s\tilde{\mu}_s = 0 \\ & \tilde{\mu}^l \geq 0\ \ ,\ \ \tilde{\mu}^u \geq 0\ \ ,\ \ \tilde{\mu}_s \geq 0 \end{array} \qquad (19)$$

Because the right-hand-side of the equality constraints is 0, the all-0 solution is feasible. Hence, an unbounded dual ray is associated to any feasible solution with positive objective function value. The corresponding feasibility cut is

$$lpl_s\tilde{\mu}_s + \textstyle\sum_{i\in\mathcal{N}}[\,(l_i - a_i)\tilde{\mu}_i^l - (u_i - a_i)\tilde{\mu}_i^u\,]y_i \leq 0\ \ . \qquad (20)$$

In the extreme ray, $\tilde{\mu}_s = 0$ cannot happen, otherwise it would be an extreme ray also for the dual of (12) with the constraint $x_s \leq -lpl_s$ removed, which would therefore be empty, which is not possible because $x = 0$ is feasible. Hence, $\tilde{\mu}_s > 0$: dividing (20) by $\tilde{\mu}_s$, and defining $\lambda = \tilde{\lambda}/\tilde{\mu}_s$, $\mu^l = \tilde{\mu}^l/\tilde{\mu}_s$, $\mu^u = \tilde{\mu}^u/\tilde{\mu}_s$ we get (17). Applying in reverse this change of multipliers, i.e., multiplying λ, μ^l, μ^u by an arbitrary $\tilde{\mu}_s > 0$, we get (20) from (17). □

The previous discussion has highlighted the well-known fact that there can be multiple ways to generate Benders cuts. As mentioned in §1, the standard Benders cuts generated may not be (indeed, often they are not) the most effective, and some research has been devoted to develop alternatives. For instance, [22] proposes a selection criteria for Benders cuts based on the correspondence between minimal infeasible subsystems of an infeasible linear optimization problem [25] and the vertices of the so-called alternative polyhedron. This boils down to adding a *normalization constraint* to the Benders subproblems. In our case (limiting as usual the discussion to the lower protection case, as the upper protection one is analogous) this leads to

$$\begin{array}{ll} \max_{\lambda,\mu} & lpl_s\mu_s + \sum_{i\in\mathcal{N}}[\,(l_i - a_i)\mu_i^l - (u_i - a_i)\mu_i^u\,]y_i \\ \text{s. to} & A^\top\lambda + \mu^l - \mu^u - e_s\mu_s = 0 \\ & \mu^l \geq 0 \;\;,\;\; \mu^u \geq 0 \;\;,\;\; \mu_s \geq 0 \\ & \sum_{i\in\mathcal{N}}(\,w_i^l\mu_i^l + w_i^u\mu_i^u\,) + w_0\mu_s = 1 \;, \end{array} \tag{21}$$

i.e., adding to (19) a normalization constraint in *arbitrary* weights w^l, w^u, and w_0. This makes the dual subproblem always bounded, and therefore easier to solve with any LP algorithm, in particular interior-point methods that often have numerical issues with unbounded instances. Also, a wise choice of the weights may provide deeper Benders cuts. The primal subproblem (the dual of (21)) is

$$\begin{array}{rll} \min_{\alpha,x} & \alpha & \\ \text{s. to} & Ax = 0 & \\ & x_i + w_i^l\alpha \geq (l_i - a_i)y_i & i \in \mathcal{N} \\ & x_i - w_i^u\alpha \leq (u_i - a_i)y_i & i \in \mathcal{N} \\ & x_s - w_0\alpha \leq -lpl_s \;, & \end{array} \tag{22}$$

which, thanks to the new variable α (the dual variable of the normalization constraint), is never infeasible. Therefore, if $\alpha^* \leq 0$, then (12) is feasible, i.e., cell s is protected, while if $\alpha^* > 0$ then (12) is infeasible and the optimal solution of (21) provides a ray, thus a Benders feasibility cut.

5 Computational results

To empirically validate the efficiency of the proposed stabilized Benders decomposition for CSP, we performed an extensive set of numerical experiments on a set of real-world general and synthetic 1H2D tables. Real-world general tables are standard instances used in the literature [9, 20]. Some of these real-world instances were discarded since they were too large and difficult for all tested methods, i.e, no feasible solution was obtained within the (one hour) time limit. Synthetic instances were obtained with a generator of random 1H2D tables introduced in [8]. This generator is governed by several parameters: the number of rows in a subtable; the number of columns per subtable; the depth of the hierarchical tree; the minimum and maximum number of rows with hierarchies for each subtable; and the probability for a cell to be marked as sensitive. The 1H2D table generator is available from `http://www-eio.upc.edu/~jcastro/generators_csp.html`. We considered asymmetric 1H2D instances, i.e., instances where $u_i = \mathtt{a} \cdot l_i$ for all $i \in \mathcal{N}$, the asymmetry parameter being $\mathtt{a} = 5$. A total of 48 randomly 1H2D instances and 15 real-world tables were considered. The 48 1H2D tables were obtained by running the generator with all the parameters fixed, except three: the number of rows per subtable ($\mathtt{r} \in \{40, 50, 60, 70\}$), the number of columns per subtable ($\mathtt{c} \in \{50, 60, 70, 80\}$) and the percentage of sensitive cells ($\mathtt{s} \in \{5, 10, 15\}$). Tables 1 and 2 report the characteristics of 1H2D synthetic and real instances respectively: the number of cells ("n"), the number of sensitive cells ("s"), the number of table relations ("m") and the number of non zero coefficients in linear constraints ("nz"). Hierarchical synthetic tables are identified by the particular combination of parameters, i.e., `r-c-s`.

The stabilized Benders approach for CSP was implemented in C++ (GNU g++ version 4.5.1) using the state-of-the-art solver CPLEX 12.5 for the solution of the SMP and the subproblems. The trust region radius κ, one of the most influential parameters of the algorithm of Figure 2, took the initial value of $0.01|\mathcal{S}|$ and it was sequentially increased at line 8 of the algorithm, taking values $\kappa \in \{0.02|S|, 0.5|S|, |S|\}$, as this particular sequence was empirically found to be the most satisfactory one for this application. The optional reset of κ at step 21 of the algorithm after a change of the stability center was not applied, as this led to faster convergence. We also tested the following different methods for the Benders subproblems (for both upper and lower protection subproblems):

- **meth1:** solve the (always feasible) primal subproblem (14);
- **meth2:** solve the (possibly unbounded) dual subproblem (19);
- **meth3:** solve the dual subproblem (19) but setting the finite target $f(\tilde{\mu}_s, \tilde{\mu}_l, \tilde{\mu}_u) \leq ||\nabla f(\tilde{\mu}_s, \tilde{\mu}_l, \tilde{\mu}_u)||$, where

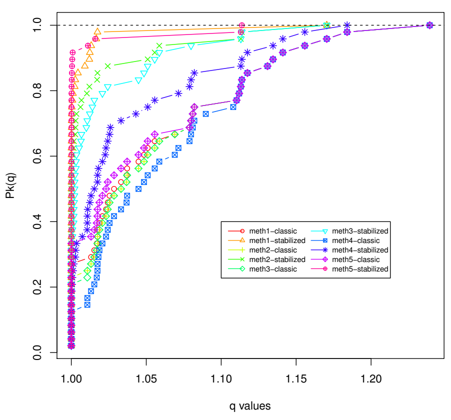
  $$f(\tilde{\mu}_s, \tilde{\mu}_l, \tilde{\mu}_u) = lpl_s \tilde{\mu}_s + \textstyle\sum_{i \in \mathcal{N}} [\, (l_i - a_i)\tilde{\mu}_i^l - (u_i - a_i)\tilde{\mu}_i^u \,] y_i \ \ ,$$

  which is an alternative way to the normalization constraint to make it bounded;
- **meth4:** solve the normalized subproblem (21), using as particular weights $w_i^l = w_i^u = w_0 = 1$, $i \in \mathcal{N}$;
- **meth5:** as in meth4 but replacing the normalization constraint with $\sum_{i \in \mathcal{N}} (w_i^l \mu_i^l + w_i^u \mu_i^u) + w_0 \mu_s \leq 1$.

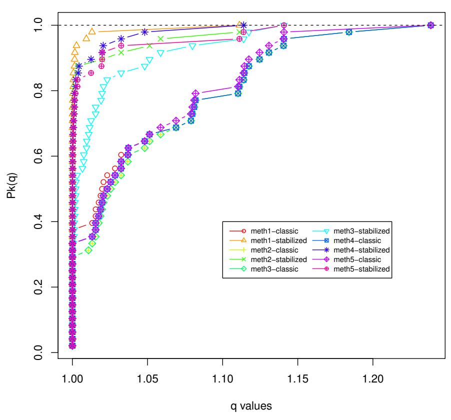
All the above five different methods for subproblems, excluding meth2, were tested using the primal simplex, dual simplex and barrier method of CPLEX.

Instance	n	s	m	nz
40-50-10	7242	705	346	14637
40-50-5	7242	352	346	14637
40-60-10	10248	1002	412	20679
40-60-5	10248	501	412	20679
40-70-10	13916	1365	480	28045
40-70-5	13916	682	480	28045
40-80-10	11583	1136	467	23409
40-80-5	11583	568	467	23409
50-50-10	9639	940	393	19431
50-50-5	9639	470	393	19431
50-60-10	13725	1344	469	27633
50-60-5	13725	672	469	27633
50-70-10	9514	931	418	19241
50-70-5	9514	465	418	19241
50-80-10	17658	1736	542	35559
50-80-5	17658	868	542	35559
60-50-5	13923	680	477	27999
60-60-5	15494	759	498	31171
60-70-5	16685	819	519	33583
60-80-5	19926	980	570	40095
70-50-5	13515	660	469	27183
70-60-5	14945	732	489	30073
70-70-5	18247	896	541	36707
70-80-5	24786	1220	630	49815
40-50-15	7242	1057	346	14637
40-60-15	10248	1503	412	20679
40-70-15	13916	2047	480	28045
40-80-15	11583	1704	467	23409
50-50-15	9639	1410	393	19431
50-60-15	13725	2016	469	27633
50-70-15	9514	1396	418	19241
50-80-15	17658	2604	542	35559
60-50-10	13923	1360	477	27999
60-50-15	13923	2040	477	27999
60-60-10	15494	1518	498	31171
60-60-15	15494	2277	498	31171
60-70-10	16685	1638	519	33583
60-70-15	16685	2457	519	33583
60-80-10	19926	1960	570	40095
60-80-15	19926	2940	570	40095
70-50-10	13515	1320	469	27183
70-50-15	13515	1980	469	27183
70-60-10	14945	1464	489	30073
70-60-15	14945	2196	489	30073
70-70-10	18247	1792	541	36707
70-70-15	18247	2688	541	36707
70-80-10	24786	2440	630	49815
70-80-15	24786	3660	630	49815

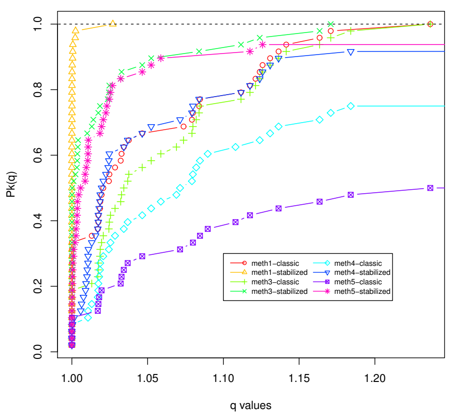
Table 1: Characteristics of synthetic 1H2D instances.



(a) Primal simplex

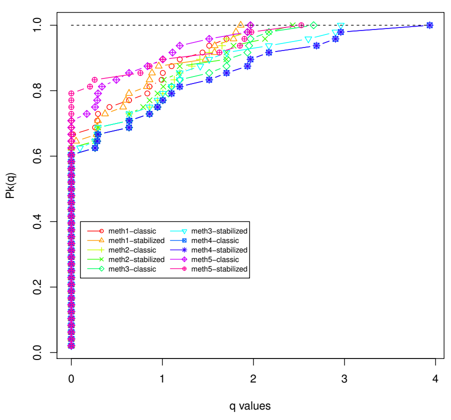


(b) Dual simplex

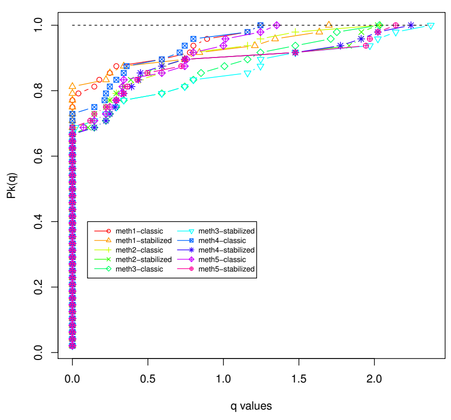


(c) Barrier

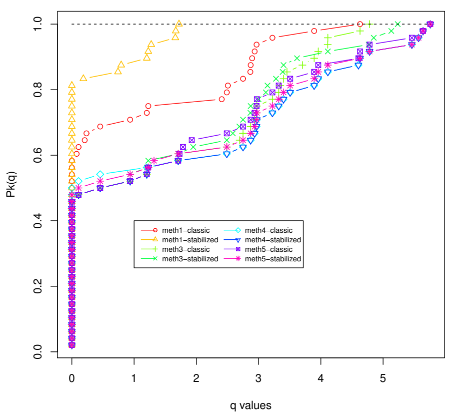
Figure 5: Performance profiles for the different combinations based on upper bound



(a) Primal simplex



(b) Dual simplex

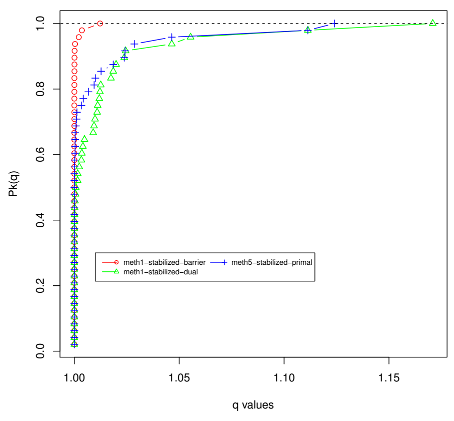


(c) Barrier

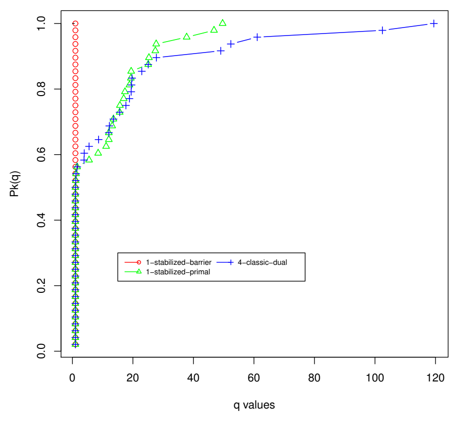
Figure 6: Performance profiles for the different combinations based on CPU time

Instance	n	s	m	nz
hier13x13x13a	2197	108	3549	11661
hier13x13x13b	2197	108	3549	11661
hier13x13x13c	2197	108	3549	11661
hier13x13x13d	2197	108	3549	11661
hier13x13x13e	2197	112	3549	11661
hier13x13x7d	1183	75	1443	5369
hier13x7x7d	637	50	525	2401
hier16	3564	224	5484	19996
hier16x16x16a	4096	224	5376	21504
hier16x16x16b	4096	224	5376	21504
hier16x16x16c	4096	224	5376	21504
hier16x16x16d	4096	224	5376	21504
hier16x16x16e	4096	224	5376	21504
table4	4992	517	2464	19968
table5	4992	517	2464	19968

Table 2: Characteristics of real tables.



(a) Upper bound



(b) CPU time

Figure 7: Performance profiles for the most effective combinations based on upper bound and CPU time

Meth2 was only solved with the primal and dual simplex, since the barrier method had difficulties in providing an extreme ray on unbounded instances.

All in all we obtained $k \in \mathcal{K}$ possible combinations depending on whether we used: 1) meth1, meth2, meth3, meth4 or meth5 for the subproblems; 2) primal simplex, dual simplex, or barrier for the subproblems; and 3) the classical Benders decomposition algorithm or the stabilized Benders decomposition algorithm for the master problem—thus, $|\mathcal{K}| = 28$. To select the best combination we first evaluated the 48 synthetic instances; this amounted to $48 \cdot 28 = 1344$ executions. All these runs were carried out on a Fujitsu Primergy RX300 server with two 3.33 GHz Intel Xeon X5680 CPUs (each CPU with 12 cores) and 144 GB of RAM, under a GNU/Linux operating system (Suse 11.4), without exploitation of multithreading capabilities. Default values were used for the CPLEX parameters (including optimality gap), unless explicitly stated.

We compare the different combinations by means of performance profiles [15]. Quality measures were the value of the objective function and CPU time (thus, in both cases, the lower, the better). Let Q_{ik} be the quality of the solution of instance i solved by combination k, i.e., the best upper bound provided by the method or the CPU time obtained; in both cases, Q_{ik} is always strictly positive for CSP. The performance ratio is thus defined as

$$v(i,k) = Q_{i,k} / \min\{Q_{i,k} \,:\, k \in \mathcal{K}\} \ ,$$

and the (cumulative) distribution function $P_k(q) : [1,\infty) \to [0,1]$ is defined as

$$P_k(q) = |\{i \in \mathcal{I} \,:\, v(i,k) \le q\}| / |\mathcal{I}| \ , \ q \ge 1 \ ,$$

where $\mathcal{I}$ is the set of instances.

Figures 5 and 6 show different performance profiles based on solution quality and total CPU time, respectively. Each figure contains three subfigures, depending on the solver used for the subproblems (primal, dual or barrier). Each subfigure shows performance profiles for the combinations “method”-”master”, where master is either “classic” and “stabilized”; for the reasons alluded to above, this amounts to 10 combinations for simplex approaches and 8 for the barrier one.

In terms of upper bounds, we conclude from Figure 5 that the best choices are: meth5-stabilized and meth1-stabilized for primal simplex; meth1-stabilized and meth4-stabilized for dual simplex; and clearly meth1-stabilized for barrier. All the best options use the stabilized Benders decomposition. In terms of total CPU time we can see in Figure 6 that the fastest variants were: meth1-stabilized, meth5-classic and meth5-stabilized for primal simplex; meth4-classic for simplex dual; and, by far, meth1-stabilized for barrier.

Figure 7 shows a last performance profile using only the best combinations according to the previous performance profiles in Figures 5 and 6. Clearly, stabilized Benders decomposition with meth1 using the barrier

		gap	iter.	CPU
meth1	classic	5.60%	94	3007
	stabilized	1.50%	137	2778
meth2	classic	6.19 %	30	3419
	stabilized	2.45 %	43	3409
meth3	classic	6.19 %	26	3489
	stabilized	2.94 %	32	3542
meth4	classic	16.58 %	102	3492
	stabilized	6.05 %	190	3569
meth5	classic	27.01 %	81	3319
	stabilized	3.96 %	137	3290

Table 3: Average gap, average Benders iterations and average CPU time for all the synthetic instances, for the five methods and two Benders variants (either classical or stabilized).

for the subproblems is the fastest combination and provides the best upper bound for more than 90% of the instances. This particular combination was thus selected to make a comparison with current state-of-the-art solvers.

To shed more light on the global effect of stabilization, Table 3 reports average numbers of gap, Benders iterations, and CPU time for all the instances of previous profiles, differentiating by method and Benders variant (either classical or stabilized). According to the table, the stabilized version always outperformed the corresponding classical one. And the less efficient is classical Benders (meth4 and meth5), the most useful is stabilization. Meth1 is in general superior to the other ones, whether stabilized or not; this is consistent with the profiles of Figure 7, and supports our choice of meth1-stabilized as the best combination.

Being average numbers, the results of Table 3 can be affected by outliers (which explain why the average CPU times are large); and in addition they don't allow to clearly understand the interaction between stabilization and normalization in variants meth3, meth4 and meth5. To have a clearer picture, detailed results for four of the larger instances, that we have chosen as representatives of the trends we have identified, are reported in Table 4, for meth1, and for the normalized meth3, meth4 and meth5. The information given is: number of Benders iterations ("it"), final upper bound computed ("UB"), total CPU time ("Total time")—with a time limit of one hour, average time of solution of master, lower protection subproblem, and upper protection subproblem per Benders iteration (respectively, "Master/it time", "LowS/it time" and "UpS/it time"), and optimality gap of reported solution ("gap"). From Table 4 we conclude:

- In some cases meth1 (i.e., only stabilization) significantly reduced the number of Benders iterations compared to the other methods that include stabilization and normalization (e.g., instance 70-80-10 required 11, 128 and 133 Benders iterations for meth3, meth4 and meth5, but only seven for meth1). It is also worth noting that in the first two instances meth5 was the best approach in terms of Benders iterations, that is, normalization was very useful for the quality of generated cuts.

- Even when meth1 performed more iterations (such as in instance 50-70-10: 62 iterations of meth1 vs nine iterations of meth5), the time of subproblems was much lower. This seems to indicate that the normalization constraint makes subproblems (in all cases) much more difficult, at least for CSP. In addition, the upper subproblems are consistently harder than the lower ones for all the methods.

- In general the total CPU time is dominated by the solution of subproblems, the master time is not significant. This readily explains why normalization was not helpful for CSP, since it made subproblems much harder. While there are clear differences in the CPU time of masters of the different methods, they end up having a negligible effect when compared to subproblem time.

It is worth noting that the normalization had much less effect than what reported in other studies [22]. A possible factor explaining this is that normalization has mostly be proven useful when using Benders cuts in the context of a branch-and-bound framework, i.e., to also cut fractional solutions. Since we adopt a more traditional Benders approach where cuts are only applied to the integer variables, normalization may play a lesser role, as indeed it turned out to.

Table 5 reports a comparison between stabilized Benders meth1 using the barrier solver (`meth1-stabilized-barrier`, selected as the best variant from the previous analysis) and a recent version—dated from 2014—of the state-of-the-art approach initially developed in [19, 20], which embeds Benders cuts within a branch-and-cut tree. This method for CSP, also implemented in C/C++, is included in the τ-Argus software, which is widely used by most European NSAs [14]. We note that this approach also uses CPLEX 12.5, but only for the solution of the different LP subproblems. If it also exploited the branch-and-cut tree of CPLEX, instead of generating its own tree, its performance would likely increase, at the expense of using callbacks. Therefore, a side benefit of our approach is that it can fully exploit CPLEX as a black-box. Table 5 shows the gap and CPU time for stabilized Benders ("A") and the approach of [19, 20] ("B"). The last two columns report the difference in gap and CPU time between both methods. A time limit of one hour was considered for all the runs. It is worth remarking that both variants use the same primal heuristic (a standard procedure first described

Instance	meth	it	UB	Total time	Master/it time	LowS/it time	UpS/it time	gap
50-70-10	1	62	175330	161.75	0.8381	0.129	1.642	0.00%
	3	74	175329	2650.08	0.3872	1.639	33.786	0.00%
	4	363	181794	—	0.0561	0.822	8.333	3.56%
	5	9	177168	—	0.0667	0.584	399.548	1.04%
40-50-15	1	20	245325	33.87	0.0475	0.060	1.586	0.00%
	3	18	245325	249.84	0.0833	0.732	13.064	0.00%
	4	341	246750	—	0.0188	1.077	8.951	0.58%
	5	13	245326	132.79	0.0608	0.500	9.654	0.00%
70-80-10	1	7	614907	71.03	0.0514	0.813	9.283	0.00%
	3	11	614894	2722.73	0.1036	22.885	224.532	0.00%
	4	128	664024	—	0.0641	3.746	23.061	7.39%
	5	133	617008	—	0.0623	3.241	22.505	0.34%
70-80-15	1	4	752873	57.12	0.1475	2.003	12.130	0.01%
	3	11	752888	—	0.1391	22.622	340.705	0.01%
	4	67	753984	—	0.0791	7.187	45.073	0.15%
	5	66	797019	—	0.0809	6.269	46.929	5.54%

— Time limit reached.

Table 4: Detailed information of stabilized Benders for four representative instances and methods 1, 3, 4 and 5.

in [28]), and therefore the differences are not due to the starting points. Both variants behaved much worse without the heuristic.

Table 5 clearly shows that, for 1H2D tables, stabilized Benders was considerably more efficient. The average gap for "A" was 0.87% whereas it was 2.51% for "B" within the same CPU time limit. In average, stabilized Benders was 1.8 times faster than "B"; however, this result is heavily skewed by the several instances reaching time limit. In several cases where "A" terminated before the time limit (which "B" never did), the speedup reached and exceeded two orders of magnitude. In nine of 48 instances (marked with †) the approach of [19, 20] did not find a feasible solution within the time limit, whereas stabilized Benders always found a solution, and most often of excellent quality. Only in four instances "B" outperformed "A"; in the remaining 44 instances, "A" was better.

Table 6 reports the same information for the real-world instances. CPU times are not reported, since the time limit was reached in all the runs. In this case the behaviour of both approaches was slightly different. In six instances (marked in bold) stabilized Benders improved the upper bound with an average gap of 2.61%. On the other hand, in other six tables "B" outperformed "A" with an average gap of 4.57%. However stabilized Benders computed a feasible solution within the limit for all the instances, whereas "B" was not able to do it in two cases. In general, we can safely state that

	meth1-stabilized-barrier (A)		State-of-the-art CSP method (B)		Difference A-B
Instance	gap A	CPU A	gap B	CPU B	Δgap
40-50-10	0.00%	190.96	4.4%	—	-4.44%
40-50-5	1.42%	—	1.7%	—	-0.25%
40-60-10	0.01%	—	0.4%	—	-0.39%
40-60-5	0.83%	—	6.3%	—	-5.48%
40-70-10	0.00%	2962.3	0.8%	—	-0.75%
40-70-5	2.08%	—	1.7%	—	0.36%
40-80-10	0.01%	3232.28	4.5%	—	-4.53%
40-80-5	1.17%	—	4.4%	—	-3.26%
50-50-10	0.00%	3517.52	1.2%	—	-1.21%
50-50-5	3.90%	—	2.7%	—	1.22%
50-60-10	0.89%	—	2.6%	—	-1.66%
50-60-5	0.95%	—	5.9%	—	-4.94%
50-70-10	0.00%	184.29	1.2%	—	-1.18%
50-70-5	5.98%	—	†	†	†
50-80-10	0.01%	143.58	†	†	†
50-80-5	1.39%	—	1.2%	—	0.23%
60-50-5	2.64%	—	10.9%	—	-8.27%
60-60-5	0.23%	—	1.4%	—	-1.18%
60-70-5	3.01%	—	7.0%	—	-4.04%
60-80-5	0.17%	—	1.0%	—	-0.84%
70-50-5	6.74%	—	15.6%	—	-8.87%
70-60-5	2.83%	—	5.2%	—	-2.35%
70-70-5	0.18%	—	0.7%	—	-0.57%
70-80-5	2.36%	—	†	†	†
40-50-15	0.00%	36.06	0.1%	—	-0.14%
40-60-15	0.00%	129.86	0.4%	—	-0.40%
40-70-15	0.00%	85.43	0.1%	—	-0.12%
40-80-15	0.00%	60.37	0.1%	—	-0.10%
50-50-15	0.01%	298.78	0.8%	—	-0.76%
50-60-15	0.00%	68.75	1.8%	—	-1.78%
50-70-15	0.01%	651.86	0.6%	—	-0.60%
50-80-15	0.01%	49.15	0.1%	—	-0.10%
60-50-10	0.00%	2282.89	3.1%	—	-3.05%
60-50-15	0.01%	185.2	0.3%	—	-0.28%
60-60-10	1.45%	—	3.9%	—	-2.47%
60-60-15	0.00%	203.57	0.1%	—	-0.08%
60-70-10	0.24%	—	1.2%	—	-0.92%
60-70-15	0.00%	30.14	0.0%	—	-0.05%
60-80-10	0.01%	230.2	†	†	†
60-80-15	0.01%	265.29	†	†	†
70-50-10	0.59%	—	1.6%	—	-1.03%
70-50-15	0.00%	35.14	0.2%	—	-0.22%
70-60-10	2.46%	—	0.8%	—	1.67%
70-60-15	0.00%	129.71	1.9%	—	-1.86%
70-70-10	0.12%	—	†	†	†
70-70-15	0.01%	67.27	†	†	†
70-80-10	0.00%	73.42	†	†	†
70-80-15	0.01%	58.93	†	†	†

— Time limit reached

† Time limit reached without a feasible solution

Table 5: Comparison between stabilized Benders method 1 using the barrier solver (`meth1-stabilized-barrier`) and the state-of-the-art method of [20] for synthetic 1H2D instances.

	meth1 stabilized) barrier (A)	state-of-the-art CSP method (B)	Difference A-B
Instance	gap A	gap B	Δgap
hier13x13x13a	**98.86%**	**98.95%**	**-0.09%**
hier13x13x13b	**28.92%**	**39.25%**	**-10.33%**
hier13x13x13c	**40.41%**	**42.33%**	**-1.92%**
hier13x13x13d	63.16%	†	†
hier13x13x13e	**42.10%**	**45.00%**	**-2.90%**
hier13x13x7d	54.08%	†	†
hier13x7x7d	17.25%	0.01%	17.24%
hier16	99.17%	99.13%	0.04%
hier16x16x16a	99.10%	99.09%	0.01%
hier16x16x16b	88.80%	88.65%	0.15%
hier16x16x16c	**92.33%**	**92.67%**	**-0.34%**
hier16x16x16d	**99.02%**	**99.12%**	**-0.10%**
hier16x16x16e	100.00%	100.00%	0.00%
table4	15.94%	11.84%	4.10%
table5	16.92%	11.06%	5.86%

† Time limit reached without a feasible solution

Table 6: Comparison between stabilized Benders method 1 using the barrier solver (`meth1-stabilized-barrier`) and the state-of-the-art method of [20] for the real tables.

stabilized Benders was competitive even for real-world instances.

Finally, we tried the Benders algorithm built-in in CPLEX 12.7 (CPLEX-Benders) using a set of small 1H2D instances. It is worth noting that, although CPLEX-Benders is a state-of-the-art implementation of Benders decomposition, it is tuned to be efficient for a wide class of general problems, where optimality cuts can be relevant, while for CSP only feasibility cuts are generated. In addition, we are not aware of which Benders acceleration strategies [31] are implemented in CPLEX-Benders; this should be taken into account when comparing the results with both codes. For this tests CPLEX was interfaced through AMPL, and we only considered the CPLEX solution time, discarding the model generation time. Table 7 reports the comparison between stabilized Benders meth1 using the barrier solver and CPLEX-Benders. The decision on the distribution of the continuous variables in the different Benders subproblems was determined automatically by CPLEX (strategy 0, where all continuous variables are in a single Benders subproblem) and by the user (strategy 1, in this case the continuous variables for each sensitive cell went to different Benders subproblems). As the results show, stabilized Benders is, by far, superior to CPLEX-Benders.

	meth1-stabilized-barrier		CPLEX-Benders strategy 1		CPLEX-Benders strategy 0	
	CPU	gap	CPU	gap	CPU time	gap
20-25-15	51.44	0.01%	—	0.09%	—	0.48%
20-30-15	2297.58	0.01%	—	1.05%	—	1.32%
20-35-15	163.14	0.01%	—	1.39%	—	0.77%
25-25-15	—	0.03%	—	6.22%	—	0.35%
25-30-15	54.87	0.01%	—	93.51%	—	5.68%
25-35-15	43.61	0.00%	—	†	—	94.98%
30-25-15	1128.73	0.01%	—	0.34%	—	0.33%
30-30-15	99.76	0.01%	‡	‡	—	94.91%
30-35-15	568.42	0.01%	‡	‡	‡	‡

† Time limit reached.

† No feasible solution was found within the time limit of 3600 seconds.

‡ Internal memory error provided by AMPL.

Table 7: Comparison between stabilized Benders method 1 using the barrier solver (`meth1-stabilized-barrier`) and CPLEX-Benders for a set of small 1H2D instances.

CPLEX-Benders always exhausted the time limit (3600 seconds). It is worth noting that one instance was solved by stabilized Benders with a 0.00% in less than one minute, whereas CPLEX-Benders only provided a 94.98% gap solution.

6 Conclusions

We presented computational results for a new variant of Benders decomposition for 0-1 problems based on a stabilization of the master problem through local branching—or trust region—constraints. Although the method is just one of those theoretically analyzed in [1], this particular variant was actually developed before that the deeper analysis of [1] was developed, and actually inspired a significant part of the results in that paper. Besides, while [1] was focused on a general theoretical convergence framework, with comparatively little attention devoted to the computational part, in this paper the method has been very thoroughly tested against state-of-the-art ones for a specific, challenging and very large real-world application, the Cell Suppression Problem in the field of data privacy. Our results clearly showed that stabilization significantly improves the performances of Benders decomposition. In particular, stabilization made subproblems much easier compared to "normalization" in Benders subproblems, which has been reported in the past to be useful for performances of the non-stabilized Benders method. Also, stabilized Benders was way superior to the Benders decomposition built-in in the latest version of CPLEX.

For structured 1H2D tables, stabilized Benders was always superior to the state-of-the-art method for CSP (which embeds Benders cuts within a branch-and-cut tree). For real-world instances, the two approaches traded blows, each one outperforming the other in roughly half of the instances; however stabilized Benders was the only one able to provide a feasible solution within the one hour time limit for all the real-world instances, which is surely a desirable feature for practitioners looking forward to using this approach. It is worth mentioning that these instances are so challenging that both methods provided poor solutions—with large gaps—after one hour of CPU time for most of the runs. Clearly, then, more work is needed in order to tackle these hard real-world instances. Several approaches are possible, such as using the conceptual tools developed in [1] to diminish the cost of the subproblems, and/or testing other forms of stabilization. Clearly, parallelization of subproblems' solution is also a promising venue. All in all, we believe that stabilization of Benders decomposition can be a useful tool for tackling hard structured 0-1 MILP problems, of which CSP is just one of very many relevant examples.

Acknowledgments

This work has been supported by MINECO–FEDER grant MTM2015-65362-R. We thank the anonymous reviewers for numerous suggestions and pointing out related references which allowed us to improve (and to fix some inaccuracies in) the original manuscript.

References

[1] W. van Ackooij, A. Frangioni, and W. de Oliveira. Inexact stabilized Benders' decomposition approaches with application to chance-constrained problems with finite support. *Computational Optimization and Applications*, 65(3), 637–669, 2016.

[2] N. Boland, M. Fischetti, M. Monaci, and M. Savelsbergh. Proximity Benders: a decomposition heuristic for stochastic programs, *Journal of Heuristics*, 22, 181-198, 2016.

[3] H. Ben Amor, J. Desrosiers, and A. Frangioni. On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics*, 157, 1167–1184, 2009.

[4] W. Ben-Ameur and J. Neto. Acceleration of cutting-plane and column generation algorithms: Applications to network design. *Networks*, 49, 3–17, 2007.

[5] J.F Benders. Partitioning procedures for solving mixed-variables programming problems. *Computational Management Science*, 2, 3–19, 2005. English translation of the original paper appeared in *Numerische Mathematik*, 4, 238–252, 1962.

[6] J.R. Birge and F. Louveaux. *Introduction to Stochastic Programming*, 1997, New York, Springer.

[7] O. Briant, C. Lemaréchal, Ph. Meurdesoif, S. Michel, N. Perrot, and F. Vanderbeck. Comparison of bundle and classical column generation. *Mathematical Programming*, 113(2), 299–344, 2008.

[8] J. Castro. A shortest paths heuristic for statistical disclosure control in positive tables. *INFORMS Journal on Computing*, 19, 520–533, 2007.

[9] J. Castro. Recent advances in optimization techniques for statistical tabular data protection. *European Journal of Operational Research*, 21, 257–269, 2012.

[10] J. Castro, S. Nasini, and F. Saldanha-da-Gama. A cutting-plane approach for large-scale capacitated multi-period facility location using a specialized interior-point method. *Mathematical Programming*, 163, 411–444, 2017.

[11] J. Castro and A. Via. Revisiting interval protection, a.k.a. partial cell suppression, for tabular data. In J. Domingo-Ferrer and M. Péjic-Bach editors, *Privacy in Statistical Databases. Lecture Notes in Computer Science 9867*, 3–14, 2016, Switzerland, Springer.

[12] A.M. Costa. A survey on Benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research*, 32, 1429–1450, 2005.

[13] C. d'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi. On Interval-Subgradient Cuts and No-Good Cuts. *Operations Research Letters*, 38, 341–345, 2010.

[14] P.-P. de Wolf, A. Hundepool, S. Giessing, J.J. Salazar, and J. Castro, τ-Argus User's Manual, Statistics Netherlands, 2014. Available online at `http://neon.vb.cbs.nl/casc/Software/TauManualV4.1.pdf`.

[15] E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91, 201–13, 2002.

[16] M. Fischetti, I. Ljubić, and M. Sinnl. Benders decomposition without separability: a computational study for capacitated facility location problems, *European Journal of Operational Research*, 253, 557–569, 2016.

[17] M. Fischetti, I. Ljubić, and M. Sinnl. Redesigning Benders decomposition for large-scale facility location. *Management Science*, 63, 2146–2162, 2017.

[18] M. Fischetti and A. Lodi. Local Branching. *Mathematical Programming*, 98, 23–47, 2003.

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

[20] M. Fischetti and J.J. Salazar. Solving the cell suppression problem on tabular data with linear constraints. *Management Science*, 47, 1008–1026, 2001.

[21] M. Fischetti and D. Salvagnin. An in-out approach to disjunctive optimization. *Lecture Notes in Computer Science*, 6140, 136–140, 2010.

[22] M. Fischetti, D. Salvagnin, and A. Zanette. A note on the selection of Benders cuts. *Mathematical Programming*, 124, 175–182, 2010.

[23] A. Frangioni and B.Gendron. A stabilized structured Dantzig-Wolfe decomposition method. *Mathematical Programming*, 140, 45–76, 2013.

[24] A.M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10, 238–260, 1972.

[25] J. Gleeson and J. Ryan. Identifying minimally infeasible subsystems of inequalities. *ORSA Journal on Computing*, 2, 61–63, 1990.

[26] J.B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms vol. II*, 1996, Berlin, Springer.

[27] A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, E. Schulte Nordholt, K. Spicer, and P.-P. de Wolf. *Statistical Disclosure Control*, 2012, Chichester, Wiley.

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

[29] T.L. Magnanti and R. Wong. Accelerating Benders decomposition: algorithmic enhancement and model selection criteria. *Operations Research*, 29, 464–484, 1981.

[30] A. Nasri, S.J. Kazempour, A.J. Conejo, and M. Ghandhari. Network-constrained AC unit commitment under uncertainty: a Benders' decomposition approach. *IEEE Transactions on Power Systems*, 31, 412–422, 2016.

[31] R. Rahmaniani, T.G. Crainic, M. Gendreau, and W. Rei. The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259, 801–817, 2017.

[32] W. Rei, J.F. Cordeau, M. Gendreau, and P. Soriano. Accelerating Benders decomposition by local branching. *INFORMS Journal on Computing*, 21, 333–345, 2009.

[33] T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167, 96–115, 2005.

[34] M. Tahanan, W. van Ackooij, A. Frangioni, and F. Lacalandra. Large-scale Unit Commitment under uncertainty. *4OR*, 13(2), 115–171, 2015.

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

[36] G. Zakeri, A.B. Philpott, and D.M. Ryan. Inexact cuts in Benders decomposition. *SIAM Journal on Optimization*, 10, 643–657, 2000.