Exact and heuristic methods for statistical tabular data protection

Daniel Baena Mirabete

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Department of Statistics and Operations Research

EXACT AND HEURISTIC METHODS FOR STATISTICAL TABULAR DATA PROTECTION

PhDTesis

Author: Daniel Baena Mirabete

Supervisor: Jordi Castro Pérez

2017

Als meus pares, Isabel i Antonio
pel seu amor incondicional.

They who can give up essential liberty to obtain a little temporary safety deserve neither liberty nor safety.

Benjamin Franklin.

Contents

List of Figures

List of Tables

Agraïments-Acknowledgments

Segurament és en aquests moments finals, just quan estic a poques hores de fer l'entrega de la meva tesis, el millor moment de recordar a totes aquelles persones que m'han acompanyat en aquesta llarga travessia i que han fet possible i realitat aquest repte personal tan important per a mi. No ha estat un camí gens fàcil, però us puc assegurar que l'he gaudit per moments, l'he patit en altres però que sobretot m'ha enriquit professionalment i més encara a nivell personal. Com amant de l'atletisme puc dir que a dia d'avui aquesta tesis és la meva gran carrera de fons realitzada amb les millor llebres possibles.

En primer lloc vull agrair a la persona més rellevant d'aquest projecte. El meu director de tesis, el Dr. Jordi Castro. Crec que no cal que en aquestes breus línies li agraeixi res a nivell científic donat que ja es dóna per suposat. Sí que vull agrair-li la seva dedicació, la seva feina, el seu suport però especialment la seva paciència. Gràcies!

En segon lloc la meva família. Als meus pares. Antonio i Isabel. Sou sense cap mena de dubte el meu pilar imprescindible. Sempre al meu costat. En tots aquests anys de la meva vida mai m'heu deixat sol, sempre heu tingut paraules d'ànims, sempre un bon consell a dir-me. I el més important, cada dia de la meva vida he sentit molt de prop el vostre amor incondicional.

Als meus germans. Sandra i Sergio. Us asseguro que heu estat un exemple per a mi. Estic immensament orgullós de ser el vostre germà petit. De ben petit, m'heu portat pel camí que tocava, sense deixar que em perdés. Ara que arribo a aquest punt i seguit del meu camí us dono les gràcies per tot.

Als meus nebots. Aitor i Roger. Els dos petitons de la família que estimo amb bogeria. Gràcies per treure cada dia que estic amb vosaltres el nen petit que tots tenim a dins. Gràcies per ensenyar-me que lo més important a la vida és ser feliç i passar-ho pipa. Gràcies pel vostre somriure diari.

Finally I would like to sincerely thank Dr. Antonio Frangioni for his dedication, kindness and attention during my 3 months in Pisa (Italy). It made me feel at home. Thank you very much for providing me with scientific knowledge that fortunately has been reflected in this thesis.

I ara si, per últim, gracies a tots els meus amics. Potser alguns de vosaltres no acabeu d'entendre que hagi perdut tants moments especials i tants caps de setmana amb vosaltres per quedar-me tancat a casa treballant. Crec que jo tampoc mai ho entendré però només se que volia fer-ho. Gràcies per respectar-ho i per no deixar mai de comptar amb mi.

MOLTES GRÀCIES A TOTS.
THANKS TO EVERYONE.

Abstract

One of the main purposes of National Statistical Agencies (NSAs) is to provide citizens or researchers with a large amount of trustful and high quality statistical information. NSAs must guarantee that no confidential individual information can be obtained from the released statistical outputs. The discipline of Statistical disclosure control (SDC) aims to avoid that confidential information is derived from data released while, at the same time, maintaining as much as possible the data utility. 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 contains aggregated information obtained by crossing one or more categorical variables from those microdata files. Several SDC methods are available to avoid that no confidential individual information can be obtained from the released microdata or tabular data. This thesis focus on tabular data protection, although the research carried out can be applied to other classes of problems. Controlled Tabular Adjustment (CTA) and Cell Suppression Problem (CSP) have concentrated most of the recent research in the tabular data protection field. Both methods formulate Mixed Integer Linear Programming problems (MILPs) which are challenging for tables of moderate size. Even finding a feasible initial solution may be a challenging task for large instances. Due to the fact that many end users give priority to fast executions and are thus satisfied, in practice, with suboptimal solutions, as a first result of this thesis we present an improvement of a known and successful heuristic for finding feasible solutions of MILPs, called feasibility pump. The new approach, based on the computation of analytic centers, is named the Analytic Center Feasbility Pump. The second contribution consists in the application of the fix-and-relax heuristic (FR) to the CTA method. FR (alone or in combination with other heuristics) is shown to be competitive compared to CPLEX branch-and-cut in terms of quickly finding either a feasible solution or a good upper bound. The last contribution of this thesis deals with general Benders decomposition, which is improved with the application of stabilization techniques. A stabilized Benders decomposition is presented, which focus on finding new solutions in the neighborhood of "good" points. This approach is efficiently applied to the solution of realistic and real-world CSP instances, outperforming alternative approaches.

The first two contributions are already published in indexed journals (*Operations Research Letters* and *Computers and Operations Research*). The third contribution is a working paper to be submitted soon.

Chapter 1

Statistical disclosure control

1.1 Introduction

1.1.1 Motivations

A large amount of data travels daily through Internet. Today, according to *Data Never Sleeps 4.0*, more than 347.000 tweets, 150 millions of emails sent, 21 millions of WhatsApp messages, 38.000 posts in Instagram, 701.000 logins in Facebook, 3 millions of videos viewed in YouTube, 203.000$ in Amazon's sales, among many others, are generated every minute in the World. We are living in the age of digital information, the age of Big Data. More than 3.400 million people are Internet users. We are constantly showing our preferences, directly or indirectly, through surveys, shopping, talks, participation in the social networks, etc. Many areas like banking, insurance, investment, pharmaceutical industry, e-commerce or search engines manage large individual customer information for their own benefit. In addition to these areas mentioned, there are sectors like official statistics or health information, where the main purpose is to provide citizens or researchers with trusted and high quality statistical outputs. In all cases, the more detailed the information you provide is, the richer and more interesting the statistical information will be. However, what about the right to individuals' privacy? The Universal Declaration of Human Rights and Organic Law of Personal Data Protection protect, through heavy fines applied to violators, that not confidential information provided by individuals can be derived to statistical outputs released. The anonymisation of individual data, for example by removing usernames and/or IP addresses, is not enough in order to ensure the privacy of individuals. Let us remind the particular case from 2006 of the company AOL Research where 650.000 detailed Internet search records, previously anonymised, were released. Despite those efforts, a particular user was identified by the New York Times. Of course, the image of the company was extensively damaged and they incurred in severe economic penalties. The discipline Statistical disclosure control (SDC) seeks to avoid that confidential information can be derived from data released whereas, at the same time maintaining as far as possible the data utility. So, the goal is always to publish data as close as possible to the original data (minimize *information loss*) but reducing the risk that someone identifies a particular person or enterprise (minimize *disclosure risk*). This discipline is also known as Statistical disclosure limitation (SDL) because the disclosure risk can be only limited, not completely avoided, unless no data is published. This is clearly shown in the risk-utility graph of Figure 1.1 (from [29]). Several SDC methods have been developed in order to minimize the disclosure risk while maximizing the data utility. SDC methods are very important because in addition to potential fines, it guarantees data quality and high response rates in surveys. If respondents feel that their privacy will be respected, they will be more likely to respond in future surveys. In this thesis, we focus on official statistics but the interested reader in the application of SDC methods to the Big Data field, can also check the recent paper [26]. In general, 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 and each attribute is recorded separately for each respondent. Tabular data contains aggregated information obtained by crossing one or more categorical variables from those microdata files.

In the following sections, we will briefly describe the most relevant aspects about microdata and tabular data protection methods.

1.1.2 Microdata protection

A microdata file contains data at the level of the individual respondent, that is, the lowest level of aggregation of the information collected. The lines or records of the microdata file corresponds to single persons, enterprises, households or others. Each record is characterized by a set of variables or attributes (such as age, gender, income, job etc.). The attributes can be classified as follows:

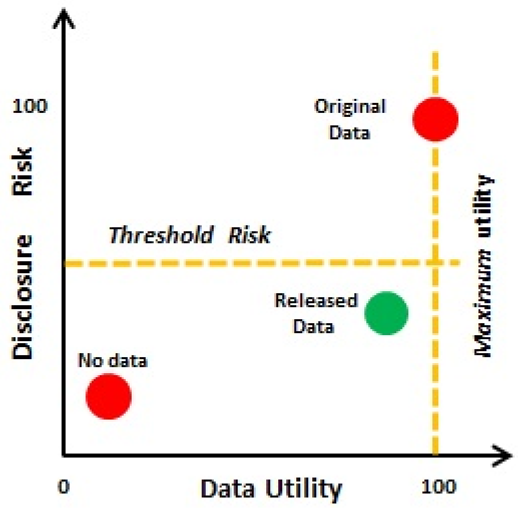


Figure 1.1: The risk-utility graph.

- Identifiers: Attributes that can be used directly to identify the respondent. For instance: names, passport numbers, addresses. The identifiers attributes are always removed before releasing any microdata file. However, it is often not enough.

- Quasi-identifiers: Also called key variables. Attributes that only combined with other quasi-identifier could be used to identify the respondent. For instance: city, job, birth date, sex and ZIP/postal. The value of any quasi-identifier by itself often does not lead to identification; however the combination of several quasi-identifiers could mean individual disclosure. For instance: male, 20 years, married, Barcelona, nurse. This combination of variables is known as key. It is not advisable to remove all quasi-identifiers attributes given that the data utility is drastically reduced. An important step in the SDC process is to detect a list of possible quasi-identifiers.

- Others: All attributes apart from identifiers and quasi-identifiers.

Each attribute (identifier, quasi-identifier or other) can be classified as confidential (or sensitive) or non-confidential (non-sensitive). Confidential variables contain sensitive information such as health, income, religion, political affiliation, etc., that should be protected by SDC methods. Non-confidential variables do not contain sensitive information. For instance: place of residence, zip code, etc.

The attributes can be categorical (i.e, they take values over a finite set, for instance gender, region or education level) or continuous (they take values on an infinite number of values in a particular domain, for instance income, height or weight). Any continuous variable can be transformed to categorical through the establishment of intervals. Any publication of a microdata file without previous pre-processing implies the maximum individual disclosure risk. Different SDC methods have been developed in order to minimize the risk that intruders can estimate sensitive information while at the same time maximizing the data utility, providing an opportunity to make a good and high quality statistical analysis. All protection methods can be classified as:

- Perturbative: The original microdata file is modified changing the value of some attributes. The perturbative SDC method must guarantee that disclosure risk is below a certain threshold agreed by NSAs. Microaggregation, Data swapping o Rank swapping, Noise addition, Rounding, Resampling, Post-randomization method or Data shuffling are some of the most well known perturbative methods.

- Non-Perturbative: The original values are not changed with these methods. However, the level of detail released fell significantly applying suppressions or global recoding. In general, the risk of identifying a respondent is reduced. Sampling, Global recoding, Top and bottom coding or Local suppression are some of the most well known non-perturbative methods.

In this thesis we focus on tabular data protection, for more information about microdata protection the interested reader is addressed to the recent research monographs [24], [25], [49] and [62].

1.1.3 Tabular data protection

Tabular data is obtained by crossing two or more categorical variables in 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, in fact, tabular data is the most common form of publishing information of NSAs. Although tabular data report aggregated information for several respondents, so

		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 1.2: 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.

they could be considered anonymized, there is a risk of disclosing individual information. Figure 1.2 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 of individuals. If there was only one individual of age between 51 and 55 in town t2, 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. The tables can be classified as positive or general tables according to the sign of cell values. Cell values in positive tables are always nonnegative while in general tables the sign can be positive or negative. Another possible classification of tables is based on their particular structure. In fact, this is the most important criteria because 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 table of Figure 1.2 shows two tables obtained from a microdata file with information of inhabitants of some region. Crossing variables age and town, the two-dimensional frequency table of Figure 1.2(b) may be obtained. Instead, Figure 1.2(a) shows a magnitude table with information about a third variable like overall salary for each range of age and town. A hierarchical table is made up of 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 of interest for NSAs. A particular class of 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, let's 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 1.3 (from [12]) 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 made up of a 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 column. Notice that there are two types of marginals: those that contain the sum of interior cells and the one that contains the sum of the marginals themselves.

The first step and one of the most important aspects in tabular data protection is to determine if a cell is considered unsafe or not. Several sensitive rules exist for that purpose, which are outlined bellow (a detailed explanation of these sensitive rules is outside the scope of this thesis. The reader interested in this field can be found more details in ([12, 24, 25, 48, 49, 62])):

- Minimum frequency rule: Used for frequency tables, a cell is considered unsafe when the cell frequency is less than a pre-specified minimum frequency n (normally $n = 3$). This rule could also be applied to magnitude tables but this is not a good practice because it doesn't take into account the contribution of each respondent to the cell value.

- (n,α) dominance rule: A cell is considered unsafe when the sum of the n largest contributions exceeds α% of the cell total. For instance, for a cell $100 = 30 + 30 + 20 + 10 + 10$ (i.e., cell of value 100 and 5 respondents with contributions $30, 30, 20, 10, 10$), if $n = 1$ and $\alpha = 50$ then the cell is non-sensitive: any respondent contribution is less than a 50% of the cell value; however if parameter $n = 2$ and $\alpha = 50$ then the cell is considered sensitive since $30 + 30 > 100 \cdot 50$. Note that (n,α) rule tries to avoid that a coalition of n respondents could obtain accurate estimates of the other respondents contribution. Some usual values are n 1 or 2 and α higher than 60.

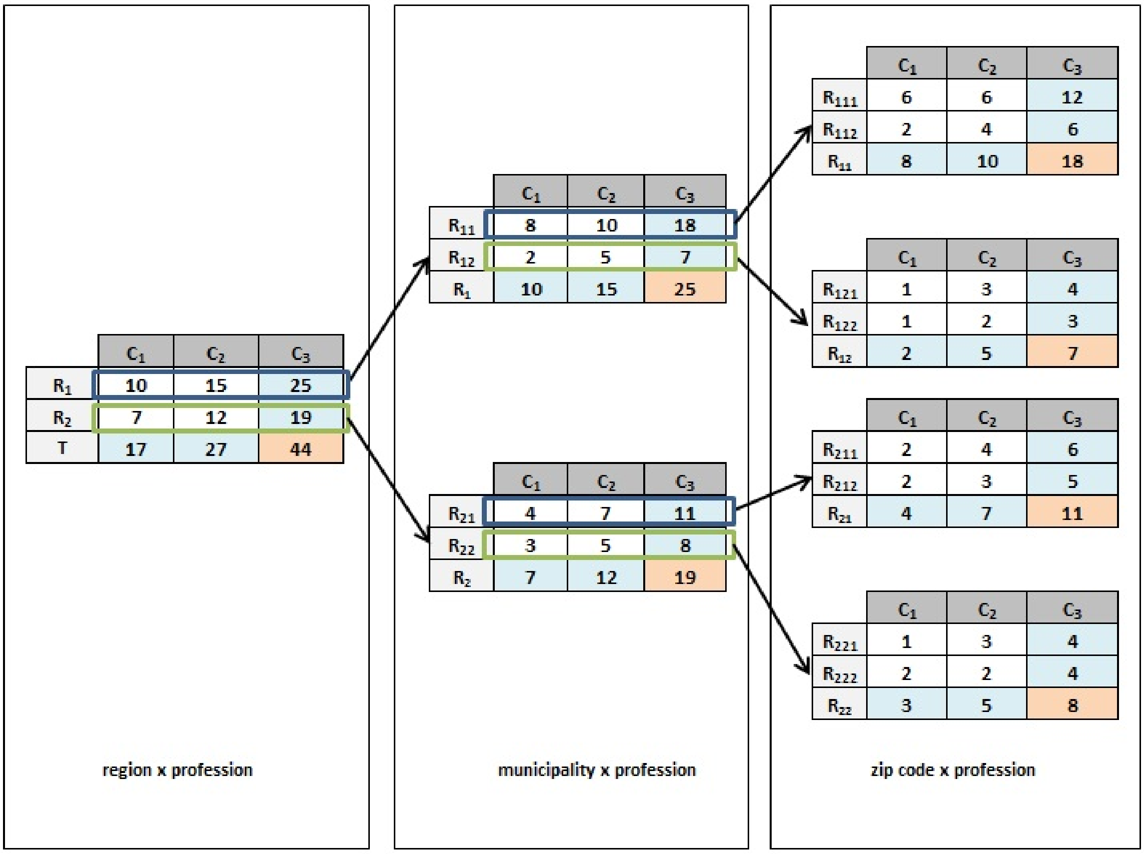


Figure 1.3: Example of 1H2D table made of different subtables: "region"×"profession", "municipality"×"profession" and "zip code"×"profession".

- $p\%$-rule: A cell is considered unsafe if some respondent may obtain an estimate of another respondent contribution within a $p\%$ precision. In practice, NSAs consider the worst case: when the respondent with the second largest contribution tries to estimate the value of the respondent with the highest contribution. For instance, for the cell $100 = 55 + 30 + 10 + 3 + 2$ (i.e., cell of value 100 and 5 respondents with contributions $55, 30, 10, 3, 2$), the second respondent knows that the value of the first respondent is at most $100 - 30 = 70$; If $p = 20\%$, since $70 > (1 + 20/100) \cdot 55 = 66$, then the cell is non-sensitive. If $p = 30$, since $70 < (1 + 30/100) \cdot 55 = 71.5$, the cell is considered sensitive.

The values of parameters n,α and p are decided by NSAs. In general, the $p\%$-rule is preferred to the (n, α) dominance rule because the (n, α) dominance rule could wrongly consider as unsafe some sensitive cells and vice-versa. Let us look at an example of [57]. Let $n = 1$ and $\alpha = 0.6(60\%)$. Then a cell with value $100 = 59 + 40 + 1$ would be declared not sensitive (because $59 < 0.6 \cdot 100$), while a cell with value $100 = 61+20+19$ would be declared sensitive (because $61 > 0.6\cdot 100$). However, for the cell declared non-sensitive, the second largest respondent gets a too tight estimation for the largest contribution $(100 - 40 = 60)$. Similarly, for the cell considered sensitive, the estimation by second respondent would be $100-20 = 80$, far from the real value. These situations could be avoided by using $n = 2$ but even in this case the $p\%$ rule is preferred. Whatever, neither $p\%$-rule nor (n, α) reflect the concentration of contributions in a proper way: for this reason, better bounds on the largest contribution could be done in cells declared as non-sensitive because they have a smaller tail (sum of small contributions) than cells declared sensitive. In [27] the authors propose a sensitivity rule based on the concentration of contributions, measured by the entropy of the relative contributions. In [44] the author claims that classical sensitive rules are not always well-suited for survey data with negative values, missing data or sampling weights. For this reason, he introduces a new class of sensitivity rule known as the Precision Threshold and Noise framework.

The second step is, of course, to minimize the risk of disclosing individual information. For this, several statistical disclosure control methods are available. In general, we can divide all statistical disclosure control methods in two categories: those that adjust the data before tables are created (pre-tabular: disclosure control techniques are applied to microdata files before crossing variables to generate tabular data) and those that consider statistical disclosure control methods once the table is created (post-tabular).

In post-tabular techniques, we find perturbative methods (i.e., they change the original values, for example: controlled tabular adjustment, rounding or controlled rounding) or non-perturbative (i.e., they do not change the original values because we only suppress data or change the table structure, for example: recoding or cell suppression). A post-tabular data protection method can be seen as a map F such that $F(T) = T'$, i.e., table T is transformed to another table T'. There are two main requirements for F: (1) the output table T' should be "safe", and (2) the quality of T' should be high (or equivalently, the information loss should be low), i.e., T' should be a good replacement for T. The disclosure risk can be analyzed through the inverse map $T = F^{-1}(T')$: if not available or difficult to compute by any *data attacker*, then we may guarantee that F is safe. Among the available post-tabular methods we find:

- *Recoding:* This technique consists in combining several categories with few respondents into a new in order to satisfy the sensitive rules above cited. For instance: a categorical variable age with several categories where the category $51-55$ has five respondents and the category $56-60$ has only one respondent. In order to preserve the privacy of respondent between $56-60$ years old, we create a new category $51-60$ with six respondents after aggregation of $51-55$ and $56-60$.

- *Random rounding:* This technique consists in rounding all cell tables to the closest multiple of a certain base number r. Rounding up or down is a random decision. Note that in order to get an additive protected table, the total cells could not be rounded to the nearest multiple of r.

- *Controlled additive rounding:* This method is an extension of the rounding method in order to guarantee both the additivity of the resulting table and the total cells are rounded to a multiple of r (but likely to a multiple which can be far from the original value, increasing the information loss). This method was initially presented by [19] and it has recently been extended by [58] using lower and upper protection levels. The resulting model is a large MILP which is solved by advanced optimization techniques like Benders decomposition.

- *Cell suppression problem (CSP):* This method is based on the suppression of a set of cells that guarantees the protection of the table with minimum loss of information.

- *Controlled tabular adjustment (CTA)*: This method consists in finding the minimum amount of perturbations to the original cells that minimize the risk of disclosing individual information from released table.

CTA and CSP are two of the most recurrent available methods. For this reason, this thesis has focused on these particular techniques, trying to improve their efficiency. In the following subsections we explain CSP and CTA in further detail. But, before starting, in order to model the different mixed-integer optimization problems, we have to modeling tables. Briefly, any positive table can be defined as:

- A set of cells a_i, $i \in \mathcal{N} = \{1, \ldots, n\}$, that satisfies $\mathcal{M} = \{1, \ldots, m\}$ linear relations $Aa = b$, $a \in \mathbb{R}^n$ being the vector of a_i's, and $A \in \mathbb{R}^{m \times n}$. These linear relations impose that the set of inner cells has to be equal to the total or marginal cell, i.e., if $\mathcal{I}_j$ is the set of inner cells of relation $j \in \mathcal{M}$, and t_j is the index of the total cell of relation j, the constraint associated to this relation is $\left(\sum_{i \in \mathcal{I}_j} a_i\right) - a_{t_j} = 0$.

- Nonnegative cell weights w_i, $i \in \mathcal{N}$, used in the definition of the objective function. These weights penalize suppressions or modifications from the original cell values in the released table. Cells weights are usually a function of the cell value, e.g., $w_i = a_i$ where the overall value perturbated or suppressed is minimized. If $w_i = 1$ the number total of perturbated or suppressed cells is minimized.

- A lower and upper bound for each cell $i \in \mathcal{N}$, respectively l_{a_i} and u_{a_i}, which can be considered publicly known.

- A set $\mathcal{S} = \{i_1, i_2, \ldots, i_s\} \subseteq \mathcal{N}$ of indices of sensitive or confidential cells.

- A lower and upper protection level for each sensitive cell, respectively, lpl_i and upl_i, $i \in \mathcal{S}$.

A table is considered feasible by NSAs if it guarantees the required protection intervals for each sensitive cell. In addition to this, another important measure in order to evaluate a SDC tabular method is the loss of statistical utility of the protected data (information loss or data utility). In [28] the information loss was measured by comparing several statistics on the original and protected microdata (like means, correlations and covariances preserved).

More details about tabular data protection can be found in the recent survey [12] and the monographs [48, 49].

Cell suppression problem

Cell suppression problem (CSP) is a statistical disclosure control method where values of some cells are not published while the original values of the others are. In particular, it consists of finding a set of additional cells (named secondary or complementary cells) that guarantees that the value of primary cells containing sensitive information (also suppressed) cannot be recompiled, with minimum loss of information. If only sensitive cells are suppressed their values could be retrieved from marginal, for this reason, additional cells (hopefully, as few as possible) are selected for secondary suppression. Once the protected file has been released, any external attacker could calculate a lower and upper bound for each sensitive cell $s \in \mathcal{S}$ by solving the following optimization problems:

$$\begin{array}{rll} \underline{a_s} = & \min & x_s \\ & \text{s. to} & Ax = b \\ & & l_i \leq x_i \leq u_i \quad i \in \mathcal{S} \cup \mathcal{P} \\ & & x_i = a_i \quad i \notin \mathcal{S} \cup \mathcal{P} \end{array} \quad \text{and} \quad \begin{array}{rll} \overline{a_s} = & \max & x_s \\ & \text{s. to} & Ax = b \\ & & l_i \leq x_i \leq u_i \quad i \in \mathcal{S} \cup \mathcal{P} \\ & & x_i = a_i \quad i \notin \mathcal{S} \cup \mathcal{P}. \end{array} \tag{1.1}$$

where $\mathcal{P}$ is the set of secondary cells to be suppressed. The sensitive cell protection is guaranteed if and only if:

$$\underline{a_s} \leq a_s - lpl_s \quad \text{and} \quad \overline{a_s} \geq a_s + upl_s \tag{1.2}$$

The classical model for CSP was originally formulated in [50]. It considers two sets of variables: (1) $y_i \in \{0, 1\}, i = 1, \ldots, n$, is 1 if cell i has to be suppressed, 0 otherwise; (2) 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_i values. The resulting model is:

$$
\begin{array}{lllll}
\min & \displaystyle\sum_{i=1}^{n} w_i y_i & & & \\
\text{s. to} & & \left.\begin{array}{rcll}
& Ax^{l,s} & = 0 & \\
(l_i - a_i)y_i \leq & x_i^{l,s} & \leq (u_i - a_i)y_i & i = 1,\ldots,n \\
& x_s^{l,s} & \leq -lpl_s & \\
& & & \\
& Ax^{u,s} & = 0 & \\
(l_i - a_i)y_i \leq & x_i^{u,s} & \leq (u_i - a_i)y_i & i = 1,\ldots,n \\
& x_s^{u,s} & \geq upl_s &
\end{array}\right\} \ \forall\, s \in \mathcal{S} & & \\
& y_i \in \{0,1\} \quad i = 1,\ldots,n. & & &
\end{array}
\qquad (1.3)
$$

The inequality constraints of (1.3) 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 (1.3) guarantee that the solutions of the linear programs (1.1) will satisfy (1.2).

From a computational point of view, CSP is very large even for tables of moderate size and number of primary cells. Note that (1.3) gives rise to a MILP problem of 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, we obtain a MILP with 8000000 continuous variables, 4000 binary variables, and 21000000 constraints. Because of that, it has been solved in the past by cutting planes or Benders decomposition approaches [33]. In chapter 4, an outline of the Benders decomposition algorithm applied to CSP is presented.

Controlled tabular adjustment

Controlled tabular adjustment [11, 21] (also known as minimum-distance controlled tabular adjustment or simply CTA) is a perturbative recent technique for the protection of any tabular data. It was empirically observed in [13] that estimates $\hat{T} = \hat{F}^{-1}(T')$, $\hat{F}^{-1}$ being an estimate of F^{-1} for CTA, were not close to T for some real tables. CTA can be considered a safe method for the tables tested. Moreover, the quality of CTA solutions has shown to be high [15, 16], higher than that provided by alternative methods in some real instances.

The goal of CTA is: given a table with any structure, to find the closest *safe*

	C_1	C_2	C_3	C_4	Total
R_1	10	15	11	9	45
R_2	8	**10**$^{(3)}$	**12**$^{(4)}$	15	45
R_3	10	12	**11**$^{(2)}$	**13**$^{(5)}$	46
Total	28	37	34	37	136

(a)

	C_1	C_2	C_3	C_4	Total
R_1	11	18	11	5	45
R_2	8	**7**	**16**	14	45
R_3	9	12	**7**	**18**	46
Total	28	37	34	37	136

(b)

Figure 1.4: Small table for optimal CTA method: (a) Original table, sensitive cells are in boldface. Symmetric protection limits lpl_i and upl_i are in brackets. Weights are cell values ($w_i = a_i$). (b) Optimum protected table, after CTA protection method is applied.

table to the original one. This is achieved by adding the minimum amount of deviations (or perturbations) to the original cells that make the released table safe. Safety is guaranteed by imposing that sensitive cells in the new protected table are far enough from the original value. This means the cell value is either above *or* below some certain values, thus a disjunctive constraint involving a binary variable is needed for each sensitive cell. The minimum amount of above or below perturbations required for each sensitive cell are named, respectively, *upper protection* and *lower protection* levels. Changes in sensitive cells force other changes in the remaining cells to guarantee that the value of total or marginal cells is preserved.

Figure 1.4 illustrates CTA on a small two-dimensional table with four sensitive cell in boldface, where symmetric lower and upper protection levels are in brackets (Table (a) from Figure 1.4). Depending on the protection sense of the sensitive cell, either lower or upper (decided in an optimum way by CTA), the value to be published for this cell will be respectively less or equal than the original cell value minus the lower protection level or greater or equal than the original cell value plus the upper protection level. Note that some non sensitive cells are modified to guarantee that total or marginal cells are preserved.

Although it is a recent approach, CTA is gaining recognition among NSAs; for instance, CTA is considered a relatively new emerging method in the recent monographs [48, 49]. We recently implemented a package for CTA in [17] in collaboration with the NSAs of Germany and the Netherlands, within a project funded by Eurostat, the Statistical Office of the European Communities. This package has been largely improved within the FP7-INFRA-2010-262608 project funded by the European Union, with the participation, among others, of the NSAs of Germany, Netherlands, Finland, Sweden and Slovenia. This CTA software is included in the τ-Argus package [47] (http://neon.vb.cbs.nl/casc/tau.htm), used by many European NSAs for the protection of tabular data. Among the recent literature on CTA variants we find [18, 45]. In recent specialized workshops on statistical disclosure control, some NSAs stated that perturbative methods, like CTA, are gaining acceptance [64], and perturbative approaches are being used for the protection of national census tables (e.g., [40] for Germany). CTA has also been used within other wider protection schemes, such as the pre-tabular protection method of [39]. In addition, some NSAs are questioning current non-perturbative protection methods because "the task of balancing confidentiality and usability [...] is nearly impossible" [60]. Therefore there is a need for new methods, and this justifies the research on CTA and other approaches. Indeed, there is no actually any protection method that fits the needs of all NSAs in the world.

From a computational point of view, the size of the CTA optimization problem is by far smaller than for other well-known protection methods, such as the cell suppression problem (CSP). Despite these nice features, CTA formulates a challenging mixed integer linear problem (MILP) for current state-of-the-art solvers (such as Cplex or XPress). Optimal (or suboptimal, e.g., with a 5% gap) solutions may require many hours of execution for medium instances; very large or massive tables can not be tackled with current technology. Even finding a feasible initial solution may be a challenging task for large instances.

Since the purpose of CTA is to find the closest safe values x_i to a_i and considering any distance ℓ, CTA can be formulated as:

$$\begin{array}{ll} \min\limits_{x} & ||x-a||_{\ell} \\ \text{s. to} & Ax = b \\ & l_{a_i} \le x_i \le u_{a_i} \quad i \in \mathcal{N} \\ & x_i \le a_i - lpl_i \text{ or } x_i \ge a_i + upl_i \quad i \in \mathcal{S}. \end{array} \tag{1.4}$$

The disjunctive constraints of (1.4) guarantee the published value is safely out of the interval $(a_i - lpl_i, a_i + upl_i)$. Problem (1.4) can also be formulated in terms of deviations from the current cell values. Defining $z_i = x_i - a_i$, $i \in \mathcal{N}$,and similarly

$l_{z_i} = l_{a_i} - a_i$ and $u_{z_i} = u_{a_i} - a_i$, (1.4) can be recast as

$$\begin{array}{ll} \min\limits_{z} & ||z||_\ell \\ \text{s. to} & Az = 0 \\ & l_{z_i} \leq z_i \leq u_{z_i} \quad i \in \mathcal{N} \\ & z_i \leq -lpl_i \text{ or } z_i \geq upl_i \quad i \in \mathcal{S}, \end{array} \tag{1.5}$$

$z \in \mathbb{R}^n$ being the vector of cell deviations. Using the ℓ_1 or the Manhattan distance and the cell weights w_i, the objective function is $\sum_{i\in\mathcal{N}} w_i|z_i|$. Since w_i are nonnegative, splitting the vector of deviations z in two nonnegative vectors $z^+ \in \mathbb{R}^n$ and $z^- \in \mathbb{R}^n$, model (1.5) with the ℓ_1 distance can thus be written as

$$\begin{array}{lll} \min\limits_{z^+,z^-,y} & \sum\limits_{i\in\mathcal{N}} w_i(z_i^+ + z_i^-) & \\ \text{s. to} & A(z^+ - z^-) = 0 & \\ & 0 \leq z_i^+ \leq \quad u_{z_i} & i \in \mathcal{N} \setminus \mathcal{S} \\ & 0 \leq z_i^- \leq -l_{z_i} & i \in \mathcal{N} \setminus \mathcal{S} \\ & upl_i\ y_i \leq z_i^+ \leq u_{z_i}\ y_i & i \in \mathcal{S} \\ & lpl_i(1-y_i) \leq z_i^- \leq -l_{z_i}(1-y_i) & i \in \mathcal{S} \\ & y_i \in \{0,1\}, & i \in \mathcal{S}, \end{array} \tag{1.6}$$

with $y \in \mathbb{R}^s$ being the vector of binary variables associated with protection directions. When $y_i = 1$ the constraints mean $upl_i \leq z_i^+ \leq u_{z_i}$ and $z_i^- = 0$, thus the protection direction is “upper”; when $y_i = 0$ we get $z_i^+ = 0$ and $lpl_i \leq z_i^- \leq -l_{z_i}$, thus the protection direction is “lower”.

1.1.4 Contributions

The research carried out in this thesis contributes to improving two important areas of optimization: heuristic techniques and decomposition methods. Even though our focus is on tabular data protection those contributions can be applied to general problems. The three main contributions are outlined below.

Firstly, we improved a known and successful heuristic for finding feasible solutions of MILPs, called feasibility pump (FP). The problem of finding a feasible solution of a MILP is known to be NP-hard problem. Moreover, many end users give priority to fast executions and are thus satisfied in practice with suboptimal solutions. Briefly, FP alternates between two sequences of points: one of feasible solutions for the relaxed problem (but not integer), and another of integer points (but not feasible for the relaxed problem). Hopefully, the procedure may eventually converge to a feasible and integer solution. Integer points are obtained from the feasible ones by some rounding procedure. We extend FP, such that the integer point is obtained by rounding a point on the (feasible) segment between the computed feasible point and the analytic center for the relaxed linear problem. Since points in the segment are closer (may be even interior) to the convex hull of integer solutions, it may be expected that the rounded point has more chances to become feasible, thus reducing the number of FP iterations.

The second contribution consists in the application of the fix-and-relax heuristic (FR) to the CTA method. Finding optimal (or suboptimal) solutions or even finding a feasible initial solution for CTA may be a complex task that requires many hours of execution. We present fix-and-relax heuristic as an efficient method applied to the CTA, either alone or in combination with other heuristics.

Finally, the last contribution of this thesis deals with Benders decomposition, successfully applied in many real-world applications, which allows to decompose the difficult original MILP in several smaller sub-problems. Despite its successful application, the convergence to optimal solution is often too slow due to well-known instability issues that limit their efficiency in very large-scale problems. This is mainly due to the fact that the solutions tend to oscillate wildly between different feasible regions, so we can move from a good point to a much worse one. This behaviour is prevented by using a stabilized Benders decomposition, which focus on finding new solutions as close as possible to well considered points.

This PhD thesis gave rise to the following publications in peer-reviewed journals, scientific conferences and research reports:

- Publications:
  - D. Baena, J. Castro, Using the analytic center in the feasibility pump, Operations Research Letters, 39 (2011) 310-317. It corresponds to Chapter 2 of this PhD thesis.
  - D. Baena, J. Castro, J. A. González, Fix-and-relax approaches for controlled tabular adjustment, Computers & Operations Research, 58 (2015) 41-52. It corresponds to Chapter 3 of this PhD thesis.
  - D.Baena, J.Castro, A. Frangioni, Stabilized Benders methods for large combinatorial optimization problems: applications to cell suppression,

working paper to be submitted. It corresponds to Chapter 4 of this PhD thesis.

- Scientific conferences:
  - D. Baena, J. Castro, J.A. González, Fix-and-relax approaches for controlled tabular adjustment, XXXV Congreso Nacional de Estadística e Investigación Operativa, Pamplona, Spain, May 2015.
  - D. Baena, J. Castro, A fix and relax heuristic for controlled tabular adjustment, 25th European Conference on Operational Research-EURO 2012, Vilnius University, Vilnius (Lithuania), July 2012. **Invited presentation.**
  - D. Baena, J. Castro, The analytic center feasibility pump, XXXIII Congreso Nacional de Estadística e Investigación Operativa, Madrid, Spain, April 2012.

Chapter 2

Using the analytic center in the feasibility pump

2.1 Introduction

In the previous chapter, we outlined the two most important methods for tabular data protection (CTA and CSP). Both protection methods are a challenging mixed integer linear problem (MILP) for current state-of-the-art solvers. Optimal (or suboptimal, e.g., with a 5% gap) solutions may require many hours of execution for medium instances; very large or massive tables can not be tackled with current technology. A big effort to reach an optimal solution may not make sense because, in practice, end users of tabular data protection techniques give priority to fast executions and are thus satisfied with suboptimal solutions. However, finding a feasible initial solution for large instances is a NP-hard problem. For this reason, heuristics are a very important tool in mixed-integer optimization to ensure feasible and hopefully good solutions for a particular problems in a reasonable computational time. Moreover, they are also very useful to warm start other methods such as branch-and-cut. It could be expected that providing a good incumbent from the beginning would significantly reduce the computational time for current state-of-the-art solvers. It is worth noting that some methods can only be applied if an initial feasible solution is known. In [7, 35] the authors proposed a new heuristic approach to compute MILP solutions, named the *feasibility pump* (FP). This heuristic turned out to be successful in finding feasible solutions even for some hard MILP instances. A slight modification of FP was suggested in [2], named the *objective feasibility pump*, in order to improve the quality of the solutions in terms of the objective value. The main difference between the two versions is that the objective FP, in contrast to the original version, takes the objective function of the MILP into account during the execution of the algorithm. FP alternates between feasible (for the linear relaxation of MILP) and integer points, hopefully converging to a feasible integer solution. The integer point is obtained by applying some rounding procedure to the feasible solution. This thesis suggests an extension of FP where all the points in a feasible segment are candidates to be rounded. The end points of this segment are the feasible point of the standard or objective FP and some interior point of the polytope of the relaxed problem, the analytic center being the best candidate (our approach will be named analytic center FP, or AC-FP). When the end point of the segment in the boundary of the polytope is considered for rounding, we obtain the standard FP algorithm. The motivation of this approach is that rounding a point of the segment closer to the analytic center may increase the chances of obtaining a feasible integer point in some instances, thus reducing the number of FP iterations. The computational results with AC-FP show that, for some instances, taking a point in the interior of the feasible segment may be more effective than the standard end point of the objective FP. A recent version of FP [34] introduced a new improved rounding scheme based on constraint propagation. Although in our research we considered as base code a freely available implementation of the objective FP, AC-FP could also be used with the new rounding scheme of [34]. Interior-point methods have been applied in the past in branch-and-bound frameworks for MILP and mixed integer nonlinear problems (MINLP) [6, 9, 52, 53]. However, as far as we know, the only previous attempt to apply them to a primal heuristic was [56]. Although AC-FP and the approach of [56] (named analytic center feasibility method (ACFM)) have the same motivation (using the analytic center for getting MILP feasible solutions), both approaches are significantly different, as shown at the end of Subsection 2.4.2. Briefly, (i) AC-FP relies on FP, while ACFM is based on an analytic center cutting plane method; (ii) AC-FP only computes one analytic center, while ACFM computes one per iteration; (iii) as a consequence of the previous point, ACFM can be computationally expensive, while AC-FP is almost as efficient as FP. Later to the publication of our work, [8] presents a FP approach improved, which efficiently also explores all rounded solutions along a line segment.

initialize $t := 0$ and $x^* := \arg\min\{c^T x : Ax = b, x \geq 0\}$
if $x^*_{\mathcal{I}}$ is integer **then** return(x^*) **end if**
$\tilde{x} := [x^*]$ (rounding of x^*)
while time < TimeLimit **do**
    $x^* := \arg\min\{\triangle\ (x, \tilde{x}) : Ax = b, x \geq 0\}$
    if $x^*_{\mathcal{I}}$ is integer **then** return(x^*) **end if**
    if $\exists j \in \mathcal{I} : [x^*_j] \neq \tilde{x}_j$ **then**
        $\tilde{x} := [x^*]$
    else
        restart
    end if
    $t := t + 1$
end while
return(FP failed)

Figure 2.1: The feasibility pump heuristic (original version).

2.2 The feasibility pump heuristic

2.2.1 The original feasibility pump

Consider a generic mixed integer linear problem (MILP) of the form:

$$
\begin{aligned}
\min_{x} \quad & c^T x \\
\text{s. to} \quad & Ax = b \\
& x \geq 0 \\
& x_j \quad integer \quad \forall j \in \mathcal{I},
\end{aligned}
\tag{2.1}
$$

where A $\in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$ and $\mathcal{I} \subseteq \mathcal{N} = \{1, \ldots, n\}$, is the subset of integer variables.

The FP heuristic starts by solving the linear programming (LP) relaxation of (2.1)

$$\min_{x}\{c^T x \ : \ Ax = b, x \geq 0\}, \tag{2.2}$$

and its solution x^* is rounded to an integer point $\tilde{x}$, which may be infeasible for (2.2). The rounding $\tilde{x}$ of a given x^*, denoted as $\tilde{x} = [x^*]$, is obtained by setting $\tilde{x}_j = [x^*_j]$ if $j \in \mathcal{I}$ and $\tilde{x}_j = x^*_j$ otherwise, where $[\cdot]$ represents scalar rounding to the nearest integer. If $\tilde{x}$ is infeasible, FP finds the closest $x^* \in P$, where

$$P = \{x \in \mathbb{R}^n \ : \ Ax = b, x \geq 0\}, \tag{2.3}$$

by solving the following LP

$$x^* = \arg\min\{\triangle\ (x, \tilde{x}) : Ax = b, x \geq 0\}, \tag{2.4}$$

$\triangle(x, \tilde{x})$ being defined (using the ℓ_1 norm) as

$$\triangle(x, \tilde{x}) = \sum_{j \in \mathcal{I}} |x_j - \tilde{x}_j|. \tag{2.5}$$

Notice that continuous variables $\tilde{x}_j$, $j \notin \mathcal{I}$, do not play any role. If $\triangle(x^*, \tilde{x}) = 0$ then $x^*_j (= \tilde{x}_j)$ is integer for all $j \in \mathcal{I}$, so x^* is a feasible solution for (2.1). If not, FP finds a new integer point $\tilde{x}$ from x^* by rounding. The pair of points $(\tilde{x}, x^*)$ with $\tilde{x}$ integer and $x^* \in P$ are iteratively updated at each FP iteration with the aim of reducing as much as possible the distance $\triangle(x^*, \tilde{x})$. An outline of the FP algorithm is showed in Figure 2.1. To avoid that the procedure gets stuck at the same sequence of integer and feasible, there is a restart procedure when the previous integer point $\tilde{x}$ is revisited (lines 7–11 of algorithm of Figure 2.1). In a restart, a random perturbation step is performed.

The FP implementation has three stages. *Stage 1* is performed just on the binary variables by relaxing the integrality conditions on the general integer variables. In *stage 2* FP takes all integer variables into account. The FP algorithm exits stage 1 and goes to stage 2 when either (a) a feasible point with respect to only the binary variables has been found; (b) the minimum $\triangle(x^*, \tilde{x})$ was not updated during a certain number of iterations; or (c) the maximum number of iterations was reached. The point $\tilde{x}$ that produced the smallest $\triangle(x^*, \tilde{x})$ is stored and passed to stage 2 as the initial $\tilde{x}$ point. When FP turns out to be unable to find a feasible solution within the provided time limit, the default procedure of the underlying MILP solver (CPLEX 12 [1] in this work) is started; this is named *stage 3*.

2.2.2 The modified objective feasibility pump

According to [2], although the original FP heuristic of [7, 35] has proved to be a very successful heuristic for finding feasible solutions of mixed integer programs, the quality of their solutions in terms of objective value tends to be poor. In the original FP algorithm of [7, 35] the objective function of (2.1) is only used at the beginning of the procedure. The purpose of the objective FP [2] is, instead of instantly discarding the objective function of (2.1), to consider a convex combination of it and $\triangle(x, \tilde{x})$, reducing gradually the influence of the objective term. The hope is that FP still converges to a feasible solution but it concentrates the search on the region of high-quality points. The modified objective function

$\triangle_\alpha(x, \tilde{x})$ is defined as

$$\triangle_\alpha(x, \tilde{x}) := (1-\alpha)\triangle(x, \tilde{x}) + \alpha\frac{||\triangle||}{||c||}c^T x, \quad \alpha \in [0,1], \tag{2.6}$$

where $||\,.\,||$ is the Euclidean norm of a vector, and $\triangle$ is the objective function vector of $\triangle(x, \tilde{x})$ (i.e., at stage 1 is the number of binary variables, and at stage 2 is the number of integer (both general integer and binary) variables). At each FP iteration α is geometrically decreased with a fixed factor $\varphi < 1$, i.e., $\alpha_{t+1} = \varphi\alpha_t$ and $\alpha_0 \in [0,1]$. Notice that the original FP algorithm is obtained using $\alpha_0 = 0$. The objective FP algorithm is basically the same as the original FP algorithm of Figure 2.1, replacing $\triangle(x, \tilde{x})$ by $\triangle_{\alpha_t}(x^*, \tilde{x})$ at line 5, performing at the beginning the initialization of α_0, and adding at the end of the loop $\alpha_{t+1} = \varphi\alpha_t$.

2.3 The analytic center feasibility method (ACFM)

ACFM [56] is a three-phase procedure that mainly relies on the analytic center cutting plane method. In phase-I it computes (i) the analytic center $\bar{x}$ of the bounded polyhedron

$$P \cap \{x : c^T x \leq z\} \cap C, \tag{2.7}$$

z being an upper bound on the objective function and C a set of valid cuts (initially empty), and (ii) the minimizer $x^*_{\min}$ and maximizer $x^*_{\max}$ of the objective function $c^T x$ subject to $x \in P$. Actually, the formulation in [56] of the problem for computing the analytic center is different from the above one, since it considers only inequalities, and it needs a reformulation of equality constraints; our approach, detailed in Section 2.4 below, directly works with the original formulation of the problem, as it can deal with equality constrained problems. Scanning the segments $\overline{\bar{x}\ x^*_{\min}}$ and $\overline{\bar{x}\ x^*_{\max}}$, phase-I tries to obtain the closest integer point to the analytic center by rounding the integer components of different segment points—let us name $\tilde{x}$ such a rounded point—and adjusting the remaining continuous components by solving

$$\min_x\{c^T x \ : \ Ax = b, x \geq 0, x_j = \tilde{x}_j \ j \in \mathcal{I}\}. \tag{2.8}$$

If (2.8) is feasible then an integer feasible solution is obtained. Whether this problem is feasible or not, phase-II is started. If phase-I found a feasible integer point, the upper bound z on the objective is updated and we go to phase-I again, to recompute the new analytic center (different from previous iteration, since z, thus (2.7), changed). If no feasible integer point was found at phase-I, then additional constraints (cuts) are added to C to move the analytic center towards the interior of the integer feasible region, and phase-I is restarted again (computing a new analytic center for the new polyhedron (2.7)). The procedure iterates Phase-I and Phase-II until some stopping criteria is satisfied (iteration limit—20 iterations in [56]—, or quality of the solution). If no feasible solution is found the procedure switches to a phase-III which is similar to the stage 3 of FP.

2.4 The analytic center feasibility pump (AC-FP)

2.4.1 The analytic center

Given the LP relaxation (2.2), its analytic center is defined as the point $\bar{x} \in P$ that minimizes the *primal potential function* $-\sum_{i=1}^{n} \ln x_i$, i.e.,

$$\begin{aligned} \bar{x} = \arg\min_{x} \quad & -\textstyle\sum_{i=1}^{n} \ln x_i \\ \text{s. to} \quad & Ax = b \\ & x > 0. \end{aligned} \tag{2.9}$$

Note that the analytic center is well defined only if P is bounded. Note also that constraints $x > 0$ could be avoided, since the domain of ln are the positive numbers. Problem (2.9) is a linearly constrained strictly convex optimization problem. It is easily seen that the $\arg\min -\sum_{i=1}^{n} \ln x_i$ is equivalent to the $\arg\max \prod_{i=1}^{n} x_i$. Therefore, the analytic center provides the point that maximizes the distance to the hyperplanes $x_i = 0, i = 1, \ldots, n$, and it is thus expected to be well centered in the interior of the polytope P. We note that the analytic center is not a topological property of a polytope, and it depends on how the polytope is represented. That is, two different sets of linear inequalities P and P' defining the same polytope may provide different analytic centers. Other centers, such as the center of gravity, are not affected by different formulations of the same polyhedron (but they are computationally more expensive). In this sense, redundant inequalities may change the location of the analytic center (i.e., if formulation P' is obtained from formulation P by adding redundant constraints, it will provide a different analytic center). Additional details can be found in [63].

The analytic center may be computed by solving the KKT conditions of (2.9)

$$\begin{array}{rcll} Ax & = & b & \\ A^T y + s & = & 0 & \\ x_i s_i & = & 1 & i = 1, \dots, n \\ (x, s) & > & 0, & \end{array} \tag{2.10}$$

$y \in \mathbb{R}^m$ and $s \in \mathbb{R}^n$ being the Lagrange multipliers of $Ax = b$ and $x > 0$ respectively. Alternatively, and in order to use an available highly efficient implementation, the analytic center was computed in this work by applying a primal-dual path-following interior-point algorithm to the barrier problem of (2.2), after removing the objective function term (i.e., setting $c = 0$):

$$\begin{array}{ll} \min\limits_{x} & -\mu \sum_{i=1}^{n} \ln x_i \\ \text{s. to} & Ax = b \\ & x > 0, \end{array} \tag{2.11}$$

where μ is a positive parameter (the parameter of the barrier) that tends to zero. The arc of solutions of (2.11) $x^*(\mu)$ is named the primal central path. The central path converges to the analytic center of the optimal set. When $c = 0$ (as in (2.11)) the central path converges to the analytic center of the feasible set P [63].

2.4.2 Using the analytic center in the feasibility pump heuristic

Once the analytic center has been computed, it can be used to (in theory infinitely) increase the number of feasible points candidates to be rounded. Instead of rounding, at each FP iteration, the feasible point $x^* \in P$, points on the segment

$$x(\gamma) = \gamma \bar{x} + (1 - \gamma) x^* \quad \gamma \in [0, 1] \tag{2.12}$$

will be considered. Note that the segment is feasible, since it is a convex combination of two feasible points.

AC-FP first considers a *stage 0* (which is later applied at each FP iteration) where several $x(\gamma)$ points are tested, from $\gamma = 0$ to $\gamma = 1$ (i.e, from x^* to $\bar{x}$). Each $x(\gamma)$ is rounded to $\tilde{x}(\gamma)$. If $\tilde{x}(\gamma)$ is feasible, then a feasible integer solution was found and the procedure is stopped at the stage 0. Otherwise the algorithm

1. initialize $t := 0$, $\alpha_0 \in [0,1]$, $\varphi \in [0,1]$, and $x^* := \arg\min\{c^T x : Ax = b, x \geq 0\}$
2. compute analytic center $\bar{x} := \arg\min\left\{-\sum_{i=1}^{n} \ln x_i : Ax = b, x > 0\right\}$
3. { *Beginning of stage 0*}
4. **for** $\gamma \in [0,1]$ **do**
5. $x(\gamma) := \gamma\bar{x} + (1-\gamma)x^*$
6. $\tilde{x}(\gamma) := [x(\gamma)]$ (rounding of $x(\gamma)$)
7.   **if** $\tilde{x}(\gamma)$ is feasible **then** return($\tilde{x}(\gamma)$) **end if**
8. **end for**
9. { *End of stage 0*}
10. select $\tilde{x}$ from the set $\{\tilde{x}(\gamma)\}$
11. **while** time < TimeLimit **do**
12. $x^* := \arg\min\{\Delta_{\alpha_t}(x, \tilde{x}) : Ax = b, x \geq 0\}$
13.   **for** $\gamma \in [0,1]$ **do**
14. $x(\gamma) := \gamma\bar{x} + (1-\gamma)x^*$
15. $\tilde{x}(\gamma) := [x(\gamma)]$ (rounding of $x(\gamma)$)
16.   **if** $\tilde{x}(\gamma)$ is feasible **then** return($\tilde{x}(\gamma)$) **end if**
17.   **end for**
18.   select $\hat{x}$ from the set $\{\tilde{x}(\gamma)\}$
19.   **if** $\hat{x}_{\mathcal{I}} \neq \tilde{x}_{\mathcal{I}}$ **then**
20. $\tilde{x} := \hat{x}$
21.   **else**
22.   restart
23.   **end if**
24. $\alpha_{t+1} := \varphi\alpha_t$
25. $t := t+1$
26. **end while**
27. return(FP failed)

Figure 2.2: The AC-FP heuristic.

proceeds with the next stage of FP, considering two different options:

a) using the point $\tilde{x}(0) = [x^*]$ (option $\gamma = 0$);

b) using the point $\tilde{x}(\gamma)$ that minimizes $||\tilde{x}(\gamma) - x(\gamma)||_\infty$ (option ℓ_∞).

If the first option is applied at each FP iteration, and no feasible $\tilde{x}(\gamma)$ for $\gamma > 0$ is found, AC-FP behaves as the standard FP algorithm. In the second option, if no feasible $\tilde{x}(\gamma)$ is found, the procedure selects the $x(\gamma)$ which is closer to $[x(\gamma)]$ according to the ℓ_∞ norm. The aim is to select the point with more chances to become both integer and feasible, in an attempt to reduce the number of FP iterations. This second option provided better results in general and it was used in the computational results of Section 3.4. It is worth to note that if the rounding of several $x(\gamma)$ points is feasible, the procedure selects the one with the lowest γ, i.e., the one closest to x^* (instead of the one closest to the analytic center $\bar{x}$), since this point was computed considering the objective function (for $\alpha > 0$). An outline of the algorithm is shown in Figure 2.2.

From Figure 2.2 it is clear that AC-FP only computes one analytic center (that of P) at line 2 of the algorithm, unlike ACFM [56] which computes one analytic center (for a modified polyhedron) at each iteration. This is computationally the most significant difference between AC-FP and ACFM: since the computation of analytic centers can be expensive, AC-FP is more efficient than ACFM. It is also seen that AC-FP and ACFM are completely different approaches: the former is an extension of FP, the latter is based on computing analytic centers of modified polyhedrons obtained by adding cutting planes to P.

Both procedures, AC-FP and ACFM, consider the feasible segment between the analytic center $\bar{x}$ and a solution of the relaxed problem (x^* in AC-FP, $x^*_{\min}$ and $x^*_{\max}$ in ACFM) for rounding purposes. It is worth to note that in AC-FP the analytic center is the same for all the iterations and x^* is different at each iteration, whereas the opposite holds for ACFM: it computes a different analytic center at each iteration whereas $x^*_{\min}$ and $x^*_{\max}$ are uniquely determined at the beginning. In addition, AC-FP and ACFM use the rounded point $\tilde{x}(\gamma)$ in a different manner. AC-FP checks if $\tilde{x}(\gamma)$ is feasible, and stops the procedure once the first feasible $\tilde{x}(\gamma)$ is found (which is indeed the criterion considered by FP). On the other hand, ACFM, which may obtain a rounded feasible point at its phase-I, keeps on iterating with phase-I and phase-II until some stopping criteria (i.e., time limit or quality of the solution) is satisfied. In addition, after obtaining the rounded point, ACFM solves (2.8) for adjusting the remaining continuous components (this is not done by AC-FP, which relies on the overall FP procedure for performing a similar adjustment at line 12 of the algorithm of Figure 2.2). Since AC-FP may obtain a feasible point at stage 0 close to the analytic center $\tilde{x}$ and far from the feasible point $x^* \in P$, this point may provide a very large objective function value. An extension would be to save this point and keep on looking for new feasible points of higher quality (as done by ACFM).

As stated in Subsection 2.3, ACFM computes two linear feasible points $x^*_{\min}$ and $x^*_{\max}$, the minimizer and maximizer of $c^T x$ within P, and it considers the two segments that join the analytic center of the current ACFM iteration with those two points. On the other hand, AC-FP only considers one segment between $\bar{x}$ and x^*. Actually, we initially also considered two segments: the current one $\overline{\bar{x}\ x^*}$, and a second one joining $\bar{x}$ with the farthest feasible point from $\bar{x}$ in the direction $\bar{x} - x^*$ (name it x^*_f). Note that this point is easily computed as $x^*_f = \bar{x} + \beta^*(\bar{x} - x^*)$, where $\beta^* = \min\{\frac{-x_i}{(\bar{x}-x^*)_i} : (\bar{x} - x^*)_i < 0, i = 1, \ldots, n\}$. The computational benefit of using x^*_f instead of $x^*_{\max}$ is that the solution of an extra LP problem is avoided. However, in practice, using the second segment $\overline{\bar{x}\ x^*_f}$ was not useful, and it was discarded in the final AC-FP implementation.

2.5 Computational results

AC-FP was implemented using the base code of the objective FP, freely available from http://www.or.deis.unibo.it/research_pages/ORcodes/FP-gen.html. The base FP implementation was extended for computing the analytic center using three different interior-point solvers, CPLEX [1], GLPK [42] and PCx [20]. The new code is available from http://www-eio.upc.es/~dbaena/sw/2010/fp_analytic_center.tgz. CPLEX integrates better with the rest of the FP code, which also relies on CPLEX, and it also turned out to be significantly more efficient than GLPK and PCx. On the other hand, even deactivating all the preprocessing options and removing the crossover postprocess, CPLEX was not always able to provide the analytic center of P because of its aggressive reduced preprocessing (which can not be deactivated as we were told by CPLEX developers). For instance, for $P = \{x : \sum_{i=1}^{n} x_i = n, x \geq 0\}$, the barrier option of CPLEX did not apply the interior-point algorithm, not providing an interior solution (i.e., it provided $x_i = n$, $x_j = 0, j \neq i$), whereas both GLPK and PCx reported the right analytic center $x_i = 1, i = 1, \ldots, n$. Of the other two solvers, PCx turned out to be much more efficient than GLPK. Indeed, PCx may handle upper bounds implicitly (i.e., $0 \leq x \leq 1$ from linear relaxations of $x \in \{0, 1\}$) in its interior-point implementation, whereas GLPK transforms the problem to the standard form (replacing $x \leq 1$ by $x + s = 1, s \geq 0$), significantly increasing the size of the Newton's system to be solved at each interior-point iteration.

The AC-FP implementation was applied to a subset of MIPLIB2003 instances, whose dimensions are shown in Table 2.1. Columns "rows", "cols", "nnz", "int", "bin" and "con" provide respectively the number of constraints, variables, nonzeros, general integer variables, binary variables, and continuous variables of the instances. Column "objective" shows the optimal objective function. Unknown optimal objectives are marked with a "?".

Table 2.2 shows the results obtained with AC-FP using PCx and CPLEX-12.1. For the two AC-FP variants, Table 2.2 reports the number of FP iterations (columns "niter"), the objective value of the feasible point found ("fobj"), the gap between the feasible and the optimal solution ("gap%"), and the FP stage where the feasible point was found ("stage"). Columns "tFP(tAC)" report separately the CPU time spent in stages 1 to 3 ("tFP") and the time for computing the analytic center before stage 0 (in brackets, "tAC"); the total time is the sum of "tFP" and "tAC". Columns "AC value" show the value of the original objective function eval-

Instance	rows	cols	nnz	int	bin	con	objective
10teams	230	2025	12150	0	1800	225	924
a1c1s1	3312	3648	10178	0	192	3456	11503.40
aflow30a	479	842	2091	0	421	421	1158
aflow40b	1442	2728	6783	0	1364	1364	1168
air04	823	8904	72965	0	8904	0	56137
air05	426	7195	52121	0	7195	0	26374
arki001	1048	1388	20439	96	415	877	7580810
atlanta-ip	21732	48738	257532	106	46667	1965	90.00
cap6000	2176	6000	48243	0	6000	0	-2451380
dano3mip	3202	13873	79655	0	552	13321	?
danoint	664	521	3232	0	56	465	65.66
disctom	399	10000	30000	0	10000	0	-5000
ds	656	67732	1024059	0	67732	0	93.52
fast0507	507	63009	409349	0	63009	0	174
fiber	363	1298	2944	0	1254	44	405935
fixnet6	478	878	1756	0	378	500	3983
gesa2-o	1248	1224	3672	336	384	504	25779900
gesa2	1392	1224	5064	168	240	816	25779900
glass4	396	322	1815	0	302	20	1200010000
harp2	112	2993	5840	0	2993	0	-73899800
liu	2178	1156	10626	0	1089	67	?
manna81	6480	3321	12960	3303	18	0	-13164
markshare1	6	62	312	0	50	12	1
markshare2	7	74	434	0	60	14	1
mas74	13	151	1706	0	150	1	11801.20
mas76	12	151	1640	0	150	1	40005.10
misc07	212	260	8619	0	259	1	2810
mkc	3411	5325	17038	0	5323	2	-563.84
mod011	4480	10958	22254	0	96	10862	-54558500
modglob	291	422	968	0	98	324	20740500
msc98-ip	15850	21143	92918	53	20237	853	19839500
mzzv11	9499	10240	134603	251	9989	0	-21718
mzzv42z	10460	11717	151261	235	11482	0	-20540
net12	14021	14115	80384	0	1603	12512	214
noswot	182	128	735	25	75	28	-41
nsrand-ipx	735	6621	223261	0	6620	1	51200
nw04	36	87482	636666	0	87482	0	16862
opt1217	64	769	1542	0	768	1	-16
p2756	755	2756	8937	0	2756	0	3124
pk1	45	86	915	0	55	31	11
pp08aCUTS	246	240	839	0	64	176	7350
pp08a	136	240	480	0	64	176	7350
protfold	2112	1835	23491	0	1835	0	-31
qiu	1192	840	3432	0	48	792	-132.87
roll3000	2295	1166	29386	492	246	428	12890
rout	291	556	2431	15	300	241	1077.56
set1ch	492	712	1412	0	240	472	54537.80
seymour	4944	1372	33549	0	1372	0	423
sp97ar	1761	14101	290968	0	14101	0	660706000
swath	884	6805	34965	0	6724	81	467.40
timtab1	171	397	829	94	64	239	764772
timtab2	294	675	1482	164	113	398	1096560
tr12-30	750	1080	2508	0	360	720	130596
vpm2	234	378	917	0	168	210	13.75

?: Unknown value

Table 2.1: Characteristics of the subset of MILP instances from MIPLIB 2003.

	AC-FP with PCx					AC-FP with CPLEX				
Instance	niter	fobj	tFP(tAC)	stage	gap%	AC value	niter	fobj	tFP(tAC)	stage	gap%	AC value

Instance	niter	fobj	tFP(tAC)	stage	gap%	AC value	niter	fobj	tFP(tAC)	stage	gap%	AC value
	AC-FP with PCx						AC-FP with CPLEX					
10teams	179	1022	26(0)	3	10.59	1020	177	1056	25(0)	3	14.27	1020
a1c1s1	0	46756.40	0(0)	0	306.43	50396.80	0	38193.60	0(0)	0	232	40504.70
aflow30a	171	3802	1(0)	2	228.13	5377.34	298	5578	2(0)	2	381.36	4714.70
aflow40b	394	8300	12(0)	3	610.09	7234.03	54	7051	2(0)	1	503.25	6635.6
air04	186	72098.00	1220(2)	3	28.43	79260.30	186	71223.99	1147(0)	3	26.87	79989.5
air05	186	37907	162(1)	3	43.73	45309	190	35798	148(0)	3	35.73	45732.10
arki001	871	7729296.21	43(0)	3	1.96	7807100	1573	7763720.15	79(0)	3	2.41	7822170
atlanta-ip	42	198.02	68(9398)	1	118.68	171.31	397	154.01	934(11)	3	70.32	159.76
cap6000	0	-2442800	1(0)	0	0.35	-596562	0	-2442800	1(0)	0	0.35	-109362
dano3mip	205	1000	1892(17)	3	?	12849.20	252	1000	1947(4)	3	?	995.15
danoint	99	76	4(0)	1	15.50	434.456	230	85.50	9(0)	3	29.75	66.77
disctom	4	-5000	3(1)	1	0	-5000	4	-5000	4(0)	1	0	-5000
ds	198	5418.56	1945(10)	3	5633.77	1053.93	0	5418.56	1(2)	0	5633.77	5418.56
fast0507	39	11884	131(4)	1	6691.43	8254.52	0	275	2(1)	0	57.71	122425
fiber	41	6481510	0(0)	1	1496.68	19694200	15	3147830	0(0)	1	675.45	45602200
fixnet6	18	38401	0(0)	1	863.91	60883.20	0	97271.70	0(0)	0	2341.58	101827
gesa2-o	20	71213100	0(0)	2	176.23	116914000	35	32635500	1(0)	2	26.59	166784000
gesa2	3	38472300	1(0)	2	49.23	124095000	47	40307000	1(0)	2	56.35	188208000
glass4	254	10500117800	2(0)	3	775	142889000000	224	5000046800	1(0)	3	316.67	8862840000
harp2	178	-40631391	3(0)	3	45.02	-50758200	59	-49759800	1(0)	1	32.67	-46262500
liu	119	3036	4(5)	1	?	9218.57	121	5876	5(0)	1	?	959.02
manna81	0	-12948	0(6)	0	1.64	-7307.16	0	-12878	0(0)	0	2.17	0
markshare1	65	603	0(0)	1	30100	30.48	0	7286	0(0)	0	364250	7286
markshare2	66	925	0(0)	1	46200	36.10	0	10512	0(0)	0	525550	10512
mas74	0	57195600000	0(0)	0	484618022.50	571956000000	0	50000000000	0(0)	0	423649728.01	1000000000000
mas76	0	26804400000	0(0)	0	67000682.38	536000000000	0	50000000000	0(0)	0	124980840.41	1000000000000
misc07	217	3935	3(0)	2	40.02	3601.66	219	3410	2(0)	2	21.34	4894.40
mkc	13	-276.96	1(0)	1	50.79	-253.58	12	38.81	1(1)	1	106.69	-95.53
mod011	23	-37482400	3(1)	1	31.30	-31430100	23	-35547800	3(0)	1	34.84	-36600000
modglob	60	21809700	1(0)	1	5.16	272286000	0	82243300	0(0)	0	296.53	142949000
msc98-ip	33	30196300	16(949)	1	52.20	29571000	29	30928000	19(22)	1	55.89	29545100
mzzv11	567	-16262	435(116)	3	25.12	-4264.40	561	-13744	484(7)	3	36.71	-4794.93
mzzv42z	23	-12736	13(147)	1	37.99	-3210.77	27	-14192	12(15)	1	30.90	-3825.70
net12	25	337	10(86)	1	57.21	325.12	25	337	8(27)	1	57.21	337
noswot	34	-15	0(0)	2	61.90	-21.82	33	-31	0(0)	2	23.81	-15.67
nsrand-ipx	883	258080	367(2)	3	404.05	761986	694	203040	265(0)	3	296.56	802647
nw04	2	18380	9(8)	1	9	50318.90	42	61640	120(2)	1	265.54	52460.90
opt1217	124	-12.11	0(0)	1	22.80	-8.23	0	0	0(0)	0	94.12	0
p2756	244	51338	7(0)	3	1542.85	139225	279	51338	7(0)	3	1542.85	164724
pk1	57	86	0(0)	1	625	34.13	0	731	0(0)	0	6000	731
pp08aCUTS	11	16390	0(0)	1	122.98	18715	0	21671.40	0(0)	0	194.82	23012.40
pp08a	15	15850	0(0)	1	115.63	21666.70	0	18439.30	0(0)	0	150.85	18778.70
protfold							307	-18.90	365(2)	3	37.81	-18.42
qiu	41	868.57	1(0)	1	748.05	722.04	0	3693.35	0(0)	0	2858.10	4188.61
roll3000	818	40048.40	65(1)	3	210.68	44336.80	175	18507	11(1)	2	43.57	38004.10
rout	79	1644.41	1(0)	1	52.56	1455.68	74	1337.27	1(0)	1	24.08	1474.90
set1ch	0	268719	0(0)	0	392.71	224714	0	216475	0(0)	0	296.92	262834
seymour	39	754	5(35)	1	78.07	728.54	0	588	0(0)	0	38.92	1345
sp97ar	63	1161990000	57(4)	1	75.87	8272000000	97	11702100000	88(1)	1	1671.15	18441700000
swath	795	34774.58	96(1)	3	7324.22	1470.14	795	34774.58	100(0)	3	7324.22	1470.14
timtab1	169	1081000	1(0)	2	41.35	1475570	819	1401240.99	3(0)	3	83.22	419539
timtab2	972	2105005.99	6(0)	3	91.96	2052380	1072	1772242.99	7(0)	3	61.62	671850
tr12-30	214	289227.99	7(0)	3	121.47	135860	221	285716	6(0)	3	118.78	75936.50
vpm2	11	29.50	0(0)	1	106.78	48.48	27	23.75	0(0)	1	67.8	14.08

?: Unknown value

Table 2.2: Computational results using AC-FP with PCx and CPLEX.

Instance	AC-FP with PCx tFP(tAC)	gap%	AC-FP with CPLEX tFP(tAC)	gap%	ACFM niter	fobj	tt(tAC)	gap%
mas74	0(0)	484618022.50	0(0)	423649728.01	7	15026.47	8.89(8.26)	**434.75**
mas76	0(0)	67000682.38	0(0)	124980840.41	1	44877.42	2.55(2.1)	**12.18**
misc07	3(0)	40.02	2(0)	**21.34**	13	4795	9.28(8.71)	70.64
noswot	0(0)	61.90	0(0)	23.81	3	-37	2.51(2.11)	**9.76**
pk1	0(0)	625	0(0)	6000	1	28.99	0.75(0.72)	**163.55**
pp08aCUTS	0(0)	122.98	0(0)	194.82	1	8458	2.81(2.25)	**15.07**
pp08a	0(0)	115.63	0(0)	150.85	1	9048.56	2.07(1.5)	**23.11**
rout	1(0)	52.56	1(0)	24.08	4	1111.88	101.95(100.58)	**3.18**
vpm2	0(0)	106.78	0(0)	67.8	6	15.5	28.43(27.31)	**12.73**

Table 2.3: Comparison of AC-FP (PCx and CPLEX) with ACFM only for the instances solved in [56].

uated at the analytic center. Differences are due to different computed analytic centers because both solvers apply very distinct preprocessing strategies.

Table 2.3 compares AC-FP with ACFM using the subset of nine MIPLIB2003 instances solved in [56]. For ACFM, Table 2.3 reports the number of ACFM iterations needed to reach the feasible solution (“niter“), the feasible solution (column “fobj”), and the gap between the solution found by ACFM and the optimal solution (column “gap%”). Column “tt(tAC)” reports the total CPU time of the ACFM algorithm, including the amount of CPU time in seconds spent on calculating the analytic centers (in brackets, “tAC”). The best result (i.e., execution with the lowest gap) is highlighted in boldface.

Table 2.4 compares AC-FP with the objective FP. For the objective FP we report the number of FP iterations (column “niter”), the objective value of feasible point found (“fobj”), the gap between the feasible and the optimal solution (“gap%”), the FP stage where the feasible point was found (“stage”) and the total CPU time (column “tt”). The best result (i.e. that with the lowest gap if obtained in stages 0–2), is highlighted in boldface. Note that for instance “swath” objective FP is considered the best approach, though the gap is greater than for AC-FP, since the solution with objective FP was found at stage 2, while AC-FP failed and it needed stage 3. This same argument was applied for instance “dano3mip”, of unknown gap. For instance “liu” AC-FP with PCx provided a better objective function, though the gap is also unknown. If two approaches provide the same gap, but one is significantly more efficient, this is marked as the best result (as in instance “ds”).

The default FP settings were used as suggested in [2]. All runs were carried on a Dell PowerEdge 6950 server with four dual core AMD Opteron 8222 3.0 GHZ

	AC-FP with PCx		AC-FP with CPLEX		objective FP				
Instance	tFP(tAC)	gap%	tFP(tAC)	gap%	niter	fobj	tt	stage	gap%
Problems with only binary variables									
10teams	26(0)	10.59	25(0)	14.27	278	1014	19	3	**9.73**
a1c1s1	0(0)	306.43	0(0)	232	351	22714.68	8	2	**97.45**
aflow30a	1(0)	228.13	2(0)	381.36	41	2355	0	1	**103.28**
aflow40b	12(0)	610.09	2(0)	503.25	21	2329	1	1	**99.32**
air04	1220(2)	28.43	1147(0)	26.87	45	58229	181	1	**3.73**
air05	162(1)	43.73	148(0)	35.73	3	26930	2	1	**2.11**
cap6000	1(0)	**0.35**	1(0)	**0.35**	31	-2442163	0	1	0.38
dano3mip	1892(17)	?	1947(4)	?	70	763.97	361	1	**?**
danoint	4(0)	15.50	9(0)	29.75	96	74	3	1	**12.50**
disctom	3(1)	**0**	4(0)	**0**	3	-5000	3	1	**0**
ds	1945(10)	5633.77	1(2)	**5633.77**	446	5418.56	9495	3	5633.77
fast0507	131(4)	6691.43	2(1)	57.71	8	184	51	1	**5.71**
fiber	0(0)	1496.68	0(0)	**675.45**	41	6481506.12	0	1	1496.68
fixnet6	0(0)	**863.91**	0(0)	2341.58	67	41304	0	1	936.77
glass4	2(0)	775	1(0)	**316.67**	374	12700154400	1	3	958.34
harp2	3(0)	45.02	1(0)	32.67	138	-60669440	3	1	**17.90**
liu	4(5)	**?**	5(0)	?	119	3286	1	1	?
markshare1	0(0)	**30100**	0(0)	364250	65	725	0	1	36200
markshare2	0(0)	**46200**	0(0)	525550	65	963	0	1	48100
mas74	0(0)	484618022.50	0(0)	423649728.01	109	16534.04	0	1	**40.10**
mas76	0(0)	67000682.38	0(0)	124980840.41	106	46242.57	1	1	**15.59**
misc07	3(0)	40.02	2(0)	**21.34**	188	3690	1	1	31.31
mkc	1(0)	50.79	1(1)	106.69	13	-288.96	0	1	**48.67**
mod011	3(1)	31.30	3(0)	34.84	12	-45633967.33	1	1	**16.36**
modglob	1(0)	**5.16**	0(0)	296.53	60	22995521.33	0	1	10.87
net12	10(86)	57.21	8(27)	**57.21**	216	337	12	2	57.21
nsrand-ipx	367(2)	404.05	265(0)	**296.56**	132	211040	5	2	312.38
nw04	9(8)	9	120(2)	265.54	10	17858	10	1	**5.91**
opt1217	0(0)	22.80	0(0)	94.12	40	-16	0	1	**0**
p2756	7(0)	**1542.85**	7(0)	**1542.85**	377	51338	2	3	**1542.85**
pk1	0(0)	625	0(0)	6000	56	36	0	1	**208.33**
pp08aCUTS	0(0)	122.98	0(0)	194.82	10	8360	0	1	**13.74**
pp08a	0(0)	115.63	0(0)	150.85	11	12010	0	1	**63.39**
protfold			365(2)	**37.81**	286	-16	90	2	46.88
qiu	1(0)	748.05	0(0)	2858.10	9	160.76	0	1	**219.34**
set1ch	0(0)	392.71	0(0)	296.92	46	95845.5	0	1	**75.74**
seymour	5(35)	78.07	0(0)	38.92	7	471	3	1	**11.32**
sp97ar	57(4)	75.87	88(1)	1671.15	9	919778417.68	4	1	**39.21**
swath	96(1)	7324.22	100(0)	7324.22	395	35951.85	14	2	**7575.56**
tr12-30	7(0)	121.47	6(0)	118.78	25	164128	1	1	**25.68**
vpm2	0(0)	106.78	0(0)	67.8	12	18.25	0	1	**30.51**
Problems with binary and general integer variables									
arki001	43(0)	1.96	79(0)	2.41	803	7719381.38	15	3	**1.83**
atlanta-ip	68(9398)	118.68	934(11)	**70.32**	454	156.01	227	3	75.52
gesa2-o	0(0)	176.23	1(0)	**26.59**	33	36205441.29	1	2	40.44
gesa2	1(0)	49.23	1(0)	56.35	33	28181419.78	0	2	**9.32**
manna81	0(6)	**1.64**	0(0)	2.17	52	-12940	2	2	1.70
msc98-ip	16(949)	**52.20**	19(22)	55.89	61	30502274.00	26	1	53.75
mzzv11	435(116)	25.12	484(7)	36.71	540	-17898	127	3	**17.59**
mzzv42z	13(147)	37.99	12(15)	30.90	25	-14502	49	1	**29.39**
noswot	0(0)	61.90	0(0)	23.81	13	-41	1	2	**0**
roll3000	65(1)	210.68	11(1)	**43.57**	793	36109.80	17	3	180.12
rout	1(0)	52.56	1(0)	**24.08**	117	1652.55	0	1	53.31
timtab1	1(0)	**41.35**	3(0)	83.22	216	1400493.99	1	2	83.13
timtab2	6(0)	91.96	7(0)	**61.62**	1222	1982037.99	2	2	80.75

?: Unknown value

Table 2.4: Comparison with objective FP.

processors (without exploitation of parallelism capabilities) and 64 GB of RAM. According to the Standard Performance Evaluation Corporation (http://www.spec.org/) the ratio of the performance of our machine (considering specfp2000 and specint2000) and that used in [56] is about 1.5. Therefore the CPU times in Table 2.3 for ACFM are those of [56] divided by 1.5.

As stated in Subsection 2.4.2, as a consequence of computing one analytic center per iteration, ACFM can be computationally expensive, and this is the most important difference from a practical point of view between ACFM and AC-FP. Indeed, as it can be observed in Table 2.3, ACFM was only tested in [56] on nine of the smaller MIPLIB instances, while we applied AC-FP to 54 (some of them much larger) instances. For example, for instance "rout" ACFM needed 101 seconds and got a solution of 1111.88 (gap of 3.18%), while AC-FP needed one second for an objective of 1337.27 (gap of 24.08%); but in other cases AC-FP outperformed ACFM both in time and objective, as in instance "misc07" where ACFM required nine seconds for an objective of 4795 (gap 70.64%), while AC-FP took two seconds for an objective of 3410 (gap 21.34%).

Although from Table 2.4, in general it can be concluded that AC-FP is inferior to the objective FP, there are some notable exceptions. For instance, for the 13 instances with both binary and general integer variables, AC-FP (either with PCx or CPLEX) obtained a solution with a lower gap than the objective FP in eight of the 13 instances; in some cases more efficiently and even being able to find a solution when the objective FP failed (i.e., it required stage 3), as for instances "roll3000" and "atlanta-ip" (in this latter case, however, at the expense of a very large CPU time). On the other hand, for problems with only binary variables AC-FP obtained solutions with a lower gap in very few instances. A possible explanation of this different behaviour in problems with and without general integer variables is that, for a binary problem, the only feasible integer points "close" to the segment $x(\gamma)$ are $\{0,1\}^n$, which in addition may be far from the center. For problems with general integer variables, the number of feasible integer solutions close to the analytic center will be, in general, much larger. For some problems with only binary variables, AC-FP behaved very poorly, as for "mas74" and "mas76" (it stopped at stage 0 in those cases). However, in other instances it was much more efficient obtaining the same gap that the objective FP, as for "ds". Note that for "ds" AC-FP with CPLEX obtained the feasible solution in one second at stage 0 (the other two variants failed, requiring stage

3). However, in that case CPLEX did not really compute the analytic center: it solved $\min_x\{0 : x \in P\}$ heuristically, instead of applying the barrier algorithm, as required. It thus considered a segment between two feasible solutions, none of them being the analytic center of P. Therefore, the idea of using a segment of feasible points is not restricted to the case where one of the endpoints is the analytic center, and it can be extended to more general situations.

Chapter 3

Fix-and-relax approaches for Controlled Tabular Adjustment

3.1 Introduction

Controlled Tabular Adjustment (CTA), as explained in Chapter 1, is a MILP which applied to very large or massive tables becomes very complex. Finding optimal or suboptimal solutions to model (1.6) within an acceptable computational time can be a challenging task to official statistics (NSAs). When the number of sensitive cells is large, the branch-and-cut scheme has shown to be inefficient, and in some cases it is even unable to provide a first feasible solution. For some massive instances—such as, e.g., those in `http://www-eio.upc.es/~jcastro/huge_sdc_instances.html`— the LPs obtained by fixing the value of binary variables—associated to the protection directions—are even not solvable with moderate computational resources. For example, the LPs derived from the six million cells instances of the above web address exhausted the memory of a 16 gigabytes workstation when solved with the CPLEX barrier solver. Unfortunately, the alternative simplex solver is even more prohibitive, but in terms of CPU time: interior-point algorithms have shown to be much more efficient than the simplex for the LPs derived from CTA [11, 12]. Quick tools to provide fast solutions to CTA are a necessity because of the increasing ability of NSAs to create more complex and huge tables from collected data.

For this reason, there is a lot of interest to speed up the solution time. The approach described in this chapter goes along these lines, with two clear objectives:

- Its first goal is to apply a fix-and-relax (FR) heuristic [23] to the MILP CTA problem. Briefly, FR partitions the set of binary variables into k clusters, and iteratively optimizes for each cluster $i = 1, \dots, k$, fixing the binary variables of clusters $j < i$ at the optimal value found in previous iterations, and relaxing the integrality of binary variables of clusters $j > i$. The effect of this partitioning of the set of binary variables is that the nodes of the branch-and-cut tree are selectively explored. Equipping this procedure with a backward repartition strategy (details will be given in Section 3.2), if the MILP is feasible then FR will always provide a feasible, hopefully good and efficient, suboptimal solution. The approach cannot guarantee the optimal solution, but in practice end users of statistical data protection techniques prefer quick suboptimal solutions than optimal costly ones, i.e., requiring too many hours, days or weeks of CPU time. FR has been successfully applied in the past mainly to scheduling problems [23, 30, 31]. In those applications, variables and constraints can naturally be partitioned according to some sequential stages, two consecutive ones being only linked by a few of the variables and constraints of each partition. Such a structure can also be found in *two dimensional tables with one hierarchical variable*, or, shortly, 1H2D tables, described in Chapter 1. This type of tables, which are of interest for NSAs, are a priori suitable for FR. Most of the instances tested in the computational results of this work are 1H2D, and, as it will be shown, FR provides good solutions in a fraction of the time required by state-of-the-art branch-and-cut solvers (to obtain equivalent solutions, i.e., with the same objective function value).

- The second objective of the work is to apply a hybrid approach combining FR and the block coordinate descent (BCD) heuristic, which was successfully applied to some classes of CTA problems in [43]. This hybrid method will be named FR+BCD. Indeed, FR is efficient for computing initial, hopefully good, feasible points, while BCD requires a feasible starting point. Therefore, both heuristics are complementary. As it will be shown in Section 3.4, BCD, warm started with the FR solutions, was able to reduce the gap of the FR solution in approximately half of the real-world CTA instances. In 25 of the 34 real-world instances FR or FR+BCD provided similar or better objective functions in less CPU time than the state-of-the-art MILP solver CPLEX. It will be seen that FR+BCD improved the

FR solutions in only 25% of these 1H2D tables. For real-world tables, this percentage increased up to 50%, making FR+BCD a competitive approach.

In the past, several approaches have been tried to solve the CTA method more efficiently. A straightforward Benders reformulation of the problem was attempted in [14], but promising results were only obtained for two-dimensional tables (i.e., tables obtained by crossing two categorical variables, whose constraints are represented by a node-arc network incidence matrix [12]). Heuristic and metaheuristic methods were attempted in [41], but they only solved small two-dimensional and three-dimensional tables of up to 625 and 8000 cells, respectively, while we consider in this thesis much more complex synthetic and real tables from the literature, of up to 200000 and 36000 cells, respectively. For instance, we generated a set of 20 two-dimensional and 20 three-dimensional tables with the same characteristics (sizes and number of sensitive cells) than those in [41]. We remark that: (1) the tables used in [41] were also randomly generated; (2) the matrix constraints only depends on the table structure (two- or three-dimensional table) so they were the same in our experiments and those in [41]; (3) although the instances are not *exactly* the same, what makes difficult (in general) a problem is the structure of the matrix constraints and the number of sensitive cells (which is associated to the number of binary variables of the optimization problem); those characteristics are the same in our experiments and those of [41]. CPLEX 12.5 found a 0% gap solution for all these two-dimensional tables with an average CPU time of 0.02 seconds (the maximum time required by an instance was 0.03 seconds). For the three-dimensional tables, the average CPU time was 0.2 seconds (the maximum time for an instance was 0.49 seconds), again for 0% gap solutions. No CPU time comparison with CPLEX was reported in [41]; it was just stated that CPLEX 8.1 could not solve the instances. Therefore, up to now, there is no conclusive evidence that those metaheuristics are helpful for the CTA problem. We also tried in the past other general metaheuristics as genetic algorithms without success: combinations or modifications of solutions are not expected to satisfy the large number of linear constraints with no particular structure of CTA. Indeed, these constraints are usually complex, and any practical approach must rely on the efficient solution of (usually difficult) linearly constrained problems (either LPs or MILPs). The approaches in this chapter rely on decomposing the problem into smaller, thus tractable, MILP instances. It is worth to note that even the LPs obtained from large CTA instances by fixing the binary variables are very difficult for today state-of-the-art solvers. Indeed, some of these instances have been included in standard LP repositories [54].

The chapter is organized as follows. Section 3.2 describes the FR heuristic for CTA; Section 3.3 outlines the BCD approach. Finally, Section 3.4 presents extensive computational results, showing the effectiveness of FR and FR+BCD for synthetic 1H2D and real-world tables.

3.2 Fix-and-relax

FR is a decomposition method based on partitioning the set of binary variables into clusters to iteratively solve a sequence of MILPs of smaller dimension than the original problem. In those smaller MILPs only a subset of variables retain their binary constraints while the rest are either fixed or relaxed. Since only a reduced subset of (non-fixed) 0-1 variables is kept integer at each FR iteration, a computational improvement is expected. FR can both be seen as an approach for obtaining (hopefully good) initial feasible solutions and primal bounds. There are other approaches for initial good solutions in MILPs, such as the feasibility pump [36], but as it will seen in Section 3.4, in practice FR outperformed them.

FR can be briefly stated as follows. The set of binary variables is partitioned into a finite set of clusters $\{V_1, \ldots, V_k\}$. The original MILP is then decomposed into k subproblems and at each iteration one of them is solved. At first iteration (counter r set to 1) the subproblem considers as binary only the variables of V_1, while the integrality of binary variables in the remaining clusters is relaxed. Continuous variables in the original MILP maintain this same status at each subproblem. Hopefully, this first subproblem will be easily solved since the cardinality of V_1 is much smaller than the number of binary variables in the original MILP. Once solved, the counter r is incremented and the next subproblem is considered. At subproblem of iteration r, $k > r > 1$, the binary variables of clusters V_i, $i < r$, are fixed to the values of optimal solutions from the previous iterations; variables of cluster V_r are considered binary, while the integrality of variables in clusters V_j, $j > r$ is relaxed. The process is repeated until $r = k$. If no subproblem is infeasible, a (hopefully good) feasible solution will be available after the solution of subproblem k. In the particular case of CTA, the set $\mathcal{S}$ of sensitive cells is partitioned into the subsets $\{V_1, \ldots, V_k\}$, and the subproblem r

1. Input: Number of clusters $k \geq 1$
2. Partition $\mathcal{S}$ into $\{V_1, \ldots, V_k\}$ clusters
3. Initialize $r = 1$ and solve CTA^1_{FR}
4. **if** CTA^1_{FR} is infeasible, STOP
5. **else** Store values of binary variables of CTA^1_{FR}, set lower bound LB, and $r \leftarrow r + 1$
6. **while** $r \leq k$ **do**
7. Solve CTA^r_{FR}
8. **if** infeasible, redefine the partition structure as in (3.2)
9. **else** Store optimal values of binary variables of CTA^r_{FR}, and $r \leftarrow r + 1$
10. **end while**
11. Return UB (solution of CTA^k_{FR}) and LB

Figure 3.1: The fix-and-relax heuristic applied to the CTA problem.

associated to (1.6)—which will be referred as (CTA^r_{FR})—is

$$
\begin{array}{lll}
\min\limits_{z^+,z^-,y} & \sum\limits_{i=1}^{n} w_i(z_i^+ + z_i^-) & \\
\text{s. to} & A(z^+ - z^-) = 0 & \\
& 0 \leq z_i^+ \leq \quad u_{z_i} & i \in \mathcal{N} \setminus \mathcal{S} \\
& 0 \leq z_i^- \leq -l_{z_i} & i \in \mathcal{N} \setminus \mathcal{S} \\
& upl_i \; y_i \leq z_i^+ \leq u_{z_i} \; y_i & i \in \mathcal{S} \\
& lpl_i(1-y_i) \leq z_i^- \leq -l_{z_i}(1-y_i) & i \in \mathcal{S} \\
& y_i = \tilde{y}_i & i \in \bigcup_{h=1,\ldots,r-1} V_h \\
& y_i \in \{0,1\} & i \in V_r \\
& y_i \in [0,1] & i \in \bigcup_{h=r+1,\ldots,k} V_h,
\end{array}
\tag{3.1}
$$

where $\tilde{y}_i$, $i \in \cup_{h=1,\ldots,r-1} V_h$, are the values of binary variables found at subproblems CTA^1_{FR}, …, CTA^{r-1}_{FR}. Although FR is a heuristic for MILP problems, it is easily switched to an optimal approach by setting $k = 1$.

It is worth noting that the first subproblem CTA^1_{FR} has two main features compared to the subsequent ones:

- The lower bound on the objective function provided by CTA^1_{FR} is a global lower bound of (1.6). On the other hand, the lower bound of subproblems $r > 1$ are just local lower bounds. The lower bound reported by the FR algorithm will then be that of CTA^1_{FR}. Note that the optimal solution of CTA^1_{FR} can be considered a lower bound of (1.6) *only* if computed with a 0% gap. However, such a gap is impractical, because the solution of CTA^1_{FR}

would take a long execution time—something to avoid, since the goal of FR is to quickly provide a decent solution. In the implementation developed, the lower bound was obtained by the CPLEX routine CPXgetbestobjval. When a problem has been solved to optimality, this routine provides the optimal solution value. Otherwise, it provides the minimum objective function value of all remaining unexplored nodes in the branch-and-cut tree.

- If CTA^1_{FR} is infeasible, then (1.6) is infeasible as well. However, if some subproblem $r > 1$ is infeasible it can not be concluded that (1.6) is infeasible; it just means that we can not fix $y_i = \tilde{y}_i$, for $i \in V_{r-1}$, at subproblem r. To overcome this drawback, when subproblem $r > 1$ is reported as infeasible, we *backtrack* to problem $r-1$, modifying the partition by joining the clusters V_{r-1} and V_r as follows:

$$\begin{cases} V_{r-1} \leftarrow V_{r-1} \bigcup V_r \\ V_i \leftarrow V_{i+1}, i = r, \ldots, k-1 \\ k \leftarrow k-1 \\ r \leftarrow r-1. \end{cases} \tag{3.2}$$

  Note that the above repartition strategy will always provide a feasible solution if (1.6) is feasible. Indeed, in the worst case, if subproblem k is infeasible and (3.2) is applied $k-1$ times, we will end up with a unique cluster, i.e., we will be solving (1.6). However, in practice, as it was observed in the computational results of Section 3.4, this repartition strategy was never needed in the instances tested.

An outline of the FR algorithm for CTA is shown in Figure 3.1.

3.3 Outline of block coordinate descent

The BCD approach applied to CTA has been described in [43]. Briefly, it consists of a sequence of CTA subproblems, each of them optimizing the objective function over the cell deviations z^+, z^- and a subset of the decision variables y, while the remaining variables y are kept fixed to some direction. Provided that we start from a feasible assignment of y, the method can move from a solution to another, hopefully better. Although there could be uncountable strategies to determine the subset of variables to be optimized, the set $\mathcal{S}$ is usually partitioned into k clusters (or blocks) and the algorithm iterates through them. However, BCD could perform indefinitely, starting again with the same or with another partition. Stopping criteria normally employed are: only one cycle of k clusters; a time limit, or a specified number of subproblems without improvement in the objective function. Since the method does not account for dual information there are no means to compute a gap for the solution. Despite this, the results of [43] showed that BCD reaches sub-optimal but still good solutions in significantly less time than branch-and-cut schemes. The algorithm is summarized in Figure 3.2.

Experience with BCD has shown that, in general, the performance of the method improves as the number of blocks decreases, and two blocks seems to be the best choice. Notice that one block would lead the method to a plain branch-and-cut, which might be computationally prohibitive. It has been observed that many tables are (sub-optimally) protected through manipulation of half of their sensitive cells in a fraction of the time needed if the whole set of sensitive cells was considered (this fraction of time being significantly less than 1/2).

Many tests indicate that rebuilding the partition of blocks at each iteration is clearly preferable to keep some pre-determined division. Actually, the best performances are obtained with a random division of the binary variables into blocks; this is the main strategy considered.

A disadvantage of BCD is the need to find a feasible initial assignment of directions to start the process (step 2 of algorithm of Figure 3.2), which may be in itself a difficult problem for large CTA instances. The heuristic approach considered in [43], which relies on the Boolean Satisfiability problem, only focuses on the constraints, and then it may provide poor quality solutions. Since FR solutions take into account the objective function, we can use it as a good warm start to BCD. This approach, named FR+BCD, will be computationally tested and seen as a very efficient option in Subsection 3.4.3.

3.4 Computational results

The FR and FR+BCD heuristics for CTA have been coded in C++, using the state-of-the-art CPLEX 12.5 branch-and-cut solver for the solution of subproblems (3.1). FR and FR+BCD were compared with the direct solution of (1.6) through plain CPLEX branch-and-cut, which will be referred as BC.

All the runs were carried out on a Dell PowerEdge 6950 server with four

instance	$\overline{n}$	$\overline{s}$	$\overline{m}$	$\overline{nz}$
Symmetric instances				
sym-40-50-5	29039	1421	1334	58793
sym-40-50-15	31753	46612	1388	64219
sym-40-50-30	29141	8556	1336	58997
sym-40-60-5	36990	1816	1521	74835
sym-40-60-15	34026	5011	1473	68906
sym-40-60-30	38040	11207	1539	76933
sym-50-50-5	40637	1989	1562	81988
sym-50-50-15	39596	5815	1541	79907
sym-50-50-30	38097	11190	1512	76908
sym-50-60-5	45555	2237	1662	91964
sym-50-60-15	44457	6550	1644	89768
sym-50-60-30	45835	13507	1666	92525
Asymmetric instances				
asym-40-50-5	125661	6157	5677	254483
asym-40-50-15	126844	18646	5700	256850
asym-40-50-30	127000	37338	5703	257162
asym-40-60-5	151166	7431	6321	306114
asym-40-60-15	149641	22069	6296	303064
asym-40-60-30	150711	44454	6314	305203
asym-50-50-5	162561	7966	6400	328284
asym-50-50-15	159766	23487	6346	322694
asym-50-50-30	160171	47094	6354	323503
asym-50-60-5	191503	9415	6982	386789
asym-50-60-15	189718	27982	6953	383218
asym-50-60-30	188742	55676	6937	381266

Table 3.1: Characteristics of symmetric/asymmetric synthetic 1H2D instances.

instance	n	s	m	nz
australia_ABS	24420	918	274	13224
bts4	36570	2260	36310	136912
cbs	11163	2467	244	22326
dale	16514	4923	405	33028
destatis	5940	621	1464	18180
hier13d4	18969	2188	47675	143953
hier13	2020	112	3313	11929
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
nine5d	10733	1661	17295	58135
osorio	10201	7	202	20402
sbs2008_C	4212	1135	2580	13806
sbs2008_E	1430	382	991	4680
table1	1584	146	510	4752
table3	4992	517	2464	19968
table4	4992	517	2464	19968
table5	4992	517	2464	19968
table6	1584	146	510	4752
table7	624	17	230	1872
table8	1271	3	72	2542
targus	162	13	63	360
toy3dsarah	2890	376	1649	9690
two5in6	5681	720	9629	34310

Table 3.2: Characteristics of real instances.

Input: Number of clusters $k \geq 1$
Set feasible initial values to y; initialize outer iteration counter: $t \leftarrow 0$
while stopping criterion not satisfied and $t \leq t_{\max}$ **do**
  Set inner iteration counter: $i \leftarrow 0$; divide y into k blocks: $y = \{y^{1,i}, \ldots, y^{k,i}\}$
  while $i < k$ **do**
    Solve (1.6) with respect to block $y^{i,i}$ fixing $y^{j,i}$ for $j \neq i$: obtain $(y^{i,i})^*$
    $y^{i,i+1} \leftarrow (y^{i,i})^*$; $y^{j,i+1} \leftarrow y^{j,i}$ for $j \neq i$
    $i \leftarrow i + 1$
  end while
  $t \leftarrow t + 1$
end while
Return the best solution found

Figure 3.2: The block coordinate descent heuristic for the CTA problem.

dual core AMD Opteron 8222 3.0 GHZ processors (without exploitation of parallelism capabilities) and 64 GB of RAM. Default values were used for the CPLEX parameters, unless explicitly stated. For the computational tests we considered a set of real-world general and synthetic 1H2D tables. Real-world general tables are standard instances used in the literature [12]. It is worth noting that some real-world instances were not included in this set since they are too difficult for both heuristic and exact MILP approaches—no feasible solution was obtained within the time limit. Synthetic instances were obtained with a generator of 1H2D tables. This generator is governed by several parameters, as, for instance, 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.es/~jcastro/generators_csp.html. We fixed all parameters, but three: the number of rows per subtable ($\mathtt{r} \in \{40, 50\}$), the number of columns per subtable ($\mathtt{c} \in \{50, 60\}$) and the percentage of sensitive cells ($\mathtt{s} \in \{5, 15, 30\}$).

We considered either symmetric and asymmetric instances, i.e., instances where $u_{a_i} = l_{a_i}$ for all $i \in \mathcal{N}$ and $u_{a_i} \neq l_{a_i}$ for some $i \in \mathcal{N}$, respectively. Asymmetric instances were obtained by considering $u_{a_i} = \mathtt{a} \cdot l_{a_i}$ for all $i \in \mathcal{N}$, where $\mathtt{a} \in \{2, 5, 10\}$ is the asymmetry parameter. For each combination of parameters we generated a sample of five instances varying the random generator seed. This amounted to 12 and 36 samples of five instances each one, for symmetric and asymmetric instances respectively. Although the asymmetry parameter slightly affects to the difficulty of the problem, both symmetric and asymmetric instances will be grouped by `r`, `c` and `s` to simplify the exposition. The reported computational results are thus averaged on five and 15 replications for symmetric and asymmetric tables, respectively.

Table 3.1 reports the characteristics of each set of symmetric/asymmetric 1H2D instances: the average number of cells ("$\overline{n}$"), the average number of sensitive cells ("$\overline{s}$"), the average number of table relations ("$\overline{m}$") and the average number of coefficients in linear constraints ("$\overline{nz}$"). Hierarchical synthetic tables are identified by the particular combination of parameters, i.e., sym-`r-c-s` for symmetric instances and asym-`r-c-s` for asymmetric ones. Table 3.2 reports the same information for real-world tables, though in this case the dimensions are not averaged. The dimensions of the MILP problems (1.6) are $2n$ continuous variables, s binary variables, and $m + 4s$ linear constraints.

3.4.1 Tuning the number of clusters in fix-and-relax

The performance of FR depends on the number of clusters k considered. We performed an empirical study of the effect of k on two particular metrics: the CPU time and the quality of the solutions provided by FR. This empirical analysis was done considering values $k \in \mathcal{K} = \{3, 5, 7, 10, 20, 30, 40, 50\}$, and using a subset of asymmetric 1H2D instances. For each combination of parameters `r-c-s` we considered a sample of three instances.

Each instance was solved $|\mathcal{K}|$ times, randomly partitioning the set $\mathcal{S}$ of sensitive cells into $k \in \mathcal{K}$ subsets. In [43], different strategies to partition the sensitive cells were tested. The first strategy divides the set $\mathcal{S}$ randomly into a number of blocks, keeping their sizes as similar as possible. The second strategy takes account the tree structure of 1H2D tables and the sensitive cells are partitioned according to their level. Curiously, this strategy was not satisfactory and the best performances are obtained with a random division of the binary variables into blocks; so this was the main strategy considered in FR framework. The stopping criterion for all the runs, i.e., subproblems (3.1), was a 5% optimality gap, which is computed by CPLEX as $(\text{UB} - \text{LB})/(|\text{UB}| + 10^{-10})$, where UB is the best integer solution (upper bound) and LB is the best achievable value from the current branch-and-cut tree (lower bound).

Figure 3.3 reports the CPU time (in seconds, averaged for the three instances of each sample) used by FR for the different $k \in \mathcal{K}$ number of clusters. Clearly, the CPU time increases with k, and the heuristic becomes prohibitive if the number of clusters is large.

The second metric, the quality of the solutions, was evaluated using the performance profile proposed in [55]. Quality was measured as the value of the objective function (thus, the lower, the better). Let Q_{tk} be the quality of the solution of instance t solved by FR with k clusters. Note that Q_{tk} for CTA is always strictly positive. The performance ratio is thus defined as

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

i.e., the ratio between the quality of the solution obtained when instance t is solved by FR with k clusters over the strategy with the best (minimum) performance for this instance. The (cumulative) distribution function $P_k(q) : [1, \infty) \rightarrow [0, 1]$ is defined as

$$P_k(q) = \frac{|\{t \in \mathcal{T} : v(t,k) \leq q\}|}{|\mathcal{T}|}, q \geq 1.$$

where $\mathcal{T}$ is the set of instances. Figure 3.4 shows the performance profiles for the different $k \in \mathcal{K}$. $P_k(q) = 1$ means FR with k clusters is able to solve all the instances within a factor q of the best possible ratio. In our case $k = 3$ is the first strategy to converge to 1 for $q \approx 1.45$ (i.e., FR with 3 blocks solves all the instances within a factor ≈ 1.45 of the best ratio). It can also be observed that $k = 3$ provides the highest quality for 80% of the instances ($P_3(1) \approx 0.8$).

3.4.2 Comparison between fix-and-relax and plain branch-and-cut

From the discussion of previous Subsection, $k = 3$ was set for FR. An optimality gap of 5% was considered for all the optimization problems, either (1.6) or FR subproblems (3.1). The time limit was set to two hours for both 1H2D and real-world instances. Note that FR subproblems are also solved by CPLEX branch-and-cut; therefore the comparison is between whether using or not the FR scheme. We will refer to these two variants as FR and BC.

Tables 3.5 and 3.6 report an exhaustive comparison between FR and BC for synthetic 1H2D and real-world instances, respectively. These tables report the

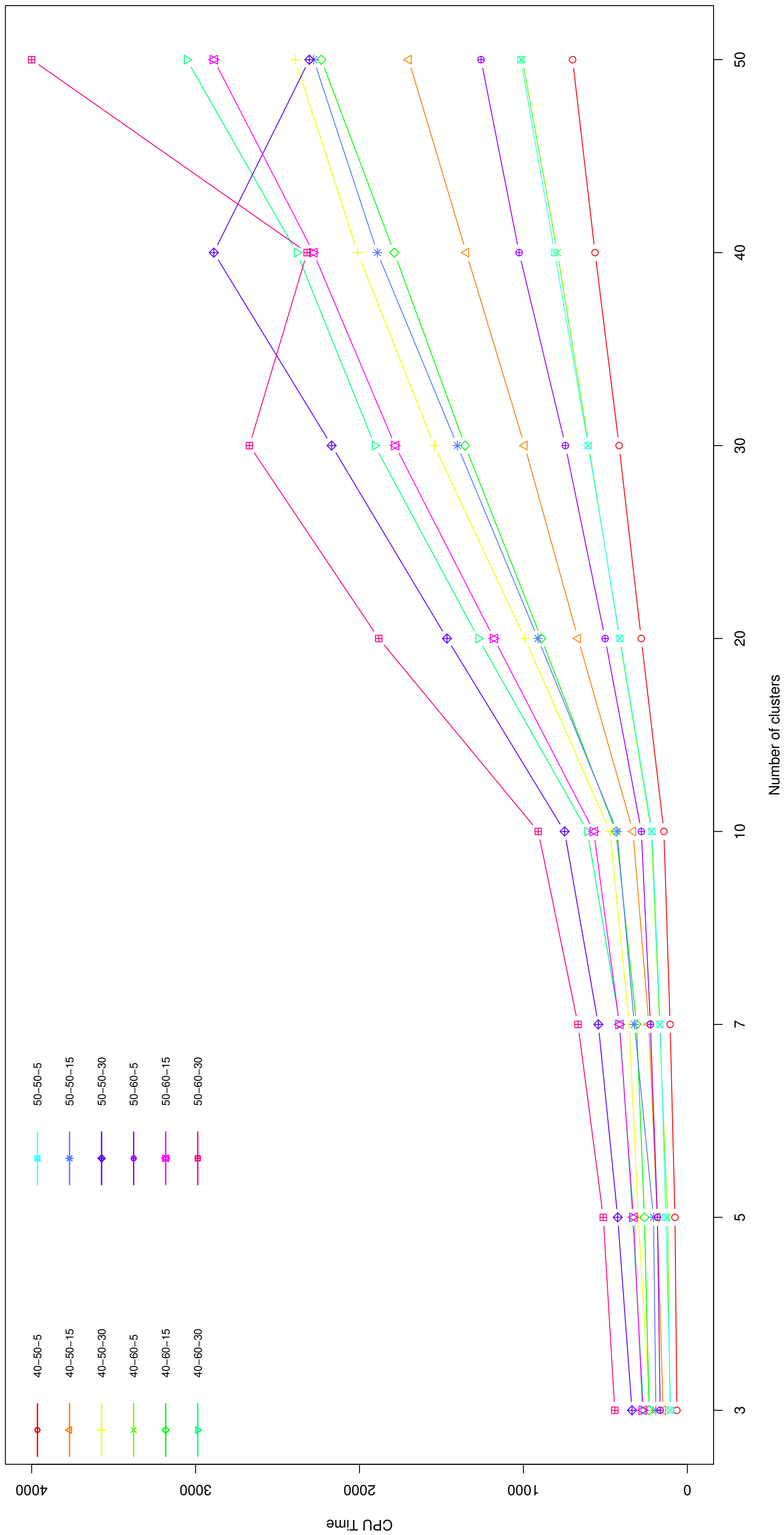


Table 3.3: CPU time for different number of clusters.

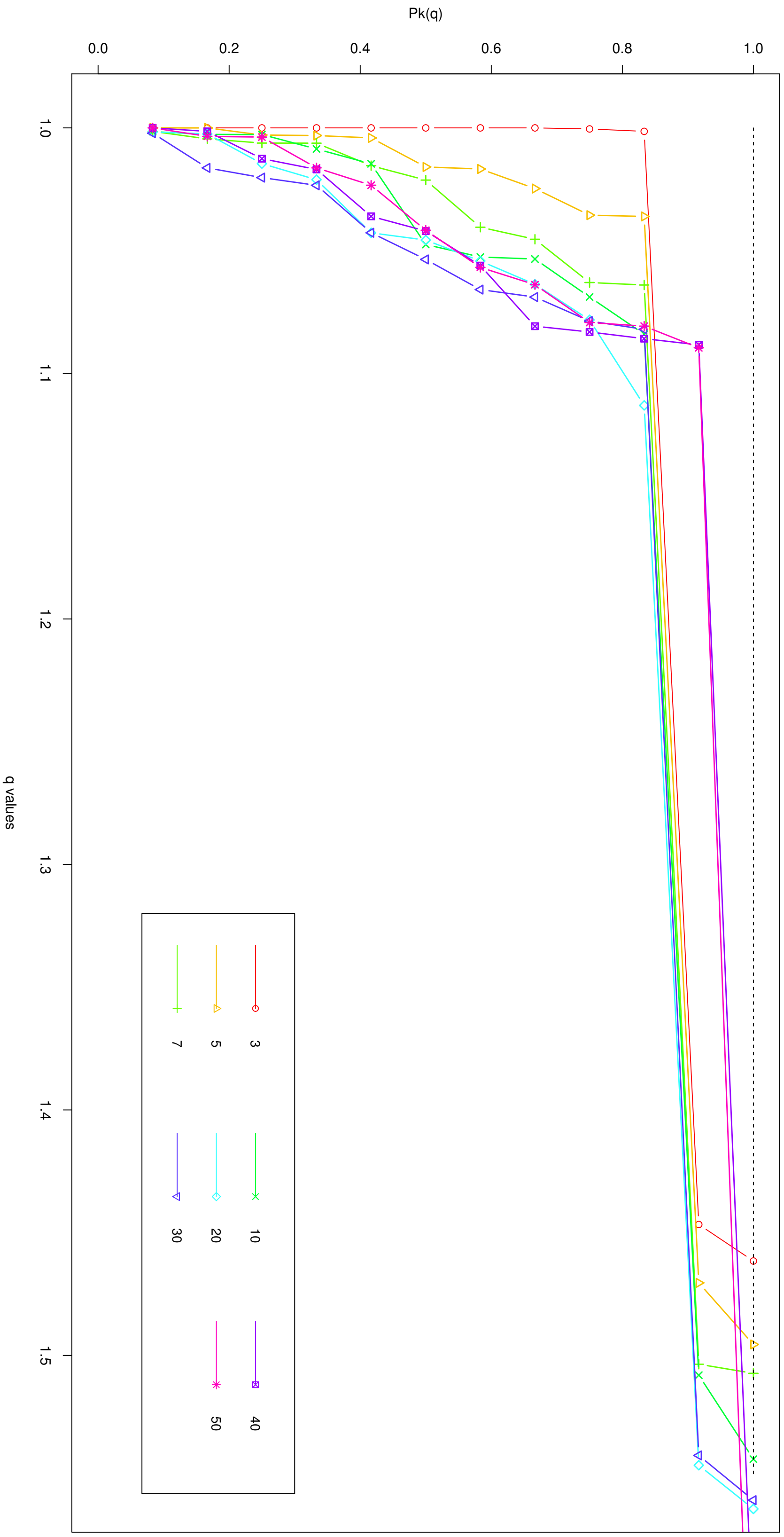


Table 3.4: Performance profile of the quality of the solution for different number of clusters.

FR CPU time (columns "T_{FR}"); the primal gap of the solution reported by FR (columns "$GAP_{FR}\%$"); the primal gap of the solution reported by BC after T_{FR} seconds of CPU time (columns "$GAP_{BC}\%$"), i.e., using the same time than FR; the difference between both primal gaps (columns "$\Delta(BC, FR)$"); the primal gap and CPU time needed by BC to compute a better solution than the feasible solution found by FR (columns "$GAP_{BC}^{up}\%$" and "T_{BC}^{up}"); and finally the difference between the time needed by BC to improve the FR solution and the time needed by FR to compute that solution (columns"$\Delta(T_{FR}, T_{BC}^{up})$"). Positive values at column $\Delta(BC, FR)$ means that FR achieved a better solution than BC in the same CPU time.

From Table 3.5 it can be concluded that FR is more efficient than BC for fast good feasible solutions of 1H2D tables. In several runs (marked with ‡) BC could not find a better solution than FR within the time limit. It is worth noting that for all the 1H2D instances FR provided solutions with gaps below 6%. For the real-world general instances of Table 3.6 the situation is slightly different. These instances are not guaranteed to have a hierarchical structure, and this may explain why FR is not as competitive as for 1H2D tables. FR provided a better gap than BC within the same CPU time in 17 of the 34 instances, and both FR and BC provided the same gap in six adittional cases. In six of these cases BC could not improve the FR solution within the two hours time limit. In the remaining instances BC outperformed FR.

3.4.3 Comparison between fix-and-relax with block coordinate descent and plain branch-and-cut

As mentioned in section 3.3, since FR can provide good feasible solutions faster in average than BC, BCD was warm started with the FR solution. This hybrid approach was named FR+BCD.

The BCD algorithm performed in all cases a loop with two clusters, each one with a half of the sensitive cells, partitioned at random. At exit, the CPU computation time and the objective function value were saved. This CPU time was added to the FR CPU time and compared to the CPU time used by BC. We also took into account whether the sensitive cells had been correctly protected in the final solutions, since accuracy errors might be present in some instances, making actually infeasible the protected table. This accuracy errors are due to

instance	T_{FR}	$GAP_{FR}\%$	$GAP_{BC}\%$	$\Delta(BC, FR)$	$GAP^{up}_{BC}\%$	T^{up}_{BC}	$\Delta(T^{up}_{BC}, T_{FR})$
asym-40-50-5	72.76	2.52	$\dagger^{(40.70,5)}$	$\dagger^{(38.18,5)}$	$\ddagger^{(1.20,13)}$	$\ddagger^{(87.76,13)}$	$\ddagger^{(15.00,13)}$
asym-40-50-15	158.75	3.55	86.67	83.11	$\ddagger^{(1.94,13)}$	$\ddagger^{(465.84,13)}$	$\ddagger^{(307.09,13)}$
asym-40-50-30	210.51	5.60	99.98	94.38	1.84	1083.47	872.96
asym-40-60-5	95.40	2.40	$\dagger^{(0.88,5)}$	$\dagger^{(-1.51,5)}$	$\ddagger^{(1.08,14)}$	$\ddagger^{(124.01,14)}$	$\ddagger^{(28.61,14)}$
asym-40-60-15	193.37	3.17	99.96	96.79	$\ddagger^{(1.69,12)}$	$\ddagger^{(945.40,12)}$	$\ddagger^{(752.03,12)}$
asym-40-60-30	314.96	5.26	99.97	94.71	2.43	1107.84	792.88
asym-50-50-5	110.54	1.82	$\dagger^{(0.08,8)}$	$\dagger^{(-1.74,8)}$	$\ddagger^{(0.62,14)}$	$\ddagger^{(135.65,14)}$	$\ddagger^{(25.11,14)}$
asym-50-50-15	186.95	3.11	86.66	83.54	$\ddagger^{(1.02,11)}$	$\ddagger^{(804.02,11)}$	$\ddagger^{(617.07,11)}$
asym-50-50-30	333.50	5.94	99.94	93.99	2.21	1704.06	1370.55
asym-50-60-5	153.14	1.23	$\dagger^{(0.45,9)}$	$\dagger^{(-0.78,9)}$	$\ddagger^{(0.85,13)}$	$\ddagger^{(163.61,13)}$	$\ddagger^{(10.46,13)}$
asym-50-60-15	282.79	3.05	93.33	90.28	$\ddagger^{(1.46,13)}$	$\ddagger^{(1569.41,13)}$	$\ddagger^{(1286.62,13)}$
asym-50-60-30	406.42	5.52	99.92	94.40	2.39	1396.89	990.47
sym-40-50-5	8.99	2.43	12.69	10.26	$\ddagger^{(1.61,2)}$	$\ddagger^{(15.09,2)}$	$\ddagger^{(6.10,2)}$
sym-40-50-15	115.34	4.31	45.79	41.48	$\ddagger^{(2.56,4)}$	$\ddagger^{(443.91,4)}$	$\ddagger^{(328.57,4)}$
sym-40-50-30	371.35	4.61	63.90	59.29	$\ddagger^{(4.36,3)}$	$\ddagger^{(2681.80,3)}$	$\ddagger^{(2310.45,3)}$
sym-40-60-5	10.52	2.79	14.44	11.65	1,33	37.76	27.24
sym-40-60-15	102.29	2.20	82.23	80.03	$\ddagger^{(-,0)}$	$\ddagger^{(-,0)}$	$\ddagger^{(-,0)}$
sym-40-60-30	800.45	4.39	12.88	8.49	$\dagger^{(4.59,3)}$	$\dagger^{(2870.02,3)}$	$\dagger^{(2069.57,3)}$
sym-50-50-5	25.47	1.94	$\dagger^{(12.20,3)}$	$\dagger^{(10.25,3)}$	$\ddagger^{(0.72,5)}$	$\ddagger^{(50.75,5)}$	$\ddagger^{(25.27,5)}$
sym-50-50-15	166.33	3.66	12.74	9.08	$\ddagger^{(1.95,2)}$	$\ddagger^{(1434.04,2)}$	$\ddagger^{(1267.71,2)}$
sym-50-50-30	511.19	3.45	66.81	63.36	$\ddagger^{(3.71,2)}$	$\ddagger^{(5049.30,2)}$	$\ddagger^{(4538.11,2)}$
sym-50-60-5	56.80	1.61	$\dagger^{(54.33,4)}$	$\dagger^{(52.72,4)}$	$\ddagger^{(0.97,4)}$	$\ddagger^{(104.60,4)}$	$\ddagger^{(47.80,4)}$
sym-50-60-15	279.63	2.47	13.60	11.14	$\ddagger^{(0.70,1)}$	$\ddagger^{(1055.10,1)}$	$\ddagger^{(775.47,1)}$
sym-50-60-30	833.45	3.95	47.62	43.66	$\ddagger^{(4.38,2)}$	$\ddagger^{(3863.53,2)}$	$\ddagger^{(3030.08,2)}$

$\dagger^{(x,y)}$ BC could not find a solution in y of the overall number of replications within T_{FR} seconds; x is the average value for the remaining successful runs.

$\ddagger^{(z,w)}$ BC could not improve the FR solution in w of the overall number of replications within the time limit; z is the average value for the remaining successful runs.

Table 3.5: Comparison between fix-and-relax and plain branch-and-cut for synthetic asymmetric and symmetric 1H2D instances.

instance	T_{FR}	$GAP_{FR}\%$	$GAP_{BC}\%$	$\Delta(BC, FR)$	$GAP^{up}_{BC}\%$	T^{up}_{BC}	$\Delta(T^{up}_{BC}, T_{FR})$
australia_ABS	6,05	73,87	3,84	-70,03	7,40	2,45	-3,6
bts4	6332,15	66,57	74,16	7,59	‡	‡	‡
cbs	2,87	100,00	100,00	0,00	0,00	2,88	0,01
dale	595,85	48,44	48,44	0,00	48,44	7199,96	6604,11
destatis	204,32	19,53	99,97	80,44	1,80	706,48	502,16
hier13d4	6410,73	82,86	99,98	17,12	‡	‡	‡
hier13	747,29	6,88	4,90	-1,98	4,90	159,24	-588,05
hier13x13x13a	542,72	5,22	5,22	0,00	4,94	690,15	147,43
hier13x13x13b	584,92	5,89	5,24	-0,65	5,24	369,13	-215,79
hier13x13x13c	542,86	5,63	4,99	-0,65	4,99	243,43	-299,43
hier13x13x13d	178,12	4,69	5,27	0,58	2,40	340,47	162,35
hier13x13x13e	336,72	5,39	4,40	-0,99	4,40	269,82	-66,9
hier13x13x7d	34,72	5,58	7,19	1,61	4,96	69,32	34,6
hier13x7x7d	2,71	4,82	12,35	7,53	‡	‡	‡
hier16	4854,46	59,48	63,07	3,59	‡	‡	‡
hier16x16x16a	2803,76	44,96	48,80	3,84	44,71	3706,65	902,89
hier16x16x16b	3401,2	33,49	99,95	66,46	31,89	6275,87	2874,67
hier16x16x16c	3488,27	40,86	50,40	9,54	39,62	4926,13	1437,86
hier16x16x16d	3776,81	57,66	63,32	5,66	57,07	5369,03	1592,22
hier16x16x16e	3862,21	46,55	46,87	0,32	‡	‡	‡
nine5d	6133,25	67,69	99,99	32,30	‡	‡	‡
osorio	1,86	0,00	0,00	0,00	0,00	1,03	-0,83
sbs2008_C	543,88	50,11	3,36	-46,75	21,77	32,24	-511,64
sbs2008_E	4,56	4,73	4,73	0,00	4,73	2,94	-1,62
table1	0,62	8,38	13,43	5,06	4,92	1,25	0,63
table3	1909,18	25,39	15,07	-10,32	17,11	408,3	-1500,88
table4	1196,86	25,39	17,30	-8,09	18,60	511,64	-685,22
table5	720,49	22,80	20,20	-2,60	20,25	216,19	-504,3
table6	1,01	7,77	39,32	31,54	3,36	2,17	1,16
table7	0,1	1,01	0,41	-0,61	0,41	0,02	-0,08
table8	0,18	2,44	0,00	-2,44	1,35	0,04	-0,14
targus	0,08	3,84	3,84	0,00	3,84	0,01	-0,07
toy3dsarah	25,47	0,34	7,29	6,94	0,34	37,46	11,99
two5in6	3010,89	66,04	99,99	33,95	63,91	7200,45	4189,56

‡ Time limit reached without improving the feasible FR solution.

Table 3.6: Comparison between fix-and-relax and plain branch-and-cut for real instances.

			Objective Function	
			FR+BCD < BC	FR+BCD > BC
	FR+BCD < BC			
		mean (sd) ΔF	−1.24 (1.08)	1.86 (1.46)
		max $\lvert\Delta F\rvert$	−4.23	7.26
		mean (sd) ΔT	−1564 (1872)	−883 (1281)
		max $\lvert\Delta T\rvert$	−6641	−6351
		N [N_{asym} ; N_{sym}]	58 [27; 31]	119 [93; 26]
Time	FR+BCD > BC			
		mean (sd) ΔF	−1.55 (1.43)	0.83 (1.07)
		max $\lvert\Delta F\rvert$	−3.89	4.17
		mean (sd) ΔT	146 (207)	49 (38)
		max $\lvert\Delta T\rvert$	508	182
		N [N_{asym} ; N_{sym}]	5 [4; 1]	58 [56; 2]

ΔF stands for $100(F_{FR+BCD} - F_{BC})/F_{BC}$. ΔT stands for $(T_{FR+BCD} - T_{BC})$, in seconds.
"sd" stands for standard deviation.

Table 3.7: Summary of results for 1H2D instances, in the comparison between FR+BCD versus BC.

the big-M constraints $z_i^+ \leq u_{z_i}\ y_i$ and $z_i^- \leq -l_{z_i}(1-y_i)$ of (1.6) and (3.1), since u_{z_i} and $-l_{z_i}$ can take very large values.

With regard to 1H2D instances, it has been observed that the extra time needed by the BCD stage is related to the number of sensitive cells, although with considerable variability especially if the table is large. The 16 tables with more than 50,000 sensitive cells consumed between 114 and 492 seconds, with a median time of 255 seconds. In 104 instances with less than 10,000 sensitive cells the median time was 29.7 seconds. Compared to the time employed by the FR stage, it took about 40% of that time (median proportion): in 18 instances out of 240 BCD lasted longer than FR, generally in tables with high density of sensitive cells.

Sixty-one tables improved the objective function after the BCD stage, and the others remained in the same value (not necessarily in the same solution). The median change in the objective function with respect to the value attained by FR was 3%, with a maximum of 10%. Improving the solution requires also more time: 54.6% of FR time, instead of 37% for the tables not improved. We observed a higher rate of success among the tables with high density of sensitive cells: an odd of 33 versus 47 for tables with 30% of sensitive cells, compared to 28 versus 132 for tables with 15% or lower proportion. The table size or the asymmetry degree in the protection levels were not related to improvement in the objective function.

Table 3.7 summarizes the results with 1H2D instances for two factors: solution times (in rows) and objective function values (in columns) between BC and

FR+BCD. The two categories for each factor are either FR+BCD outperfomed BC ("FR+BCD $<$ BC", i.e., less CPU time or a lower objective function value) or the opposite ("FR+BCD $>$ BC"). Each of the four cells shows the number of instances ("N"), and some statistics (mean, standard deviation, maximum) about the change in both factors: Δ_F is the percentage change in the objective function, Δ_T is the absolute change in the time. Comparing left versus right columns, we can see small differences between the percentage changes in objective function values (they range from -4.23% to 7.26%). However, comparing above versus below rows, we can see large differences with respect to solution times: 1564 and 883 seconds in favor of FR+BCD (177 cases) against 146 and 49 seconds (63 cases) in favor of BC.

For the real-world tables, FR+BCD got better solutions than FR in 18 instances after a BCD cycle, whereas it did not improve the FR objective function in 16 cases. Table 3.8 reports the results obtained. Columns "$F_\cdot$" and "$T_\cdot$" provide, respectively, the objective function and CPU solution times for each method, BC, FR and FR+BCD or BCD. Column "$\Delta(F_{FR}, F_{FR+BCD})$" provides the relative change (as a percentage) in the objective function between the FR and FR+BCD solutions. The rows are ordered by $\Delta(F_{FR}, F_{FR+BCD})$; the first instance shows a negative change because the solution reached by FR was actually not feasible, due to slight deviations in some sensitive cells beyond their protection levels, but undetectable with the (already tight) infeasibility tolerance in use by the solver (cf. big-M issue discussed above). In general, FR already provided a good solution for instances which could not be improved by BCD; this FR solution was close to the one obtained by BC, but it was computed faster. On the other hand, it is remarkable that most of the instances where the BCD cycle could improve the solution were difficult for the BC scheme, which used to exhaust the time limit.

Table 3.9 summarizes the results for the real-world instances with respect to CPU time and objective function values. The structure of this table is similar to that of Table 3.7, but with an additional central column. This central column corresponds to instances without relevant differences in the objective function value (i.e., $(F_{FR+BCD} - F_{BC})/F_{BC}$ less than 5%). Each cell of Table 3.9 reports the number and names of its instances. The first row includes the instances that were solved faster with FR+BCD than with BC, and the instances that could not be solved in the 2-hour time limit by BC, but they could by FR+BCD.

instance	F_{BC}	F_{FR}	F_{FR+BCD}	T_{BC}	T_{FR}.	T_{BCD}	$\Delta(F_{FR}, F_{FR+BCD})$
table6‡	28331962414	29686800000	29899600000	2.17	1	0.38	-0.72
dale	256	256	256	7199.96	595.9	596.62	0
hier13	434834824.5	444063000	444063000	1312.7	747.3	8.18	0
hier13d4	5.11488e+12	6143970000	6143970000	7201.23	6410.7	2769.53	0
hier13x13x13a	434834824.5	436127000	436127000	895.44	542.7	6.16	0
hier13x13x13b	44385.67	44865.8	44865.8	1444.94	584.9	7.14	0
hier13x13x13c	368036.2	370564	370564	1561.69	542.9	8.47	0
hier13x13x13d	414115.44	424074	424074	340.48	178.1	6.89	0
hier13x13x13e	4644973.87	4693570	4693570	269.83	336.7	6.83	0
hier13x7x7d	594401	593370	593370	29.19	2.7	0.24	0
hier16	591756145.8	556221000	556221000	7200.41	4854.5	130.81	0
hier16x16x16b	74891.53	76700.4	76700.4	7200.44	3401.2	121.52	0
osorio	13	13	13	1.8	1.9	1.35	0
sbs2008_E	109959.57	109960	109960	2.95	4.6	0.13	0
table7	9970266227	10031200000	10031200000	0.04	0.1	0.1	0
toy3dsarah	5.0747e+14	5.07506e+14	5.07506e+14	37.46	25.5	0.22	0
hier13x13x7d	1684140	1695250	1686430	241.81	34.7	2.59	0.52
table8	439	450	445	0.09	0.2	0.18	1.11
hier16x16x16a	529703489.9	532145000	525466000	7200.31	2803.8	372.61	1.26
targus	1103759.75	1103760	1088480	0.02	0.1	0.08	1.38
nine5d	6.20788e+12	1215790000	1191810000	7200.51	6133.3	2389.56	1.97
table5	10154665.5	11094800	10837600	7200.21	720.5	258.66	2.32
hier16x16x16c	604844.28	620633	601548	7200.39	3488.3	672.13	3.08
table4	10290147784	11843300000	11433900000	7200.27	1196.9	142.92	3.46
hier16x16x16e	9543201.06	9485820	9077750	7200.41	3862.2	692.34	4.3
table1	2.93185e+13	3.04227e+13	2.89576e+13	1.45	0.6	0.54	4.82
two5in6	707133564.8	751514000	713214000	7200.45	3010.9	169.86	5.1
hier16x16x16d	752648610.3	765577000	725869000	7200.31	3776.8	789.54	5.19
table3	1.20849e+12	1.39866e+12	1.29248e+12	7200.24	1909.2	171.17	7.59
destatis	234541294	286199000	241329000	2528.91	204.3	29.83	15.68
bts4	4114851966	3180710000	2592590000	7200.6	6332.2	6002.29	18.49
sbs2008_C	320835.46	621448	459655	66.34	543.9	0.43	26.03
australia_ABS	651	2396	746	2.91	6.05	4.46	68.86
cbs†	0	268	0	2.88	2.9	1.79	100

‡ This negative improvement is due to unprotected cells in FR solution.
† cbs instance has a global optimum of zero because all the sensitive cells have null weights in the objective function.

Table 3.8: Comparison between plain branch-and-cut and FR+BCD with real-world instances. Instances ordered by $\Delta(F_{FR}, F_{FR+BCD})$.

		Objective Function		
		FR+BCD < BC	FR+BCD ≈ BC	FR+BCD > BC
Time	FR+BCD < BC	bts4 hier16 nine5d table3 table4 table5 [N=6]	dale destatis hier13 hier13d4 hier13x13x13a hier13x13x13b hier13x13x13c hier13x13x13d hier13x13x7d hier13x7x7d hier16x16x16a hier16x16x16b hier16x16x16c hier16x16x16d hier16x16x16e table1 table6 toy3dsarah two5in6 [N=19]	[N=0]
	FR+BCD > BC	[N=0]	cbs hier13x13x13e osorio sbs2008_E table7 table8 targus [N=7]	australia_ABS sbs2008_C [N=2]

Table 3.9: Summary of results for real instances, in the comparison between FR+BCD versus BC.

Moreover, some instances were not suitably protected: bts4, dale, table1, table3, table4, table5, table6 and table7 present some sensitive cells unprotected in the BC solution; dale, table5 and table6 had the same problem with FR, and BCD was in trouble as well with bts4 and table6: in general, FR+BCD dealt better than BC with these difficult instances.

Nine tables were solved faster with the pure BC scheme, but it is worth noting that only two (sbs2008_C and hier13x13x13e) can be considered as challenging, since they needed more than one minute to be solved, whereas four (osorio, table7, table8 and targus) have few sensitive cells and could be solved very quickly by both FR+BCD and BC.

To sum up, Table 3.9 shows that the combination FR+BCD is competitive with BC in the solution's quality, and, in addition, it protects the table in significantly less time.

3.4.4 Comparison between fix-and-relax and other heuristics

Current state-of-the-art MILP solvers can be turned into heuristic approaches by tuning some of their pre-build heuristics. For a fair comparison, FR is tested in this section against feasibility pump (FP), relaxation induced neighborhod search (RINS), and FR+BC (warm starting CPLEX from the FR solution) with and without polishing.

Fix-and-relax and feasibility pump heuristics

As we discussed in Chapter 2, FP [36] is considered an efficient heuristic for the fast computation of hopefully good initial feasible solutions to MILPs. We used the *objective feasibility pump* (oFP) [2], which is more efficient than FP in terms of quality of the solution and the *analytic center feasibility pump* (AC-FP) [4], also introduced in Chapter 2, as a good alternative in some MILP instances (either in time or quality of the solution). Table 3.10 shows a comparison between FR and these FP variants for real instances. It reports the primal gap of the FR and FP solutions (columns "$GAP_{FR}\%$" and "$GAP_{FP}\%$", respectively), the CPU time required by FR and FP to compute the feasible solution (columns "T_{FR}" and "T_{FP}", respectively), and the difference between both methods in CPU times and gaps (columns "$\Delta(T_{FP}, T_{FR})$" and "$\Delta(FP, FR)$", respectively). We ran both oFP and AC-FP. Table 3.10 only shows the result of the best FP variant, i.e., the one that provides the lowest gap, and in case of equal gaps, the fastest one. The best FP variant is clearly marked in the table.

It is clearly seen that FR outperformed FP for CTA in terms of quality of the solution. In most cases, FR provided a better gap than FP by a big difference. Only in three instances FP was better. FP reached the time limit without a feasible solution in two instances. However, FP is in general faster than FR in order to find a feasible solution. It can be concluded that, for the CTA problem, FR instead of FP should be used for finding good feasible solutions within a reasonable short time.

Fix-and-relax and RINS and local branching heuristics

RINS [22] is a heuristic that explores a neighborhood of the current incumbent solution and the continuous relaxation at a node h of the BC tree to try to find a new and improved incumbent. CPLEX BC incorporates RINS, allowing the user to control how often to apply the heuristic through a frequency parameter f. A value $f > 0$ means that RINS is applied at nodes $h = 0, f, 2f, ...$ while for $f = 0$ CPLEX automatically decides when to apply the heuristic. The results of Table 3.6 were obtained with $f = 0$; as it was shown in that table, FR outperformed BC

instance	$GAP_{FR}\%$	T_{FR}	$GAP_{FP}\%$	T_{FP}	$\Delta(T_{FP}, T_{FR})$	$\Delta(FP, FR)$
australia_ABS	73,87	6,05	$95{,}90^{AC-FP}$	26	19,95	22,03
bts4	66,57	6332,15	$74{,}09^{oFP}$	552	-5780,15	7,52
cbs	100,00	2,87	$100{,}00^{oFP}$	20	17,13	0,00
dale	48,44	595,85	$98{,}52^{oFP}$	27	-568,85	50,08
destatis	19,53	204,32	$21{,}93^{AC-FP}$	222	17,68	2,40
hier13d4	82,86	6410,73	†	†	†	†
hier13	6,88	747,29	$58{,}96^{oFP}$	126	-621,29	52,08
hier13x13x13a	5,22	542,72	$63{,}36^{AC-FP}$	122	-420,72	58,14
hier13x13x13b	5,89	584,92	$53{,}36^{AC-FP}$	235	-349,92	47,47
hier13x13x13c	5,63	542,86	$54{,}01^{oFP}$	217	-325,86	48,38
hier13x13x13d	4,69	178,12	$99{,}86^{oFP}$	132	-46,12	95,17
hier13x13x13e	5,39	336,72	$99{,}87^{oFP}$	128	-208,72	94,48
hier13x13x7d	5,58	34,72	$60{,}49^{AC-FP}$	13	-21,72	54,91
hier13x7x7d	4,82	2,71	$73{,}56^{oFP}$	2	-0,71	68,73
hier16	59,48	4854,46	$68{,}36^{AC-FP}$	2852	-2002,46	8,89
hier16x16x16a	44,96	2803,76	$99{,}99^{oFP}$	4537	1733,24	55,03
hier16x16x16b	33,49	3401,2	$99{,}91^{oFP}$	3742	340,8	66,42
hier16x16x16c	40,86	3488,27	$99{,}93^{oFP}$	3937	448,73	59,07
hier16x16x16d	57,66	3776,81	$66{,}88^{AC-FP}$	2706	-1070,81	9,22
hier16x16x16e	46,55	3862,21	$81{,}19^{oFP}$	4430	567,79	34,64
nine5d	67,69	6133,25	†	†	†	†
osorio	0,00	1,86	$27{,}65^{oFP}$	0	-1,86	27,65
sbs2008_C	50,11	543,88	$82{,}64^{oFP}$	12	-531,88	32,53
sbs2008_E	4,73	4,56	$74{,}15^{oFP}$	2	-2,56	69,42
table1	8,38	0,62	$2{,}17^{oFP}$	0	-0,62	-6,20
table3	25,39	1909,18	$100{,}00^{oFP}$	323	-1586,18	74,61
table4	25,39	1196,86	$96{,}81^{AC-FP}$	379	-817,86	71,42
table6	7,77	1,01	$9{,}05^{oFP}$	0	-1,01	1,28
table7	1,01	0,1	$69{,}21^{oFP}$	0	-0,1	68,20
table8	2,44	0,18	$6{,}51^{oFP}$	0	-0,18	4,07
targus	3,84	0,08	$0{,}92^{oFP}$	0	-0,08	-2,92
toy3dsarah	0,34	25,47	$65{,}07^{oFP}$	5	-20,47	64,73
two5in6	66,04	3010,89	$61{,}91^{AC-FP}$	5234	2223,11	-4,13

† Time limit reached without finding a feasible feasibility pump solution.
^{oFP}: best solution provided by oFP.
$^{AC-FP}$: best solution provided by AC-FP.

Table 3.10: Comparison between fix-and-relax and feasibility pump for real instances.

in a considerable percentage of real-world instances. Table 3.11 adds a comparison between FR and BC with $f = 50$. The meaning of columns is the same as in Table 3.6. The value of f, either 0 or 50, is reported in the new column $RINS_f$. We only considered the subset of real-world instances whose BC tree had more than 50 nodes. From Table 3.11 it can be concluded that FR still outperforms BC with the RINS heuristic using $f = 50$.

We additionally tried RINS frequencies $f \in \{100, 150, 200\}$, obtaining *exactly* the same results (they are thus omitted in Table 3.11). As stated above, CPLEX always applies the RINS heuristic at node 0 for any $f > 0$. We noted that, since RINS is an expensive heuristic, it exhausted most of the allowed time (that of the FR heuristic) at node 0, making irrelevant the particular value of f. Therefore, at least for this particular application, RINS $f = 0$ seems to be the best choice. Indeed, we noted that when $f = 0$ CPLEX does not apply RINS to node 0 in many instances.

The local branching (LBr) heuristic also explores the neighborhood of an incumbent solution, but by adding constraints based on the number of binary variables flipping their values with respect the incumbent [32]. Running CPLEX with the LBr heuristic, and setting as time limit the CPU time of FR, we only observed differences with RINS $f = 0$ for five instances of Table 3.11: hier13x13x7d (solutions of 6.2% and 7.2% gaps for LBr and RINS, respectively), hier13x7x7d (24.1% gap for LBr, $12,4\%$ gap for RINS), hier16 (61.2% for LBr, 63.0% for RINS), hier16x16x16a (44.4% for LBr, 49.0% for RINS), and hier16x16x16d (62.4% for LBr, $63,0\%$ for RINS). LBr only clearly outperformed RINS $f = 0$ in hier16x16x16a; for that instance, LBr was also more efficient than FR.

3.4.5 Using fix-and-relax to warm start branch-and-cut

Table 3.12 shows the results obtained with FR+BC (i.e., warm starting BC with the FR solution) on 1H2D tables. The table reports the CPU computation time and gap (as a percentage) of the FR solution (columns "T_{FR}" and "$GAP_{FR}\%$"). The same information is provided for the FR+BC solution using a 1% optimality gap (columns "T_{FR+BC}" and $GAP_{FR+BC}\%$); and for CPLEX BC without starting point with the same 1% optimality gap (columns T_{BC} and $GAP_{BC}\%$). Columns $\Delta(FR+BC, BC)$ and $\Delta(T_{FR+BC}, T_{BC})$ give the difference in gap and CPU time between the FR+BC and BC solutions. A time limit of one hour was considered for these runs. Some FR+BC or BC executions were unable to find a solution of

instance	T_{FR}	$GAP_{FR}\%$	$RINS_f$	$GAP_{BC}\%$	$\Delta(BC,FR)$	$GAP_{BC}^{up}\%$	T_{BC}^{up}	$\Delta(T_{FR},T_{BC}^{up})$
bts4	6332,15	66,57	0	74,16	7,59	‡	‡	‡
			50	100	33,43	‡	‡	‡
dale	595,85	48,44	0	48,44	0	48,44	7199,96	6604,11
			50	48,44	0	‡	‡	‡
destatis	204,32	19,53	0	99,97	80,44	1,8	706,48	502,16
			50	99,97	80,44	1,78	3090,16	2885,84
hier13	747,29	6,88	0	4,9	-1,98	4,9	159,24	-588,05
			50	99,98	93,1	4,9	1239,3	492,01
hier13x13x13a	542,72	5,22	0	5,22	0	4,94	690,15	147,43
			50	99,99	94,77	4,94	1426,84	884,12
hier13x13x13b	584,92	5,89	0	5,24	-0,65	5,24	369,13	-215,79
			50	99,98	94,09	4,87	2742,67	2157,75
hier13x13x13c	542,86	5,63	0	4,99	-0,65	4,99	243,43	-299,43
			50	99,98	94,34	4,99	3405,16	2862,3
hier13x13x7d	34,72	5,58	0	7,19	1,61	4,96	69,32	34,6
			50	38,93	33,34	4,84	158,96	124,24
hier13x7x7d	2,71	4,82	0	12,35	7,53	‡	‡	‡
			50	24,1	19,28	4,53	20,09	17,38
hier16	4854,46	59,48	0	63,07	3,59	‡	‡	‡
			50	100	40,52	‡	‡	‡
hier16x16x16a	2803,76	44,96	0	48,8	3,84	44,71	3706,65	902,89
			50	100	55,03	43,87	7200,57	4396,81
hier16x16x16d	3776,81	57,66	0	63,32	5,66	57,07	5369,03	1592,22
			50	100	42,33	56,08	7200,59	3423,78
hier16x16x16e	3862,21	46,55	0	46,87	0,32	‡	‡	‡
			50	99,96	53,41	46,18	7200,52	3338,31
table3	1909,18	25,39	0	15,07	-10,32	17,11	408,3	-1500,88
			50	100	74,61	16,78	2303,34	394,16
table4	1196,86	25,39	0	17,3	-8,09	18,6	511,64	-685,22
			50	100	74,61	14,64	2163,97	967,11
table5	720,49	22,8	0	20,2	-2,6	20,25	216,19	-504,3
			50	100	77,2	16,96	1441,96	721,47

‡ Time limit reached without improving the feasible fix-and-relax solution.

Table 3.11: Comparison between FR and BC with frequency RINS f equal to 0 and 50 for some real-world instances.

instance	T_{FR}	$GAP_{FR}\%$	T_{FR+BC}	$GAP_{FR+BC}\%$	T_{BC}	$GAP_{BC}\%$	$\Delta(FR+BC, BC)$	$\Delta(T_{FR+BC}, T_{BC})$
asym-40-50-5	72.76	1.89	264.45	0.47	306.46	0.35	0.12	−42.01
asym-40-50-15	158.75	3.13	$\dagger^{(1223.55,1)}$	0.71	$\dagger^{(1357.15,3)}$	0.67	0.04	−133.60
asym-40-50-30	**210.51**	**5.41**	$\dagger^{(\mathbf{1606.53,2})}$	**0.82**	$\dagger^{(\mathbf{2062.43,3})}$	**1.04**	**−0.23**	**−455.90**
asym-40-60-5	95.40	1.75	259.40	0.40	231.36	0.40	0.00	28.04
asym-40-60-15	**193.37**	**3.11**	$\dagger^{(\mathbf{1352.60,2})}$	**0.93**	$\dagger^{(\mathbf{1430.58,4})}$	**1.02**	**−0.09**	**−77.98**
asym-40-60-30	314.96	5.07	$\dagger^{(2181.63,6)}$	1.40	$\dagger^{(2040.04,6)}$	1.21	0.19	141.59
asym-50-50-5	110.54	1.53	$\dagger^{(560.24,1)}$	0.43	476.49	0.35	0.08	83.75
asym-50-50-15	**186.95**	**2.86**	$\dagger^{(\mathbf{1403.85,3})}$	**0.90**	$\dagger^{(\mathbf{1533.78,4})}$	**0.93**	**−0.03**	**−129.93**
asym-50-50-30	333.50	5.81	$\dagger^{(2417.91,5)}$	1.52	$\dagger^{(2284.80,5)}$	1.74	−0.22	133.11
asym-50-60-5	153.14	1.06	268.43	0.64	286.51	0.48	0.16	−18.08
asym-50-60-15	**278.72**	**2.73**	$\dagger^{(\mathbf{1263.52,3})}$	**1.24**	$\dagger^{(\mathbf{1408.72,3})}$	$\ddagger^{(\mathbf{8.67,2})}$	**−7.43**	**−145.21**
asym-50-60-30	416.11	5.54	$\dagger^{(2768.88,8)}$	1.60	$\dagger^{(2665.53,7)}$	$\ddagger^{(1.74,1)}$	−0.14	103.35
sym-40-50-5	8.99	2.00	41.51	0.40	29.10	0.68	−0.28	12.40
sym-40-50-15	115.34	4.21	1114.50	0.75	720.22	0.80	−0.06	394.29
sym-40-50-30	371.35	4.62	$\dagger^{(3097.98,4)}$	1.93	$\dagger^{(3131.25,3)}$	1.77	0.16	−33.27
sym-40-60-5	10.52	2.68	32.82	0.63	60.05	0.52	0.11	−27.23
sym-40-60-15	102.29	2.16	1731.73	0.86	1381.75	0.82	0.04	349.98
sym-40-60-30	**800.45**	**4.40**	$\dagger^{(\mathbf{3600,5})}$	**3.22**	$\dagger^{(\mathbf{3600,5})}$	**6.55**	**−3.33**	**0**
sym-50-50-5	25.47	1.88	103.45	0.50	85.12	0.57	−0.07	18.33
sym-50-50-15	166.33	3.66	$\dagger^{(2370.98,1)}$	0.97	$\dagger^{(1728.63,1)}$	0.75	0.22	642.35
sym-50-50-30	**511.19**	**3.45**	$\dagger^{(\mathbf{3600,5})}$	**2.54**	$\dagger^{(\mathbf{3600,5})}$	**4.46**	**−1.92**	**0**
sym-50-60-5	56.80	1.59	80.64	0.54	134.79	0.36	0.17	−54.15
sym-50-60-15	279.63	2.46	2347.58	0.91	1894.95	0.74	0.17	452.63
sym-50-60-30	**833.45**	**3.93**	$\dagger^{(\mathbf{3600,5})}$	**3.54**	$\dagger^{(\mathbf{3600,5})}$	**6.31**	**−2.77**	**0**

$\dagger^{(x,y)}$ a solution within 1% optimality gap could not be found in y of the overall number of replications within the time limit of 3600 seconds; x is the average CPU time for the remaining successful runs.

$\ddagger^{(z,w)}$ no feasible solution was found in w of the overall number of replications within the time limit of 3600 seconds; z is the average gap for the remaining runs.

Table 3.12: Using the fix-and-relax solution to warm start CPLEX branch-and-cut.

1% optimality gap within this time limit; these are clearly marked in Table 3.12. BC could not find a feasible solution within the time limit for three instances, which are also clearly marked in the table. In those situations the average gap reported in Table 3.12 may be greater than 1%.

FR+BC provided a lower gap than BC in 12 of 24 cases. In addition, in six of these 12 cases the CPU time of FR+BC was inferior. These six successful FR+BC executions are marked in boldface in Table 3.12. These results are not entirely satisfactory, since it could be expected that providing a good incumbent from the beginning would significantly reduce the computational burden for all the instances, by pruning portions of the search space. In fact, we found reported similar experiences. In `http://www2.isye.gatech.edu/~rcarvajal3/2012/2012-12-24_effect-of-information` the author presents an experiment with instances from MIPLIB 2010 [51] where providing the optimal solution as a warm start can actually be harmful for the performance of the solver.

We also applied the CPLEX polishing heuristic to the FR starting point. This heuristic, which can be very time consuming, tries to exploit an initial feasible solution provided to BC by solving an alternative branch-and-cut. We ran FR+BC with and withoug polishing. Activating the polishing the gap was improved in 92% of the executions; however the average gap reduction was 0.4%. On the other hand, in 83% of the executions the polishing significantly increased the CPU time: an average increment of 59%. In the remaining 17% of executions the CPU time was reduced, in average, a 18%. From these figures, it can be concluded that the polishing is in general very time consuming for CTA, and it is not worth the gap reduction provided.

Chapter 4

Stabilized Benders methods for large combinatorial optimization problems: applications to cell suppression

In previous chapters we focused on heuristic techniques applied to challenging MILPs. The current one deals with Benders decomposition which, although is widely used in many real-world applications, it suffers from well-known instability issues that limit its efficiency. Benders decomposition is an iterative method which decomposes the original MILP in several smaller subproblems theoretically easier to solve and provides the optimal solution after a finite number of iterations. Despite this, the convergence to the optimum is often too slow due to the fact that the solutions tend to oscillate wildly among different feasible regions by jumping from a good point, i.e. close to optimality, to a much worse one. This chapter addresses this issue and proposes a stabilized Benders decomposition (SBD) in order to prevent this behaviour. In particular, we focus on finding new solutions inside trust feasible regions, i.e. neighbourhoods of well considered points, where we expect to find better solutions.

The chapter is structured as follows: In Section 4.1 we recall the classical Benders algorithm. In Section 4.2 we present the SBD applied to MILP problems. In section 4.3 we apply the SBD to the cell suppression problem. Finally, numerical results are presented in Section 4.4, showing that stabilization techniques allow to find better feasible solutions with the same computational effort.

4.1 Benders decomposition

Briefly, Benders decomposition [5], is an iterative method that allows to decompose the original MILP in several smaller subproblems (referred to relaxed master and slaves) and after a finite number of iterations the method provides an optimal solution. Originally, this method was suggested for problems with two types of variables where one of them are considered as "complicating variables". In MILP models complicating variables are the binary/integer ones. We consider the following MILP primal problem (P) in variables (x, y):

$$(P) \qquad \begin{array}{ll} \min & c^T x + d^T y \\ \text{s. to} & Dx + Fy = b \\ & x \geq 0 \\ & y \in Y, \end{array}$$

where y are the binary/integer complicating variables, $c, x \in \mathbb{R}^{n_1}$, $d, y \in \mathbb{R}^{n_2}$, $D \in \mathbb{R}^{m \times n_1}$ and $F \in \mathbb{R}^{m \times n_2}$. For binary problems, we have $Y = \{0,1\}^{n_2}$. Problem (P) can be formulated in the equivalent form:

$$(P') \qquad \min_y \ \left\{ d^T y + \min_x \left\{ c^T x | Dx = b - Fy, x \geq 0, \right\} y \in Y \right\}.$$

Writing the dual form of the inner minimization problem, called slave problem:

$$(SP_D) \qquad \max_u \left\{ u^T(b - Fy) | D^T u \leq c, u \in \mathbb{R}^m \right\},$$

the problem (P') can be formulated as:

$$(P'') \qquad \min_y \ \left\{ d^T y + \max_u \left\{ u^T(b - Fy) | D^T u \leq c, u \in \mathbb{R}^m, \right\} y \in Y \right\},$$

where the feasible region of (SP_D) does not depend on the value of y, which only affects the objective function. Depending on the result of the (SP_D), we have two possible scenarios:

1. If (SP_D) is unbounded, for some y fixed, then there must exist an $v \geq 0$ verifying $D^T v \leq c$ for which $v^T(b - Fy) > 0$; v is a ray or extreme direction representing an unbounded direction in the dual polyhedron.

2. If (SP_D) is feasible for a given y, then we get an extreme point u of the dual polyhedron such that $u^T(b - Fy) \leq 0$.

Enumerating extreme points (u), rays (v) and introducing variable θ, one can write the original problem (P) as follows:

$$
(MP) \qquad \begin{array}{lll} \min & \theta & \\ \text{s. to} & \theta \geq d^T y + u^{iT}(b - Fy) & i = 1, \ldots, p \\ & v^{jT}(b - Fy) \leq 0 & j = 1, \ldots, t \\ & y \in Y. & \end{array}
$$

Problem (MP) is impractical since p and t can be very large and in addition the extreme points and rays are unknown. Instead, the method considers a relaxation of MP with a subset of the extreme points and rays. The relaxed Benders problem (called master problem) is the following:

$$
(RMP_r) \qquad \begin{array}{lll} \min & \theta & \\ \text{s. to} & \theta \geq d^T y + u^{iT}(b - Fy) & i \in I \subseteq \{1, \ldots, p\} \\ & v^{jT}(b - Fy) \leq 0 & j \in J \subseteq \{1, \ldots, t\} \\ & y \in Y. & \end{array}
$$

Although the subindex $_r$, which denotes the iteration number, does not appear in the definition of (RMP_r), we keep it for a consistent notation with $(RSMP_r)$, the stabilized master version, which is defined below. Initially $I = J = \emptyset$ and iteratively new and non repeated constraints are added to the relaxed master problem (RMP_r): a feasibility cut $v^{jT}(b - Fy) \leq 0$ when (SP_D) is unbounded, and an optimality cut $\theta < d^T y + u^{iT}(b - Fy)$ if (SP_D) is bounded but $\theta < d^T y + u^{iT}(b - Fy)$. Otherwise, the optimal solution to original problem (P) is found. In summary, the steps of the Benders algorithm are:

Benders algorithm

1: Initially $I = \emptyset$ and $J = \emptyset$. Let (θ_r^*, y_r^*) be the solution of current master problem (RMP_r), and (θ^*, y^*) the optimal solution of (MP).

2: Solve master problem (RMP_r) obtaining θ_r^* and y_r^*. At first iteration, $\theta_r^* = -\infty$ and y_r is any feasible point in Y.

3: Solve subproblem (SP_D) using $y = y_r^*$.

4: **if** (SP_D) has finite optimal solution in vertex u^{i0} **then**

5: **if** $\theta_r^* = d^T y_r^* + u^{i_0 T}(b - F y_r^*)$ **then**
6: STOP. Optimal solution is $y^* = y_r^*$ with cost $\theta^* = \theta_r^*$.
7: **else if** $\theta_r^* < d^T y_r^* + u^{i_0 T}(b - F y_r^*)$ **then**
8: This solution violates constraint $\theta > d^T y + u^{i_0 T}(b - F y)$ of (MP).
9: Add this new constraint to (RMP_r): $I \leftarrow I \cup \{i_0\}$.
10: **end if**
11: **else if** (SP_D) is unbounded along segment $u^{i_0} + \lambda v^{j_0}$ **then**
12: This solution violates constraint $v^{j_0 T}(b - F y) \leq 0$ of (MP).
13: Add this new constraint to (RMP_r): $J \leftarrow J \cup \{j_0\}$.
14: Vertex may also be added: $I \leftarrow I \cup \{i_0\}$.
15: **end if**
16: Go to step 2.

Notice that, convergence of Benders decomposition is always guaranteed with a maximum of $r = p + t$ number of iterations. In practice, the number of required iterations may be excessive due, among other causes, to instability issues. In order to overcome this drawback, we have developed a specialized stabilization Benders decomposition. This strategy is described in detail next.

4.2 Stabilizing Benders through local branching constraints

At each iteration of Benders decomposition, one solves the current master problem relaxation and sends the optimal solution to the slave problem. Depending on the result: 1) we stop because we have found the optimal solution; 2) if no optimal solution is found, a cut is generated. The main cause for slow convergence is due to the generation of weak Benders cuts as a result of obtaining "bad" points y_r^* when we solve the master problem (RMP_r) [46]. The idea behind the stabilized Benders decomposition is to search new solutions y_r^* as close as possible to properly chosen points, so called stability center points.

For binary MILPs, the stabilization can be done by adding linear constraints that restrict the feasible region of relaxed master problems RMP_r. This is made possible by using the Hamming distance defined from a stability center point $(\bar{y})$, not necessarily feasible, and a radius $K_r \geq 1$ [59]. This restricted feasible region of size K_r is called trust region (TR). Note that K_r can be either a constant or dynamically updated at each iteration r. TR is defined by a well-known local branching constraint which limits the "switching" of binary variables only at most K_r [32]. These local branching constraints prevent the master problem solution from moving too far from the stability center point $\bar{y}$. The local branching constraint is defined as follows:

$$\triangle(y,\bar{y}) = \sum_{j\in\Omega}(1-y_j) + \sum_{j\in\{1,\ldots,n_2\}\setminus\Omega}(y_j) \leq K_r$$

where $\Omega := \{j \in \{0,1\}^{n_2} : \bar{y}_j = 1\}$. The local branching constraint can be used as a branching criterion within an enumerative scheme. Indeed, given the stabilized center point $\bar{y}$, the feasible region space to explore can be partitioned by means of the disjunction $\triangle(y,\bar{y}) \leq K_r$ or $\triangle(y,\bar{y}) \geq K_r + 1$ (reverse local branching constraints). Let us define the new relaxed stabilized master problem as:

$$
(RSMP_r) \qquad
\begin{array}{lll}
\min & \theta & \\
\text{s. to} & \theta \geq d^T y + u^{iT}(b - Fy) & i \in I \subseteq \{1,\ldots,p\} \\
 & v^{jT}(b - Fy) \leq 0 & j \in J \subseteq \{1,\ldots,t\} \\
 & \triangle(y,\bar{y}) \leq K_r \quad \text{or} \quad \triangle\,(y,\bar{y}) \geq K_r + 1 & \\
 & y \in Y. &
\end{array}
$$

The main benefits of stabilization techniques are [3, 10, 38]:

- Reduction of the total computational time because fewer iterations are required. Moreover, relaxed master problems of smaller feasible region and theorically easier need to be solved.
- The search for solutions around a good point considered increases the chance of finding better feasible solutions.

An outline of the stabilized Benders framework is shown in Figure 4.1. The algorithm finds an initial feasible solution (x_0^*, y_0^*) and an upper bound $\rho^{ub} = c^T x_0^* + d^T y_0^*$ by solving the original problem (P) with a primal heuristic. This point y_0^* is used as stability center. A local branching constraint, based on this stability center and an initial $1 \leq K_r \leq |S|$, is added to the master Benders problem ($RSMP_r$). At each iteration we solve the master ($RSMP_r$) to obtain a new solution y_r^* and a lower bound θ_r^*. Note that this lower bound is only local because of a set of local branching constraints are present in the problem. If

(SP_D) has finite optimal solution in vertex u^{i_0} and $\theta_r^* = d^T y_r^* + u^{i_0 T}(b - F y_r^*)$ we delete the last local branching constraint based on the stability center $\bar{y}$ because there is no better solution in this trust region and we update the stability center with the current y_r^* (lines 25-26 of algorithm of Figure 4.1). A new local branching constraint considering the new stability center is added (step 4 of algorithm of Figure 4.1). However, if $\theta_r^* < d^T y_r^* + u^{i_0 T}(b - F y_r^*)$ an optimality cut $\theta > d^T y + u^{i_0 T}(b - F y)$ is added to master $(RSMP_r)$ (steps 29-30 of algorithm of Figure 4.1). In this case, we have to expand the space to explore. Adding the reverse local branching constraint $\triangle(y, \bar{y}) \geq K_r + 1$ (lines 12-13 of the algorithm of Figure 4.1) we ensure that the master problem will not explore in the previous neighborhood already explored. Hopefully, this makes the master problem easier to solve because of reducing the feasible region. If (SP_D) is unbounded along segment $u^{i_0} + \lambda v^{j_0}$ a feasibility cut $v^{j_0 T}(b - F y) \leq 0$ is added to master $(RSMP_r)$ (steps 34-36 of algorithm of Figure 4.1). The convergence is only guaranteed when $K_r \geq n_2$ and $RSMP_r$ is infeasible. At this point we'll have explored all the feasible space and we have a global optimal solution. However a significant improvement has been carried out in our implementation. Every time we obtain a better feasible solution we drop all the stabilization constraints (i.e, we solve RMP_r instead of $RSMP_r$) in order to obtain a valid global lower bound (step 21 of algorithm of Figure 4.1). If we are in the optimal case, we can stop (step 23 of algorithm of Figure 4.1). We want to highlight one of the most important steps of the algorithm, when a new K_r is chosen (step 11 of algorithm of Figure 4.1). This is a nontrivial step and there are different possible rules according to the problem at hand.

In [61], authors propose a different stabilization technique based on level stabilization where the new y_r^* points are chosen within a certain level set: the new value of the objective function is strictly better that the one we already have. They define a level parameter (LP) which forces to $c^T x + d^T y \leq \rho^{ub} - LP$.

4.3 Application to data privacy: the cell suppression problem

In Chapter 1 we introduced the cell suppression problem (CSP), one of the most used statistical disclosure control methods. As we have already discussed, CSP formulates a very large MILP problem of n binary variables, $2n|\mathcal{S}|$ continuous

1. Initial Heuristic: Let (x_0^*, y_0^*) be the initial solution of the original problem (P) by solving a primal heuristic (assuming (P) is feasible).
2. Let $\rho^{ub} = c^T x_0^* + d^T y_0^*$
3. Initialize stability center $\bar{y} := y_0^*$, $r = 1$ and choose $K_r \geq 1$
4. Add local branching constraint $\triangle(y, \bar{y}) \leq K_r$ to master $(RSMP_r)$
5. Solve Master $(RSMP_r)$
6. **if** $(RSMP_r)$ is infeasible **then**
7.   **if** $K_r \geq n_2$ **then**
8.   STOP. We have found the optimal solution ρ^{up} of (P) problem
9.   **end if**
10. $r \leftarrow r + 1$
11.   Choose K_r : $\quad K_{r-1} \leq K_r \leq n_2$
12.   Delete last local branching constraint $\triangle(y, \bar{y}) \leq K_{r-1}$
13.   Add the reverse local branching constraint $\triangle(y, \bar{y}) \geq K_{r-1}$
14.   GOTO line 4
15. **else**
16.   Let (θ_r^*, y_r^*) be the solution of $(RSMP_r)$. θ_r^* is a local lower bound
17.   Solve subproblem (SP_D) using $y = y_r^*$
18.   **if** (SP_D) is feasible in vertex u^{i_0} **then**
19. $\rho^{ub} = d^T y_r^* + u^{i_0 T}(b - F y_r^*)$
20.   **if** $\theta_r^* = \rho^{ub}$ **then**
21. ρ_*^{lb} = global lower bound by solving (RMP_r) with current sets I and J.
22.   **if** $\rho_*^{lb} = \rho^{ub}$ **then**
23.   STOP. We have found the optimal solution ρ^{up} of (P) problem
24.   **end if**
25. $\bar{y} := y_r^*$
26.   Delete last local branching constraint $\triangle(y, \bar{y}) \leq K_r$
27.   GOTO line 4
28.   **else if** $\theta_r^* < \rho^{ub}$ **then**
29.   This solution violates constraint $\theta > d^T y + u^{i_0 T}(b - Fy)$ of (MP)
30.   Add this new constraint to $(RSMP_r)$: $I \leftarrow I \cup \{i_0\}$
31.   GOTO line 7
32.   **end if**
33.   **else if** (SP_D) is unbounded along segment $u^{i_0} + \lambda v^{j_0}$ **then**
34.   This solution violates constraint $v^{j_0 T}(b - Fy) \leq 0$ of (MP).
35.   Add this new constraint to $(RSMP_r)$: $J \leftarrow J \cup \{j_0\}$.
36.   Vertex may also be added: $I \leftarrow I \cup \{i_0\}$.
37.   GOTO line 5
38.   **end if**
39. **end if**

Figure 4.1: The stabilized Benders method through local branching constraints

variables and $2(m + 2n)|\mathcal{S}|$ constraints. Trying to solve it with state-of-the-art MILP solvers becomes impractical even for tables of moderate size. Because of that, a Benders decomposition approach was suggested in the past for its solution [33]. The master Benders problem for CSP can be defined as:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} w_i y_i \\ \text{s. to} \quad & y_s = 1 \quad \forall s \in \mathcal{S} \\ & y_i \in \{0,1\} \quad i = 1, \ldots, n \\ & {v^j}^T y \geq \beta^j \quad j \in J, \end{aligned} \tag{4.1}$$

where $v^j \in \mathbb{R}^n$ and $\beta^j \in \mathbb{R}$ are the left and right hand sides of protection cuts (initially $J = \emptyset$). Note that primary cells are always suppressed even for $J = \emptyset$. The following slightly updated master CSP Benders is solved for the stabilized variant:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} w_i y_i \\ \text{s. to} \quad & y_s = 1 \quad \forall s \in \mathcal{S} \\ & y_i \in \{0,1\} \quad i = 1, \ldots, n \\ & {v^j}^T y \geq \beta^j \quad j \in J, \\ & \triangle(y, \bar{y}) \leq K_r \quad \text{or} \quad \triangle(y, \bar{y}) \geq K_r + 1. \end{aligned} \tag{4.2}$$

In order to guarantee that deviations $x^{l,s}$ and $x^{u,s}$ (supraindices are suppressed to simplify the notation) satisfy the first group of constraints of (1.3) and that, therefore, the suppression pattern $y_i, i = 1, \ldots, n$ 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 (1.3), the subproblems can be reduced to a feasibility problem. The subproblem for lower protection is

$$\begin{aligned} \min \quad & 0 \\ \text{s. to} \quad & Ax = 0 \\ & x_i \geq (l_i - a_i) y_i \quad i = 1, \ldots, n \\ & x_i \leq (u_i - a_i) y_i \quad i = 1, \ldots, n \\ & x_s \leq -lpl_s. \end{aligned} \tag{4.3}$$

while for upper protection is

$$\begin{array}{ll} \max & 0 \\ \text{s. to} & Ax = 0 \\ & x_i \geq (l_i - a_i)y_i \quad i = 1, \dots, n \\ & x_i \leq (u_i - a_i)y_i \quad i = 1, \dots, n \\ & x_s \geq upl_s. \end{array} \tag{4.4}$$

Alternatively, the two previous subproblems can be formulated as:

$$\begin{array}{rlll} -lpl_s \geq \min & x_s & & \\ \text{s. to} & Ax = 0 & & [\lambda] \\ & x_i \geq (l_i - a_i)y_i & i = 1, \dots, n & [\mu_l] \\ & x_i \leq (u_i - a_i)y_i & i = 1, \dots, n & [\mu_u], \end{array} \tag{4.5}$$

for lower protection and

$$\begin{array}{rlll} upl_s \leq \max & x_s & & \\ \text{s. to} & Ax = 0 & & [\lambda] \\ & x_i \geq (l_i - a_i)y_i & i = 1, \dots, n & [\mu_l] \\ & x_i \leq (u_i - a_i)y_i & i = 1, \dots, n & [\mu_u], \end{array} \tag{4.6}$$

for upper protection, λ, μ_l and μ_u being the set of Lagrange multipliers (also known as dual variables) of each group of constraints. Problems (4.5) and (4.6) have always a solution: (i) it is feasible, since $x = 0$ (no deviation) is a feasible but non optimal solution; (ii) it is not unbounded, since $x_s \geq l_s - a_s > -\infty$ (e.g., if table is positive then $l_s = 0$) and $x_s \leq u_s - a_s < \infty$. By LP duality, the dual of (4.5) is:

$$\begin{array}{ll} \max & 0\lambda + \sum_{i=1}^{n} (l_i - a_i) y_i \mu_{l_i} - \sum_{i=1}^{n} (u_i - a_i) y_i \mu_{u_i} = \\ & = \sum_{i=1}^{n} \left((l_i - a_i)\mu_{l_i} - (u_i - a_i)\mu_{u_i} \right) y_i \\ \text{s. to} & A^T \lambda + \mu_l - \mu_u = e_s \\ & \mu_l \geq 0, \quad \mu_u \geq 0, \end{array} \tag{4.7}$$

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

$$-lpl_s \geq \sum_{i=1}^{n} ((l_i - a_i)\mu_{l_i} - (u_i - a_i)\mu_{u_i})\, y_i. \tag{4.8}$$

If (4.8) holds for all $s \in \mathcal{S}$, then the suppression pattern y guarantees lower protection levels. If, for some $s \in \mathcal{S}$, (4.8) is not satisfied, then it is added to J, the set of protection constraints of the master problem. Similarly, we check whether the suppression pattern $y_i, i = 1, \ldots, n$ satisfies upper protection level upl_s for each cell $s \in \mathcal{S}$. If

$$upl_s \leq \sum_{i=1}^{n} (-(l_i - a_i)\mu_{l_i} + (u_i - a_i)\mu_{u_i})\, y_i, \tag{4.9}$$

is not satisfied for some $s \in \mathcal{S}$, (4.9) is added to J. Iteratively, Benders decomposition applied to CSP solves the master problem in variables $y_i, i = 1, \ldots, n$ and provides a suppression pattern. The protection is checked by solving $2|\mathcal{S}|$ subproblems (lower and upper sense per primary cell). If all primaries are protected, then the suppression pattern is optimal. Otherwise, a feasibility cut is added to the master problem, and the master is solved again. It is worth to note the equivalence between either using (4.5), (4.6) or (4.3), (4.4). The standard Benders or cutting plane procedure considers (4.3) and (4.4). Let's look at it in the case of lower protection. Considering Lagrange multipliers $\tilde{\lambda} \in \mathbb{R}^m$, $\tilde{\mu}_l \in \mathbb{R}^n$, $\tilde{\mu}_l \in \mathbb{R}^n$, and $\tilde{\mu}_s \in \mathbb{R}$ for the constraints of (4.3), its dual is:

$$\begin{aligned}
\max \quad & 0\tilde{\lambda} + \sum_{i=1}^{n}(l_i - a_i)y_i\tilde{\mu}_{l_i} - \sum_{i=1}^{n}(u_i - a_i)y_i\tilde{\mu}_{u_i} - (-lpl_s\tilde{\mu}_s) = \\
& = lpl_s\tilde{\mu}_s + \sum_{i=1}^{n}((l_i - a_i)\tilde{\mu}_{l_i} - (u_i - a_i)\tilde{\mu}_{u_i})\, y_i \\
\text{s. to} \quad & A^T\tilde{\lambda} + \tilde{\mu}_l - \tilde{\mu}_u = e_s\tilde{\mu}_s \\
& \tilde{\mu}_l \geq 0, \quad \tilde{\mu}_u \geq 0, \quad \tilde{\mu}_s \geq 0.
\end{aligned} \tag{4.10}$$

To avoid an unbounded solution we have to impose that there is no extreme ray in (4.10), that is,

$$lpl_s\tilde{\mu}_s + \sum_{i=1}^{n}((l_i - a_i)\tilde{\mu}_{l_i} - (u_i - a_i)\tilde{\mu}_{u_i})\, y_i \leq 0. \tag{4.11}$$

Dividing (4.11) by $\tilde{\mu}_s$, and defining $\lambda = \tilde{\lambda}/\tilde{\mu}_s$, $\mu_l = \tilde{\mu}_l/\tilde{\mu}_s$, $\mu_u = \tilde{\mu}_u/\tilde{\mu}_s$, (4.11) is equivalent to (4.8). Similarly, applying this change of multipliers, (4.10) is equivalent to (4.7) (aside of the constant term lpl_s, which appears in the primal formulation (4.5)).

4.3.1 Adding a normalization constraint to the subproblem

In this thesis we have considered an alternative selection criteria for Benders cuts. Note that the classical Benders approach uses a completely random selection policy, so many times the cuts generated are not the most effective. In [37], the authors have considered an alternative selection criteria for Benders cuts based on the correspondence between minimal infeasible subsystems of an infeasible LP and the vertices of the so-called alternative polyhedron. Computational results have shown a great performance. Following [37] we can apply this theory to our CSP problem by adding a normalization constraint to the Benders subproblems. Let's see it in the particular case of lower protection subproblem (4.10). Adding the normalization constraint we obtain the following dual subproblem (tildes of λ and μ are removed to simplify the notation):

$$
\begin{array}{ll}
\max & lpl_s\mu_s + \sum_{i=1}^{n} \left((l_i - a_i)\mu_{l_i} - (u_i - a_i)\mu_{u_i}\right) y_i \\
\text{s. to} & A^T\lambda + \mu_l - \mu_u - e_s\mu_s = 0 \\
 & \mu_l \geq 0, \quad \mu_u \geq 0, \quad \mu_s \geq 0 \\
 & \sum_{i=1}^{n} (w_{l_i}\mu_{l_i} + w_{u_i}\mu_{u_i}) + w_0\mu_s = 1.
\end{array}
\tag{4.12}
$$

The normalization constraint in (4.12) has two main benefits: (1) it makes the dual subproblem always bounded, such that it can be solved by any algorithm (either simplex or interior-point—interior-point methods do not work well with unbounded problems in this context since they do not provide a unbonded ray); (2) the subproblem may provide a deeper Bender's cut depending on the weights w_{l_i}, w_{u_i}, $i = 1, \ldots, n$, and w_0 in the normalization constraint.

By considering Lagrange multipliers $x \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$ for, respectively, the first group of equality constraints and the normalization constraint of (4.12), the associated primal subproblem (the normalized variant of (4.3)) is:

$$
\begin{aligned}
\min \quad & \alpha \\
\text{s. to} \quad & Ax = 0 \\
& x_i + w_{l_i}\alpha \geq (l_i - a_i)y_i \quad i = 1, \ldots, n \\
& x_i - w_{u_i}\alpha \leq (u_i - a_i)y_i \quad i = 1, \ldots, n \\
& x_s - w_0\alpha \leq -lpl_s.
\end{aligned}
\tag{4.13}
$$

Thanks to the normalization constraint and variable α, (4.12) and (4.13) are, never unbounded and infeasible, respectively, so their optimal objective values coincide. Therefore,

- if the optimal solution α^* is 0, then (4.13) is feasible (i.e., cell s is protected);
- if $\alpha^* > 0$, then (4.13) is infeasible and the optimal solution of (4.12) provides a ray, thus a Bender's infesibility cut.
- if $\alpha^* < 0$, then (4.13) is also feasible (cell s is protected).

4.4 Computational results

In this section we describe a series of computational experiments designed to empirically validate the efficiency of the proposed stabilized Benders decomposition for CSP. We have implemented all tested variants with GNU g++, using the state-of-the-art solver CPLEX 12.5. All the 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, unless explicitly stated. The numerical experiments have been performed on a set of real-world general and synthetic 1H2D tables. Real-world general tables are standard instances used in the literature [12]. We discarded some instances since they are too difficult for all tested variants (i.e, no feasible solution was obtained within the time limit). Synthetic instances were obtained with a generator of random 1H2D tables. 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.es/~jcastro/generators_csp.html`. We fixed all parameters, but 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 ($\mathtt{s} \in \{5, 10, 15\}$).

We considered asymmetric instances, i.e., instances where $u_{a_i} = \mathtt{a} \cdot l_{a_i}$ for all $i \in \mathcal{N}$. The asymmetry parameter considered is $\mathtt{a} = 5$. A total of 48 randomly 1H2D instances and 15 real-world tables were considered.

Tables 4.1 and 4.2 report the characteristics of each 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-a`. The default optimality gap of CPLEX was considered for all the optimization problems.

The parameter K_r takes the initial value of 1% of the total number of binary cells and it is sequentially increased to $K_r = [2\%|S|, 50\%|S|, 100\%|S|]$ when either an infeasible stabilized master problem ($RSMP_r$) is obtained or an optimal solution is already found within this trust region (note that this optimal solution for $RSMP_r$ could not be feasible for the CSP problem). We consider the following versions for the Benders subproblem (we only refer to subproblems for lower protection to shorten the explanation, though it should be undestand that upper protection subproblems are also solved):

- **meth1, meth2:** We solve the primal (4.5) and dual (4.7) subproblems, obtaining in both cases optimality Benders cuts.

- **meth3:** We solve the unbounded dual subproblems (4.10) but setting a 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:

$$f(\tilde{\mu}_s, \tilde{\mu}_l, \tilde{\mu}_u) = lpl_s\tilde{\mu}_s + \sum_{i=1}^{n} \left((l_i - a_i)\tilde{\mu}_{l_i} - (u_i - a_i)\tilde{\mu}_{u_i}\right) y_i.$$

  This target actually allows us to avoid feasibility cuts.

- **meth4:** We solve the normalized subproblem (4.12) where the normalization constraint is $\sum_{i=1}^{n}(w_{l_i}\mu_{l_i} + w_{u_i}\mu_{u_i}) + w_0\mu_s = 1$.

Instance	n	s	m	nz
40_50_10_5	7242	705	346	14637
40_50_5_5	7242	352	346	14637
40_60_10_5	10248	1002	412	20679
40_60_5_5	10248	501	412	20679
40_70_10_5	13916	1365	480	28045
40_70_5_5	13916	682	480	28045
40_80_10_5	11583	1136	467	23409
40_80_5_5	11583	568	467	23409
50_50_10_5	9639	940	393	19431
50_50_5_5	9639	470	393	19431
50_60_10_5	13725	1344	469	27633
50_60_5_5	13725	672	469	27633
50_70_10_5	9514	931	418	19241
50_70_5_5	9514	465	418	19241
50_80_10_5	17658	1736	542	35559
50_80_5_5	17658	868	542	35559
60_50_5_5	13923	680	477	27999
60_60_5_5	15494	759	498	31171
60_70_5_5	16685	819	519	33583
60_80_5_5	19926	980	570	40095
70_50_5_5	13515	660	469	27183
70_60_5_5	14945	732	489	30073
70_70_5_5	18247	896	541	36707
70_80_5_5	24786	1220	630	49815
40_50_15_5	7242	1057	346	14637
40_60_15_5	10248	1503	412	20679
40_70_15_5	13916	2047	480	28045
40_80_15_5	11583	1704	467	23409
50_50_15_5	9639	1410	393	19431
50_60_15_5	13725	2016	469	27633
50_70_15_5	9514	1396	418	19241
50_80_15_5	17658	2604	542	35559
60_50_10_5	13923	1360	477	27999
60_50_15_5	13923	2040	477	27999
60_60_10_5	15494	1518	498	31171
60_60_15_5	15494	2277	498	31171
60_70_10_5	16685	1638	519	33583
60_70_15_5	16685	2457	519	33583
60_80_10_5	19926	1960	570	40095
60_80_15_5	19926	2940	570	40095
70_50_10_5	13515	1320	469	27183
70_50_15_5	13515	1980	469	27183
70_60_10_5	14945	1464	489	30073
70_60_15_5	14945	2196	489	30073
70_70_10_5	18247	1792	541	36707
70_70_15_5	18247	2688	541	36707
70_80_10_5	24786	2440	630	49815
70_80_15_5	24786	3660	630	49815

Table 4.1: Characteristics of synthetic 1H2D instances.

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 4.2: Characteristics of real tables.

- **meth5:** As in meth4 but using the following normalized constraint: $\sum_{i=1}^{n}(w_{l_i}\mu_{l_i}+w_{u_i}\mu_{u_i})+w_0\mu_s \leq 1$.

All the different versions mentioned above are tested using the simplex (primal and dual) and the barrier method. Notice that, we have $c \in \mathcal{C}$ possible combinations depending whether: 1) we use meth1, meth2, meth3, meth4 or meth5; 2) we use primal, dual or barrier; and finally 3) type of Benders master problem used ($RSMP_r$ or RMP_r). In order to compare all the combinations we will make use of performance profiles proposed in [55]. Quality was measured as the value of the objective function (thus, the lower, the better) and CPU time. Let Q_{ic} be the quality of the solution or total CPU time of instance i solved by combination c. Note that Q_{ic} for CSP is always strictly positive. The performance ratio is thus defined as:

$$v(i,c)=\frac{Q_{i,c}}{\min\{Q_{i,c}: c\in\mathcal{C}\}},$$

i.e., the ratio between the quality of the solution or total CPU time obtained when instance i is solved by combination c over the combination with the best (minimum) performance for this instance. The (cumulative) distribution function $P_c(q):[1,\infty)\rightarrow[0,1]$ is defined as:

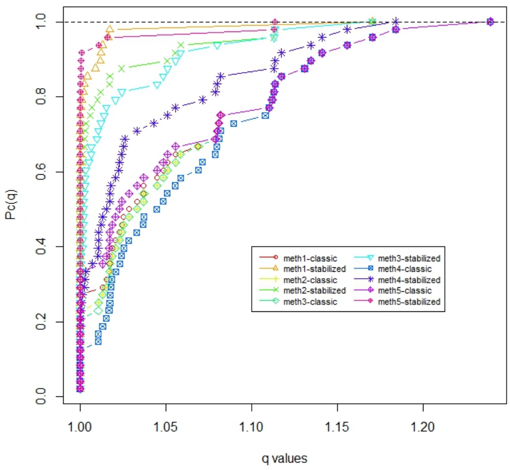
$$P_c(q)=\frac{|\{i\in\mathcal{I}: v(i,c)\leq q\}|}{|\mathcal{I}|}, q\geq 1,$$

where $\mathcal{I}$ is the set of instances. Figures 4.2 and 4.3 show different performance profiles based on quality of the solution and total cpu time, respectively. $P_c(q) = 1$ means combination c is able to solve all the instances within a factor q of the best possible ratio. In terms of quality of the solution we can see in Figure 4.2 that the best choice when we use primal simplex is meth1; meth4 and meth5 for dual simplex; and clearly meth1 in the case of using barrier. It is important to highlight that in all cases, the best option is to use the stabilized Benders decomposition. In terms of total CPU time we can see in Figure 4.3 that the fastest variant for simplex primal is meth1 using stabilization and meth4 using classic Benders with simplex dual. Finally, the best option when we use the barrier solver is meth1 using stabilized Benders decomposition.

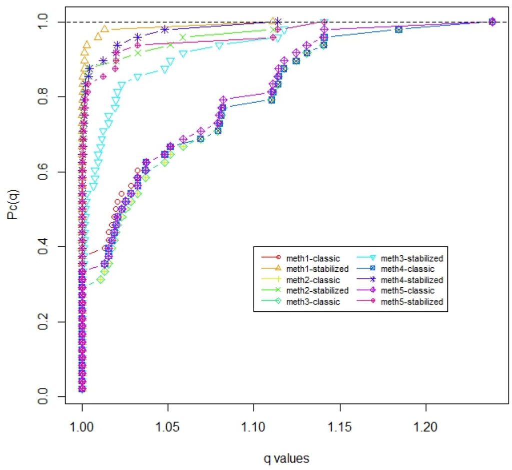
Selecting only the best combinations according to the performance profiles carried out previously (Figures 4.2 and 4.3), we do a last performance profile with the aim of finding the most effective combination. In our particular case, as we can see in Figure 4.4, the best combination is the stabilized Benders decomposition with meth1 using barrier. Clearly, this combination is the fastest and it provides the highest quality for more than 90% of the instances.

Table 4.3 reports a comparison between stabilized Benders meth1 using the barrier solver (`meth1-stabilized-barrier`) and the use of the state-of-the-art classical Benders developed in [33]. Indeed, in the approach of [33] classical Benders cuts were embedded in a branch-and-cut tree. Table shows the gap (gap_A) and total CPU time (TT_A) for stabilized strategy and gap (gap_B) and total CPU time (TT_B) for the state-of-the-art classical Benders. Moreover, the columns ($\Delta(gap_A, gap_B)$ and ($\Delta(TT_A, TT_B)$ show the difference gap and CPU time, respectively, between both methods. A time limit of one hour was considered for these runs.

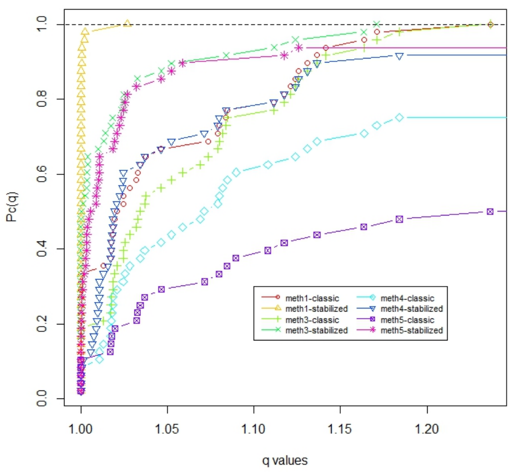
From Table 4.3 it can be concluded that the stabilized Benders (meth1) using the barrier solver is more efficient than the state-of-the-art classical Benders developed in [33]. Notice that the average gap for stabilized Benders CSP is 0,87% whereas for state-of-the-art classical Benders is 2,51% within the same CPU time limit. We have to emphasize that stabilized strategy is 1.8 times faster than the state-of-the-art classical Benders option. In nine of 48 instances (marked with †) the state-of-the-art classical Benders did not find a feasible solution within the time limit (3600 seconds). Only in four instances the state-of-the-art classical Benders outperformed our stabilized Benders algorithm. In the remaining 92%



(a) Primal simplex

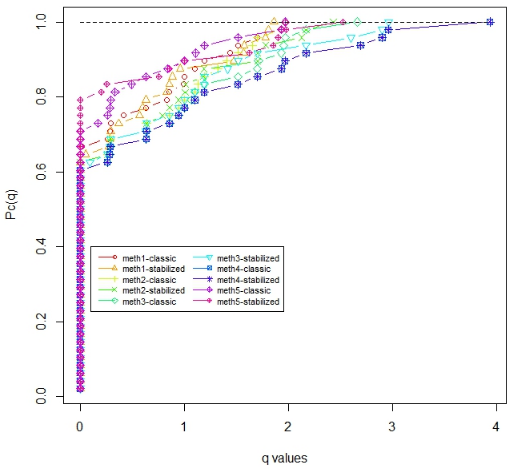


(b) Dual simplex

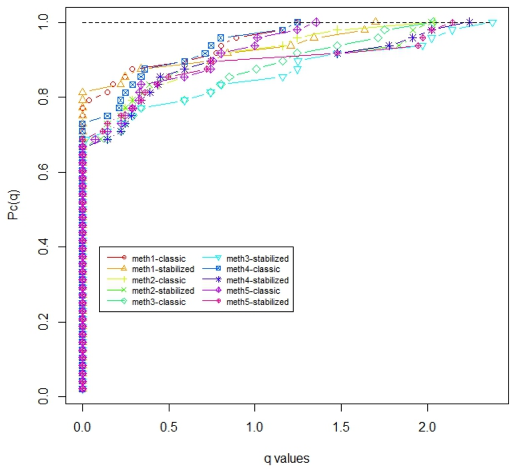


(c) Barrier

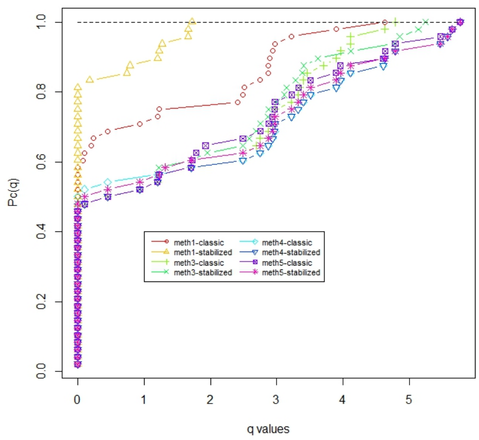
Figure 4.2: Performance profiles for the different combinations based on upper bound



(a) Primal simplex

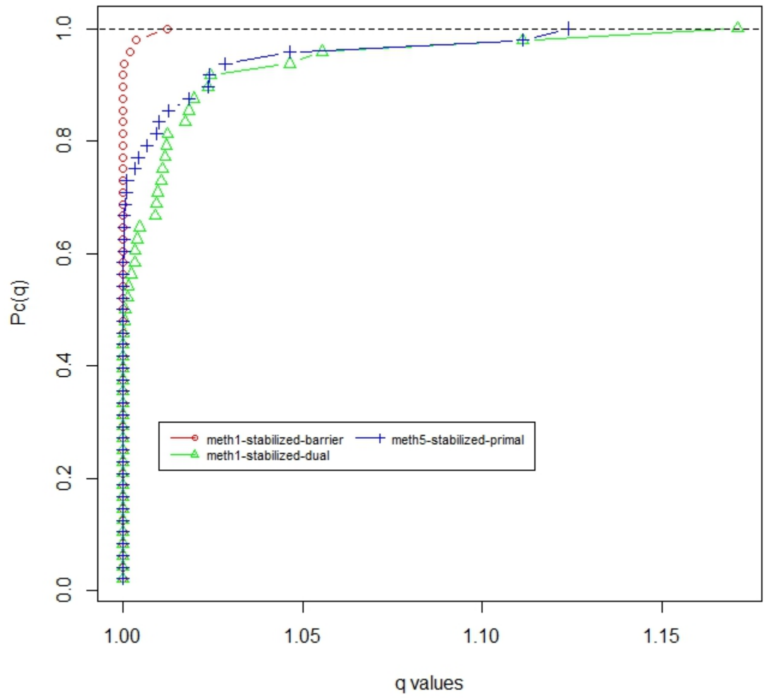


(b) Dual simplex

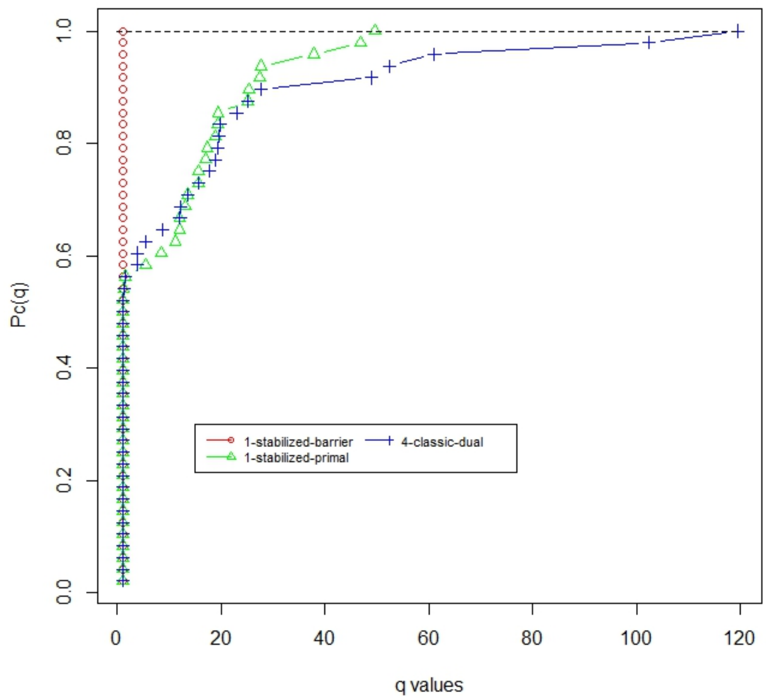


(c) Barrier

Figure 4.3: Performance profiles for the different combinations based on CPU time



(a) PF based on upper bound



(b) PF based on CPU time

Figure 4.4: Performance profiles for the most effective combinations based on upper bound and CPU time

	meth1-stabilized barrier (A)		State-of-the-art classical Benders CSP (B)		Comparison A-B	
Instance	gap_A	TT_A	gap_B	TT_B	$\Delta(gap_A, gap_B)$	$\Delta(TT_A, TT_B)$
40_50_10_5	0,00%	190,96	4,4%	3707,09	-4,44%	3516,13
40_50_5_5	1,42%	3501,72	1,7%	3742,58	-0,25%	240,86
40_60_10_5	0,01%	3600,01	0,4%	3704,29	-0,39%	104,28
40_60_5_5	0,83%	3600,05	6,3%	3716,31	-5,48%	116,26
40_70_10_5	0,00%	2962,3	0,8%	3654,57	-0,75%	692,27
40_70_5_5	2,08%	3600,12	1,7%	3673,18	0,36%	73,06
40_80_10_5	0,01%	3232,28	4,5%	3635,94	-4,53%	403,66
40_80_5_5	1,17%	3600,94	4,4%	4245,02	-3,26%	644,08
50_50_10_5	0,00%	3517,52	1,2%	3692,48	-1,21%	174,96
50_50_5_5	3,90%	3604,37	2,7%	3805,14	1,22%	200,77
50_60_10_5	0,89%	3605,38	2,6%	3651,88	-1,66%	46,5
50_60_5_5	0,95%	3606,32	5,9%	3673,98	-4,94%	67,66
50_70_10_5	0,00%	184,29	1,2%	3633,31	-1,18%	3449,02
50_70_5_5	5,98%	3600,06	†	†	†	†
50_80_10_5	0,01%	143,58	†	†	†	†
50_80_5_5	1,39%	3600,2	1,2%	3667,05	0,23%	66,85
60_50_5_5	2,64%	3605,02	10,9%	3675,43	-8,27%	70,41
60_60_5_5	0,23%	3602,64	1,4%	3661,03	-1,18%	58,39
60_70_5_5	3,01%	3600,03	7,0%	3643,12	-4,04%	43,09
60_80_5_5	0,17%	3600,04	1,0%	3667,78	-0,84%	67,74
70_50_5_5	6,74%	3606,18	15,6%	3690,57	-8,87%	84,39
70_60_5_5	2,83%	3601,47	5,2%	3665,87	-2,35%	64,4
70_70_5_5	0,18%	3603,38	0,7%	3753,13	-0,57%	149,75
70_80_5_5	2,36%	3605,29	†	†	†	†
40_50_15_5	0,00%	36,06	0,1%	3714,27	-0,14%	3678,21
40_60_15_5	0,00%	129,86	0,4%	3644,78	-0,40%	3514,92
40_70_15_5	0,00%	85,43	0,1%	3656,03	-0,12%	3570,6
40_80_15_5	0,00%	60,37	0,1%	3627,67	-0,10%	3567,3
50_50_15_5	0,01%	298,78	0,8%	3651,9	-0,76%	3353,12
50_60_15_5	0,00%	68,75	1,8%	3635,97	-1,78%	3567,22
50_70_15_5	0,01%	651,86	0,6%	3643,28	-0,60%	2991,42
50_80_15_5	0,01%	49,15	0,1%	3630,19	-0,10%	3581,04
60_50_10_5	0,00%	2282,89	3,1%	3646,38	-3,05%	1363,49
60_50_15_5	0,01%	185,2	0,3%	3664,49	-0,28%	3479,29
60_60_10_5	1,45%	3600,02	3,9%	3644,95	-2,47%	44,93
60_60_15_5	0,00%	203,57	0,1%	3636,57	-0,08%	3433
60_70_10_5	0,24%	3608,02	1,2%	3647,35	-0,92%	39,33
60_70_15_5	0,00%	30,14	0,0%	3608,37	-0,05%	3578,23
60_80_10_5	0,01%	230,2	†	†	†	†
60_80_15_5	0,01%	265,29	†	†	†	†
70_50_10_5	0,59%	3607,47	1,6%	3661,02	-1,03%	53,55
70_50_15_5	0,00%	35,14	0,2%	3634,4	-0,22%	3599,26
70_60_10_5	2,46%	3604,34	0,8%	3654,57	1,67%	50,23
70_60_15_5	0,00%	129,71	1,9%	3640,4	-1,86%	3510,69
70_70_10_5	0,12%	3609,82	†	†	†	†
70_70_15_5	0,01%	67,27	†	†	†	†
70_80_10_5	0,00%	73,42	†	†	†	†
70_80_15_5	0,01%	58,93	†	†	†	†

† Time limit reached without finding a feasible solution

Table 4.3: Comparison between stabilized Benders meth1, using the barrier solver (`meth1-stabilized-barrier`) and the use of the state-of-the-art classical Benders software developed in [33], for random 1H2D instances.

	meth1 stabilized) barrier (A)	state-of-the-art classical Benders CSP(B)	Comparison A-B
Instance	gap_A	gap_B	$\Delta(gap_A, gap_B)$
hier16	99,17%	99,13%	0,04%
hier16x16x16a	99,10%	99,09%	0,00%
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%
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%

† Time limit reached without finding a feasible solution

Table 4.4: Comparison between stabilized Benders meth1, using the barrier solver (`meth1-stabilized-barrier`) and the use of the state-of-the-art classical Benders developed in [33] for a subset of real tables.

of instances, stabilized method outperformed state-of-the-art classical Benders. The average gain of gap is about 1,95% so it is clearly seen that for synthetic 1H2D instances the stabilized Benders is a competitive approach to find good feasible solutions.

For the real-world general instances of Table 4.4 the situation is slightly different. These instances are not guaranteed to have a hierarchical structure, and this may explain why stabilized Benders decomposition is not as competitive as for synthetic 1H2D tables. However, we have six instances (marked in bold) where stabilized Benders improved the upper bound with an average gap of 2.61%. On the other hand, in other six tables the state-of-the-art classical Benders CSP outperformed stabilized Benders with an average gap of 4.57%. In two instances, the state-of-the-art classical Benders did not find any feasible solution within the time limit of 3600 seconds.

Finally, we have tried the Benders algorithm built in CPLEX 12.7 (CPLEX-

	meth1-stabilized barrier		CPLEX-Benders strategy 1		CPLEX-Benders strategy 0	
	CPU time	gap	CPU time	gap	CPU time	gap
20_25_15_5_1	51,44	0,01%	3596.43	0,09%	3596,5	0,48%
20_30_15_5_1	2297,58	0,01%	3596.50	1,05%	3596,44	1,32%
20_35_15_5_1	163,14	0,01%	3596.33	1,39%	3596,39	0,77%
25_25_15_5_1	3600,11	0,03%	3596.41	6,22%	3596,32	0,35%
25_30_15_5_1	54,87	0,01%	3596.56	93,51%	3596,5	5,68%
25_35_15_5_1	43,61	0,00%	3596.47	†	3596,55	94,98%
30_25_15_5_1	1128,73	0,01%	3596.25	0,34%	3596,38	0,33%
30_30_15_5_1	99,76	0,01%	‡	‡	3596,56	94,91%
30_35_15_5_1	568,42	0,01%	‡	‡	‡	‡

† no feasible solution was found within the time limit of 3600 seconds;
‡ internal memory error provided by AMPL;

Table 4.5: Comparison between stabilized Benders meth1, using the barrier solver (`meth1-stabilized-barrier`) and and CPLEX-Benders for a set of small 1H2D instances.

Benders) using a set of small 1H2D instances. For this test CPLEX was interfaced through AMPL, and we only considered the CPLEX solution time, not the model generation time. Table 4.5 reports a comparison with stabilized Benders meth1 using the barrier solver (`meth1-stabilized-barrier`). The decision on the distribution of the continuous variables in the different Benders subproblems can be determined automatically by CPLEX (strategy 0, where all continuous variables are in a single Benders subproblem) or for ourselves (strategy 1, where the continuous variables go to different Benders subproblems). As the results show, it is not competitive with the stabilized Benders. CPLEX-Benders always exhausted the time limit (3600 seconds) and even failed in several instances. It is worth noting that some instances were solved by stabilized Benders with a 0.00% gap in 43 seconds, whereas CPLEX-Benders only provided a 94.98% gap solution in 3600 seconds.

Chapter 5

Conclusions and future directions

Along the preceding chapters we have presented different methods of Operations Research in order to find optimal or suboptimal good solutions of mixed integer linear problems in a reasonable computational time, where even finding a feasible solution may be a challenging task for large instances. All methods have been coded by ourselves in C++, using commercial state-of-the-art solvers, and they were applied to real-world problems from the privacy in statistical databases field. In particular we focused on statistical tabular data protection. However, they can be applied to other real-world problems. Now is the time to summarize the main conclusions and comment future research directions. We highlight the following conclusions:

5.1 Conclusions

- This thesis contributes to improve two important fields of mixed integer optimization: (1) heuristic methods to find good initial feasible solutions in a short computational time, although not optimal. (2) a successful contribution to the exact Benders decomposition method.

- The first contribution (AC-FP) suggests an extension of the successful *feasibility pump* heuristic (FP), applied to general mixed integer linear problems, where candidate points to be rounded are found in a segment of feasible points, one of the extremes being the analytic center. The objective FP is a particular case where the endpoint associated to the solution of the relaxed problem is selected as the point to be rounded. AC-FP is also compared with the recent analytic center feasibility method (ACFM), which also uses the analytic center for obtaining MILP feasible solutions. Computational results show that AC-FP may outperform FP and ACFM in some MILP instances, either in solution time or quality of the solution. The three approaches (FP, ACFM and AC-FP) have their own benefits and disadvantages. FP is likely the fastest approach, and in general it provides good (if not the best) solutions in most instances; however it does not exploit the concept of analytic center, which may be beneficial in some instances. ACFM seems to provide better points, but it is computationally expensive and it was only possible to test on small instances. AC-FP is not computationally as expensive as ACFM (it only needs to compute one analytic center), and in some MILP instances outperforms FP (either in time or quality of the solution); however, for binary problems AC-FP seems not to be competitive against FP (the analytic center seems not to be helpful when we optimize within the unit cube).

- The second contribution is the application of the fix-and-relax heuristic (FR) to the statistical tabular data protection method named *Controlled tabular adjustment* (CTA). FR, either alone or in combination with other heuristics such as BCD, has shown to be an efficient approach. FR has been particularly successful applied to a class of hierarchical tables named 1H2D, being competitive against plain CPLEX branch-and-cut (BC), feasibility pump heuristic (FP) or RINS heuristics. For general real-world tables, FR and FR+BCD outperformed BC in 73% of the instances tested. Promising results were also obtained in a reduced set of instances by warm starting BC with the FR solution.

- Stabilized Benders applied to CSP was shown to be an excellent strategy compared to the state-of-the-art classical Benders of [30]. In 92% of the synthetic 1H2D tables, stabilized Benders outperformed classical Benders in terms of both CPU time and gap of the feasible solution found. With stabilized strategy, the average GAP was 0.87% whereas for commercial CSP Benders was 2.51%. Moreover, the stabilized Benders was 1.8 faster than classical Benders. For real-world general tables the stabilized approach was not as competitive as for the 1H2D case, probably due to the absence of a hierarchical structure. However, it is worth noting that stabilized Benders can be a promising approach because, when applied to real-world tables, the average gap was lower (2.61%) than for classical Benders (4.57%).

5.2 Future directions

We can mention the following points as future works:

- Combining FR with other heuristics, or embedding FR in exact approaches, like Benders reformulation, is part of the further work to be done in this field.

- All approaches in the thesis dealt with post-tabular data protection, i.e., the protection methods are applied to the tables once they have been created. It would be interesting to study if similar ideas to the ones developed in the thesis are valid for: (1) pre-tabular data protection methods (which focus on modifying the microdata files, and then using this modified microfiles to create protected the tables); (2) microdata protection methods that solve some sort of optimization problem (i.e., microaggregation).

- The application of classical and stabilized Benders for optimal CTA. Preliminary works [14] for small-medium two-dimensional tables show that it can be a promising approach for more complex tables.

- A different line of research would be to apply the tools developed in this thesis (in particular AC-FP and stabilized Benders) to problems from other fields (e.g., logistics, production planning, etc).

- The heuristic AC-FP could also be used with the recent rounding scheme based on constraint propagation suggested in [34].

5.3 Our contributions

The following publications in peer-reviewed journals, scientific conferences and research reports have been the base of this work and had resulted from this thesis.

- Publications:

- D. Baena, J. Castro, Using the analytic center in the feasibility pump, Operations Research Letters, 39 (2011) 310-317. Corresponding to Chapter 2.
- D. Baena, J. Castro, J. A. González, Fix-and-relax approaches for controlled tabular adjustment, Computers & Operations Research, 58 (2015) 41-52. Corresponding to Chapter 3.
- D.Baena, J.Castro, A. Frangioni, Stabilized Benders methods for large combinatorial optimization problems: applications to cell suppression, working paper to be submitted. Corresponding to chapter 4.

• Scientific conferences:

- D. Baena, J. Castro, J.A. González, Fix-and-relax approaches for controlled tabular adjustment, XXXV Congreso Nacional de Estadística e Investigación Operativa, Pamplona, Spain, May 2015.
- D. Baena, J. Castro, A fix and relax heuristic for controlled tabular adjustment, 25th European Conference on Operational Research-EURO 2012, Vilnius University, Vilnius (Lithuania), July 2012. **Invited presentation.**
- D. Baena, J. Castro, The analytic center feasibility pump, XXXIII Congreso Nacional de Estadística e Investigación Operativa, Madrid, Spain, April 2012.

Bibliography

[1] ILOG CPLEX 12.1. *User's Manual*. IBM, 2010.

[2] T. Achterberg and T. Berthold. Improving the feasibility pump. *Mathematical Programming*, 4:77–86, 2007.

[3] H. Ben Amor, J. Desrosiers, and A. Frangioni. On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics*, 157(6): 1167–1184, 2009.

[4] D. Baena and J. Castro. Using the analytic center in the feasibility pump. *Operations Research Letters*, 39:310–317, 2011.

[5] J.F Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[6] H.Y. Benson. Mixed integer nonlinear programming using interior-point methods. *Optimization Methods and Software*, 26:6:911–931, 2011.

[7] L. Bertacco, M. Fischetti, and A. Lodi. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization*, 4:63–76, 2007.

[8] N.L. Boland, A.C. Eberhard, F.G. Engineer, M. Fischetti, M.W.P. Savelsbergh, and A. Tsoukalas. Boosting the feasibility pump. *Mathematical Programming Computation*, 6:255–279, 2014.

[9] P. Bonami, L.T. Biegler, A.R. Conn, G. Cornuejols, I.E. Grossman, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A.Wachter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5:186–204, 2008.

[10] 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.

[11] J. Castro. Minimum-distance controlled perturbation methods for large-scale tabular data protection. *European Journal of Operational Research*, 171:39–52, 2006.

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

[13] J. Castro. On assessing the disclosure risk of controlled adjustment methods for statistical tabular data. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 20:921–941, 2012.

[14] J. Castro and D. Baena. Using a mathematical programming modeling language for optimal CTA. *Lecture Notes in Computer Science*, 5262:1–12, 2008.

[15] J. Castro and S. Giessing. Testing variants of minimum distance controlled tabular adjustment. *In: Monographs of Official Statistics, Eurostat-Office for Official Publications of the European Communities, Luxembourg*, 92-79-01108-1:333–343, 2006.

[16] J. Castro and J.A. González. Assessing the information loss of controlled tabular adjustment in two-way tables. *Lecture Notes in Computer Science*, 8744:11–23, 2014.

[17] J. Castro, J.A. González, and D. Baena. User's and programmer's manual of the RCTA package. *Technical Report DR 2009/01, Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya*, 2009.

[18] J. Castro, A. Frangioni, and C. Gentile. Perspective reformulations of the CTA problem with l_2 distances. *Operations Research*, 62(4):891–909, 2014.

[19] L.H. Cox and J.A. George. Controlled rounding for tables with subtotals. *Annals of Operations Research*, 20:141–157, 1989.

[20] J. Czyzyk, S. Mehrotra, M. Wagner, and S.J. Wright. PCx: an interior-point code for linear programming. *Optimization Methods and Software*, 11: 397–430, 1999.

[21] R.A. Dandekar and L.H. Cox. Synthetic tabular data: an alternative to complementary cell suppression. *Manuscript, Energy Information Administration, U.S. Department of Energy*, 2002.

[22] E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102: 71–90, 2005.

[23] C. Dillenberger, L.F. Escudero, A. Wollensak, and W. Zhang. On practical resource allocation for production planning and scheduling with period overlapping setups. *European Journal of Operational Research*, 75:275–286, 1994.

[24] J. Domingo-Ferrer and L. Franconi. *Lecture Notes in Computer Science. Privacy in Statistical Databases. Privacy in Statistical Databases*, 4302, 2006, Springer, Berlin.

[25] J. Domingo-Ferrer and Y. Saigin. *Lecture Notes in Computer Science*, 5262, 2008, Springer, Berlin.

[26] J. Domingo-Ferrer and J. Soria-Comas. Anonymization in the time of big data. *Lecture Notes in Computes Science*, 9867:57–68, 2016.

[27] J. Domingo-Ferrer and V. Torra. A critique of the sensitivity rules usually employed for statistical table protection. *Int. J. of Unc.,Fuzziness and Knowledge Based Systems*, 10:545–556, 2002.

[28] J. Domingo-Ferrer, J. Mateo-Sanz, and V. Torra. Comparing SDC methods for microdata on the basis of information loss and disclosure risk. *Proceedings of ETK-NTTS, Eurostat*, pages 807–826, 2001.

[29] G. Duncan, S. Keller-McNulty, and S.Stokes. Disclosure risk vs. data utility: the R-U confidentiality map. *Technical Report, Statistical Sciences Group, Los Alamos National Laboratory*, 2001.

[30] L.F. Escudero and J. Salmerón. On a fix-and-relax framework for a class of project scheduling problems. *Annals of Operations Research*, 140:163–188, 2005.

[31] D. Ferreira, R. Morabito, and S. Rangel. Relax and fix heuristics to solve one-stage one-machine lot-scheduling models for small-scale soft drink plants. *Computers & Operations Research*, 37:684–691, 2010.

[32] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98: 23–47, 2003.

[33] M. Fischetti and J.J. Salazar-González. Solving the cell suppression problem on tabular data with linear constraints. *Journal of the American Statistical Association*, 95:916–928, 2000.

[34] M. Fischetti and D. Salvagnin. Feasibility pump 2.0. *Mathematical Programming*, 1:201–222, 2009.

[35] M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Mathematical Programming*, 104:91–104, 2005.

[36] M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Mathematical Programming*, 104:91–104, 2005.

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

[38] A. Frangioni and B.Gendron. A stabilized structured dantzig-wolfe decomposition method. *Mathematical Programming*, 140:45–76, 2013.

[39] S. Giessing. Pre-tabular perturbation with controlled tabular adjustment: some considerations. *Lecture Notes in Computer Science*, 8744:48–61, 2014.

[40] S. Giessing and J. Höhne. Eliminating small cells from census counts tables: some considerations on transition probabilities. *Lecture Notes in Computer Science*, 6344:52–65, 2010.

[41] F. Glover, L.H. Cox, R. Patil, and J.P. Kelly. Integrated exact, hybrid and metaheuristic learning methods for confidentiality protection. *Annals of Operations Research*, 183:47–73, 2011.

[42] GNU. *Linear Programming Kit v. 4.43.* Reference Manual, 1997.

[43] J.A. González and J. Castro. A heuristic block coordinate descent approach for controlled tabular adjustment. *Computers & Operations Research*, 38: 1826–1835, 2011.

[44] D. Gray. Precision threshold and noise: An alternative framework of sensitivity measures. *Lecture Notes in Computes Science*, 9867:15–27, 2016.

[45] M.S. Hernández and J.J. Salazar-González. Enhanced controlled tabular adjustment. *Computers & Operations Research*, 43:61–67, 2014.

[46] J.B. Hiriart-Urruty and C. Lemaréchal. Convex analysis and minimization algorithms ii. *Springer-Verlag*, pages 45–76, 1996.

[47] A. Hundepool. The Argus software in CENEX. *Lecture Notes in Computer Science*, 4302:334–346, 2006.

[48] A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, R. Lenz, J. Naylor, E.S Nordholt, G. Seri, and P.P. De Wolf. Handook on statistical disclosure control. *Network of Excellence in the European Statistical System in the field of Statistical Disclosure Control. Available on-line at http://neon.vb.cbs.nl/casc/SDC_Handbook.pdf*, 1.2, 2010.

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

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

[51] T. Koch et al. Miplib 2010. mixed integer programming library version 5. *Mathematical Programming Computation*, 3:103–163, 2011.

[52] J.E. Mitchell. Fixing variables and generating classical cutting planes when using an interior point branch and cut method to solve integer programming problems. *European Journal of Operational Research*, 97:139–148, 1997.

[53] J.E. Mitchell and M.J. Todd. Solving combinatorial optimization problems using Karmarkar's algorithm. *Mathematical Programming*, 56:245–284, 1992.

[54] H. Mittelmann. Decision tree for optimization software. *http://plato.asu.edu/guide.html*, 2014.

[55] J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[56] J. Naoum-Sawaya and S. Elhedhli. An interior point cutting plane heuristic for mixed integer programming. *Computers & Operations Research*, 38 (9): 1335–1341, 2011.

[57] D.A Robertson and R. Either. Cell suppression: experience and theory. *Lecture Notes in Computes Science*, 2316:8–20, 2002.

[58] J.J. Salazar-González. Controlled rounding and cell perturbation: statistical disclosure limitation methods for tabular data. *Mathematical Programming*, 105:583–603, 2006.

[59] 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(1):96–115, 2005.

[60] K. Soininvaara, T. Oinonen, and A. Nissinen. Balancing confidentiality and usability: protecting sensitive data in the case of inward foreign affiliates statistics (FATS). *Lecture Notes in Computer Science*, 8744:338–349, 2014.

[61] 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.

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

[63] Y. Ye. *Interior Point Algorithms. Theory and Analysis.* Wiley, 1997.

[64] L. Zayatz. Work session on statistical data confidentiality. *U.S. Census Bureau, communication at Joint UNECE/Eurostat*, 2009, Bilbao (Basque Country, Spain).