CHALMERS
UNIVERSITY OF TECHNOLOGY

Comparison and Analysis of Three Approaches to Cell Suppression in Statistical Tables

Master's thesis in Engineering Mathematics and Computational Science

JOHAN KARLSSON
ALFRED OLSSON

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Master's thesis

Comparison and Analysis of Three Approaches to Cell Suppression in Statistical Tables

Johan Karlsson Alfred Olsson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2015

Comparison and Analysis of Three Approaches to Cell Suppression in Statistical Tables
Johan Karlsson
Alfred Olsson

Supervisors: Lars-Erik Almberg, Statistics Sweden, Örebro
Karin Kraft, Statistics Sweden, Örebro
Examiner: Ann-Brith Strömberg, Department of Mathematical Sciences,
Chalmers University of Technology and University of Gothenburg

Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2015

Comparison and Analysis of Three Approaches to Cell Suppression in Statistical Tables

Johan Karlsson
Alfred Olsson
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Statistical disclosure control aims to ensure that no sensitive information can be acquired from published statistical data. One method to achieve this for statistical tables is cell suppression, in which cells that are regarded as sensitive are suppressed, i.e., removed from the table. Tables generally include marginal totals from which the missing values can be computed, wherefore additional, non-sensitive, cells must be suppressed. The selection of a set of suppressions such that no sensitive information can be derived, while the loss of information is minimized, is modelled as an optimization problem, called the cell suppression problem. For tables with complex structures an optimization solver is required to solve this problem. In use at the statistical agency Statistics Sweden is the commercial solver Xpress. This thesis was motivated by the question of whether an open source solver is a viable alternative.

The traditional approach, used at Statistics Sweden, is so-called complete cell suppression, in which sensitive values are removed completely from the table. This results in an integer linear optimization problem. Two other approaches, partial cell suppression and combined cell suppression, in which suppressed cell values are replaced by intervals covering the actual values, have been suggested. These approaches, resulting in linear and mixed integer linear optimization problems, respectively, are largely untested in the literature. We have studied the three approaches theoretically and compared them computationally for test instances of varying size, using the commercial solver CPLEX and the open source solver GLPK.

Our test results show that GLPK performs well on small instances when using complete cell suppression. For partial cell suppression, GLPK managed to solve only a subset of the small instances. For combined cell suppression, none of the test instances were solved by GLPK. For complete and partial cell suppression, CPLEX found optimal solutions to all instances tested, but for combined cell suppression solutions were obtained only for the smaller instances. Our results also indicate that combined cell suppression yields solutions with the desirable qualities of both the other approaches, but at the cost of large computation times. We conclude that for large problem instances, a commercial solver is required for solving the cell suppression problem in a reasonable time.

Keywords: cell suppression, statistical disclosure control, open source, integer programming, linear optimization, mixed integer linear programming

Acknowledgements

We would like to thank our supervisors at SCB, Lars-Erik Almberg and Karin Kraft, for the opportunity to perform this thesis work. We are especially grateful for the swift process from when the idea was first conceived to when the work was initiated. We also extend our thanks to our examiner Ann-Brith Strömberg for her mathematical guidance.

Johan Karlsson and Alfred Olsson
Gothenburg, October 2015

Contents

Chapter 1

Introduction

One of the main assignments for national statistical institutes is to gather and publish data that can be used for decision-making, debate and research. One problem that might appear when doing this is that part of the collected data contains sensitive information. It is very important for statistical institutes not to reveal this sensitive information both since there are laws regulating that data on individual respondents should not be derivable from published data and to ensure that respondents participating in surveys can trust that their information is safe. That participants feel safe is of course crucial in order to be able to ensure high quality of future surveys.

Statistical disclosure control seeks to prevent unauthorized people from gaining sensitive knowledge from published statistical data. This methodology is not only used by statistical institutes but is also applied to fields such as health information and e-commerce, where it is important to protect information.

Data is often published in tables and several methods have been developed to prevent people from gathering sensitive information from such tables (see [12]). One such method is *cell suppression*, in which cell values that are considered sensitive according to some criterion are removed from the table. However, after all sensitive cells have been removed it might still be possible to calculate these values from marginal totals. Therefore, the removal of additional (non-sensitive) cells is necessary. The goal of the *cell suppression problem* (CSP) is to determine which cells should be suppressed in order to make the table safe while keeping the information loss to a minimum.

1.1 Background

The national statistical agency Statistics Sweden (Swedish: Statistiska centralbyrån, SCB) uses the software τ-ARGUS (see [23]), developed and maintained by Statistics Netherlands, for disclosure control of statistical tables. In τ-ARGUS the user can tabulate the gathered microdata, choose a sensitivity rule for deciding which cells are considered sensitive and then apply a method for producing a safe table. The method used by SCB for magnitude tables is complete cell suppression, which is one of several possible approaches to cell suppression.

As a large national agency, the tables SCB work with often exhibit complex structures with hierarchies and multiple dimensions. In order to find the optimal set of suppressions for such instances τ-ARGUS requires a linear programming solver. In use at SCB is the

optimization solver Xpress (see [8]) which is a commercial software and as such has a yearly license fee. Naturally, it is of interest to SCB whether it is possible to cut down on the cost of optimization software by using an open source solver instead.

Other approaches to cell suppression, called partial cell suppression and combined cell suppression, have been proposed in [10]. In that article, preliminary results on a slightly modified model for partial cell suppression suggested that this approach could be both computationally efficient and reduce the information loss necessary to protect the sensitive data. The mathematical model for combined cell suppression has only been outlined theoretically and not thoroughly tested.

1.2 Purpose

The purpose of this thesis is twofold. The primary question prompting the realization of this project was to investigate whether open source software is a viable alternative to Xpress for solving the CSP. Secondly, we shall compare three different variations of cell suppression—complete, partial and the combination thereof—and explore how they relate to each other from a theoretical viewpoint.

The three cell suppression problems will be compared computationally and test results will be presented for the commercial solver CPLEX (see [13]) and the open source solver GLPK (see [11]).

1.3 Delimitations

This thesis is concerned only with magnitude tables with continuous response variables, as opposed to tables with integer data. These concepts are explained in the introduction to Chapter 2. Also, the mathematical models of the CSP that are formulated aim to protect tables only from external attackers and not from internal attackers, terms that will be made precise in Section 2.1. These delimitations are discussed more in depth in Section 3.1.1. Finally, the reason CPLEX is used instead of Xpress is that the performance of these two solvers is comparable (see [19]), and that CPLEX is immediately available for usage at Chalmers University of Technology, whereas Xpress is not.

1.4 Thesis outline

This thesis is divided into six chapters. Chapter 1, contains the introduction, motivation, and aim of the work. In the following two chapters, theory is prestented. Chapter 2 is concered with the theory of primary suppressions, focusing on discussing what data should be sensitive and analyzing the most common sensitivity rules. The second part of the theory, Chapter 3, deals with secondary suppressions and what should be expected from a safe table. The complete, partial and combined CSP are presented in Sections 3.2, 3.3, and 3.4, respectively. In Chapter 4 the algorithms for solving these models are presented and in Chapter 5 the algorithms' performance on seven different problem instances is evaluated and compared. Conclusions are drawn in the sixth and final chapter of the thesis.

Chapter 2

Primary suppressions

A table consists of a number of spanning variables and one response variable. The spanning variables contain information about the traits of the respondents, for instance age group, geographic location, or business area, while the response variable is the trait the table displays, e.g, income, turnover, or number of empolyees. Keeping this vague definition in mind, there exist essentially three different types of tables: *non-hierarchical*, *hierarchical*, and *linked*. Simple examples of the three types are illustrated in Figures 2.1–2.3. A non-hierarchical table is self-explanatory and is usually considered as an "ordinary table". A hierarchical table is similar, with the difference that at least one of the spanning variables contains different levels. For the table in Figure 2.2 the hierarchy is present in that the column spanning variable can be divided into subcategories: Aa and Ab for A, and Ba, Bb, and Bc for B. A linked table is simply two or more tables (hierarchical, non-hierarcichal and of any dimension) that have cells in common. For an example consider Figure 2.3 and note that the totals of A and B are present in both tables but then divided into two different row spanning variables. This creates "links" between the two tables and when considered as one table it is referred to as a linked table. In this context the tables that make up the linked table are called *subtables*.

If each cell in a table contains the number of respondents within a certain category the table is called a *frequency* or *counting table*, and if each cell contains the sum of a particular quantity for each respondent belonging to that cell the table is called a *magnitude table*.

The three types of tables in Figures 2.1–2.3 can be represented by one mathematical entity. We say that a *table* is a vector $a = [a_1 \dots a_N]^T$, where $a_i \in \mathbb{R}$ for all i, that satisfies the constraints

$$\begin{cases} Ma = b, \\ l_i \leq a_i \leq u_i, \quad i = 1, \dots, N, \end{cases}$$

	1	2	Total
A	250	100	350
B	300	200	500
Total	550	300	850

Figure 2.1: A 2D non-hierarchical table.

	1	2	Total
A	250	100	350
Aa	100	50	150
Ab	150	50	200
B	300	200	500
Ba	115	60	175
Bb	175	125	300
Bc	10	15	25
Total	550	300	850

Figure 2.2: A 2D hierarchical table with one hierarchical spanning variable.

	1	2	Total
A	250	100	350
Aa	100	50	150
Ab	150	50	200
B	300	200	500
Ba	115	60	175
Bb	175	125	300
Bc	10	15	25
Total	550	300	850

	I	II	Total
A	150	200	350
B	55	445	500
Total	205	645	850

Figure 2.3: A linked table with one hierarchical and one non-hierarchical subtable.

where $M \in \mathbb{R}^{m \times N}$ and $b \in \mathbb{R}^m$ represent linear relations between cell values and $l_i \in \mathbb{R}$ and $u_i \in \mathbb{R}$, the lower and upper *external bounds*, define intervals in which the table entries a_i are known to lie. Typically, tables contain cells whose values are the sums of other values in the same table, in which case $M \in \{0, \pm 1\}^{m \times N}$ and $b = \mathbf{0}$.

Example 2.1. For the table in Figure 2.1, $Ma = b$ is equivalent to the equations

$$\begin{aligned} a_{11} + a_{12} - a_{13} &= 0, \\ a_{21} + a_{22} - a_{23} &= 0, \\ a_{31} + a_{32} - a_{33} &= 0, \\ a_{11} + a_{21} - a_{31} &= 0, \\ a_{12} + a_{22} - a_{32} &= 0, \\ a_{13} + a_{23} - a_{33} &= 0. \end{aligned}$$

⧫

For hierarchical or linked tables, usually a larger number of equations need to be fulfilled. The equations for the table in Figure 2.3 are given in Example 2.2, with a_{ij} denoting the values in the left subtable and $\widetilde{a}_{ij}$ denoting the values in the right subtable.

Example 2.2. For the table in Figure 2.3, $Ma = b$ is equivalent to the equations

Row equations for left subtable

$$a_{11} + a_{12} - a_{13} = 0,$$
$$a_{21} + a_{22} - a_{23} = 0,$$
$$a_{31} + a_{32} - a_{33} = 0,$$
$$a_{41} + a_{42} - a_{43} = 0,$$
$$a_{51} + a_{52} - a_{53} = 0,$$
$$a_{61} + a_{62} - a_{63} = 0,$$
$$a_{71} + a_{72} - a_{73} = 0,$$
$$a_{81} + a_{82} - a_{83} = 0,$$

Column equations for left subtable

$$a_{11} + a_{41} - a_{81} = 0,$$
$$a_{12} + a_{42} - a_{82} = 0,$$
$$a_{31} + a_{43} - a_{83} = 0,$$

Hierarchy equations for left subtable

$$a_{21} + a_{31} - a_{11} = 0,$$
$$a_{51} + a_{61} + a_{71} - a_{41} = 0,$$
$$a_{22} + a_{32} - a_{12} = 0,$$
$$a_{52} + a_{62} + a_{72} - a_{42} = 0,$$
$$a_{23} + a_{33} - a_{13} = 0,$$
$$a_{53} + a_{63} + a_{73} - a_{43} = 0,$$

Row equations for right subtable

$$\widetilde{a}_{11} + \widetilde{a}_{12} - \widetilde{a}_{13} = 0,$$
$$\widetilde{a}_{21} + \widetilde{a}_{22} - \widetilde{a}_{23} = 0,$$
$$\widetilde{a}_{31} + \widetilde{a}_{32} - \widetilde{a}_{33} = 0,$$

Column equations for right subtable

$$\widetilde{a}_{11} + \widetilde{a}_{21} - \widetilde{a}_{31} = 0,$$
$$\widetilde{a}_{12} + \widetilde{a}_{22} - \widetilde{a}_{32} = 0,$$
$$\widetilde{a}_{13} + \widetilde{a}_{23} - \widetilde{a}_{33} = 0,$$

Link equations

$$\widetilde{a}_{11} + \widetilde{a}_{12} - a_{13} = 0,$$
$$\widetilde{a}_{21} + \widetilde{a}_{22} - a_{43} = 0,$$
$$\widetilde{a}_{31} + \widetilde{a}_{32} - a_{83} = 0.$$

♦

Note that the equations to the left in Example 2.2 are also equivalent to $Ma = b$ for the table in Figure 2.2 since this table is identical to the left subtable of Figure 2.3.

2.1 Sensitivity rules

The first thing to be decided in the procedure of protecting a table is which cells should be considered sensitive, and therefore should not be published (if the table has no sensitive cells it can be published as-is and there is no CSP to solve). All sensitive cells must be suppressed, and these suppressions are called *primary suppressions*. Several articles, for instance [4] and [20], discuss different rules for the classification of sensitive cells, *sensitivity rules*, and how well these rules perform. This is a crucial part of cell suppression, since if the wrong cells are protected the whole procedure is meaningless. The purpose of the following subsections is to give an overview of the most common rules in the literature. First, some concepts need to be introduced.

Generally, a cell value in a table is the sum of one or more numbers. For instance, the table in Figure 2.4 gives the turnover of companies by type of business and geographic location. Clearly, there may be several companies in the same type of business and

	1	2	Total
A	120+80+40+10	55+45	350
B	280+15+5	99+99+2	500
Total	550	300	850

Figure 2.4: A 2D non-hierarchical table displaying turnover for companies by type of business (A and B) and location (1 and 2) where each term in the cells A1, A2, B1, and B2 is the turnover of a single company.

location. Each term in a sum that makes up a cell value is referred to as a *contribution* and the entity from which it originates is called a *contributor*. Note that the contributions to a marginal cell are all of the contributions to the table entries that sum up to the corresponding marginal value.

Example 2.3. In Figure 2.4, the value of cell A1 is the total turnover of companies in business type A in geographic location 1. There are four such companies (contributors) with turnover (contributions) 120, 80, 40, and 10, respectively. The marginal cell Total1 contains the contributions 280, 120, 80, 40, 15, 10 and 5, i.e., all the contributions of A1 and B1. ⧫

An entity that attempts to disclose sensitive information from a table is referred to as an *attacker*. If the attacker has no more knowledge than the published table, including its structure, and the external bounds, it is referred to as an *external attacker*. An attacker that, in addition to this knowledge, is also a contributor to at least one cell of the table is called an *internal attacker*.

The table in Figure 2.4 will be the main example used for examining the different sensitivity rules. We argue that a good rule should make sure that the cells A2, B1, and B2 in Figure 2.4 are sensitive, with the following motivations. If cell A2, with value 100, is published, then the company with turnover 45, assumed to be the attacker, will know that the other companies in the same business area and location have a total turnover of 55. If the attacker knows that there is only one other company that contributes to A2, then they know that company's turnover exactly. The converse of course goes for the company with turnover 55, assuming they have similar knowledge. For B1 and B2 the situation is similar but exact disclosure may not be possible. If the attacker is the second largest contributor to one of these cells, then this internal attacker can find a very good upper bound on the largest contributor to the same cell. This is possible since all contributions must be non-negative (turnover cannot be negative) and one contribution is known by the attacker. The value of cell B1 is 300, and the second largest contribution is 15, so the largest contribution can be at most 300−15=285. For B2, the largest contribution is at most 200−99=101. Further, it is reasonable to assume that this attacker has knowledge about other companies in the same industry. If the second largest contributor to B1 knows that there is one contributor to the cell that is significantly larger than all the others, they also know that the upper bound of 285 on the largest contribution is a good approximation of the real value, which is 280. Similarly for B2, a contributor to this cell may know that there are two contributions that are almost equal and that the rest are small.

Clearly this reasoning is dependent on the example at hand, where the response

variable is the turnover of companies and some amount of knowledge beyond the table can be assumed. Otherwise the attackers would merely have upper bounds on individual contributions and no way of knowing that they are in fact good approximations. However, cell suppression is very commonly used for magnitude tables with enterprise statistics, which validates the argumentation.

Precisely defining the meaning of sensitivity in the context of cell suppression is not simple but, with the reasoning above as a basis, what one generally wants to avoid is the possibility that someone can gain information about an individual respondent. We shall now study three different sensitivity rules, starting with the minimum frequency rule.

2.1.1 Minimum frequency rule

The *minimum frequency rule* is extremely simple. A cell is considered sensitive if it has fewer than a prespecified number of contributors (see [12, pp. 140–141]). If the minimum frequency is chosen to be 3 we get that the only sensitive cell in the table in Figure 2.4 is A2. Hence, the intuitive sensitivity of B1 and B2 is not captured by this rule. Of course, a frequency $f > 3$ would capture this sensitivity, but this will only work for specific examples since an example like B1, with one large contribution and $f - 1$ small contributions, can always be constructed. Therefore, we quickly abandon the minimum frequency rule and move on to the more interesting (n, k)-dominance rule.

2.1.2 (n, k)-dominance rule

The idea behind the (n, k)-*dominance rule* is to make sure that cells with contributors having large enough contributions are regarded as sensitive, i.e., to check if the sum of the n largest contributions to a cell is more than $k\%$ of the total cell value, denoted C. Mathematically this means that a cell is sensitive if

$$c_1 + c_2 + \ldots + c_n > \frac{k}{100}C$$

where $c_1, \ldots, c_n$ denote the n largest contributors to the cell. Consider the $(1, 90)$-dominance rule. This rule simply states that a cell is sensitive if the contribution of the largest contributor is more than 90% of the cell value. This would imply that B1, but no other cells, from Figure 2.4 would be considered sensitive. In order for B2 to be sensitive according to a $(1, k)$-dominance rule it must hold that $k/100 < 99/200 = 0.495$. This is probably not a good idea, since it implies that all cells to which one contributor contributes with more than 50% of the cell value would be sensitive.

The alternative to lowering k is to take more contributors into account, i.e., to increase the value of n. Applying a $(2, 90)$-dominance rule will indeed make A2, B1, and B2 sensitive. So, it seems like a suitable rule for this simple example. However, it turns out that also this rule suffer from inconsistency, which is illustrated in Example 2.4.

Example 2.4. Consider the hypothetical cells 46+45+9 and 80+9+4+4+3. Applying a $(2, 90)$-dominance rule, the first cell will be labeled as sensitive while the second will not. On the other hand, the best upper bound one contributor can get on one other contributor's contribution in the first cell is 100−45=55 and for the second cell 100−9=91. So the best upper bound on a contribution in the first cell is within $(55 - 46)/46 \approx 19.6\%$ of the true value and for the second cell it is within $(91 - 80)/80 \approx 13.8\%$. ♦

This example implies that, in terms of one attacker from the same cell, the first cell is actually safer than the second one. So in the context that no one should be able to gain information about an individual respondent, it is inconsistent to label the first cell as sensitive and the second as safe. In other words, a rule that leads to the first cell being sensitive should also lead to the second cell being sensitive. This issue illustrates that measuring dominance and sensitivity is not necessarily equivalent. This subtle problem, unlike the one illustrated in the $(1, k)$-dominance case, cannot be circumvented by changing the values of the parameters. This problem is a property of the rule itself. As an example, for a $(3, 90)$-dominance rule the cells 80+9+4+4+3 and 46+45+8+1 can be constructed to illustrate the inconsistency. As we shall see, the rule discussed next does not suffer from this inconsistency.

2.1.3 (p, q)-prior-posterior rule

The third rule is the so-called (p, q)-*prior-posterior rule* which involves two parameters $p > 0$ and $q > 0$, chosen such that $p < q$. It is assumed that without access to the table, an attacker can estimate the contribution of any contributor to within $q\%$ of its actual value. A cell is sensitive if one (or more) contributions can be estimated by an individual attacker, with access to the table, to within $p\%$. In order to formulate this mathematically it might seem necessary to compute upper and lower bounds on all contributions for every possible attacker (who might also be a contributor to some cell) but this turns out to not be the case. Since a cell is sensitive if any contribution can be estimated too closely by any attacker it suffices to check the best upper and lower bound that can be computed on some contribution. The best bounds are obtained if the second largest contributor of a cell attempts to approximate the largest contribution of the same cell (see [4]). For a cell with n_c contributions $c_1 \geq c_2 \geq \ldots \geq c_{n_c} \geq 0$, the second largest contributor (with contribution c_2) can obtain an upper bound U on the value of c_1 by subtracting from the cell value their own contribution and the lower bound of each of the other contributions, i.e.,

$$U = \sum_{i=1}^{n_c} c_i - c_2 - \left(1 - \frac{q}{100}\right) \sum_{i=3}^{n_c} c_i = c_1 + \frac{q}{100} \sum_{i=3}^{n_c} c_i.$$

According to the (p, q)-prior-posterior rule this cell is sensitive if $U < c_1 + (p/100)c_1$ and safe otherwise, which simplifies to the condition

$$q \sum_{i=3}^{n_c} c_i < pc_1. \tag{2.1}$$

A lower bound L on c_1 can be calculated by the second largest contributor as

$$L = \sum_{i=1}^{n_c} c_i - c_2 - \left(1 + \frac{q}{100}\right) \sum_{i=3}^{n_c} c_i = c_1 - \frac{q}{100} \sum_{i=3}^{n_c} c_i.$$

The condition $L > c_1 - (p/100)c_1$ will yield precisely the same criterion as for the upper bound and we conclude that, using the (p, q)-prior-posterior rule a cell is sensitive if and only if (2.1) holds. Note that this criterion only makes sense when all contributions are

Cell	min. freq., $f = 3$	$(1, 90)$-dom.	$(2, 90)$-dom.	$(20, 50)$-prior-post.
A1	safe	safe	safe	safe
A2	sensitive	safe	sensitive	sensitive
B1	safe	sensitive	sensitive	sensitive
B2	safe	safe	sensitive	sensitive
A Total	safe	safe	safe	safe
B Total	safe	safe	safe	safe
1 Total	safe	safe	safe	safe
2 Total	safe	safe	safe	safe
Table Total	safe	safe	safe	safe

Table 2.1: The result of applying different sensitivity rules to the table in Figure 2.1.

non-negative, since otherwise the definitions of the upper and lower bounds above would not be correct. See [12, p. 148] for a discussion regarding negative contributions.

The result of applying this rule with $p = 20$ and $q = 50$ to the example in Figure 2.4 is presented in Table 2.1 together with the results for a minimum frecuency rule with $f = 3$, a $(1, 90)$-dominance rule, and a $(2, 90)$-dominance rule. As can be seen, the problematic cells A2, B1, and B2 are labeled as sensitive by the (p, q)-prior-posterior rule. Further, the (p, q)-prior-posterior rule does not have the weakness of inconsistency that the (n, k)-dominance rule has. Indeed, all of the cells constructed to illustrate the inconsistency are labeled as sensitive by a (p, q)-prior-posterior rule with $p = 20$ and $q = 50$. For a discussion about why the inconsistency cannot exist for the (p, q)-prior posterior rule as well as further discussion of these rules, see [20].

A rule that is similar to the (p, q)-prior-posterior rule is the so-called $p\%$ rule for which it is assumed that the only a priori information an attacker has is that all contributions are non-negative. With this rule, a cell is considered sensitive if the upper bound of a contribution is within $p\%$ of the actual value. Since the lower bound of a contribution is always zero, a criterion for the lower bound is not applicable. It can easily be derived that the $p\%$ rule and the $(p, 100)$-prior-posterior rule yield the exact same criteria for sensitive cell (since $q = 100$ means precisely that the lower bound on a contribution is zero). For this reason the $p\%$ rule is considered a special case of the (p, q)-prior-posterior rule.

Note here that it has been assumed that several contributors to a table do not cooperate as attackers. The (p, q)-prior-posterior rule and $p\%$ rules can be generalized to remove this assumption by changing the lower summation limit in the left hand side of (2.1). If the second through r largest contributors of a cell try to approximate the largest contribution, the cell is sensitive if

$$q \sum_{i=r+2}^{n_c} c_i < pc_1.$$

See [18, p. 8] for further details.

The (p, q)-prior-posterior and $p\%$ rules are the most widely used in real applications since the minimum frequency and (n, k)-dominance rules have obvious shortcomings. That said, the minimum frequency rule can be used in combination with a (p, q)-prior-posterior or $p\%$ rule to yield additional protection for cells with few contributors.

Chapter 3

Secondary suppressions

Generally, if a table is published with only the sensitive cells omitted, then it will still be possible to calculate the missing values using the marginal totals. Consider the table in Figure 3.1 and assume the two suppressed cells are sensitive. Owing to the marginal totals in this table, it is easy to exactly calculate the missing values. For example, the value of cell B1 is given by solving the equation $x + 120 = 200$, i.e., B1 has the value 80. Therefore, in order to ensure that a table is safe for publication it is necessary to suppress some cells in addition to those that are sensitive. These suppressions are called *secondary suppressions*. The aim of the cell suppression problem is to choose these secondary suppressions in such a way that the table as a whole is safe while the loss of information is minimized. The choice is influenced by which sensitivity rule is used and what paramenter values are chosen. In the next section we discuss what constitutes a safe table, and in Section 3.4.1 we discuss the concept of information loss.

	1	2	Total
A	×	100	150
B	×	120	200
C	70	80	150
Total	200	300	500

Figure 3.1: A table with two sensitive cells suppressed (replaced by ×).

3.1 Safe tables

Assume that for the table in Figure 3.1 the cells A2 and B2 have been chosen as secondary suppressions, as illustrated in Figure 3.2. Suppose that an attacker is interested in the

	1	2	Total
A	×	×	150
B	×	×	200
C	70	80	150
Total	200	300	500

Figure 3.2: The table in Figure 3.1 with two secondary suppressions added.

value of cell B1. With no further information except that contributions to cell values are non-negative, the attacker cannot compute the exact value. However, it is clear from the marginal total of column 1 that the value cannot be higher than 130. So the attacker knows that the value of cell B1 is in the interval [0,130]. Is this cell sufficiently protected? For the (p, q)-prior-posterior and $p\%$ rules, the standard criterion in the literature for a *protected cell* is the following:

> The lower and upper bounds on the cell value that an attacker can compute should be such that, for both these values, the cell would be considered safe according to the sensitivity rule.

We shall say that a table is *safe* if every sensitive cell is protected according to this criterion. Next, we derive explicit expressions for the bounds.

Recall that the criterion (2.1) for a cell being sensitive was derived by letting the second largest contributor to a cell calculate an upper bound on the cell's largest contribution. A cell with value $C = \sum_{i=1}^{n_c} c_i$ is sensitive if

$$U = C - c_2 - \left(1 - \frac{q}{100}\right) \sum_{i=3}^{n_c} c_i < \left(1 + \frac{p}{100}\right) c_1$$

or, expressed in terms of the cell value, if

$$C < \left(1 + \frac{p}{100}\right) c_1 + c_2 + \left(1 - \frac{q}{100}\right) \sum_{i=3}^{n_c} c_i.$$

Now, let $T_\mathrm{u} \geq 0$ and $T_\mathrm{l} \geq 0$ denote the distances from the cell value C to its upper and lower bounds, respectively. With $C + T_\mathrm{u}$ as an upper bound on the cell value, the upper bound on c_1 that the second largest contributor can calculate is instead given by

$$C + T_\mathrm{u} - c_2 - \left(1 - \frac{q}{100}\right) \sum_{i=3}^{n_c} c_i.$$

Thus, according to the criterion for protected cell formulated above, a cell is protected if it holds that

$$C + T_\mathrm{u} - c_2 - \left(1 - \frac{q}{100}\right) \sum_{i=3}^{n_c} c_i \geq \left(1 + \frac{p}{100}\right) c_1$$

or, equivalently, that

$$C + T_\mathrm{u} \geq \left(1 + \frac{p}{100}\right) c_1 + c_2 + \left(1 - \frac{q}{100}\right) \sum_{i=3}^{n_c} c_i. \tag{3.1}$$

An analogous requirement for the lower bound on the cell value is that

$$C - T_\mathrm{l} \leq \left(1 - \frac{p}{100}\right) c_1 + c_2 + \left(1 + \frac{q}{100}\right) \sum_{i=3}^{n_c} c_i \tag{3.2}$$

should hold. From (3.1) and (3.2) it also follows that the smallest allowed values of T_u and T_l, the values which yield equalities, are both equal to

$$\frac{p}{100} c_1 - \frac{q}{100} \sum_{i=3}^{n_c} c_i. \tag{3.3}$$

So, are the two sensitive cells A1 and B1 in the table in Figure 3.2 protected? We shall look at three examples with different contributions to the cell values. Using the $p\%$ rule with $p = 10$ and assuming that the attacker is any individual contributor to the table, we compute the bounds on the values of the sensitive cells.

Example 3.1. First, assume the cell contributions are as follows.

	1	2	Total
A	(30+20)	(40+30+30)	150
B	(65+10+5)	(40+40+40)	200
C	30+20+20	30+30+20	150
Total	200	300	500

Figure 3.3: The first example of individual cell contributions to the table in Figure 3.2, with parentheses marking the suppressed cells.

To begin with, we establish that the cells A1 and B1 are indeed sensitive. Given all table values, the contributor with contribution 20 in cell A1 will find the upper bound of the largest contribution in the same cell as $50 - 20 = 30$. This upper bound is equal to, and thus obviously within 10% of, the largest contribution. In fact, according to the (p, q) and $p\%$ rule a cell with only one or two contributors is always sensitive, which is obvious from (2.1). For B1 the second largest contributor finds the upper bound $80 - 10 = 70 < 1.1 \cdot 65 = 71.5$, meaning B1 is sensitive as well.

The inequalities (3.1) and (3.2) with $p = 10$ and $q = 100$ give that the lower bound on A1 must be at most $0.9 \cdot 30 + 20 = 47$ and the upper bound at least $1.1 \cdot 30 + 20 = 53$. For B1 the lower bound should be lower than or equal to $0.9 \cdot 65 + 10 + 2 \cdot 5 = 78.5$ and the upper bound greater than or equal to $1.1 \cdot 65 + 10 = 81.5$. The lowest upper bounds on these cell values are obtained if the largest contributor to one cell attempts to disclose the value of the other. The largest contributor to cell A1 would obtain an upper bound on B1 as

$$200 - (30 + 20 + 20) - 30 = 100$$

which is greater than the required 81.5. The largest contributor to B1 can calculate an upper bound on A1 as

$$200 - (30 + 20 + 20) - 65 = 65$$

which is also greater than the required value 53. Studying Figure 3.3 and recalling the assumptions of the $p\%$ rule, it is straightforward to see that no individual contributor can calculate lower bounds that are too close, so the table is indeed protected. ⧫

Example 3.2. For the second example, assume the following cell contributions.

	1	2	Total
A	(50)	(40+30+30)	150
B	(80)	(40+40+40)	200
C	30+20+20	30+30+20	150
Total	200	300	500

Figure 3.4: The second example of individual cell contributions to the table in Figure 3.2, with parentheses marking the suppressed cells.

In this instance, there is only one contributor to cell A1 and one to B1 (the cells are so-called *singletons*), which as we shall see, can be problematic. Once again these two cells are sensitive but we omit the calculations. The contributor to cell A1 can calculate 80 as an upper bound on B1 and similarly, the contributor to B1 can find 50 as an upper bound on A1. Here, the disclosure was actually exact so clearly the table is not safe. Whether the respective contributors would know that the disclosure was exact depends on whether they know that they are singletons. From the formulation of the $p\%$ rule (and also the (p, q)-prior-posterior rule) it is not immediately clear if they have this information, but if the table concerns, for example, turnover of companies it might be reasonable to assume that they do. ⧫

Example 3.3. As the last example, consider the following contributions.

	1	2	Total
A	(48+2)	(40+30+30)	150
B	(70+8+2)	(40+40+40)	200
C	30+20+20	30+30+20	150
Total	200	300	500

Figure 3.5: The third example of individual cell contributions to the table in Figure 3.2, with parentheses marking the suppressed cells.

This instance is similar to the preceding example with singletons, but a bit more subtle. Even though there are no singletons, the largest contributor to A1 can estimate B1 too closely. The largest contributor to cell A1 knows that the value of A1 is at least 48. Therefore the upper bound of B1 is

$$200 - (30 + 20 + 20) - 48 = 82.$$

This is less than 85 which is required based on (3.1) with $p = 10$ and $q = 100$ and thus this table is not safe for publication.

So the largest contributor to A1 finds the upper bound 82 for cell B1 and this is also the upper bound on the largest single contribution to cell B1. Contrarywise, the second largest contributor to B1 can only obtain

$$200 - (30 + 20 + 20) - 8 = 122$$

as an upper bound on the largest contribution. This may seem contradictory since it was claimed in Section 2.1.3 that it is the second largest contributor to a cell that can make the best estimate of the largest contribution in the same cell. However, the claim was made with the assumption that the complete table was available, which is not the case in this example where some cells are suppressed. ⧫

3.1.1 Further discussion

Example 3.3 has another interesting feature. The table was not considered safe since one attacker could approximate a cell value too closely. Even so, seen as an approximation of an individual contribution, that approximation was not too close according to the

p% rule (with $p = 10$) since $82 \geq 1.1 \cdot 70 = 77$. This brings about the question of why the standard definition of a safe table is formulated to protect cell values rather than the underlying contributions. Indeed, in [4] Daalmans and de Waal criticize this. With the aim of protecting contributions, they formulate a new criterion that also allows for negative contributions and attackers contributing to several cells. Further, they construct an auditing problem that is used to determine whether a table is sufficiently protected according to this new criterion. They do not, however, offer a method that can produce tables that are safe according to this criterion.

In [9], Fischetti and Salazar-González construct a method for the CSP that works for general magnitude tables (hierarchical, linked, and of any dimension) with continuous response variables, with the aim of protecting cell values. The method only takes into account external attackers. Salazar-González extends the method in [21] to work for a general attacker. This improved method should theoretically be general enough to be able to protect a table from both external and internal attackers and, if individual protection levels (i.e, how closely an attacker is allowed to estimate a cell value) are defined for each attacker, it should be possible to obtain a suppressed table in which contributions, rather than cell values, are protected. The drawback is that this method involves solving several optmization subproblems (the role of which will be explained in Section 3.2) for each attacker, and the number of attackers may be huge. The author does not discuss how to choose the protection levels or what attackers should be considered. Tables with integer response variables are discussed briefly in [22]. A solution process is suggested but not tested.

The models implemented for this thesis were first presented in [9] and [10] and aim to protect a table, with continuous response variable, from external attackers only. It is very important to be aware of precisely what is accomplished by these models. No consideration is taken to internal attackers (entities that are themselves contributors to the tables). This means that some of the protection could be undone if one (or more) contribution is known exactly. This problem is especially prominent if the known contribution is also known to be a singleton, as demonstrated above. The software in use at SCB today for disclosure control, τ-ARGUS, is based on the same mathematical techniques that are presented in Section 3.2. Essentially, τ-ARGUS can protect tables only against external attackers, but options are offered to apply extra singleton protection for a few specific situations (see [7, pp. 20–22] for details). Such measures have not been included in the implementation for this thesis.

3.2 Complete cell suppression

To be able to construct a mathematical model that works for a general magnitude table with continuous response variable we need to introduce a few sets. Let $I = \{1, 2, \ldots, N\}$ denote the index set of table $a = [a_1 \ldots a_N]^T \in \mathbb{R}^N$, let $I_S \subseteq I$ denote the set of sensitive cells, as determined by some sensitivity rule, and let $I_{SUP} = \{i \in I : a_i \text{ is supressed}\} \subseteq I$. The set I_{SUP} is referred to as a *suppression pattern* of table a.

Example 3.4. As an example we study the tables in Figure 3.6, with cells numbered row-wise from left to right.

	1	2	3	Total
A	**255**	90	45	390
B	290	230	65	585
Total	545	320	110	975

(a)

	1	2	3	Total
A	×	90	×	390
B	×	230	×	585
Total	545	320	110	975

(b)

Figure 3.6: (a): A table in which bold face indicates a sensitive cell; (b): the same table with a suppression pattern (suppressed cells are marked by ×).

The table in Figure 3.6(a) has $N = 12$ cells, i.e., $I = \{1, 2, \ldots, 12\}$. A1 is the only sensitive cell, which means that $I_{\mathrm{S}} = \{1\}$. In Figure 3.6(b), the same table is given with the suppression pattern I_{SUP}=$\{1, 3, 5, 7\}$. ♦

It is clear from Figure 3.6(b) that the values in Figure 3.6(a) is not the only possible set of values for the suppressed cells which will ensure that all relations of the table are fulfilled (for this simple table the only relations are the marginal totals). Generally, given a table $a \in \mathbb{R}^N$ with linear relations represented by a matrix $M \in \mathbb{R}^{m \times N}$ and a vector $b \in \mathbb{R}^m$, and external bounds l_i and u_i, $i \in I$, a vector $y = [y_1 \ldots y_N]^T \in \mathbb{R}^N$ is a *consistent table* with respect to a and a suppression pattern $I_{\mathrm{SUP}} \subseteq I = \{1, \ldots, N\}$, if the constraints

$$My = b, \tag{3.4a}$$
$$y_i = a_i, \qquad i \in I \setminus I_{\mathrm{SUP}}, \tag{3.4b}$$
$$l_i \leq y_i \leq u_i, \qquad i \in I_{\mathrm{SUP}}, \tag{3.4c}$$

are all satisfied. The largest possible value of a suppressed cell $i \in I_{\mathrm{SUP}}$ in any consistent table is denoted $\overline{y}_i$. Analogously, the smallest value possible is denoted $\underline{y}_i$. The interval defined by these two values, i.e., $[\underline{y}_i, \overline{y}_i]$, is the range of feasible values for cell i.

Example 3.5. We return to the table in Figure 3.6(b) and assume the external bounds $l_i = 0$ and $u_i = 1000$, $i \in I$. The values $\overline{y}_i$, $i \in I_{\mathrm{SUP}}$=$\{1, 3, 5, 7\}$, are found by solving the optimization problem

$$\begin{aligned}
\max \quad & y_i, \\
\text{s.t.} \quad & [y_2\ y_4\ y_6\ y_8\ y_9\ y_{10}\ y_{11}\ y_{12}] = [90\ 390\ 230\ 585\ 545\ 320\ 110\ 975], \\
& y_1 + y_2 + y_3 - y_4 = 0, \\
& y_5 + y_6 + y_7 - y_8 = 0, \\
& y_9 + y_{10} + y_{11} - y_{12} = 0, \\
& y_1 + y_5 - y_9 = 0, \\
& y_2 + y_6 - y_{10} = 0, \\
& y_3 + y_7 - y_{11} = 0, \\
& y_4 + y_8 - y_{12} = 0, \\
& 0 \leq y_j \leq 1000, \quad j = 1, 3, 5, 7,
\end{aligned}$$

for each $i \in I_{\mathrm{SUP}}$. Analagously, $\underline{y}_i$, $i \in I_{\mathrm{SUP}}$, are found by replacing the maximization

by minimization. This yields the following intervals on the suppressed cells

$$\begin{aligned} y_1 &\in [190,\ 300], \\ y_3 &\in [0,\ 110], \\ y_5 &\in [245,\ 355], \\ y_7 &\in [0,\ 110]. \end{aligned}$$ ♦

Recall that an external attacker has knowledge about the table structure (M), the published cell values (a_i, $i \in I \setminus I_{\text{SUP}}$) and the external bounds (l_i and u_i, $i \in I$). This means that $\overline{y}_i$ and $\underline{y}_i$ are precisely the upper and lower bounds of the interval that an external attacker can compute on a suppressed cell value. Generally, the interval that an external attacker can compute for a suppressed cell i is called the *suppression interval* of cell i.

Now, we can address the issue of how to determine whether a table is safe with respect to a suppression pattern. As pointed out in the previous section, a sensitive cell is considered protected if the upper and lower bounds on the cell value are sufficiently large and small, respectively. In order to model this we introduce three non-negative real numbers, *protection levels*, for each sensitive cell $s \in I_{\text{S}}$, denoted P_s^{U} (upper protection level), P_s^{L} (lower protection level) and P_s^{SL} (sliding protection level). For a given suppression pattern, a sensitive cell $s \in I_{\text{S}}$ is protected if the following three conditions hold:

$$\overline{y}_s \geq a_s + P_s^{\text{U}}, \tag{3.5a}$$

$$\underline{y}_s \leq a_s - P_s^{\text{L}}, \tag{3.5b}$$

$$\overline{y}_s - \underline{y}_s \geq P_s^{\text{SL}}. \tag{3.5c}$$

The conditions (3.5a), (3.5b), and (3.5c) will be referred to as the upper, lower, and sliding protection level requirements, respectively. Intuitively, these conditions mean that an attacker should not be able to approximate a sensitive cell too closely, or, in other words, that the range of feasible values for a sensitive cell is sufficiently large. If the standard criterion for a protected cell, discussed in Section 3.1, is used, P_s^{L} and P_s^{U} should be set equal to the expression in (3.3) in which case, typically, $P_s^{\text{SL}} = 0$. However, it is sometimes preferable to use $P_s^{\text{U}} = P_s^{\text{L}} = 0$ and $P_s^{\text{SL}} > 0$ in order to minimize the risk that the real cell value is in the middle of the suppression interval.

3.2.1 Model outline and attacker subproblems

The *complete cell suppression problem* must ensure that the suppression interval of every sensitive cell is sufficiently wide. Mathematically, this means that (3.5a)–(3.5c) hold. Also, the suppressed table should retain as much information as possible. As mentioned, information loss is discussed thoroughly in Section 3.4.1.

Let $a \in \mathbb{R}^N$ be a table with linear relations represented by $M \in \mathbb{R}^{m \times N}$ and $b \in \mathbb{R}^m$, external bounds u_i and l_i, $i \in I$, a set of sensitive cells $I_{\text{S}} \subseteq I$, and protection levels P_s^{U}, P_s^{L}, and P_s^{SL}, $s \in I_{\text{S}}$. In order to model the complete CSP, we introduce the vector $x = [x_1 \dots x_N]^T$, where each x_i is a binary variable that equals 1 if cell i is suppressed and 0 otherwise, i.e., x corresponds to a suppression pattern $I_{\text{SUP}} = \{i \in I : x_i = 1\}$. Also, we introduce a weight w_i, $i \in I$, for every cell representing the cost of suppressing

cell i. Now we can outline a model for the complete CSP:

$$\min \quad \sum_{i=1}^{N} w_i x_i, \tag{3.6a}$$

$$\text{s.t.} \quad \begin{cases} \text{the suppression pattern associated with } x \text{ satisfies the} \\ \text{upper protection level requirement (3.5a) for all } s \in I_{\mathrm{S}}, \end{cases} \tag{3.6b}$$

$$\begin{cases} \text{the suppression pattern associated with } x \text{ satisfies the} \\ \text{lower protection level requirement (3.5b) for all } s \in I_{\mathrm{S}}, \end{cases} \tag{3.6c}$$

$$\begin{cases} \text{the suppression pattern associated with } x \text{ satisfies the} \\ \text{sliding protection level requirement (3.5c) for all } s \in I_{\mathrm{S}}, \end{cases} \tag{3.6d}$$

$$x \in \{0,1\}^N. \tag{3.6e}$$

Note here that as a measure of information loss (3.6a) only makes sense if the weights w_i are positive. Otherwise it would not be disadvantageous to suppress cells, which is not a situation that is desirable. The constraints (3.6b)–(3.6d) depend on $\overline{y}_i$ and $\underline{y}_i$, which in turn depend on the suppression pattern corresponding to x. So how do we turn this outline into a mathematical model expressed only in the x_i variables? We shall describe a model first suggested by Fischetti and Salazar-González in [9].

The first step is to formulate the optimization problems used to calculate the upper limit $\overline{y}_s$ and lower limit $\underline{y}_s$ of the suppression interval for each cell $s \in I_{\mathrm{S}}$. For a fixed $x = \widehat{x}$, $\overline{y}_s$ is obtained by solving the *attacker subproblem* associated with the upper protection level:

$$\max_{y} \quad y_s, \tag{3.7a}$$

$$\text{s.t.} \quad My = b, \tag{3.7b}$$

$$y_i \le a_i + (u_i - a_i)\widehat{x}_i, \qquad i \in I, \tag{3.7c}$$

$$y_i \ge a_i - (a_i - l_i)\widehat{x}_i, \qquad i \in I, \tag{3.7d}$$

where the constraint (3.7b) guarantees that the structure of the table a is preserved and (3.7c) and (3.7d) make sure that the non-suppressed cells, for which $\widehat{x}_i = 0$, are fixed to their original table values and that the suppressed ones, for which $\widehat{x}_i = 1$, are within the external bounds. Note that these constraints are equivalent to the conditions in the definition of consistent table (3.4). Analogously, for a fixed $x = \widehat{x}$, $\underline{y}_s$ is obtained by solving the attacker subproblem associated with the lower protection level:

$$\min_{y} \quad y_s, \tag{3.8a}$$

$$\text{s.t.} \quad My = b, \tag{3.8b}$$

$$y_i \le a_i + (u_i - a_i)\widehat{x}_i, \qquad i \in I, \tag{3.8c}$$

$$y_i \ge a_i - (a_i - l_i)\widehat{x}_i, \qquad i \in I. \tag{3.8d}$$

These subproblems are used to check if a cell is protected, i.e., if (3.5a)–(3.5c) hold, for a given $\widehat{x}$. However, as we shall see, there is more knowledge to be gained, by studying the linear programming (LP) duals of the attacker subproblems.

3.2.2 Dual subproblems

The LP duals of the subproblems (3.7) and (3.8) will be used to derive constraints, expressed in the x_i variables, that impose the upper, lower, and sliding protection level requirements. For an introduction to linear programming duality see [1, Ch. 10].

Imposing the upper protection level requirement

Before formulating the LP dual of the maximization subproblem (3.7) we rewrite (3.7d) to $-y_i \leq -a_i + (a_i - l_i)\widehat{x}_i$ in order to get non-negative dual variables. The dual problem is then

$$\min_{\gamma,\,\alpha,\,\beta} \quad \gamma^T b + \sum_{i=1}^{N} \Big(\alpha_i(a_i + (u_i - a_i)\widehat{x}_i) - \beta_i(a_i - (a_i - l_i)\widehat{x}_i)\Big), \tag{3.9a}$$

$$\text{s.t.} \quad M^T\gamma + \alpha - \beta = e_s, \tag{3.9b}$$

$$\alpha \geq \mathbf{0},\ \beta \geq \mathbf{0}, \tag{3.9c}$$

where $\gamma \in \mathbb{R}^m$, $\alpha \in \mathbb{R}^N$, and $\beta \in \mathbb{R}^N$ denote the vectors of dual variables associated with (3.7b), (3.7c) and (3.7d), respectively, and e_s denotes the unit vector with a 1 in position s. The objective function in (3.9a) can be rewritten as

$$\gamma^T b + \alpha^T a - \beta^T a + \sum_{i=1}^{N} \Big(\alpha_i(u_i - a_i)\widehat{x}_i + \beta_i(a_i - l_i)\widehat{x}_i\Big).$$

By using $Ma = b$ and (3.9b) we get the following equalities for the first three terms in the objective function

$$\begin{aligned}\gamma^T b + \alpha^T a - \beta^T a &= \gamma^T M a + \alpha^T a - \beta^T a,\\ &= (\gamma^T M + \alpha^T - \beta^T)a,\\ &= e_s^T a,\\ &= a_s.\end{aligned}$$

Thus, the objective function in (3.9a) can be reduced to

$$a_s + \sum_{i=1}^{N} \Big(\alpha_i(u_i - a_i) + \beta_i(a_i - l_i)\Big)\widehat{x}_i. \tag{3.10}$$

Further, duality theory tells us that for a maximization problem the primal objective function value at a primal feasible solution is always less than or equal to the dual objective function value at a dual feasible solution, which implies

$$\overline{y}_s \leq a_s + \sum_{i=1}^{N} \Big(\alpha_i(u_i - a_i) + \beta_i(a_i - l_i)\Big)\widehat{x}_i \tag{3.11}$$

for all γ, α, and β fulfilling the constraints (3.9b) and (3.9c). Recall the upper protection level requirement (3.5a), which says that $\overline{y}_s \geq a_s + P_s^{\mathrm{U}}$ needs to hold in order for cell s to be protected. This, together with (3.11), yields the inequality

$$\sum_{i=1}^{N} \Big(\alpha_i(u_i - a_i) + \beta_i(a_i - l_i)\Big)x_i \geq P_s^{\mathrm{U}} \tag{3.12}$$

which has to hold for all (γ, α, β) satisfying (3.9b) and (3.9c) in order for the upper protection level requirement to be fulfilled for cell s.

Imposing the lower protection level requirement

The procedure to derive an expression for the lower protection level requirement is analogous. We rewrite (3.8d) to $-y_i \leq -a_i + (a_i - l_i)\widehat{x}_i$ and we also change (3.8a) to $\max -y_s$. The LP dual of (3.8) is then expressed as

$$\min_{\widetilde{\gamma},\widetilde{\alpha},\widetilde{\beta}} \quad \widetilde{\gamma}^T b + \sum_{i=1}^{N} \Big(\widetilde{\alpha}_i(a_i + (u_i - a_i)\widehat{x}_i) - \widetilde{\beta}_i(a_i - (a_i - l_i)\widehat{x}_i)\Big), \tag{3.13a}$$

$$\text{s.t.} \quad M^T\widetilde{\gamma} + \widetilde{\alpha} - \widetilde{\beta} = -e_s, \tag{3.13b}$$

$$\widetilde{\alpha} \geq \mathbf{0},\ \widetilde{\beta} \geq \mathbf{0}. \tag{3.13c}$$

Analogously with the derivation of (3.10), the objective function in (3.13a) can be written as

$$-a_s + \sum_{i=1}^{N} \Big(\widetilde{\alpha}_i(u_i - a_i) + \widetilde{\beta}_i(a_i - l_i)\Big)\widehat{x}_i.$$

Since the primal is a maximization problem, the dual objective value at a dual feasible solution will be greater than or equal to the primal objective value at a primal feasible solution, which gives

$$-\underline{y}_s \leq -a_s + \sum_{i=1}^{N} \Big(\widetilde{\alpha}_i(u_i - a_i) + \widetilde{\beta}_i(a_i - l_i)\Big)\widehat{x}_i \tag{3.14}$$

for all $\widetilde{\gamma}$, $\widetilde{\alpha}$, and $\widetilde{\beta}$ satisfying (3.13b) and (3.13c). Using this together with (3.5b), i.e., $\underline{y}_s \leq a_s - P_s^{\mathrm{L}}$, yields

$$\sum_{i=1}^{N} \Big(\widetilde{\alpha}_i(u_i - a_i) + \widetilde{\beta}_i(a_i - l_i)\Big)x_i \geq P_s^{\mathrm{L}} \tag{3.15}$$

which needs to hold for all ($\widetilde{\gamma}$, $\widetilde{\alpha}$, $\widetilde{\beta}$) satisfying (3.13b) and (3.13c) in order for the lower protection level requirement of cell s to be fulfilled.

Imposing the sliding protection level requirement

Lastly, we need an analogous condition for the sliding protection level requirement (3.5c), i.e., the condition $\overline{y}_s - \underline{y}_s \geq P_s^{\mathrm{SL}}$. By use of the inequalities (3.11) and (3.14) we get that

$$\begin{aligned}\overline{y}_s + (-\underline{y}_s) \leq\ & a_s + \sum_{i=1}^{N} \Big(\alpha_i(u_i - a_i) + \beta_i(a_i - l_i)\Big)\widehat{x}_i \\ & - a_s + \sum_{i=1}^{N} \Big(\widetilde{\alpha}_i(u_i - a_i) + \widetilde{\beta}_i(a_i - l_i)\Big)\widehat{x}_i\end{aligned}$$

Using this together with (3.5c), i.e., $\overline{y}_s - \underline{y}_s \geq P_s^{\mathrm{SL}}$, we end up with

$$\sum_{i=1}^{N} \Big((\alpha_i + \widetilde{\alpha}_i)(u_i - a_i) + (\beta_i + \widetilde{\beta}_i)(a_i - l_i)\Big)x_i \geq P_s^{\mathrm{SL}}. \tag{3.16}$$

for all $(\gamma,\ \alpha,\ \beta)$ satisfying (3.9b) and (3.9c) and for all $(\widetilde{\gamma},\ \widetilde{\alpha},\ \widetilde{\beta})$ satisfying (3.13b) and (3.13c).

The inequalities (3.12), (3.15), and (3.16) impose the upper, lower, and sliding protection level requirements, but define an infinite number of constraints on x. However, it is shown in Appendix A that for these inequalities, the extreme points of the polyhedra defined by (3.9b)–(3.9c) and (3.13b)–(3.13c) are sufficient in order to impose all the protection level requirements.

3.2.3 The complete CSP

Finally, we have arrived at the following mathematical formulation of the complete CSP

$$\min_{x} \quad \sum_{i=1}^{N} w_i x_i, \tag{3.17a}$$

$$\text{s.t.} \quad \begin{cases} \sum_{i=1}^{N} \Big(\alpha_i(u_i - a_i) + \beta_i(a_i - l_i)\Big)x_i \geq P_s^{\mathrm{U}} \text{ for all} \\ \text{extreme points } (\gamma,\ \alpha,\ \beta) \text{ of the set defined by (3.9b–c),} \end{cases} \quad s \in I_{\mathrm{S}}, \tag{3.17b}$$

$$\begin{cases} \sum_{i=1}^{N} \Big(\widetilde{\alpha}_i(u_i - a_i) + \widetilde{\beta}_i(a_i - l_i)\Big)x_i \geq P_s^{\mathrm{L}} \text{ for all} \\ \text{extreme points } (\widetilde{\gamma},\ \widetilde{\alpha},\ \widetilde{\beta}) \text{ of the set defined by (3.13b–c),} \end{cases} \quad s \in I_{\mathrm{S}}, \tag{3.17c}$$

$$\begin{cases} \sum_{i=1}^{N} \Big((\alpha_i + \widetilde{\alpha}_i)(u_i - a_i) + (\beta_i + \widetilde{\beta}_i)(a_i - l_i)\Big)x_i \geq P_s^{\mathrm{SL}} \\ \text{for all extreme points } (\gamma,\ \alpha,\ \beta) \text{ and } (\widetilde{\gamma},\ \widetilde{\alpha},\ \widetilde{\beta}) \text{ of the} \\ \text{sets defined by (3.9b–c) and (3.13b–c), respectively,} \end{cases} \quad s \in I_{\mathrm{S}}, \tag{3.17d}$$

$$x \in \{0,1\}^N. \tag{3.17e}$$

The model (3.17) is called the complete CSP because suppressed cells are completely removed from the table. As demonstrated, after protecting the table using (3.17) it is still possible to calculate a suppression interval on each sensitive cell value. This fact inspired another approach, called partial cell suppression, which is described next.

3.3 Partial cell suppression

Partial cell suppression is an approach to cell suppression that is slightly different from complete cell suppression. In this approach all variables are continuous and represent the difference between the real cell value and the bounds of the suppression interval. The idea is that instead of suppressing cells completely, i.e., substituting them with some arbitrary symbol, each cell that needs to be suppressed is assigned an interval that covers the actual value. To get an idea of what this means, recall Figure 3.6(a), which shows an example table with one sensitive cell, and Figure 3.6(b) which shows the same table protected by complete cell suppression. Using partial cell suppression the protected version could instead be the table in Figure 3.7.

As shown in the previous section, an attacker can compute intervals for cell values that have been suppressed. A table is safe if the suppression intervals of the sensitive cells are sufficiently large. Using partial cell suppression it is possible to control the

	1	2	3	Total
A	[240, 274]	90	[26, 60]	390
B	[271, 305]	230	[50, 84]	585
Total	545	320	110	975

Figure 3.7: The table in Figure 3.6(a) protected using partial cell suppression.

widths of the suppression intervals so that a table can be published with intervals that are no larger than is necessary for the table to be safe. In this section we shall describe a model for the partial CSP that was introduced by Fischetti and Salazar-González in [10] and that is in the same spirit as their model for the complete CSP.

For a table a we introduce the non-negative variables, $z^+ = [z_1^+ \dots z_N^+]$ and $z^- = [z_1^- \dots z_N^-]$. The published table will contain the intervals $[a_i - z_i^-,\ a_i + z_i^+]$. The case $z_i^+ = z_i^- = 0$ is allowed and means that the actual value is published. Further, the inequalities $a_i + z_i^+ \leq u_i$ and $a_i - z_i^- \geq l_i$ should hold, since it would be meaningless to publish intervals that are wider than what is given by the external bounds u_i and l_i.

3.3.1 Model outline and attacker subproblems

The goal now is to find a set of intervals for which the suppressed table fulfills the upper, lower, and sliding protection level requirements, while the information loss is minimized. We shall first introduce the attacker subproblems for the partial CSP. With $(z^+,\ z^-) = (\widehat{z}^+,\ \widehat{z}^-)$, an attacker now knows that the actual value of a sensitive cell $s \in I_{\mathrm{S}}$ is in the interval $[a_s - \widehat{z}_s^-,\ a_s + \widehat{z}_s^+]$, so the maximal possible value of y_s, denoted $\overline{y}_s$, is given by

$$\max_y \quad y_s, \tag{3.18a}$$

$$\text{s.t.} \quad My = b, \tag{3.18b}$$

$$y_i \leq a_i + \widehat{z}_i^+, \qquad i \in I, \tag{3.18c}$$

$$y_i \geq a_i - \widehat{z}_i^-, \qquad i \in I, \tag{3.18d}$$

and the minimum possible value, denoted $\underline{y}_s$, by

$$\min_y \quad y_s, \tag{3.19a}$$

$$\text{s.t.} \quad My = b, \tag{3.19b}$$

$$y_i \leq a_i + \widehat{z}_i^+, \qquad i \in I, \tag{3.19c}$$

$$y_i \geq a_i - \widehat{z}_i^-, \qquad i \in I. \tag{3.19d}$$

Then, all protection levels for cell s are fulfilled if

$$\overline{y}_s \geq a_s + P_s^{\mathrm{U}}, \tag{3.20a}$$

$$\underline{y}_s \leq a_s - P_s^{\mathrm{L}}, \tag{3.20b}$$

$$\overline{y}_s - \underline{y}_s \geq P_s^{\mathrm{SL}}, \tag{3.20c}$$

hold. The bounds in the subproblems, (3.18c)–(3.18d) or (3.19c)–(3.19d), together with (3.20a)–(3.20c), imply $z_s^+ \geq P_s^{\mathrm{U}}$, $z_s^- \geq P_s^{\mathrm{L}}$, and $z_s^+ - z_s^- \geq P_s^{\mathrm{SL}}$ for all sensitive cells $s \in I_{\mathrm{S}}$.

The partial CSP should aim to minimize the loss of information, computed as

$$\sum_{i=1}^{N}(w_i^+ z_i^+ + w_i^- z_i^-)$$

for some weights $w_i^+ > 0$ and $w_i^- > 0$, $i \in I$. We outline the partial CSP, similarly to the complete CSP in (3.6), as

$$\begin{aligned}
\min \quad & \sum_{i=1}^{N}(w_i^+ z_i^+ + w_i^- z_i^-), && && \text{(3.21a)}\\
\text{s.t.} \quad & P_s^{\mathrm{U}} \le z_p^+ \le u_s - a_s, && s \in I_{\mathrm{S}}, && \text{(3.21b)}\\
& 0 \le z_i^+ \le u_i - a_i, && i \in I \setminus I_{\mathrm{S}}, && \text{(3.21c)}\\
& P_s^{\mathrm{L}} \le z_s^- \le a_s - l_s, && s \in I_{\mathrm{S}}, && \text{(3.21d)}\\
& 0 \le z_i^- \le a_i - l_i, && i \in I \setminus I_{\mathrm{S}}, && \text{(3.21e)}\\
& z_s^+ + z_s^- \ge P_s^{\mathrm{SL}}, && s \in I_{\mathrm{S}}, && \text{(3.21f)}\\
& \text{(3.20a)–(3.20c) hold for all } s \in I_{\mathrm{S}}. && && \text{(3.21g)}
\end{aligned}$$

We need to find out how to impose (3.20a)–(3.20c) for the model (3.21), but first there is an important observation to make.

If all weights w_i^+ and w_i^-, $i \in I$, are positive, then in an optimal solution to (3.21) the interval $[a_j - z_j^-,\, a_j + z_j^+]$ for every suppressed cell $j \in I_{\mathrm{SUP}}$ will coincide with the suppression interval $[\underline{y}_j,\, \overline{y}_j]$. To realize why this is, consider a feasible solution $(\check{z}^+,\, \check{z}^-)$ to (3.21) for which all protection level requirements are fulfilled. Assume that for some cell $k \in I_{\mathrm{SUP}}$, $a_k + \check{z}_k^+ = \overline{y}_k$ and $a_k - \check{z}_k^- = \underline{y}_k$ do not both hold. Then, either $a_k + \check{z}_k^+ > \overline{y}_k$ or $a_k - \check{z}_k^- < \underline{y}_k$ (or both) must hold, because of the bounds in the subproblems (3.18) and (3.19). This means that the objective value of (3.21) could be improved by decreasing $\check{z}_k^+$ or $\check{z}_k^-$, while also preserving the suppression intervals for all cells, and therefore $(\check{z}^+,\, \check{z}^-)$ cannot be optimal. In simplistic terms: we do not publish an interval that is wider than the interval an external attacker can compute. Hence, an optimal solution to the partial CSP with positive weights fulfills

$$\begin{aligned}
\overline{y}_s &= a_s + z_s^+, \qquad s \in I_{\mathrm{S}}, && \text{(3.22a)}\\
\underline{y}_s &= a_s - z_s^-, \qquad s \in I_{\mathrm{S}}. && \text{(3.22b)}
\end{aligned}$$

The first of the equalities, (3.22a), implies the upper protection level requirement (3.20a) and the second, (3.22b), implies the lower protection level requirement (3.20b). Together, they imply the sliding protection level requirement (3.20c).

3.3.2 Dual subproblems

We will use duality theory in the same fashion as for the complete CSP to construct a model for the partial CSP, expressed only in terms of the z_i^+ and z_i^- variables.

Imposing the upper protection level requirement

After rewriting (3.18d) to $-y_i \le -a_i + \widehat{z}_i^-$ the LP dual of the maximization subproblem (3.18) becomes

$$\min_{\gamma,\,\alpha,\,\beta} \quad \gamma^T b + \sum_{i=1}^{N} \Big(\alpha_i(a_i + \widehat{z}_i^{+}) - \beta_i(a_i - \widehat{z}_i^{-})\Big), \tag{3.23a}$$

$$\text{s.t.} \quad M^T\gamma + \alpha - \beta = e_s, \tag{3.23b}$$

$$\alpha \geq \mathbf{0},\ \beta \geq \mathbf{0}. \tag{3.23c}$$

Identically to the derivation of (3.10), by using $Ma = b$ and (3.23b), we get the equality

$$\gamma^T b + \alpha^T a - \beta^T a = a_s.$$

Thus, the objective function in (3.23a) reduces to

$$a_s + \sum_{i=1}^{N}(\alpha_i\widehat{z}_i^{+} + \beta_i\widehat{z}_i^{-}).$$

From duality theory we know that in a maximization problem, the objective function of the primal is less than or equal to the objective function of the dual, wherefore

$$\overline{y}_s \leq a_s + \sum_{i=1}^{N}(\alpha_i\widehat{z}_i^{+} + \beta_i\widehat{z}_i^{-})$$

with equality attained only for an optimal solution to the dual. Thus, we can express the equality (3.22a) as

$$a_s + \widehat{z}_s^{+} = a_s + \sum_{i=1}^{N}(\alpha_i^*\widehat{z}_i^{+} + \beta_i^*\widehat{z}_i^{-})$$

for an optimal solution $(\gamma^*,\ \alpha^*,\ \beta^*)$ to (3.23). Then we also have that

$$\widehat{z}_s^{+} \leq \sum_{i=1}^{N}(\alpha_i\widehat{z}_i^{+} + \beta_i\widehat{z}_i^{-}) \tag{3.24}$$

for all γ, α, and β satisfying (3.23b) and (3.23c). For arbitrary but fixed values of α and β that are feasible in (3.23), (3.22a) will be satisfied for all z^+ and z^- that also satisfy (3.24). Finally, since the upper protection level requirement (3.20a) follow from (3.22a), we conclude that (3.24) defines a set of constraints that guarantee that all upper protection levels are fulfilled.

Imposing the lower protection level requirement

The procedure is similar for the lower protection level requirement. After changing the objective of the minimization subproblem (3.19) to max $-y_s$ and the lower bound (3.19d) to $-y_i \leq a_i - \widehat{z}_i^{-}$ we obtain the LP dual

$$\min_{\widetilde{\gamma},\,\widetilde{\alpha},\,\widetilde{\beta}} \quad \widetilde{\gamma}^T b + \sum_{i=1}^{N} \Big(\widetilde{\alpha}_i(a_i + \widehat{z}_i^{+}) - \widetilde{\beta}_i(a_i - \widehat{z}_i^{-})\Big), \tag{3.25a}$$

$$\text{s.t.} \quad M^T\widetilde{\gamma} + \widetilde{\alpha} - \widetilde{\beta} = -e_s, \tag{3.25b}$$

$$\widetilde{\alpha} \geq \mathbf{0},\ \widetilde{\beta} \geq \mathbf{0}. \tag{3.25c}$$

We can derive that the equality (3.22b) is equivalent to

$$-a_s + \widehat{z}_s^- = -a_s + \sum_{i=1}^{N}(\widetilde{\alpha}_i^* \widehat{z}_i^+ + \widetilde{\beta}_i^* \widehat{z}_i^-)$$

for an optimal dual solution $(\widetilde{\gamma}^*,\ \widetilde{\alpha}^*,\ \widetilde{\beta}^*)$ to (3.25). From this we get the set of constraints

$$z_s^- \leq \sum_{i=1}^{N}(\widetilde{\alpha}_i z_i^+ + \widetilde{\beta}_i z_i^-) \tag{3.26}$$

for all $\widetilde{\gamma}$, $\widetilde{\alpha}$, and $\widetilde{\beta}$ satisfying (3.25b) and (3.25c), which guarantee the lower protection level requirement (3.20b). As was the case for the complete CSP, it is sufficient to consider only the extreme points of the polyhedra defined by (3.23b)–(3.23c) and (3.25b)–(3.25c) in the inequalities (3.24) and (3.26) (see Appendix A).

3.3.3 The partial CSP

We can now formulate the partial CSP as

$$\begin{aligned}
\min_{z^+,\, z^-} \quad & \sum_{i=1}^{N}(w_i^+ z_i^+ + w_i^- z_i^-), && && \text{(3.27a)}\\
\text{s.t.} \quad & P_s^{\mathrm{U}} \leq z_s^+ \leq u_s - a_s, && s \in I_{\mathrm{S}}, && \text{(3.27b)}\\
& 0 \leq z_i^+ \leq u_i - a_i, && i \in I \setminus I_{\mathrm{S}}, && \text{(3.27c)}\\
& P_s^{\mathrm{L}} \leq z_s^- \leq a_s - l_s, && s \in I_{\mathrm{S}}, && \text{(3.27d)}\\
& 0 \leq z_i^- \leq a_i - l_i, && i \in I \setminus I_{\mathrm{S}}, && \text{(3.27e)}\\
& z_s^+ + z_s^- \geq P_s^{\mathrm{SL}}, && s \in I_{\mathrm{S}}, && \text{(3.27f)}\\
& \begin{cases} \sum_{i=1}^{N}(\alpha_i z_i^+ + \beta_i z_i^-) - z_s^+ \geq 0 \text{ for all} \\ \text{extreme points } (\gamma,\ \alpha,\ \beta) \text{ of the set defined by (3.23b–c)} \end{cases} && s \in I_{\mathrm{S}}, && \text{(3.27g)}\\
& \begin{cases} \sum_{i=1}^{N}(\widetilde{\alpha}_i z_i^+ + \widetilde{\beta}_i z_i^-) - z_s^- \geq 0 \text{ for all} \\ \text{extreme points } (\widetilde{\gamma},\ \widetilde{\alpha},\ \widetilde{\beta}) \text{ of the set defined by (3.25b–c),} \end{cases} && s \in I_{\mathrm{S}}. && \text{(3.27h)}
\end{aligned}$$

3.4 Combined cell suppression

In this section we derive a third model for the CSP, called the combined CSP, which is a combination of the complete CSP and the partial CSP. But first, we discuss information loss, which will serve as a motivation for the combined CSP.

3.4.1 Information loss

One crucial part of optimization is to choose an objective function that minimizes or maximizes the right quantity. In many classical optimization problems, such as the traveling salesperson problem and the set covering problem, the objective function is quite easily determined. In the former problem the aim is typically to minimize the length of the route and in the latter to minimize the number of sets. For the CSP the

question of the objective function is a bit more complicated. This is because the concept of an optimal suppression pattern is not trivial. In vague terms, an optimal solution to the CSP should be a safe table where the loss of information is as small as possible. The main difficulty lies in defining the concept of information loss. To illustrate that this is indeed difficult we will examine the following four natural ways of defining information loss for the CSP:

I. the number of suppressed cells,

II. the sum of the suppressed cell values,

III. the sum of the widths of the suppression intervals, and

IV. the number of suppressed contributions.

There are also several other ways of measuring the information loss which will not be discussed here. We refer the interested reader to [12]. The definitions I–IV of information loss can be contradictory, as is shown in Example 3.6.

Example 3.6. We borrow two tables from [20, Fig. 11 and 12], reproduced in Figure 3.8, where suppressed cells are replaced by their suppression intervals.

[99, 101]	[0, 2]	101
[0, 2]	[0, 2]	2
101	2	103

(a)

[51, 77]	[0, 26]	77
[0, 26]	[0, 26]	26
77	26	103

(b)

Figure 3.8: Tables from [20] with the suppressed cells replaced by their suppression intervals.

According to definition I and II the information loss is equal in Figures 3.8(a) and 3.8(b), since both tables contain four suppressed cells and the sum of the suppressed cells equals 103. According to definition III, however, the table in Figure 3.8(a) has an information loss of $(101-99)+(2-0)+(2-0)+(2-0)=8$ while the table in Figure 3.8(b) has an information loss of $(77-51)+(26-0)+(26-0)+(26-0)=128$. ♦

So while definitions I and II say that the tables have an equal loss of information, definition III says that the difference in information loss is large. Definition IV could easily provide a third view on the information loss difference, if for instance, the total number of contributors to Figure 3.8(a) is twelve and Figure 3.8(b) has eight contributors. Then we have different definitions saying that the information loss is either equal, a lot larger for Figure 3.8(a), or a little larger for Figure 3.8(b).

Turning to the models for the CSP derived in Sections 3.2 and 3.3 we need to investigate what kind of information loss the objective functions (3.17a) and (3.27a) can represent. In the case of complete cell suppression the objective function can measure the information loss as in I, II, or IV by setting the weights w_i to 1, the corresponding cell value a_i, or the number of contributors to cell i, respectively. However, since all variables are binary, the objective function cannot represent the sum of the widths of the suppression intervals and therefore the complete CSP cannot measure information loss

as in III. On the other hand, using partial cell suppression we can measure information loss according to definition III (by setting all weights w_i^+ and w_i^- in (3.27a) to 1) but not according to I, II, and IV. One way to interpret this difference of the objective functions is that the complete CSP has a 'yes' or 'no' answer to whether a cell is suppressed, while the partial CSP tells us "how suppressed" a cell is (in terms of the interval widths).

The definition of information loss is important and the models define it differently. So, which one of them is the best? What properties characterize a good solution to the CSP? The answers to both questions are non-trivial. Too many suppressed cells is not desirable since this will lower the practical usefulness of the data, and for the same reason the suppression intervals should not be too large. The complete CSP does not take the width of the intervals into account while the partial CSP cannot take the number of suppressions into account. This does indeed lead to large suppression intervals in complete cell suppression and to a large number of suppressions in partial cell suppression, as we shall see in Chapter 5. So, none of them fulfill our demands, wherefore we next formulate the model proposed in [10], which combines the complete and partial cell suppression approaches.

3.4.2 The combined CSP

As mentioned above, the idea behind combined cell suppression is to construct a model with an objective function that can take both suppression interval widths and number of suppressions into account. This is done by summing the objective functions of the complete and the partial CSP, according to

$$\min \quad \sum_{i=1}^{N}(w_i^+ z_i^+ + w_i^- z_i^- + w_i x_i).$$

The combined model is similar to the partial CSP but has constraints that link the binary variables to the continuous ones. The subproblems are the same as in partial cell suppression. We end up with the following model, referred to as the combined CSP

$$\min_{z^+,\, z^-,\, x} \quad \sum_{i=1}^{N}(w_i^+ z_i^+ + w_i^- z_i^- + w_i x_i), \tag{3.28a}$$

$$\text{s.t.} \quad P_s^{\mathrm{U}} \le z_s^+ \le (u_s - a_s)x_s, \qquad s \in I_{\mathrm{S}}, \tag{3.28b}$$

$$0 \le z_i^+ \le (u_i - a_i)x_i, \qquad i \in I \setminus I_{\mathrm{S}}, \tag{3.28c}$$

$$P_s^{\mathrm{L}} \le z_s^- \le (a_s - l_s)x_s, \qquad s \in I_{\mathrm{S}}, \tag{3.28d}$$

$$0 \le z_i^- \le (a_i - l_i)x_i, \qquad i \in I \setminus I_{\mathrm{S}}, \tag{3.28e}$$

$$z_s^+ + z_s^- \ge P_s^{\mathrm{SL}}, \qquad s \in I_{\mathrm{S}}, \tag{3.28f}$$

$$\begin{cases} \sum_{i=1}^{N}(\alpha_i z_i^+ + \beta_i z_i^-) - z_s^+ \ge 0 \text{ for all} \\ \text{extreme points } (\gamma,\, \alpha,\, \beta) \text{ of the set defined by (3.23b–c)} \end{cases} \qquad s \in I_{\mathrm{S}}, \tag{3.28g}$$

$$\begin{cases} \sum_{i=1}^{N}(\widetilde{\alpha}_i z_i^+ + \widetilde{\beta}_i z_i^-) - z_s^- \ge 0 \text{ for all} \\ \text{extreme points } (\widetilde{\gamma},\, \widetilde{\alpha},\, \widetilde{\beta}) \text{ of the set defined by (3.25b–c)} \end{cases} \qquad s \in I_{\mathrm{S}}, \tag{3.28h}$$

$$x \in \{0,1\}^N. \tag{3.28i}$$

As a final remark: To the complete and the combined CSP it is possible to add a constraint $\sum_{i=1}^{N} x_i \leq S$ meaning, that a maximum of S suppressions are allowed. Also, to the combined CSP a constraint $z_i^+ + z_i^- \geq W_i x_i$ can be added in order to impose a minumum width $W_i > 0$ for the interval of every suppressed cell. Both of these constraints are of limited use since they require prior knowledge of what suppressions, and how many, are necessary to protect a table, and generally this is not known.

Chapter 4

Algorithms

In this chapter algorithms for solving the complete, partial and combined CSP are presented. They are all based on the cut-and-branch procedure first described in [22]. As shown in the previous chapter, each model contains sets of constraints which are composed of one constraint per extreme point of the feasible set of the dual of an attacker subproblem. Since the number of extreme points may be huge for large CSP instances, it would be impractical to include all these constraints in the model. The basic idea for all the algorithms presented is to decompose the CSP into two subproblems for each sensitive cell, and one master problem. The master problem is initially either of the models derived in the Chapter 3, but with the complicating constraints removed, which leaves (3.17a,e), (3.27a–f), and (3.28a–f,i). The subproblems are precisely the attacker subproblems used when deriving the different models. These will be used to generate constraints that are necessary in an optimal solution to the CSP, which are added to the master problem as needed during the course of the algorithm. The full details of each algorithm are given in the following sections but in order to help the understanding a general overview is given below.

First the master problem is initialized and solved without any complicating constraints. This solution will almost certainly not be a feasible solution to the CSP, since all constraints that impose the protection level requirements have been removed. Then the attacker subproblems are solved for each sensitive cell and every upper, lower, and sliding protection level is checked. A protection level that is not fulfilled, together with the dual solution to the corresponding subproblem, induces a constraint that is violated by the current solution to the master problem. To exemplify, in complete cell suppression we would solve the maximization problem (3.7) for a sensitive cell s and check if $\overline{y}_s \geq a_s + P_s^{\mathrm{U}}$. If this does not hold, then a constraint in (3.17b), with the dual solution to the subproblem inserted, is a valid constraint that is violated by the current solution to the master problem. After the protection levels are checked for all sensitive cells, the induced constraints are added to the master problem, which is then resolved, starting a new iteration of the algorithm. The algorithm proceeds until the master problem yields a solution for which all protection level requirements are fulfilled. The procedure is illustrated in Figure 4.1.

The next section presents the full algorithm for the complete CSP in more detail. Since the algorithms for all three models are essentially the same, the reasoning therein holds for the partial and the combined CSP algorithms as well.

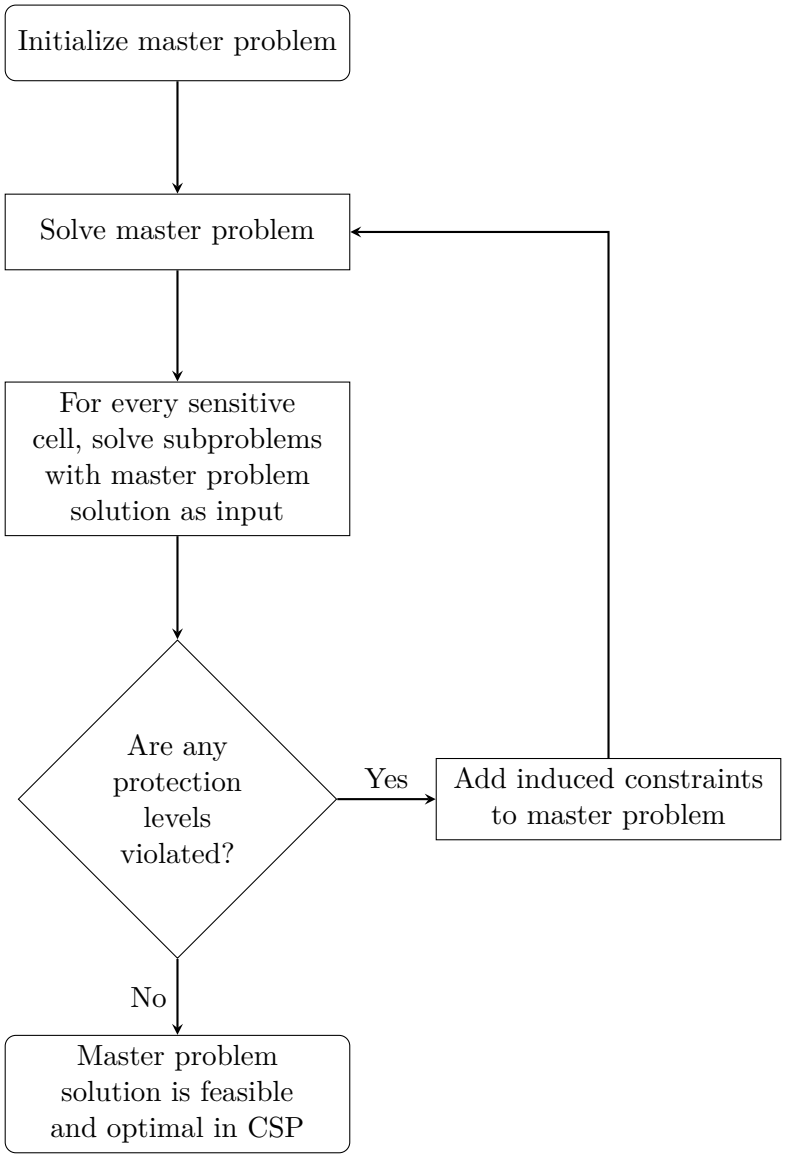

Figure 4.1: Flowchart of the general algorithm used to solve the CSP.

4.1 Algorithm for the complete CSP

In the case of the complete CSP (3.17) the initial master problem is given by

$$\min_{x} \quad \sum_{i=1}^{N} w_i x_i, \tag{4.1a}$$

$$\text{s.t.} \quad x_s = 1, \qquad s \in I_{\mathrm{S}}, \tag{4.1b}$$

$$x \in \{0,1\}^N, \tag{4.1c}$$

i.e., the model (3.17a,e) with the additional constraint (4.1b), which says that all sensitive cells must be suppressed. This constraint is implicit in (3.17) but is added here in order to give the algorithm a non-trivial starting point. Let $\widehat{x}$ be the solution to (4.1). The subproblems, two for each sensitive cell $s \in I_{\mathrm{S}}$, are given by

$$\max_{y} \quad y_s, \tag{4.2a}$$

$$\text{s.t.} \quad My = b, \tag{4.2b}$$

$$y_i \le a_i + (u_i - a_i)\widehat{x}_i, \qquad i \in I, \tag{4.2c}$$

$$-y_i \le -a_i + (a_i - l_i)\widehat{x}_i, \qquad i \in I, \tag{4.2d}$$

and

$$\max_{y} \quad -y_s, \tag{4.3a}$$

$$\text{s.t.} \quad My = b, \tag{4.3b}$$

$$y_i \le a_i + (u_i - a_i)\widehat{x}_i, \qquad i \in I, \tag{4.3c}$$

$$-y_i \le -a_i + (a_i - l_i)\widehat{x}_i, \qquad i \in I. \tag{4.3d}$$

with optimal objective values $\overline{y}_s$ and $-\underline{y}_s$, respectively. Let $(\widehat{\gamma}^s,\ \widehat{\alpha}^s,\ \widehat{\beta}^s)$ and $(\widehat{\widehat{\gamma}}^s,\ \widehat{\widehat{\alpha}}^s,\ \widehat{\widehat{\beta}}^s)$ denote the solutions to the duals of (4.2) and (4.3), respectively. If some protection level is not fulfilled for $\widehat{x}$, meaning that there is one or more $s \in I_{\mathrm{S}}$ for which at least one of the constraints $\overline{y}_s \ge a_s + P_s^{\mathrm{U}}$, $\underline{y}_s \le a_s - P_s^{\mathrm{L}}$, and $\overline{y}_s - \underline{y}_s \ge P_s^{\mathrm{SL}}$ does not hold, $\widehat{x}$ does not yield a safe table and is therefore not a feasible solution to the complete CSP (3.17). In this case, the dual solutions to the subproblems are used to construct constraints of the form (3.17b), (3.17c), or (3.17d) that are then added to the master problem. By construction, these constraints are equivalent to demanding that a solution to the complete CSP yields a safe table. This ensures that if the solution to the master problem is not safe, constraints will be added that make the current solution to the master problem infeasible, which guarantees that the master problem in the next iteration will have a different solution. Note that the subproblems are always feasible and have finite optimal solutions, since all variables are subject to upper and lower bounds and since the original table is always a feasible solution.

It has been shown (see Appendix A) that it is sufficient to consider the set of extreme points of the dual feasible set of the subproblems in the constraints (3.17b)–(3.17d). This means that only a finite number of constraints are necessary to generate and that the algorithm will terminate finitely, presuming that the solutions found to the subproblems

are extreme points to the respective dual feasible sets. This will be the case, e.g., if the simplex algorithm is used.

The algorithm terminates when no protection level requirements are violated by the solution to the master problem. This solution is then both feasible and optimal in the complete CSP. It is feasible since the constraints generated guarantee that the resulting suppressions yield a safe table (all sensitive cells are protected). It is optimal since in each iteration the master problem is a relaxation of the complete CSP (some constraints are not included), and once an optimal solution to a relaxed problem is feasible in the full problem it must also be optimal in the full problem.

Algorithm 4.1 solves (3.17).

Algorithm 4.1.

Step 0: Initialize the master problem (4.1).

Step 1: Solve the master problem to obtain $\widehat{x}$.

Step 2: For each sensitive cell $s \in I_{\mathrm{S}}$, solve the subproblems (4.2) and (4.3) with $\widehat{x}$ as input in order to obtain $\overline{y}_s$, $\underline{y}_s$, $(\widehat{\gamma}^s,\ \widehat{\alpha}^s,\ \widehat{\beta}^s)$, and $(\widehat{\widetilde{\gamma}}^s,\ \widehat{\widetilde{\alpha}}^s,\ \widehat{\widetilde{\beta}}^s)$.

Step 3: For each $s \in I_{\mathrm{S}}$:

- **a.** If $\overline{y}_s < a_s + P_s^{\mathrm{U}}$, add the induced constraint, (3.17b) with $\alpha = \widehat{\alpha}^s$ and $\beta = \widehat{\beta}^s$, to the master problem.
- **b.** If $\underline{y}_s > a_s - P_s^{\mathrm{L}}$, add the induced constraint, (3.17c) with $\widetilde{\alpha} = \widehat{\widetilde{\alpha}}^s$ and $\widetilde{\beta} = \widehat{\widetilde{\beta}}^s$, to the master problem.
- **c.** If $\overline{y}_s - \underline{y}_s < P_s^{\mathrm{SL}}$, add the induced constraint, (3.17d) with $\alpha = \widehat{\alpha}^s$, $\beta = \widehat{\beta}^s$, $\widetilde{\alpha} = \widehat{\widetilde{\alpha}}^s$, and $\widetilde{\beta} = \widehat{\widetilde{\beta}}^s$, to the master problem.

If no constraints were added in steps **3a**, **3b**, or **3c**, terminate since $\widehat{x}$ is a safe, and thus an optimal, solution to (3.17). Otherwise, go to **Step 1**.

We make two remarks on this algorithm, that are also applicable in the case of partial and combined cell suppression.

First, in each iteration the master problem has to be solved to optimality, if the final solution should be guaranteed to be optimal in the CSP. To see why, consider the case where the master problem is *not* solved to optimality. If the given, suboptimal, solution to the master problem is to suppress every cell in the table, this solution will obviously yield a safe table (assuming that a feasible solution to the CSP exists), so the algorithm will terminate. However, this solution is almost certainly not optimal in the CSP.

Secondly, if the master problem is solved to optimality in each iteration, then before the algorithm terminates not a single feasible solution to the CSP is produced, since the master problems that are solved are relaxations of the CSP. Thus, the algorithm cannot be aborted prematurely with the expectation that a solution to the master problem will be feasible but suboptimal in the CSP. This could be considered a drawback, since one might be more interested in quickly finding a feasible solution rather than finding an optimal one. Also, for large problem instances the algorithm might not be able to

find the optimal solution within a reasonable amount of time. For this reason it would be desirable to have a heuristic that could build a feasible solution to the CSP from a solution to the master problem. In [9] a heuristic that fills this purpose is presented but it has not been implemented in the context of this thesis.

4.1.1 Strengthened constraints

It is possible to strengthen the constraints (3.17b)–(3.17d). Defining the constants $d_i \geq 0$ and $D \geq 0$ appropriately, these constraints can all be expressed as

$$\sum_{i=1}^{N} d_i x_i \geq D. \tag{4.4}$$

As pointed out in [9], because the coefficients d_i are non-negative and the variables x_i are binary, if (4.4) holds then

$$\sum_{i=1}^{N} \min\{d_i, D\} x_i \geq D \tag{4.5}$$

will also hold. The proof is simple. Let $x \in \{0,1\}^N$ satisfy (4.4) and let $T = \{i \in \{1,\ldots,N\} : d_i > D\}$. If $x_i = 0$ for all $i \in T$, then it holds that

$$\sum_{i=1}^{N} \min\{d_i, D\} x_i = \sum_{i=1}^{N} d_i x_i \geq D.$$

Otherwise, there is at least one $j \in T$ such that $x_j = 1$ and we have

$$\sum_{i=1}^{N} \min\{d_i, D\} x_i \geq D x_j = D.$$

For binary variables x_i, (4.4) and (4.5) are equivalent, but (4.5) is stronger when the integrality is relaxed, in the sense that (4.5) cuts off more of the feasible set of the relaxation than (4.4) does. This is of interest since the integrality of x_i is relaxed in the solution process of the master problem (4.1). Note that this strengthening is only possible for the complete CSP and not for the partial or combined CSP.

4.2 Algorithm for the partial CSP

For the partial CSP (3.27) the initial master problem is given by

$$\min_{z^+,\, z^-} \quad \sum_{i=1}^{N}(w_i^+ z_i^+ + w_i^- z_i^-), \tag{4.6a}$$

$$\text{s.t.} \quad P_s^{\mathrm{U}} \le z_s^+ \le u_s - a_s \qquad s \in I_{\mathrm{S}}, \tag{4.6b}$$

$$0 \le z_i^+ \le u_i - a_i \qquad i \in I \setminus I_{\mathrm{S}}, \tag{4.6c}$$

$$P_s^{\mathrm{L}} \le z_s^- \le a_s - l_s \qquad s \in I_{\mathrm{S}}, \tag{4.6d}$$

$$0 \le z_i^- \le a_i - l_i \qquad i \in I \setminus I_{\mathrm{S}}, \tag{4.6e}$$

$$z_s^+ + z_s^- \ge P_s^{\mathrm{SL}} \qquad s \in I_{\mathrm{S}}, \tag{4.6f}$$

with solution $(\widehat{z}^+,\ \widehat{z}^-)$. The two subproblems are given by

$$\max_{y} \quad y_s, \tag{4.7a}$$

$$\text{s.t.} \quad My = b, \tag{4.7b}$$

$$y_i \le a_i + \widehat{z}_i^+ \qquad i \in I, \tag{4.7c}$$

$$-y_i \le -a_i + \widehat{z}_i^- \qquad i \in I, \tag{4.7d}$$

and

$$\max_{y} \quad -y_s, \tag{4.8a}$$

$$\text{s.t.} \quad My = b, \tag{4.8b}$$

$$y_i \le a_i + \widehat{z}_i^+ \qquad i \in I, \tag{4.8c}$$

$$-y_i \le -a_i + \widehat{z}_i^- \qquad i \in I, \tag{4.8d}$$

with optimal objective values $\overline{y}_s$ and $-\underline{y}_s$ and dual solutions $(\widehat{\gamma}^s,\ \widehat{\alpha}^s,\ \widehat{\beta}^s)$ and $(\widehat{\widetilde{\gamma}}^s,\ \widehat{\widetilde{\alpha}}^s,\ \widehat{\widetilde{\beta}}^s)$, respectively. Algorithm 4.2 solves (3.27).

Algorithm 4.2.

Step 0: Initialize the master problem (4.6).

Step 1: Solve the master problem to obtain $(\widehat{z}^+,\ \widehat{z}^-)$.

Step 2: For each sensitive cell $s \in I_{\mathrm{S}}$, solve the subproblems (4.7) and (4.8) with $(\widehat{z}^+,\ \widehat{z}^-)$ as input in order to obtain $\overline{y}_s$, $\underline{y}_s$, $(\widehat{\gamma}^s,\ \widehat{\alpha}^s,\ \widehat{\beta}^s)$, and $(\widehat{\widetilde{\gamma}}^s,\ \widehat{\widetilde{\alpha}}^s,\ \widehat{\widetilde{\beta}}^s)$.

Step 3: For each $s \in I_{\mathrm{S}}$:

- **a.** If $\overline{y}_s < a_s + \widehat{z}_s^+$, add the induced constraint, (3.27g) with $\alpha = \widehat{\alpha}^s$ and $\beta = \widehat{\beta}^s$, to the master problem.
- **b.** If $\underline{y}_s > a_s - \widehat{z}_s^-$, add the induced constraint, (3.27h) with $\widetilde{\alpha} = \widehat{\widetilde{\alpha}}^s$ and $\widetilde{\beta} = \widehat{\widetilde{\beta}}^s$, to the master problem.

If no constraints were added in steps **3a** or **3b**, terminate since $(\widehat{z}^+,\ \widehat{z}^-)$ is a safe, and thus an optimal, solution to (3.27). Otherwise, go to **Step 1**.

4.3 Algorithm for the combined CSP

Finally, for the combined CSP (3.28) we have the initial master problem

$$\min_{z^+,\, z^-,\, x} \quad \sum_{i=1}^{N}(w_i^+ z_i^+ + w_i^- z_i^- + w_i x_i), \tag{4.9a}$$

$$\text{s.t.} \quad P_s^{\mathrm{U}} \le z_s^+ \le (u_s - a_s)x_s \qquad s \in I_{\mathrm{S}}, \tag{4.9b}$$

$$0 \le z_i^+ \le (u_i - a_i)x_i \qquad i \in I \setminus I_{\mathrm{S}}, \tag{4.9c}$$

$$P_s^{\mathrm{L}} \le z_s^- \le (a_s - l_s)x_s \qquad s \in I_{\mathrm{S}}, \tag{4.9d}$$

$$0 \le z_i^- \le (a_i - l_i)x_i \qquad i \in I \setminus I_{\mathrm{S}}, \tag{4.9e}$$

$$z_s^+ + z_s^- \ge P_s^{\mathrm{SL}} \qquad s \in I_{\mathrm{S}}, \tag{4.9f}$$

$$x_s = 1 \qquad s \in I_{\mathrm{S}}, \tag{4.9g}$$

$$x \in \{0,1\}^N. \tag{4.9h}$$

with solution $(\widehat{z}^+,\ \widehat{z}^-,\ \widehat{x})$. The subproblems are identical to the subproblems in partial cell suppression, (4.7) and (4.8). Algorithm 4.3 solves (3.28).

Algorithm 4.3.

Step 0: Initialize the master problem (4.9).

Step 1: Solve the master problem to obtain $(\widehat{z}^+,\ \widehat{z}^-,\ \widehat{x})$.

Step 2: For each sensitive cell $s \in I_{\mathrm{S}}$, solve the subproblems (4.7) and (4.8) with $(\widehat{z}^+,\ \widehat{z}^-)$ as input in order to obtain $\overline{y}_s$, $\underline{y}_s$, $(\widehat{\gamma}^s,\ \widehat{\alpha}^s,\ \widehat{\beta}^s)$, and $(\widehat{\widehat{\gamma}}^s,\ \widehat{\widehat{\alpha}}^s,\ \widehat{\widehat{\beta}}^s)$.

Step 3: For each $s \in I_{\mathrm{S}}$:

- **a.** If $\overline{y}_s < a_s + \widehat{z}_s^+$, add the induced constraint, (3.27g) with $\alpha = \widehat{\alpha}^s$ and $\beta = \widehat{\beta}^s$, to the master problem.
- **b.** If $\underline{y}_s > a_s - \widehat{z}_s^-$, add the induced constraint, (3.27h) with $\widetilde{\alpha} = \widehat{\widehat{\alpha}}^s$ and $\widetilde{\beta} = \widehat{\widehat{\beta}}^s$, to the master problem.

If no constraints were added in steps **3a** or **3b**, terminate since $(\widehat{z}^+,\ \widehat{z}^-,\ \widehat{x})$ is a safe, and thus an optimal, solution to (3.28). Otherwise, go to **Step 1**.

4.4 A note on problem complexity

A noteworthy difference between the three formulations (3.17), (3.27), and (3.28) is that the complete CSP is an integer linear program (ILP), the partial CSP is a continuous linear program (LP) and the combined CSP is a mixed integer linear program (MILP). It is well known that LPs can be solved in polynomial time using, e.g., an interior point method (see [17]). This means that the partial CSP, in contrast to the complete and the combined CSP, belongs to a class of problems that are in some sense "easy". The complete CSP has been shown to belong to the class of NP-hard problems (see [15] or [16]) and it is not known if there exists a polynomial time algorithm for this type of problems. The combined CSP is at least as hard (meaning it is also NP-hard), in the sense that the complete CSP can be viewed as a special case of the combined CSP. The combined CSP is reduced to the complete CSP by removing the continuous variables from the objective function (setting $w_i^+ = w_i^- = 0$) and interpreting any z_i^+ or z_i^- that are greater than 0 to mean that the cell should be completely suppressed (that is, replaced by some arbitrary symbol).

Chapter 5

Tests and results

This chapter presents results for the three models and a number of test instances, using both the commercial solver CPLEX and the open source solver GLPK. The results will be used to compare solutions to the models, and to compare the performance of GLPK and CPLEX for solving the CSP. First, a breakdown of the problem instance details and the chosen parameter values are given in Section 5.1, followed by the computational results regarding the solutions to the complete, partial, and combined CSP in Section 5.2. The solutions are compared according to two criteria: (i) the number of suppressions and (ii) the sum of the widths of the suppression intervals. In Section 5.3 computation times are presented. We first compare the computation times for the different models and then the performance of GLPK is evaluated.

The algorithms used to solve the complete, partial, and combined CSP, as presented in Chapter 4, were implemented in C++ and the optimization solvers used were CPLEX 12.1 and GLPK 4.55. The simplex algorithm was used for all LP problems that arose in the solution process and the strengthened constraints derived in Section 4.1.1 were used when solving the complete CSP. Tests were run under Red Hat Enterprise Linux 6.6 on a PC with Intel Core i5-4570S and 16GB RAM.

5.1 Problem instances and parameters

The seven tables used for testing are all based on data obtained from SCB. The data are synthetic, created to possess properties similar to those of real data. The tables are magnitude tables in which the response variable is the turnover of companies. There are two linked tables, each consisting of three hierarchical subtables. The other five tables are non-linked and hierarchical. The subtables of the two linked tables, as well as the non-linked tables, are two-dimensional. We remark that none of the tables contain empty cells, i.e., cells with value zero. For more detailed information about the problem instances, see Table 5.1, which presents the total number of cells, the number of sensitive cells, and the number of rows in the matrix M. The instances 1B and 2C are subtables from the linked tables 1LINKED and 2LINKED, respectively. The instances 1ALT, 2ALT, and 3ALT are based on the same data as 1LINKED and 2LINKED but structured differently from these two.

Whether a cell is sensitive or not has been determined using the $p\%$ rule with $p = 10$. In the complete CSP, the protection levels P_s^{L} and P_s^{U} are both given by (3.3) with $p = 10$

Instance	No. of cells	No. of sensitive cells	No. of rows in M
1LINKED	222+597+359	2+46+32	653
1B	703	47	356
2C	401	61	203
3ALT	778	71	299
2LINKED	402+636+364	45+96+60	663
1ALT	1428	124	925
2ALT	1996	435	1184

Table 5.1: Data for the seven problem instances used to test and compare the three cell suppression models (each term for 1LINKED and 2LINKED corresponds to one subtable).

and $q = 100$, and P_s^{SL} are all 0. In the partial and combined CSP, both P_s^{L} and P_s^{U} are set to 0, while P_s^{SL} is set to two times the value given by (3.3) (again with $p = 10$ and $q = 100$). The protection levels were chosen in this way for the partial and combined CSP, since if only lower and upper protection levels are used in these models it will often be the case that the resulting intervals are symmetric with the actual cell value in the middle. The external bounds were chosen as $l_i = 0$ and $u_i = 11a_i$, meaning that the variables y_i in the subproblems are bounded by zero and eleven times the actual cell value.

For the complete and the partial CSP, two different objective functions are tested, called *Unity* and *Value*. Unity means that all weights in the objective function are set to 1: $w_i = 1$ in the complete CSP and $w_i^+ = w_i^- = 1$ in the partial CSP. In the complete CSP this means that the number of suppressions is minimized and in the partial CSP that the sum of the suppression interval widths is minimzed. In the other objective function, Value, each weight is set to the corresponding cell value: $w_i = a_i$ in the complete CSP and $w_i^+ = w_i^- = a_i$ in the partial CSP. This choice favors the suppression of cells with small values. The model variants for the complete and partial CSP are referred to as *Complete-Unity*, *Complete-Value*, *Partial-Unity*, and *Partial-Value*.

In the combined CSP, these two variants of objective functions are combined in two ways. One is referred to as *Unity-Unity*, meaning that the weights of both the continuous and binary variables are set to 1: $w_i = w_i^+ = w_i^- = 1$. The other is referred to as *Unity-Value*, where the weights on the continuous variables are set to 1 and the weights on the binary variables are set to the corresponding cell values: $w_i^+ = w_i^- = 1$ and $w_i = a_i$. The model variants for the combined CSP are referred to as *Combined-Unity-Unity* and *Combined-Unity-Value*.

In Sections 5.2 and 5.3.1 we compare the solution quality and computation times, respectively, of the six model variants for the seven problem instances. This comparison is made with solutions obtained using CPLEX. For the complete and partial CSP, optimal solutions are presented for all instances and objective functions. For the combined CSP, the computation time required to obtain feasible solutions proved excessive for some problem instances. For this reason, no feasible solutions were obtained for 2LINKED, 1ALT, or 2ALT using combined cell suppression. Instances 1LINKED, 1B, 2C, and 3ALT were solved to optimality with Combined-Unity-Value. The same four instances were solved with Combined-Unity-Unity but, on account of the large amount of time required

to find optimal solutions to the master problem (initially (4.9)), a non-zero relative mixed integer programming (MIP) gap tolerance was used when solving the master problem in each iteration of Algorithm 4.3. Recall that the master problem is a MILP and therefore a branch and bound procedure is necessary to solve this problem. In CPLEX the relative MIP gap is defined as |bestnode − bestinteger|/|1e-10+bestinteger| (see [14, p. 92]). For the combined CSP master problem, which is a minimization problem, 'bestinteger' is the lowest objective value of any solution found so far in the branch and bound tree that fulfills the integer requirements (4.9h) and 'bestnode' is the highest lower bound for the objective value, verified over all branches of the tree. The branch and bound procedure terminates when the relative MIP gap is lower than the tolerance, which, in Combined-Unity-Unity, was set to $2 \cdot 10^{-5}$ for 1LINKED, $5 \cdot 10^{-5}$ for 1B, and 10^{-6} for 2C and 3ALT.

5.2 Solution quality

This section presents and discusses the quality of the solutions, measured by the number of suppressions and the sum of the suppression interval widths. Detailed data for Figures 5.1–5.6 are given in Appendix B, Tables B.1–B.5. These data were obtained using CPLEX.

First, Figure 5.1 yields the total number of suppressions, primary and secondary, of each model variant for each instance, normalized by the minimum number required (as obtained by Complete-Unity). In partial and combined cell suppression every cell that is replaced by an interval is counted as one suppression. It is rather well known, and pointed out by Fischetti and Salazar-González in [10], that the partial CSP tends to suppress more cells than the complete CSP. Our results are in line with that notion.

For each instance, Complete-Unity gives the smallest number of suppressions, which is inevitable since the objective function in this model is defined to minimize the number of suppressions. Both Partial-Unity and Partial-Value yield more suppressions than Complete-Unity and Complete-Value. The reason for this is, as discussed in Section 3.4.1, that the partial CSP does not take into account how many cells are suppressed. Partial-Value suppressed fewer cells than Partial-Unity for all instances except 3ALT and 2LINKED. When cells have different weights in the partial CSP (as in Partial-Value) the intervals that are necessary in order to protect the sensitive cells will be concentrated to the cells with small weights. In Partial-Unity, all cells have the same weights and the intervals do not concentrate to any specific cells. This can lead to a larger number of suppressions in Partial-Unity than in Partial-Value as is the case in 1LINKED, 1B, 2C, 1ALT, and 2ALT. However, it can also be the case that Partial-Value needs to suppress a large number of cells with small values in order to produce a safe table when Partial-Unity can instead suppress a small number of cells with larger values, as in 3ALT and 2LINKED.

The combined CSP variants, Combined-Unity-Unity and Combined-Unity-Value, are both similar to Partial-Unity in that the weights on all the continuous variables equal 1. The difference is that in Combined-Unity-Unity and Combined-Unity-Value, each suppression inflicts a penalty in the objective function, owing to the binary variables representing whether cells are suppressed or not. It is clear from Figure 5.1 that this results in a smaller number of suppressions for the combined CSP variants than for the partial CSP

variants. The exceptions are instances 1LINKED and 1B for Combined-Unity-Unity, due to the fact that these were not solved to optimality. Note that 2C and 3ALT were also not solved to optimality for Combined-Unity-Unity, but the relative MIP gap tolerance in the master problem was smaller than in 1LINKED and 1B (10^{-6} compared to $2 \cdot 10^{-5}$ and $5 \cdot 10^{-5}$, respectively). How optimal solutions to Combined-Unity-Unity would compare to the other model variants in terms of number of suppressions is not clear. What can be said for our instances is that, since the optimal solutions to Partial-Unity and the suboptimal solutions to Combined-Unity-Unity have equal sums of the suppression interval widths, as seen in Table B.2, and since a set of intervals (i.e., z_i^+ and z_i^-) that are feasible in Partial-Unity are also feasible in Combined-Unity-Unity, optimal solutions to Combined-Unity-Unity would not have more suppressions than Partial-Unity.

Figure 5.2 presents the sum of the widths of all suppression intervals for each model variant and instance, normalized by the minimum width required (as obtained by Partial-Unity). Obviously, the sum of the suppression interval widths for both variants of the complete CSP is much larger than that of each variant of the partial and the combined CSP. This is certainly to be expected since the complete CSP model includes no concept of suppression interval width and hence cannot take it into account. Even so, Complete-Value results in smaller widths than Complete-Unity, since suppressing cells with small values results in smaller suppression intervals than suppressing cells with large values. This can be explained by simply considering two non-negative numbers with unknown values but known sum. The intervals in which the unkown numbers may lie, under the restriction of preserving the sum, depend on the (known) sum of the two numbers, mean-

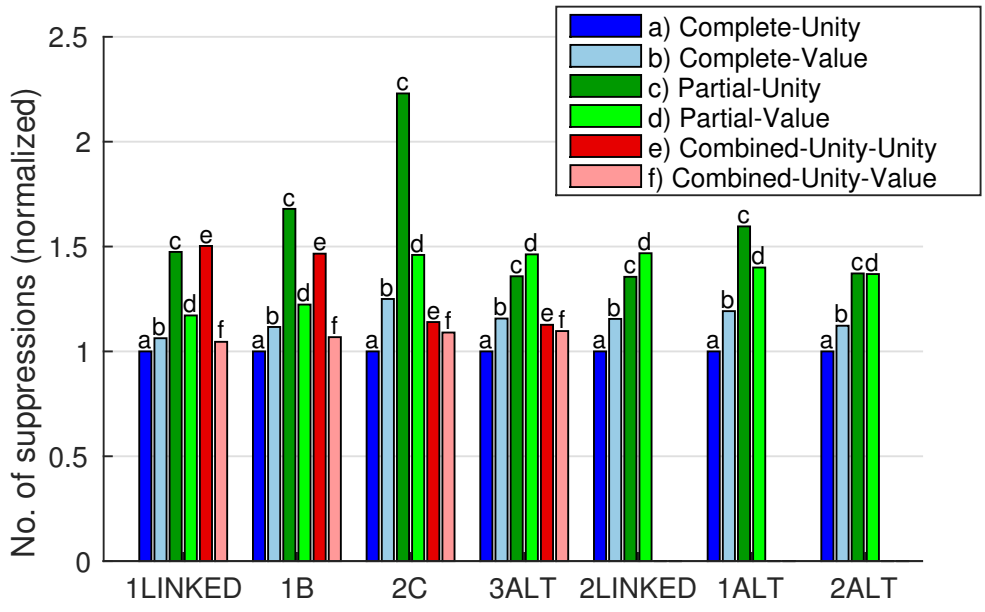

Figure 5.1: Number of suppressions obtained for each instance and model variant, normalized by the minimum number required for each instance (detailed data are found in Table B.1). 2LINKED, 1ALT, and 2ALT were not solved with Combined-Unity-Unity and Combined-Unity-Value.

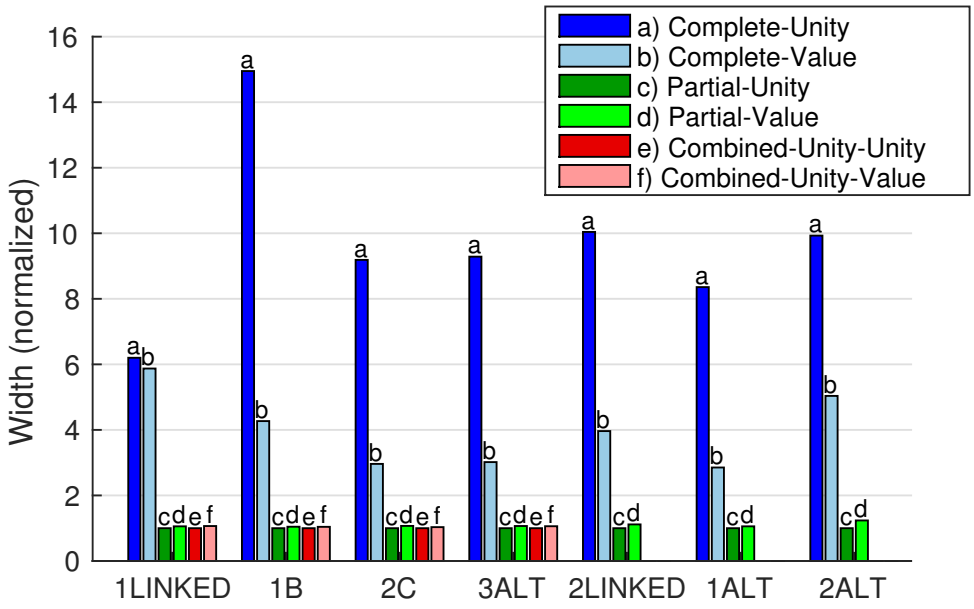

Figure 5.2: Sum of the suppression interval widths obtained for each instance and model variant, normalized by the minimum width required for each instance (detailed data are found in Table B.2).

ing that suppressing cells with large values introduces more uncertainty than suppressing cells with small values.

The sum of the suppression interval widths are very similar for Partial-Unity, Partial-Value, Combined-Unity-Unity and Combined-Unity-Value. The objective function in Partial-Unity is defined to minimize the width, wherefore this model results in the smallest width. However, it is the case that Combined-Unity-Unity has also minimized the width for each instance, which can be seen by comparing Partial-Unity and Combined-Unity-Unity in Table B.2.

It is clear from the results presented in this section that the principal undesirable characteristic of partial suppression, i.e., the numerous suppressions, can be alleviated by introducing the binary variables used in combined cell suppression. Also, partial cell suppression and combined cell suppression result in much smaller suppression intervals than complete cell suppression. If the number of suppressions and the sum of the suppression interval widths are deemed appropriate measures of solution quality, then combined cell suppression is superior. However, when discussing complete cell suppression, the significance of suppression interval widths must be interpreted with great care, for the following reason.

For all tests presented in this chapter we have assumed the external bounds $l_i = 0$ and $u_i = 11a_i$, meaning that $y_i \in [0, 11a_i]$. Still, it is possible that the exact same solutions may be found to the complete CSP if the upper bound u_i is altered (either decreased or increased). Suppose that there is a suppressed cell j and that y_j in some attacker subproblem attains its maximal value, i.e., $y_j = u_j$, according to (3.7c) or (3.8c). It might be the case that y_j could be allowed to take either a lower or a higher value

without affecting the solution to the complete CSP. So, if u_j is changed, the solution to the complete CSP could be the same but the suppression interval of cell j, $[\underline{y}_j, \overline{y}_j]$, would change. In this sense, identical solutions to the complete CSP can have different suppression interval widths depending on the chosen parameter values. Keeping this in mind, it still holds that the suppression intervals must be smaller in partial and combined cell suppression than in complete cell suppression, since in the partial and combined CSP the suppression intervals, effectively the values of the z_i^+ and z_i^- variables, are always chosen as small as possible in order for the resulting table to be safe.

5.3 Computation time

Having compared the solutions to all three models, for some possible objective functions, it is of course relevant how the computation times differ. First, the model variants are compared in Section 5.3.1. All solutions for this comparison were obtained with CPLEX. Then, the performance of GLPK is evaluated in Section 5.3.2.

5.3.1 CPLEX

For each of the seven instances and six model variants, Figures 5.3 and 5.4 show the CPU time and wall time, respectively, used by CPLEX. These figures clearly illustrate that the combined CSP is very computationally heavy. The four instances for which the combined CSP was solved, 1LINKED, 1B, 2C, and 3ALT, required huge computation times compared to the other models. This is especially pronounced for Combined-Unity-Unity even though it was not solved to optimality for any instance.

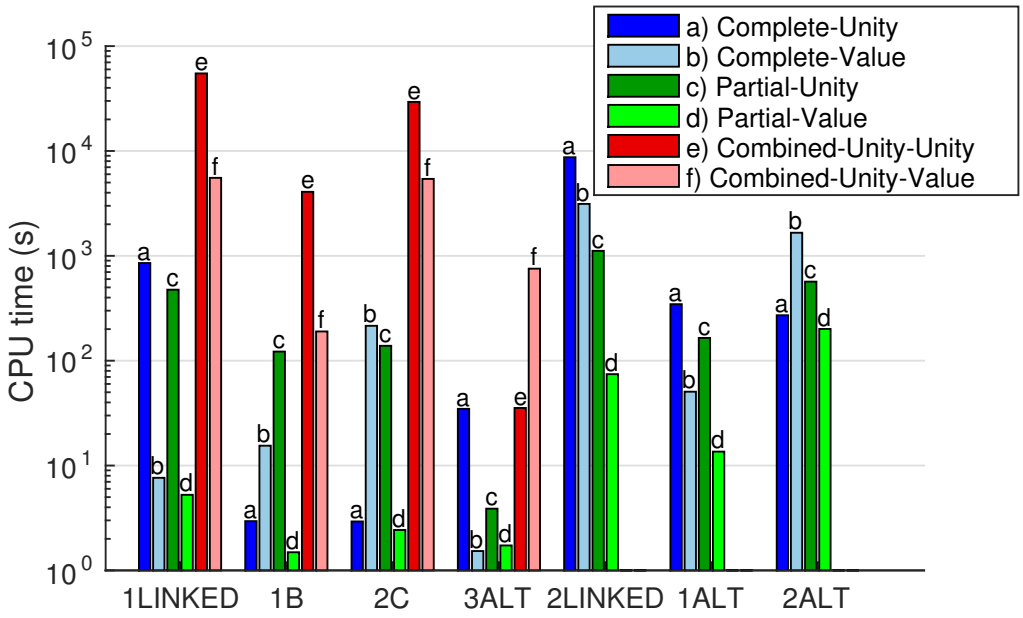

Figure 5.3: CPU time in seconds used by CPLEX to solve each instance and model variant (detailed data are found in Table B.3).

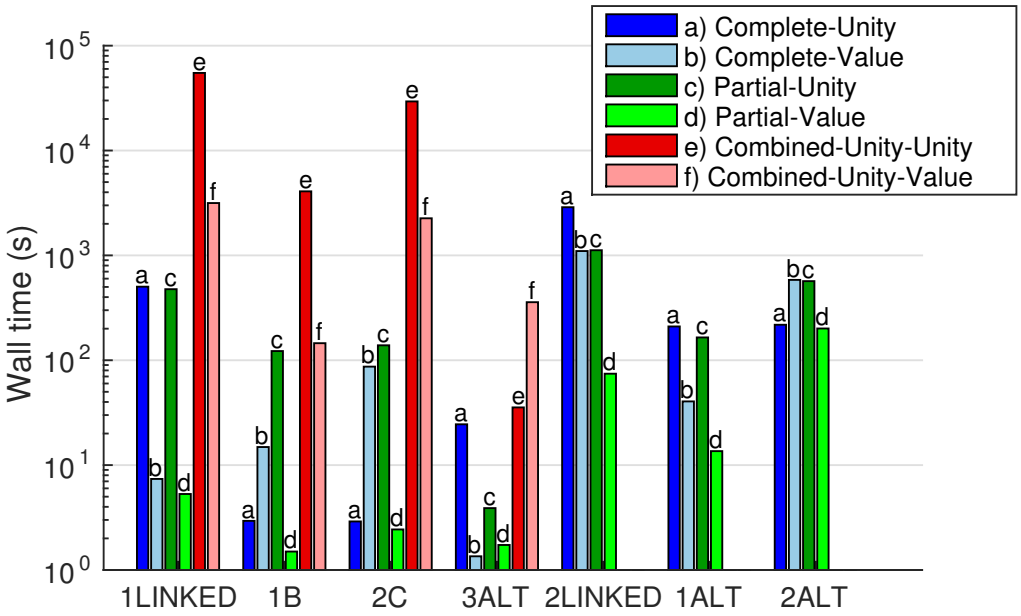

Figure 5.4: Wall time in seconds used by CPLEX to solve each instance and model variant (detailed data are found in Table B.4).

Comparing Figures 5.3 and 5.4 we see that for Complete-Unity and Complete-Value, the wall times are shorter than the CPU times, meaning that CPLEX can benefit from using multiple CPU cores in the complete CSP (for detailed timings, see Tables B.3 and B.4). Combined-Unity-Value also has this benefit but, intriguingly, Combined-Unity-Unity, Partial-Unity, and Partial-Value do not.

It is obvious that the relations between the computation times of the model variants depend on the specific problem instance. Measured both in CPU time and wall time, Complete-Unity is faster than Complete-Value for 1B, 2C, and 2ALT but considerably slower for 1ALT and the two linked tables, 1LINKED and 2LINKED. Partial-Value is the fastest for every instance except 3ALT where Complete-Value is less than 0.5 seconds faster (both CPU time and wall time). Particularly, for 2LINKED the difference between Partial-Value and the other models is immense (again, see Tables B.3 and B.4). It is noteworthy that Partial-Unity, which is an LP, for some instances requires as much, or more, computation time than either variant of the complete CSP, which is an ILP (compare with, e.g., Complete-Value for 1LINKED or Complete-Unity for 2C). This is partly explained by Figure 5.5, which presents the number of constraints generated in the subproblems for each model variant and instance. For Partial-Unity it is necessary to generate a significantly larger number of constraints than for Complete-Unity and Complete-Value before a solution to the master problem is feasible (and optimal) in the CSP. We point out that Partial-Value, which is also an LP, is faster than both Complete-Value (except for 3ALT) and Complete-Unity even though it, like Partial-Unity, generates more constraints. However, it is also the case that Partial-Unity generates significantly more constraints than Partial-Value, which may account for some of the difference in computation times for these two model variants. Compared to the

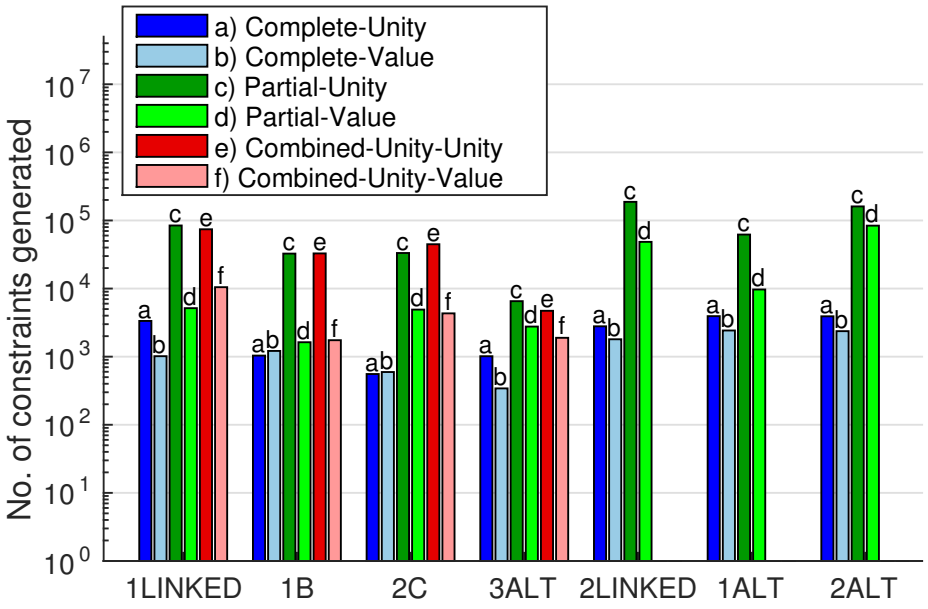

Figure 5.5: Total number of constraints generated in the subproblems for each instance and model variant using CPLEX (detailed data are found in Table B.5).

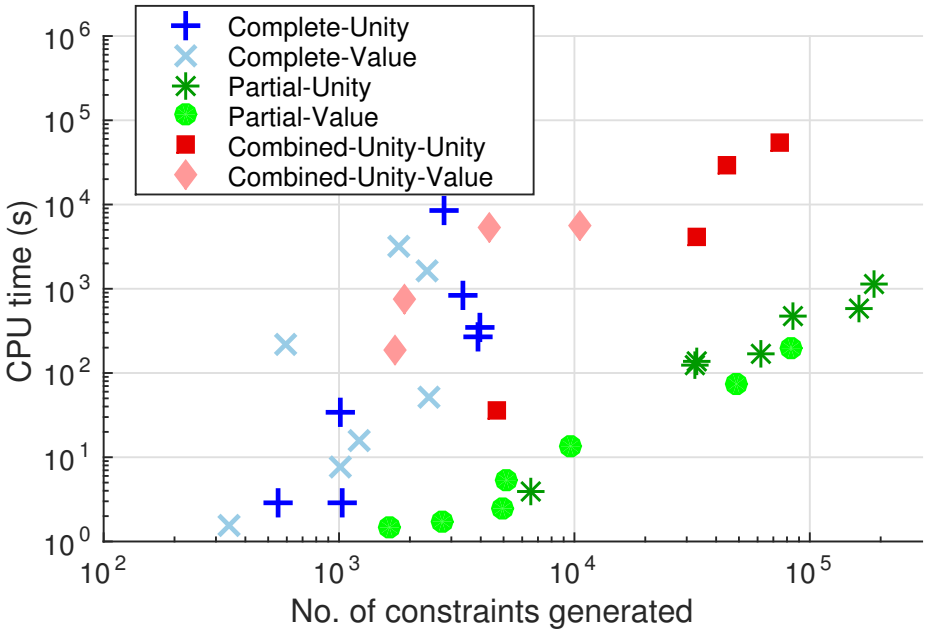

Figure 5.6: CPU time in seconds plotted against the number of constraints generated for each instance and model variant using CPLEX (detailed data are found in Tables B.3 and B.5).

complete CSP variants, the number of constraints generated by Combined-Unity-Unity and Combined-Unity-Value is very large as well.

Finally, Figure 5.6 presents the CPU time used plotted against the number of constraints generated in the subproblems for each instance and model variant. In this figure we see that, at least ostensibly, the computation times for the partial CSP are more correlated to the number of constraints generated than for the complete and combined CSP. We can also note that for an approximately equal number of generated constraints, the computation time is lower for the partial CSP than for the complete and the combined CSP.

It should be noted that during testing, it has been observed that the external bounds can have a rather large impact on the computation times. A higher upper bound often leads to a lower computation time for all model variants.

5.3.2 GLPK

Turning to the open source solver GLPK, the performance is naturally not on par with that of CPLEX and many test instances could not be solved within reasonable time. For the combined CSP, no instance was solved because after the first few iterations of the solution process the computation time per iteration increased drastically, even when a large relative MIP gap tolerance was used in the master problem. It seems likely that this is due to numerical difficulties introduced by the constraints (3.28b)–(3.28e), in which the coefficients on the x variables can be very large. For the partial CSP, GLPK sometimes ended in an infinite loop when solving the last iteration of the master problem even though the optimal solution had been found. This is probably a case of degeneracy cycling, which can cause the simplex algorithm to never terminate if the optimal solution is degenerate (see [1, pp. 243–244]).

GLPK was tested on the smaller instances 1LINKED, 1B, 2C, and 3ALT. Table 5.2 shows the computation times for the model variants that were solved, together with the corresponding times for CPLEX, and Table 5.3 shows the ratio between the computation times of the two solvers (times for GLPK divided by times for CPLEX). For the instances that were solved, GLPK was considerably slower than CPLEX but not necessarily prohibitively slow. Using Complete-Value, each instance was solved in five minutes or less, and using Complete-Unity, 1B and 2C were solved in less than seven and three minutes, respectively. Complete-Unity was too slow for 1LINKED and 3ALT and the computations were aborted. For the partial CSP variants, GLPK is less reliable. Partial-Unity was too slow for 1LINKED and the computation was aborted. Partial-Unity and Partial-Value ended with infinite loops solving 1B and 1LINKED, respectively.

Model variant	1LINKED	1B	2C	3ALT
Complete-Unity				
GLPK	n/a	384.93 | 385.08	160.59 | 160.72	n/a
CPLEX	853.73 | 503	2.95 | 2.94	2.93 | 2.90	34.74 | 24.5
Complete-Value				
GLPK	232.43 | 232.65	106.42 | 106.48	300.29 | 300.42	168.03 | 168.19
CPLEX	7.64 | 7.38	15.47 | 14.87	215.17 | 86.92	1.53 | 1.35
Partial-Unity				
GLPK	n/a	n/a	4037.73 | 4040.79	70.27 | 70.31
CPLEX	475.56 | 476	122.18 | 122.34	138.65 | 138.82	3.88 | 3.89
Partial-Value				
GLPK	n/a	5.42 | 5.42	1.94 | 1.94	5.59 | 5.60
CPLEX	5.26 | 5.30	1.49 | 1.50	2.43 | 2.43	1.73 | 1.73

Table 5.2: CPU time in seconds (left number) and wall time in seconds (right number) for four instances and all complete and partial cell suppression model variants. Computation times for the instances and model variants that were not solved are marked with n/a (not available).

Model variant	1LINKED	1B	2C	3ALT
Complete-Unity	n/a	130.48 | 130.98	54.81 | 55.42	n/a
Complete-Value	30.42 | 31.52	6.88 | 7.16	1.40 | 3.46	109.82 | 124.59
Partial-Unity	n/a	n/a	29.12 | 29.11	18.11 | 18.07
Partial-Value	n/a	3.64 | 3.61	0.80 | 0.80	3.23 | 2.24

Table 5.3: GLPK CPU time divided by CPLEX CPU time (left number) and GLPK wall time divided by CPLEX wall time (right number) for four instances and each complete and partial CSP model variant.

Chapter 6

Conclusions and further research

In the introduction to this thesis the question of whether an open source solver is a viable alternative to the commercial solver Xpress when solving the cell suppression problem was asked. In order to answer this question tests were run on seven different test instances for one well known open source solver, GLPK. The main indication of our results is that GLPK is considerably slower than CPLEX (see Tables 5.2 and 5.3)—a commercial solver comparable to Xpress—which is expected, but also that GLPK has greater difficulties with solving problems that exhibit poor numerical properties. The complete CSP model is, for our test instances, rather well behaved and some of the smaller instances were solved using GLPK. The larger instances, however, required an unreasonable amount of time to solve. The partial and combined CSP models proved to be even more problematic. GLPK could not solve the combined CSP even for the smaller instances, since this problem is too computationally heavy, and GLPK is unreliable for the partial CSP. For some choices of objective function in the partial CSP, some instances were solved quickly, whereas other instances ended with an infinite loop, likely due to degeneracy cycling. There exist several methods that can resolve degeneracy cycling for the simplex algorithm, e.g., Bland's rule (see [2]). From the results presented in this thesis, it can be concluded that an open source solver could possibly replace a commercial one for the complete and partial CSP applied to small problem instances.

Concerning the solution quality of the three models formulated, the first thing to keep in mind is that it is not clear how information loss should be defined and that this might depend on the specific data of a given application. Still, intuitively, a small number of suppressions is desired. Also, the suppression intervals should preferrably be small since a published table is likely to be more useful if the known interval of a suppressed cell is small. With these considerations, complete, partial, and combined CSP solutions were compared in terms of number of suppressed cells and the sum of the suppression interval widths. Assuming that these are proper measures of information loss, the combined CSP yields the best solutions (see Figures 5.1 and 5.2). On the other hand, at least with the implementation made for this thesis, combined cell suppression is extremely slow compared to complete and partial cell suppression (see Figures 5.3 and 5.4). So, as is, combined cell suppression is promising but probably too slow to be of practical use. Whether a better implementation could make it a more tractable method for solving the CSP is an open question.

In the comparison between complete and partial cell suppression, the results dis-

tinctly show that the complete CSP yields a smaller number of suppressions but larger suppression intervals, while the opposite is true for the partial CSP (see Figures 5.1 and 5.2). Recalling that the sum of the suppression interval widths should be interpreted with care in complete cell suppression, it is still a fact that the partial CSP yields smaller intervals. The computation times for the model variants Complete-Unity, Complete-Value, and Partial-Unity show no obvious trends: one model can be very fast for one particular instance but slow for another. Partial-Value generally required lower computation times and was faster by a large margin for some instances (see Figures 5.3 and 5.4). Choosing between complete and partial cell suppression is mainly a question of what is deemed more important: a small number of suppressions or small suppression intervals, or possibly something else entirely.

There are also considerations of less mathematical nature that should be taken into account when discussing wheter to completely remove cell values or replace them by intervals. It is of utmost importance that respondents feel assured that their data will be safe when they participate in surveys. It may be more obvious that the data is protected if a $\times$ is published than if an interval is given. That said, it is also the case that a general user of published statistical material may not be experienced in mathematical optimization. Such a user, confronted with a statistical table containing a complete suppression might assume that there is no way of knowing the missing value even approximately. In this situation, partial and combined cell suppression would have the benefit of very directly and in a simple manner providing the user with an approximate value.

Interesting questions for potential future research include whether heuristic methods could be used to quickly find feasible solutions to the partial or combined CSP. A heuristic for the complete CSP exists (see [5]) which could potentially be adapted for the partial and combined CSP. Investigation could also be made into how a table's structure (hierarchies, links and the distribution of sensitive cells) affects the computational difficulty of the CSP. Possibly, the matrix M could be partitioned into submatrices with specific properties, thereby allowing efficient network optimization algorithms to be used. This has been done (see [3]) for 2D tables with one hierachical spanning variable (such as the table in Figure 2.2) and for 2D and 3D tables without hierarchies, but not generalized to linked tables or tables with higher dimensions and several hierachical spanning variables.

Bibliography

[1] Andréasson, N., Evgrafov, A., Patriksson, M., Gustavsson, E. & Önnheim, M. 2013, *An Introduction to Continuous Optimization*, 2nd edition, Studentlitteratur, Lund.

[2] Bland, R.G. 1977, New finite pivoting rules for the simplex method, *Mathematics of Operations Research*, vol. 2, no. 2, pp. 103–107.

[3] Castro, J. 2012, Recent advances in optimization techniques for statistical tabular data protection, *European Journal of Operational Research*, vol. 216, no. 2, pp. 257–269.

[4] Daalmans, J. & de Waal, T. 2010, An improved formulation of the disclosure auditing problem for secondary cell suppression, *Transactions on Data Privacy*, vol. 3, no. 3, pp. 217–251.

[5] de Wolf, P.P. 2002, HiTaS: a heuristic approach to cell suppression in hierarchical tables. In: Domingo-Ferrer, J. ed. *Inference Control in Statistical Databases*, Springer, Berlin, Heidelberg, pp. 34–58.

[6] de Wolf, P.P. & Giessing, S. 2009, Adjusting the τ-ARGUS modular approach to deal with linked tables, *Data & Knowledge Engineering*, vol. 68, no. 11, pp. 1160–1174.

[7] de Wolf, P.P., Hundepool, A., Giessing, S., Salazar-González, J.J. & Castro, J. 2014, τ-ARGUS Version 4.1 User's Manual, Statistics Netherlands, , P.O. Box 24500, 2490 HA The Hague, The Netherlands.

[8] FICO® Xpress Optimization Suite, `http://www.fico.com/en/products/fico-xpress-optimization-suite` (2015-10-28).

[9] Fischetti, M. & Salazar-González, J.J. 2001, Solving the cell suppression problem on tabular data with linear constraints, *Management Science*, vol. 47, no. 7, pp. 1008–1027.

[10] Fischetti, M. & Salazar-González, J.J. 2003, Partial cell suppression: a new methodology for statistical disclosure control, *Statistics and Computing*, vol. 13, no. 1, pp. 13–21.

[11] GLPK (GNU Linear Programming Kit), `https://www.gnu.org/software/glpk/` (2015-10-28).

[12] Hundepool, A., Domingo-Ferrer, J., Franconi, L., Giessing, S., Schulte Nordholt, E., Spicer, K. & de Wolf, P.P. 2012, *Statistical Disclosure Control*, John Wiley & Sons, Chichester (Wiley Series in Survey Methodology).

[13] IBM CPLEX Optimizer, `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/` (2015-10-28).

[14] IBM ILOG CPLEX V12.1 Parameters Reference Manual 2009, IBM Corporation, 590 Madison Avenue, New York, NY 10022, United States.

[15] Kao, M. 1996, Data security equals graph connectivity, *SIAM Journal on Discrete Mathematics*, vol. 9, no. 1, pp. 87–100.

[16] Kelly, J., Golden, B. & Assad, A. 1992, Cell suppression: disclosure protection for sensitive tabular data, *Networks*, vol. 22, no. 4, pp. 397–417.

[17] Khachiyan, L.G. 1980, Polynomial algorithms in linear programming, *USSR Computational Mathematics and Mathematical Physics*, vol. 20, no. 1, pp. 53–72.

[18] Loeve, A. 2001, Notes on sensitivity measures and protection levels, Research paper, Statistics Netherlands.

[19] Meindl, B. & Templ, M. 2013, Analysis of commercial and free and open source solvers for the cell suppression problem, *Transactions on Data Privacy*, vol. 6, no. 2, pp. 147–159.

[20] Robertson, D.A. & Ethier, R. 2002, Cell suppression: experience and theory. In: Domingo-Ferrer, J. ed. *Inference Control in Statistical Databases*, Springer, Berlin, Heidelberg, pp. 8–20.

[21] Salazar-González, J.J. 2002, Extending cell suppression to protect tabular data against several attackers. In: Domingo-Ferrer, J. ed. *Inference Control in Statistical Databases*, Springer, Berlin, Heidelberg, pp. 34–58.

[22] Salazar-González, J.J. 2010, Branch-and-cut versus cut-and-branch algorithms for cell suppression. In: Domingo-Ferrer, J & Magkos, E. ed. *Privacy in Statistical Databases*, Springer, Berlin, Heidelberg, pp. 29–40.

[23] τ-ARGUS, `http://neon.vb.cbs.nl/casc/..%5Ccasc%5Ctau.htm` (2015-10-28).

Appendix A

Reducing the number of constraints

We shall show that in the inequalities (3.12), (3.15), (3.16), (3.24), and (3.26) it suffices to consider the extreme points of the dual feasible set of the attacker subproblems in order to impose all protection level requirements. The sets are defined by (3.9b)–(3.9c), (3.13b)–(3.13c), (3.23b)–(3.23c) and (3.25b)–(3.25c). For $M \in \mathbb{R}^{m\times N}$, $\gamma \in \mathbb{R}^m$ $\alpha \in \mathbb{R}^N$, and $\beta \in \mathbb{R}^N$, these can be expressed as

$$\begin{aligned} &M^T\gamma + \alpha - \beta = e, \\ &\alpha \geq \mathbf{0}^N,\ \beta \geq \mathbf{0}^N, \end{aligned} \tag{A.1}$$

where $e \in \{e_s, -e_s\}$. The set defined by (A.1) is a polyhedron, denoted $\mathcal{P}$, which can be expressed as

$$\mathcal{P} = \left\{ \begin{bmatrix} \gamma \\ \alpha \\ \beta \end{bmatrix} \in \mathbb{R}^{m+2N} : A \begin{bmatrix} \gamma \\ \alpha \\ \beta \end{bmatrix} \leq \begin{bmatrix} e \\ -e \\ \mathbf{0}^N \\ \mathbf{0}^N \end{bmatrix} \right\}$$

with

$$A = \begin{bmatrix} M^T & I^{N\times N} & -I^{N\times N} \\ -M^T & -I^{N\times N} & I^{N\times N} \\ \mathbf{0}^{N\times N} & -I^{N\times N} & \mathbf{0}^{N\times N} \\ \mathbf{0}^{N\times N} & \mathbf{0}^{N\times N} & -I^{N\times N} \end{bmatrix}.$$

Further, the inequalities (3.12), (3.15), (3.16), (3.24), and (3.26) can be written as

$$d_\alpha^T\alpha + d_\beta^T\beta \geq D \tag{A.2}$$

where $d_\alpha \geq \mathbf{0}$, $d_\beta \geq \mathbf{0}$ and $D \geq 0$. To exemplify, for (3.12) we have $d_\alpha = \text{diag}(u-a)x$, $d_\beta = \text{diag}(a-l)x$ and $D = P_s^{\text{U}}$, where $\text{diag}(v)$ denotes the diagonal matrix containing the elements of the vector v. For fixed x or $(z^+,\ z^-)$, we will show that if (A.2) holds for all extreme points of $\mathcal{P}$, it holds for every point in $\mathcal{P}$. Under the assumption that $M \in \mathbb{R}^{m\times N}$, where $N > m$, has rank m, A has full column rank. Note that if M does not have rank m it contains redundant rows which can be removed. We have the following representation theorem.

Theorem A.1. *Let $A \in \mathbb{R}^{k_1 \times k_2}$ and $b \in \mathbb{R}^{k_1}$. Let $P \coloneqq \{x \in \mathbb{R}^{k_2} : Ax \leq b\}$, Q denote the convex hull of the extreme points of P, and $C \coloneqq \{x \in \mathbb{R}^{k_2} : Ax \leq \mathbf{0}^{k_1}\}$. If rank $A = k_2$ then $P = Q + C \coloneqq \{x \in \mathbb{R}^{k_2} : x = v + u$ for some $v \in Q$ and $u \in C\}$.*

See for example [1, pp. 53–54] for a proof. Applying Theorem A.1 yields that $\mathcal{P} = \mathcal{Q} + \mathcal{C}$ where $\mathcal{Q}$ denotes the convex hull of the extreme points of $\mathcal{P}$ and

$$\mathcal{C} = \left\{ \begin{bmatrix} \gamma \\ \alpha \\ \beta \end{bmatrix} \in \mathbb{R}^{m+2N} : A \begin{bmatrix} \gamma \\ \alpha \\ \beta \end{bmatrix} \leq \mathbf{0}^{4N} \right\}.$$

For clarity, we note that $\mathcal{C}$ can also be expressed as

$$\mathcal{C} = \left\{ \begin{bmatrix} \gamma \\ \alpha \\ \beta \end{bmatrix} \in \mathbb{R}^{m+2N} : M^T\gamma + \alpha - \beta = \mathbf{0}^N,\ \alpha \geq \mathbf{0}^N,\ \beta \geq \mathbf{0}^N \right\}. \tag{A.3}$$

Hence, every point in $\mathcal{P}$ can be expressed as the sum of a point in $\mathcal{Q}$ and a point in $\mathcal{C}$. But every point in $\mathcal{Q}$ can be expressed as a convex combination of the extreme points of $\mathcal{Q}$ and every point in $\mathcal{C}$ can be expressed as a linear combination of the extreme directions of $\mathcal{C}$. Denote extreme points of $\mathcal{Q}$ by

$$\begin{bmatrix} \gamma_{\mathcal{Q}}^i \\ \alpha_{\mathcal{Q}}^i \\ \beta_{\mathcal{Q}}^i \end{bmatrix}, \quad i = 1, \ldots, \kappa_1,$$

where κ_1 is the number of extreme points. Further, denote the extreme directions of $\mathcal{C}$ by

$$\begin{bmatrix} \gamma_{\mathcal{C}}^j \\ \alpha_{\mathcal{C}}^j \\ \beta_{\mathcal{C}}^j \end{bmatrix}, \quad j = 1, \ldots, \kappa_2,$$

where κ_2 is the number of extreme directions. Thus, every point in $\mathcal{P}$ can be expressed as

$$\begin{bmatrix} \gamma \\ \alpha \\ \beta \end{bmatrix} = \sum_{i=1}^{\kappa_1} \lambda_i \begin{bmatrix} \gamma_{\mathcal{Q}}^i \\ \alpha_{\mathcal{Q}}^i \\ \beta_{\mathcal{Q}}^i \end{bmatrix} + \sum_{j=1}^{\kappa_2} \mu_j \begin{bmatrix} \gamma_{\mathcal{C}}^j \\ \alpha_{\mathcal{C}}^j \\ \beta_{\mathcal{C}}^j \end{bmatrix}$$

where $\lambda_1, \ldots, \lambda_{\kappa_1} \geq 0$ such that $\sum_{i=1}^{\kappa_1} \lambda_i = 1$ and $\mu_1, \ldots, \mu_{\kappa_2} \geq 0$. Now we can rewrite the left-hand side of (A.2) as

$$d_\alpha^T\alpha + d_\beta^T\beta = d_\alpha^T \left(\sum_{i=1}^{\kappa_1} \lambda_i \alpha_{\mathcal{Q}}^i + \sum_{j=1}^{\kappa_2} \mu_j \alpha_{\mathcal{C}}^j \right) + d_\beta^T \left(\sum_{i=1}^{\kappa_1} \lambda_i \beta_{\mathcal{Q}}^i + \sum_{j=1}^{\kappa_2} \mu_j \beta_{\mathcal{C}}^j \right).$$

For all i, the coefficients d_α, d_β, $\alpha_{\mathcal{Q}}^i$, and $\beta_{\mathcal{Q}}^i$ are non-negative by assumption, and for all j, the coefficients $\alpha_{\mathcal{C}}^j$ and $\beta_{\mathcal{C}}^j$ are non-negative because of the definition (A.3) of $\mathcal{C}$. Thus

the following manipulations can be made

$$\begin{aligned} d_\alpha^T \alpha + d_\beta^T \beta &\geq d_\alpha^T \sum_{i=1}^{\kappa_1} \lambda_i \alpha_{\mathcal{Q}}^i + d_\beta^T \sum_{i=1}^{\kappa_1} \lambda_i \beta_{\mathcal{Q}}^i \\ &= \sum_{i=1}^{\kappa_1} \lambda_i \left(d_\alpha^T \alpha_{\mathcal{Q}}^i + d_\beta^T \beta_{\mathcal{Q}}^i \right) \\ &\geq \min_{i=1,\ldots,\kappa_1} \left(d_\alpha^T \alpha_{\mathcal{Q}}^i + d_\beta^T \beta_{\mathcal{Q}}^i \right) \sum_{i=1}^{\kappa_1} \lambda_i \\ &= \min_{i=1,\ldots,\kappa_1} \left(d_\alpha^T \alpha_{\mathcal{Q}}^i + d_\beta^T \beta_{\mathcal{Q}}^i \right). \end{aligned}$$

Since we have assumed that (A.2) holds for all extreme points of $\mathcal{P}$, we conclude that

$$d_\alpha^T \alpha + d_\beta^T \beta \geq \min_{i=1,\ldots,\kappa_1} \left(d_\alpha^T \alpha_{\mathcal{Q}}^i + d_\beta^T \beta_{\mathcal{Q}}^i \right) \geq D$$

which means that (A.2) holds for all points in $\mathcal{P}$.

Appendix B

Result tables

Model variant	1LINKED	1B	2C	3ALT	2LINKED	1ALT	2ALT
Complete-Unity	175	103	100	134	329	250	686
Complete-Value	186	115	125	155	380	298	770
Partial-Unity	258	173	223	182	446	399	941
Partial-Value	205	126	146	196	483	350	939
Combined-Unity Unity*	263	151	114	151	n/a	n/a	n/a
Combined-Unity Value	183	110	109	147	n/a	n/a	n/a

*Master problem relative MIP gap tolerance $2 \cdot 10^{-5}$ for 1LINKED, $5 \cdot 10^{-5}$ for 1B, and 10^{-6} for 2C and 3ALT.

Table B.1: Number of suppressions for each instance and model variant using CPLEX.

Model variant	1LINKED	1B	2C	3ALT
Complete-Unity	17.2423482	28.3697370	204.0447465	81.2936786
Complete-Value	16.3297076	8.0999652	65.7830270	26.4115329
Partial-Unity	2.7806461	1.89757392	22.2080002	8.75365272
Partial-Value	2.93774604	1.9825934	23.75345768	9.33581636
Combined-Unity Unity*	2.7806461	1.89757392	22.2080002	8.75365272
Combined-Unity Value	2.96362734	1.97632202	22.92805248	9.27856334

*Master problem relative MIP gap tolerance $2 \cdot 10^{-5}$ for 1LINKED, $5 \cdot 10^{-5}$ for 1B, and 10^{-6} for 2C and 3ALT.

Model variant	2LINKED	1ALT	2ALT
Complete-Unity	335.3078217	48.8765260	234.4279953
Complete-Value	132.3576353	16.6801674	118.9257130
Partial-Unity	33.39228014	5.84831024	23.60951334
Partial-Value	37.25232462	6.17920594	29.18775085
Combined-Unity Unity	n/a	n/a	n/a
Combined-Unity Value	n/a	n/a	n/a

Table B.2: Sum of the suppression interval widths, divided by 10^7, obtained for each instance and model variant using CPLEX. Values are displayed exactly as reported by CPLEX.

Model variant	1LINKED	1B	2C	3ALT	2LINKED	1ALT	2ALT
Complete-Unity	853.73	2.95	2.93	34.74	8709.06	345.91	271.26
Complete-Value	7.64	15.47	215.17	1.53	3128.73	50.62	1660.4
Partial-Unity	475.56	122.18	138.65	3.88	1115.51	164.87	567.44
Partial-Value	5.26	1.49	2.43	1.73	74.38	13.58	200.66
Combined-Unity Unity*	54909.5	4078.56	29383.6	35.42	n/a	n/a	n/a
Combined-Unity Value	5543.03	190.02	5415.46	754.52	n/a	n/a	n/a

*Master problem relative MIP gap tolerance $2 \cdot 10^{-5}$ for 1LINKED, $5 \cdot 10^{-5}$ for 1B, and 10^{-6} for 2C and 3ALT.

Table B.3: CPU time in seconds used by CPLEX to solve each instance and model variant.

Model variant	1LINKED	1B	2C	3ALT	2LINKED	1ALT	2ALT
Complete-Unity	503	2.94	2.90	24.5	2880	210	218
Complete-Value	7.38	14.87	86.92	1.35	1100	40.5	584
Partial-Unity	476	122.34	138.82	3.89	1120	165	568
Partial-Value	5.30	1.50	2.43	1.73	74.41	13.59	200.82
Combined-Unity Unity*	54938.7	4080.87	29397.9	35.50	n/a	n/a	n/a
Combined-Unity Value	3156.3	145.42	2257.99	357.81	n/a	n/a	n/a

*Master problem relative MIP gap tolerance $2 \cdot 10^{-5}$ for 1LINKED, $5 \cdot 10^{-5}$ for 1B, and 10^{-6} for 2C and 3ALT.

Table B.4: Wall time in seconds used by CPLEX to solve each instance and model variant.

Model variant	1LINKED	1B	2C	3ALT	2LINKED	1ALT	2ALT
Complete-Unity	3345	1037	556	1015	2791	3935	3915
Complete-Value	1016	1214	595	342	1801	2423	2369
Partial-Unity	84377	32616	33278	6531	187621	62094	160776
Partial-Value	5156	1629	4910	2766	48492	9690	83918
Combined-Unity Unity*	74219	32819	44765	4715	n/a	n/a	n/a
Combined-Unity Value	10477	1745	4323	1892	n/a	n/a	n/a

*Master problem relative MIP gap tolerance $2 \cdot 10^{-5}$ for 1LINKED, $5 \cdot 10^{-5}$ for 1B, and 10^{-6} for 2C and 3ALT.

Table B.5: Number of constraints generated in the subproblems for each instance and model variant using CPLEX.