Universitat Autònoma
de Barcelona

Categorical Data Protection on Statistical Datasets and Social Networks

A dissertation submitted by
Jordi Marés Soler
at Universitat Autònoma de Barcelona (UAB)
to fulfill the degree of **PhD** in Computer Science.

Bellaterra, September 12th, 2013

Director: **Dr. Vicenç Torra Reventós**
Tutor: **Dr. Jordi Herrera Joancomartí**

Elaborated at: Institut d'Investigació en Intel·ligència Artificial
Consejo Superior de Investigaciones Científicas (IIIA-CSIC)

Acknowledgements

Ha estat un llarg i inesperat camí el que m'ha conduït a realitzar aquest treball i no hauria estat possible sense l'ajuda i confiança que m'heu donat entre tots.

En primer lloc vull donar les gràcies a la persona que ha estat clau per mi: el meu director de tesi, en Vicenç Torra. Gràcies per donar-me tota la confiança des del primer dia i ajudar-me a aconseguir reptes que mai se m'havien passat pel cap, per exemple, un doctorat. Això si, sempre recordaré lo malament que ho vaig passar a la primera entrevista que vaig tenir amb ell on el primer comentari va ser *veig que les teves notes no son gaire bones en algunes coses....* No obstant, va valer la pena passar-ho malament ja que al final es pot dir que ell ha estat la persona mes important en el meu desenvolupament professional. Gràcies un altre cop Vicenç.

Vull agrair a les segents persones que també han fet que aquest camí fos menys pesant. Gràcies als meus companys Dani, Sergi i Marc per compartir bons i mals moments cada dia. També gràcies als meus ex-companys Albert, Sergio i Isaac per fer-me sentir tant còmode en els meus primers dies a l'IIIA. Tito, gràcies per la teva ajuda tècnica sempre que feia falta i per organitzar aquells grans partidets de bàsquet. També agrair a tot el personal d'administració i a la Montse Calderón que sempre m'han solucionat amb molta efectivitat tots els problemes burocràtics que he tingut. Gràcies també al Dani Polak per la seva simpatia i amistat. A més a més, voldria agrair a tots i cadascun dels membres de l'IIIA el fet que m'hagin fet sentir com a casa entre la meva família. Per mi ha estat un orgull haver pogut aportar el meu granet de sorra en aquesta institució i haver compartit temps amb tots els seus membres.

I cannot forget thanking Dr. Natalie Shlomo for hosting me in The University of Manchester and sharing with me hours of work. I have learnt a lot from her and part of this thesis is accomplished thanks to her. I would also like to thank all the administrative department in the CCSR, with a special mention for Pip, for helping me in all the administrative troubles I had in Manchester. In addition, I want also to thank Simone, Oscar, Liting and Fabrizio for the amazing time I had with them in Manchester, making me feel like home.

També agrair a les següents persones les quals també han estat importants en molts moments: Josep Domingo, Guille i Jordi Herrera.

Per acabar, no em puc deixar d'agrair a la meva família el seu recolzament incondicional en tot moment que ha estat un factor clau per aconseguir aquesta

meta. Gràcies al meu germà Santi que sempre ha estat un model a seguir pel qué fa a esforçar-se per aconseguir creixer creient sempre en tu mateix. Gràcies a les meves nebodes Laia i Mariona per alegrar-me els dies sempre amb la seva felicitat infantil. Gràcies a l'Elizabeth per suportar-me tant en els bons moments com en els dolents i mostrar-me la seva estima incondicional. Finalment l'agraïment més gran és per als meus pares Consol i Jordi, per haver-me educat d'una manera exemplar i pel suport que m'han donat al llarg de tots els anys de la meva vida. Sense ells res d'això hauria estat possible, per tant, aquest èxit també és el seu èxit.

Per tot això i molt més, moltes gràcies a tots!

Jordi Marés Soler

Abstract

The continuous growth of public sensitive data has increased the risk of breaking the privacy of people or institutions in those datasets. This growing is, nowadays, even faster because of the expansion of the Internet. This fact makes very important the assessment of the performance of all the methods used to protect those datasets. In order to check the performance there exist two kind of measures: the information loss and the disclosure risk.

Another area where privacy has an increasing role is the one of social networks. They have become an essential ingredient of interpersonal communication in the modern world. They enable users to express and share common interests, comment upon everyday events with all the people with whom they are connected. Indeed, the growth of social media has been rapid and has resulted in the adoption of social networks to meet specific communities of interest.However, this shared information space can prove to be dangerous in respect of user privacy issues. In addition to explicit "posts" there is much implicit semantic information that is not explicitly given in the posts that the user shares. For these and other reasons, the protection of information pertaining to each user needs to be supported.

This thesis shows some new approaches to face these problems. The main contributions are:

- The development of an approach for protecting microdata datasets based on evolutionary algorithms which seeks automatically for better protections in terms of information loss and disclosure risk.
- The development of an evolutionary approach to optimize the transition matrices used in the Post-Randomization masking method which performs better protections.
- The definition of an approach to deal with categorical microdata protection based on a pre-clustering approach achieving protected data with better utility.
- The definition of a way to extract both implicit and explicit information from a real social network like Twitter as well as the development of a protection method to deal with this information and some new measures to evaluate the protection quality.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Data privacy became a very important issue since the first data publications in order to preserve the disclosure of sensitive information about individuals or institutions, but it has become even more important with the advances made in technology during the last few decades. Nowadays the number of available datasets for statistical studies is growing more and more so the amount of sensitive data like the income and health illnesses is also growing. To avoid that a data release causes the disclosure of sensitive information statistical agencies and data owners need to be very careful. They must preserve the privacy of the people involved on this data.

Most of available databases are defined in terms of microdata (records described in terms of attributes). The two most popular type of attributes are continuous and categorical. Continuous attributes are those that are numerical and allow to perform arithmetic calculations with their values, for example the income and the age. Categorical attributes are those that take values over a finite range and standard arithmetic operations do not make sense, for example the days of the week and the eyes color. Taking this into account it is easy to see that categorical attributes are the most difficult to protect.

In addition, recently it has appeared another place to be specially aware of the privacy issues: the social networks. Social networks have been adopted massively by people as a way to communicate between them and, as most sociologists agree, this online interaction will not fade away. People use these networks to share their feelings, emotions, and meet people with the same interests or hobbies. As a result, social networks are plenty of sensitive information about each user. Therefore, it is dangerous to collect this kind of data and publish it without protection. However, this kind of information would be very valuable if published. Usually, in social networks each user has its own public profile where he shows information about himself. However, it should be noticed that there is some information that is not given explicitly but can be inferred such as the

user's main topics of interest.

All these issues are studied in the Statistical Disclosure Control (SDC) discliline. SDC is concerned with the anonymization of the statistical data containing confidential information about individual entitites such as individuals or insititutions. The aim of SDC is to prevent third parties working from this data to recognize individuals and disclosing confidential information about them. Here, we understand third parties as the data users outside the statistical agencies (e.g. policy makers, academic researchers and general public).

SDC researchers have been working in the development of several data masking methods having as a final objective the construction of a masked dataset or a masked social graph able to be released mantaining the privacy of the data respondents minimising the loose of information. Masking methods are not enough. They also need a way to measure the performance of each masking method as well as the quality of the protection of a dataset. There exists two kind of measures to do this: the information loss and the disclosure risk.

Information loss measures check the quantity of harm inflected to the original data by the masking method, that is, it measures the amount of original information that has been lost during the masking process. There exist two different families of information loss measures: general measures and specific measures. General information loss measures roughly reflect the amount of information loss for a reasonable range of data uses. On the other hand, specific information loss measures evaluate the loss of statistical utility for a particular data analysis. Normally, the first kind of measures are used to compare protection methods and the second ones are used to evaluate in an accurate way the real effect of a protection method for a concrete statistical analysis.

Disclosure risk evaluates the privacy of the respondents against possible malicious uses that third parties (sometimes called intruders) could do with the released information. Disclosure risk measures evaluate the number of respondents whose identity is revealed. Normally, these measures are computed in several scenarios where the intruder has partial knowledge of the original data. In order to compute the disclosure risk, general methods for re-identification are used. These methods find relationships (i.e. links) between the protected data and the partial knowledge which the intruder is assumed to have.

The problem here is that information loss and disclosure risk measures are inversely related. That is, if we perform an aggressive protection we will obtain a high information loss but a low disclosure risk. However, if we perform no protection (or a very light protection) we will obtain a low information loss and a high disclosure risk. Then, the requirement for a protection to be considered a good protection is to have a balanced and minimised pair of values for both measures.

1.2 Contributions

The work presented in this thesis contributes in two related lines of research.

The first contribution is in the area of statistical disclosure control methods,

specifically in the set of methods dealing with categorical data. We introduce two protection methods based on evolutionary algorithms to deal with statistical microdata datasets. Those methods are splitted in two groups, one working directly with datasets and one dealing with category transition matrices used in the Post-Randomization Method. The first group of methods show that it is possible to come up with better protections just by embedding the microdata dataset inside an evolutionary algorithm which alters the data. In the case of the second group of methods, they show that it is also possible to improve the performance of the Post-Randomization Method (so also improve the protection quality of the masked dataset) just by embedding the transition matrix, that this method is based on, in an evolutionary algorithm which keeps altering the probability values inside data. Finally, we show that it is possible to add interesting properties such as invariance to the transition matrices in the evolutionary algorithm having then better data utility in the masked dataset.

The next contribution in the same area is the introduction a categorical microdata protection approach based on a pre-clustering step. We also introduce a new protection method based on the state-of-the-art protection method called Mondrian. Finally, we show the performance of this approach in two different use cases: standard statistical studies case, and clustering studies case.

The last contribution is in the area of user's privacy in social networks. We describe a set of techniques to deal with privacy for the information the users have in their social netwoks profiles. In the profiles of the social networks' users there are public fields containing explicit information like username and location. However, we show that there exist also hidden implicit information that can be extracted from the users posts. Furthermore, we propose a protection method dealing with information inside the social graph's nodes based on the k-anonymity principle. To test the performance of our method we propose a set of new analytical measures specifically developed to work with the data inside the graph's nodes. All this work is based on real samples that we extracted from the Twitter social network using an ad-hoc method. With all of this we show that it is possible to improve the privacy in social networks' graphs.

1.3 Document Structure

This thesis is organized in three parts introducing some preliminaries and related work in Chapter 2, showing our contributions in Chapters 3-6, and giving some conlusions and future directions in Chapter 7.

Chapter 2. We introduce preliminar basic concepts and methods needed to follow the content of the work presented afterwards. This chapter is divided in five sections:

- **Evolutionary Algorithms**. The chapter starts explaining the background of evolutionary algorithms as well as the basic general concepts like individual representations, different crossover and mutation

operators, individual selection methods, termination criteria and how to control the population evolution.

- **Genetic Programming**. We review a particular evolutive technique which deals with a population of executable programs with the goal of improving them under a certain environment. We review the main type of algorithms used in this field, how to represent the programs, how to alter them using genetic operators and other specific aspects like how to control the selection, population evolution and termination criteria.
- **Microdata Protection Methods.** We continue reviewing some of the state-of-the-art protection methods for statisitical categorical microdata datasets like Microaggregation, Post-Randomization Method, Mondrian, Rank Swapping, Global Recoding, Top Coding and Bottom Coding.
- **Microdata Evaluation Measures.** We introduce the state-of-the-art evaluation measures used in this thesis to evaluate the quality of the protections. The explanation is splitted in two parts, one for the information loss measures such as distance-based information loss, contingency table-based information loss and entropy-based information loss. The other part is for the disclosure risk measures with the interval disclosure, distance-based record linkage and probabilistic record linkage methods.
- **Clustering Partitions Similarity Measures.** We provide a review of the indices we used in this thesis to compute similarity between different clustering partitions.
- **Social Networks Users Privacy Protection Methods and Measures.** We review state-of-the-art approaches to deal with privacy in the social networks. In addition we provide some existing measures to deal with the information loss and disclosure risk in social networks.
- **Datasets.** We introduce the datasets used in the experiments shown in this thesis.

Chapter 3. We describe a new approach to deal with microdata privacy based on an evolutionary algorithm that seeks for better protections in an automatic and autonomous way. We present an evolutionary approach which works directly on the microdata dataset and it generates new better protections for both categorical and contiuous data.

Chapter 4. We introduce evolutive approaches to improve the performance of Post-Randomization Method protections by optimizing the matrices this protection method is based on.

- **General Evolutionary Approach for Better PRAM Matrices**. In the first section we introduce an evolutionary approach which works

on the matrices used by the Post-Randomization Method when protecting a categorical microdata dataset. This method produces new matrices that, used in the Post-Randomization Method, produces better protections.

- **Genetic Programming Approach for Better PRAM Matrices Equations.** In the second section we describe a genetic programming approach which deals with analytical equations to create Post-Randomization Method matrices and improve them in an evolutive way. The new equations lead to better protections.

Chapter 5. We introduce a categorical microdata protection approach based on performing clustering before protecting a dataset.

Chapter 6. We present a new approach to deal with user's information in social networks working from an extracted sample of Twitter real social network. In addition we introduce some new privacy measures based on user's information to use in social graphs.

Chapter 7. We finish this thesis with some concluding remarks and we also provide some future directions to continue the lines of research shown in this work.

Chapter 2

Preliminaries

This chapter introduces the basic concepts and methods used in the thesis. In Section 2.1 we present the evolutionary algorihms explaining the history of where these algorithms come from and their basic concepts. Next, in Section 2.3 we review some of the state-of-the-art categorical microdata protection methods. Section 2.4 shows the state-of-the-art measures to evaluate the protection quality in statistical microdata. In Section 2.5 we review measures to perform comparisons between clustering partitions. Then, in Section 2.6 we review state-of-the-art protection and protection analysis method for social networks. Finally, in Section 2.7 presents the datasets used in the experiments of this thesis.

2.1 Evolutionary Algorithms

Evolutionary Algorithms are based on the model of biological evolution that was formulated for the first time by Charles Darwin.

The Darwinian theory of evolution explains the adaptive change of species by the principle of natural selection. Under that principle, the species which have better adaptation to their environment are favoured for survival and further evolution. Darwin also explained that another important factor for evolution is the occurrence of small, apparantly random and undirected variations between the phenotypes, i.e., the manner of response and physical embodiment of parents and their offspring. These mutations prevail through selection, if they prove their worth in light of the current environment; otherwise, they perish. Population size grows exponentially under advantageous environmental conditions. This process is generally limited by finite resources. When resources are no longer sufficient to support all the individuals of a population, those organisms are at a selective advantage which exploit resources more effectively.

This Darwinian theory is based on genes as transfer units of heredity which are occasionally changed by mutations. Selection acts on the individual, which expresses in its phenotype its total genetic information, as well as the interaction of the genotype with the environment in determining the phenotype.

In a biological context, the term adaptation denotes a general advantage in ecological or physiological efficiency of an individual over the one achieved by other members of the population, and at the same time it denotes the process of attaining this state. The overall meaning of adaptation is often used synonymously with fitness. Furthermore, the term adaptation bears the question "to what?". Basically, the answer is to any major kind of environment or, in a broader sense, an ecological niche.

In the evolutionary framework, the fitness of an individual is measured only by its propensity to survive and reproduce in a particular environment. Furthermore, natural selection is no active driving force, but differential survival and reproduction within a population makes up selection. Selection is simply a name for the ability of those individuals that have outlasted the struggle for existence to bring their genetic information to the next generation.

Putting all together, we can imagine an extremely complex, unknown functional dependence which maps genomes to fitness measures judging phenotypical expressions of genotypes. During evolution, genotypes producing phenotypes of increasing biological fitness are created by means of the process of mutation and recombination on the genotype and selection on the phenotype.

During the biological evolution, an optimization of the fitness takes place, and even if we assume a constant adaptive surface that is not changed according to the positions of individuals themselves, the combination of mutation and recombination allows in principle for leaving a smaller hill of the landscape and therefore prevents evolution from getting stuck on suboptimal hills.

The above point of view provides the basis for the idea to use a simulated evolutionary process for the purpose of solving an optimization problem, where the goal is to find a set of parameters (which might be interpreted as genotype as well as phenotype) such that a certain quality criterion is maximized or minimized.

John Holland was the firt to use the term *genetic algorithm*. He authored the book *Adaptation in Natural and Artificial Systems* in 1975 [Holland, 1975], which was very important in creating a rich field of research and application that goes much wider than the original genetic algorithm. These first evolutionary algorithms were focused on mutation considering them as variation of hill-climbing methods. However, Holland introduced a very innovative idea: the idea of recombination. Nowadays, in order to cover the developments of the last years in this field many people use the term *evolutionary algorithms*.

Holland's influence in the development of the topic has been very important, but several other scientists with different backgrounds were also involved in developing similar ideas. Ingo Recherberg [Rechenberg, 1973] and Hans-Paul Schwefel [Schwefel, 1977] developed the idea of *Evolutionsstrategie* (evolution strategy) in the 1960s and Bremmerann, Fogel and others implemented their idea what they called *evolutionary programming*.

1975 was a crucial year for the genetic algorithms development. During that year Holland's book was published and Ken De Jong, one of the Holland's graduate students, completed his doctoral thesis [De Jong and Kenneth, 1975].

De Jong was the first to provide a thorough treatment of the genetic algorithms capabilities in optimization.

After De Jong's work, further studies were followed by others and David Goldberg, another graduate student of Holland, produced the first thesis on the application of genetic algorithms in the gas pipeline optimization. In addition, he also published a very influential book called *Genetic Algorithms in Search, Optimization, and Machine Learning* [Goldberg, 1989]. This was the final catalyst in setting off a sustained development of genetic algorithms theory and applications.

In [De Jong and Kenneth, 1993] De Jong cautioned that optimization may be misplaced. He said that genetic algorithms are not really function optimizers, and that this is in some ways incidental to the main theme of adaptation. Nervertheless, using genetic algorithms for optimization is very popular, and frequently successful in real applications.

It is interesting in this regard to compare some of the ideas being put forward in the 1960s in the field of *operational research* (OR). OR workers begun to develop techniques that seemed able to provide 'good' solutions, even if the quality was not provably optimal (or even near-optimal). Such methods became known as *heuristics*. A popular technique was *neighbourhood search*. The basic idea is to explore neighbours of an existing solution (i.e. these being defined as solutions obtainable by a specified operation on the base solution). One of the most influential papers in this context was the one published by Lin [Lin, 1965], who found excellent solutions to the traveling salesman problem by investigating neighbourhoods formed by breaking any 3 links of a tour and re-connecting them.

This was not the genetic algorithm developed by Holland, but there are clear resonances. Much later, after genetic algorithms had become more widely known, Lin's ideas were re-discovered as multi-parent recombination and consensus operators.

2.1.1 Evolutionary Algorithms Basic Concepts

Assume we have a discrete search space χ, and a function $f : \chi \mapsto \mathbb{R}$. The general problem is to find $min_{x \in \chi} f$ where x is a vector of decision variables, and f is the objective function. We assume here that the problem is one of minimization, but the modifications necessary for a maximization problem is obvious. Such a problem is commonly called a discrete or combinatorial optimization problems (COP).

One of the distinctive features of the genetic algorithm approach is to allow the separation of the representation of the problem from the actual variables in which it was originally formulated. In line with biological usage of the terms, it has become customary to distinguish the genotype (the encoded representation of the variables) from the phenotype (the set of variables themselves). That is, the vector x is represented by a string s, of length l, made up of symbols drawn from an alphabet A, using a mapping $c : A' \mapsto \chi$.

In practice, we may need to use a search space $S \subseteq A'$ to reflect the fact that some strings in the image of A' under c may represent invalid solutions to the

original problem. The string length l depends on the dimensions of both χ and A, and the elements of the string correspond to 'genes', and the values those genes can take to 'alleles'. This is often designated as the *genotype-phenotype mapping*. Thus the optimization probflem becomes one of finding $min_{s \in S} g(s)$ where the function $g(s) = f(c(s))$.

Biological analogy was the original motivation for the genetic algorithm approach where in the selective breeding of plants or animals offspring are sought that have certain desirable characteristics which are determined at the genetic level by the way the parents' chromosomes combine. In the case of genetic algorithms, a population of strings is used, and these strings are often referred to in the genetic algorithms literature as *chromosomes*. The recombination of strings is carried out using simple analogies of genetic *crossover* and *mutation*, and the search is guided by the results of evaluating the objective function f (fitness function) for each string in the population. Based on this evaluation, strings that have higher $fitness$ (i.e. represent better solutions) can be identified, and these are given more opportunity to breed.

The optimization process can be described as the generic outline of an evolutionary algorithm presented in Algorithm 1.

Algorithm 1 Outline of a Generic Evolutionary Algorithm

Input: $P(0) = \{X_i'\}$ initial population.
Output: $P(t) = \{X_j'\}$ generation t
$t \Leftarrow 0$
evaluate($P(0)$)
while stopping($P(t)$) $\neq true$; **do**
  $P_s(t) \Leftarrow$ select($P(t)$)
  $P(t+1) \Leftarrow$ alter($P_s(t)$)
  evaluate($P(t+1)$)
  $t \Leftarrow t+1$
end while
return $P(t)$

2.1.2 Evolutionary Algorithms Key Elements

In the general algorithm there are several points to be customised in order to adapt the algorithm to each problem needs: the individuals representation, the termination criteria, the selection method, the crossover-mutation conditions, the type of crossover and mutation to perform, and the way to construct the new generation at the end of each one.

Individuals Representation

The way to represent the individuals inside the algorithm is a key point because it will guide the way we perform operations and, having a proper representation, will make our problem easier to deal with. There exist

several kind of representations but here we discuss the most important kind of problems to represent them.

The first one is a binary problem, this is a problem whose variables have only two possible values. In this case it is clear that we can define a solution as a binary string of length n. In this case there is no distinction between genotype and phenotype. However, this kind of problems are not necessarily easy to solve with a genetic algorithm and the presence of constraints is likely to cause difficulties.

The second kind of problem to represent is a discrete but not binary problem. There are cases in which a discrete alphabet of higher cardinality than 2 might be appropiate. In this case a k-ary coding is natural. A solution is represented by a string of length n where each gene corresponds to a variable and the alleles, drawn from $1, \ldots, k$, represent the different k possible values for each variable.

Finally, the last kind of problem is a non-binary problem. This kind of problem has integer or real-valued variables. In such cases, transformation to a binary string is required first. Here values are not merely labels like in the previous case, instead, they are meaningful as numbers. Such problems assume firstly that we know the domain of each of our decision variables, and secondly that we have some idea of the precision with which we need to specify our eventual solution. Given these two ingredients, we can determine the number of bits needed for each decision variable, and then form a chromosome.

One well known technique used to deal with binary representations avoiding abrupt value changes when altering bits is to use gray-coded values. Gray code representation is a modification of the binary representation where two successive values differ in only one bit. For example, the binary value 111000 represents the decimal number 56 and the same one in Gray code representation represents the decimal value 47. Altering for example the second bit on the left hand side leads to the value 101000 which in binary represents the decimal value 40, but in Gray code represents the decimal value 48. Discussion of gray coding can be found in [Caruana and Schaffer, 1988].

In addition, the authors in [Greiner et al., 2005] showed that the use of Gray code in evolutionary algorithms allows to obtain faster and more accurate solutions than regular binary representation.

The conversion of a binary number to a gray coded representation has several steps. First of all, the most significant bit of the gray coded representation is the same than in the binary number. Then, add using modulo 2 the next significant bit of the binary number to the next significant bit of the binary number to obtain the next gray coded bit. Repeat this last step until all bits of the binary have been converted.

On the other hand, to convert a Gray coded representation to a binary number start by copying into the binary number the most significant bit

of the Gray coded value. Then, add using modulo 2 the next significant bit of the binary number to the last bit in of the gray code. Repeat the last step until all bits of the gray code have been converted.

To convert a binary number with $b_1, b_2, \ldots, b_{n-1}, b_n$ as the bit sequence to its corresponding binary reflected Gray code, start at the right with the bit b_n. If the b_{n-1} bit is 1, then g_n is $1 - b_n$, otherwise, g_n is b_n. Then proceed to b_{n-1}. Continue up to the first bit b_1, which is kept unchanged since b_0 is assumed to be 0. The resulting number $g_1, \ldots, g_n$ is the Gray code of the binary number.

Then, to convert a Gray code back to its represented binary number, start again at the right with the nth digit and compute the expression shown in the following expression.

$$[h!]\Sigma_n \equiv \sum_{i=1}^{n-1} g_i \pmod 2 \tag{2.1}$$

If Σ_n is 1, b_n is $1 - g_n$; otherwise, b_n is g_n. Repeat this procedure for all bits in Gray coded value and the resulting bits sequence $b_1, \ldots, b_n$ is the binary number.

Termination

Unlike simple neighbourhood search methods that terminate when a local optimum is reached, genetic algorithms are stochastic search methods that could in principle run forever. In practice, a termination criteria is needed. Common approaches are to set a limit on the number of fitness evaluations or the computer clock time, to track the population's diversity and stop when this falls below a preset threshold, or to reach a certain level of fitness in the population.

Selection

The basic idea of selection is that it should be related to fitness, and the original scheme for its implementations is commonly known as the *roulette-weel* method. It uses a probability distribution for selection in which the selection probability of a given string is proportional to its fitness. Equation 2.2 shows this idea where f_i represents the fitness evaluation value for individual i.

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{2.2}$$

Another technique is *tournament selection* in which a set of τ chromosomes is randomly chosen and compared, the best one being selected for parenthood. One potential advantage of tournament selection is that it only needs a preference ordering between pairs or groups of strings, and it can thus cope with situations where there is no formal objective function at all (in other words, it can deal with purely subjective objective function).

Crossover-Mutation Conditions

Given the stress on recombination in Holland's original work, it might be thought that crossover should be used in each generation, but in fact there is no reason to suppose that it has to be so. Furthermore, there exist two strategies to be followed to generate new offspring: the crossover-AND-mutation and the crossover-OR-mutation.

In the first strategy both operators are tried to be applied to each individual trying first to apply crossover and then applying mutation. However, it should be noticed that, as both operators need their respective rates in this strategy, in some cases nothing happens at all because no operation has been applied as the rates are not reached. However, the second strategy always perform either crossover or mutation but not both.

The mechanism for implementing such choices is customarily a randomized rule, whereby the operation is carried out if a uniform random generated value passes a threshold value. In the case of crossover, this is often called the *crossover rate*. For mutation, we have a choice between describing the number of mutations per string, or per bit. This is often called *mutation rate*.

In the -OR- case, there is further possibility of modifying the relative proportions of crossover and mutation as the search progresses. Davis [Davis and Mitchell, 1991] has argued that different rates are approapiate at different times: high crossover at the start, high mutation as the population converges. He has further suggested that the operator proportions could be adapted online, in accordance with their track record in finding new high-quality chromosomes.

Crossover Operator

Crossover simply consists on replacing some of the genes in one parent by corresponding genes of the other. Suppose we have 2 strings $(a_1, a_2, a_3, a_4, a_5, a_6)$ and $(b_1, b_2, b_3, b_4, b_5, b_6)$ each consisting of values for 6 variables which represent two possible solutions to a problem. In the *one-point crossover*, one crossover point is selected among the whole string values. Then, the offspring are constructed taking the right hand side of each parent and joining them with the left-hand side of the other parent. In our case, if the crosspoint was 3, the offspring would be $(a_1, a_2, a_3, b_4, b_5, b_6)$ and $(b_1, b_2, b_3, a_4, a_5, a_6)$.

Two-point crossover is very similar. Two crosspoints are chosen at random and two new offspring are produced by combining the pieces of the original parents. For instance, if the crosspoints were 2 and 4, the offspring solutions would be $(a_1, a_2, b_3, b_4, a_5, a_6)$ and $(b_1, b_2, a_3, a_4, b_5, b_6)$. A similar prescription can be given for m-point crossover where $m > 1$.

Eshelman et al. [Eshelman et al., 1989] provided an investigation of multipoint crossovers examining the biasing effect of traditional one-point crossover, and considered a range of alternatives. He also stated that

the one-point crossover limits the change of information between the parents. In [Eshelman et al., 1989] the possibilities of changing these biases, in particular by using multi-point crossover, were investigated and empirical evidence strongly supported the suspicion that one-point crossover is not the best option.

Another alternative is the so-called *uniform crossover* whichs removes any bias by making the crossover process completely random. This can be seen most easily by observing that the crossover operator itself can be written as a binary string or mask like 110011 which represents the two-point crossover used above. The 1's means that the alleles are taken from the first parent, while the 0's means they come from the other parent. Then, by generating the pattern of 0's and 1's stochastically using a Bernoulli distribution we get the uniform crossover.

This idea was first used by Syswerda [Sywerda, 1989], who implicitly assumed the Bernoulli parameter $p = 0.5$. De Jong and Spears [De Jong and Spears, 1992] produced a theoretical analysis that was able to characterize the amount of disruption introduced by a given crossover operator exactly. In particular, the amount of disruption in the uniform crossover can be tuned by choosing different values of p.

Mutation Operator

The main idea behind mutation operator is to randomly select a subset of genes and change their alleles. This concept is simpler than crossover, and again, this can easily be represented as a bit-string. We generate a mask such as 010001 using a Bernoulli distribution at each locus using a small value of p. Then, the mask works like in the uniform crossover case.

However, there are different ways of implementing this simple idea. The naive idea would be to draw a random number for every gene in the string and compare it to the mutation rate, but this is potentially expensive in terms of computation if the strings are long and the population is large. An efficient alternative is to draw a random variate from a Poisson distribution with parameter λ, where λ is the average number of mutation per chromosome. A common value for λ is 1 (in other words, if l is the string length, the bit-wise mutation rate is $1/l$), which in 1964 [Bremermann et al., 1966] was shown to be in some sense an optimal mutation rate. Then, having m mutations, we draw m random numbers without replacement uniformly distributed between 1 and l in order to specify the location where mutation has to take place.

In case of binary strings, mutation simply means complementing the chosen bit(s). More generally, when there are several possible allele values for each gene, if we decide to change a particular allele, we must provide some means of deciding what its new value should be. This could be a random choice, but if there is some ordinal relation between allele values, it may be more sensible to restrict the choice to alleles that are close to the current value, or at least to bias the probability distribution in their favour.

Next Generation Selection

Holland's original genetic algorithm assumed a generational approach: selection, recombination and mutataion were applied to a population of M chromosomes until a new set of M' individuals had been generated. This set then becomes the new population. From an optimization point of view this seems an odd thing to do, we may have spent considerable effort obtaining a good solution only to run the risk of throwing it away and preventing it from taking part in further reproduction.

For this reason, De Jong [De Jong and Kenneth, 1975] introduced the concepts of *elitism* and *population overlaps*. An elitism strategy ensures the survival of the best individual so far by preserving it and replacing only the $M - 1$ members of the population with new strings. Overlapping populations take this a stage further by replacing only a fraction G (the generation gap) of the population at each generation. Finally, taking this to its logical conclusion produces the so-called steady-state or incremental strategies, in which only one new chromosome (or sometimes a pair) is generated at each stage.

One of the keys to have a good performance (in nature as well as in genetic algorithms) is to mantain the diversity of the population as long as possible. The effect of selection is to reduce diversity, and some methods can reduce diversity very quickly. This can be mitigated by having larger populations, or by having greater mutation rates, but there are also other techniques often employed.

A popular approach commonly linked with steady-state or incremental genetic algorithms, is to use a no-duplicates policy [Davis and Mitchell, 1991]. This means that the offspring are not allowed into the population if they are merely clones of existing individuals.

2.2 Genetic Programming

The goal of having computers that automatically solve problems is central to artificial intelligence, machine learning, and the broad area encompassed by what Turing called "machine intelligence" [Turing, 1948][Turing, 1950].

Genetic Programming (GP) is an evolutionary computation technique that automatically solves problems without requiring the user to know or specify the form or structure of the solution in advance. At the most abstract level, GP is a systematic, domain-independent method for getting computers to solve problems automatically starting from a high-level statement of what needs to be done.

GP stochastically transforms populations of programs into new, hopefully better, populations of programs. It is a random process so it can never guarantee results, like in the nature. However, this GP's essential randomness can lead it to escape traps which deterministic methods may be captured by and this has been very successful at evolving novel and unexpected ways of solving problems.

The creation of the initial random population is performed so as to create syntactically valid, executable programs. After the genetic operations are performed on the current generation of the population, the population of offspring replaces the old one. The tasks of measuring fitness, selection, and genetic operations are then iteratively repeated over many generations. The computer program resulting from this simulated process can be the solution to a given problem or a sequence of instructions for constructing the solution.

There exist two main types of GP algorithms: the steady-state algorithm and the generational algorithm. Steady-state GP algorithm is described in Algorithm 2. This first algorithm works by randomly selecting some individuals from the population and then keeping the best of the selected for the next generation's population and replace the worst by the generated offspring under a certain replacement conditions. This approach ensures that we are not going to loose a good solution from the population.

Algorithm 2 Steady-State GP Algorithm.

Randomly create an *initial population* of programs from the available primitives.
while an acceptable solution is found or some other stopping condition is met **do**
  Select a random subset of the population to take part in the tournament.
  Execute each program in the subset to assess their fitness.
  Select the winner(s) of the tournament using the selection algorithm.
  Create new individual program(s) by applying *genetic operations* on the winner(s).
  Replace the loosers in the tournament by the new offspring.
end while
return the best-so-far individual.

Instead, generational GP algorithm (described in Algorithm 3) works by generating an entirely new population at each generation. It substitutes the old generation by the offspring obtained by applying genetic operators to each existing individual. No individuals are retained between generations.

2.2.1 Individuals and Population Initialization Methods

Population contains analytically valid and executable programs. Here we explain the state-of-the-art methods to initialize tree-based individuals and populations.

Recall that trees are built from basic units called functions and terminals. We shall assume that the terminal and functions allowable in the program trees have been selected already. There exist two different methods to initialize tree-based structures in common use: *grow* and *full*.

Grow.

Grow produces trees of irregular shape because nodes are selected randomly from the union of the function and terminal sets throughout the

Algorithm 3 Generational GP Algorithm.

Randomly create an *intial population* of programs from the avaliable primitives.
while an acceptable solution is found or some other stopping condition is met **do**
  Execute each program in the population to assess their fitness.
  while new population is not fully populated **do**
    Select an individual(s) from the population.
    Create new individual program(s) by applying *genetic operations*.
    Insert the new offspring into the new population
  end while
  Replace the old population by the new one.
end while
return the best-so-far individual.

entire tree (except the root node, which uses only the functional set). Once a branch contains a terminal node, that branch has ended, even if the maximum depth has not been reached (See left-hand draw in Figure 2.1 for a graphical example).

Full.

Full, instead, produces trees where each branch goes to the full maximum depth. This is because, in this case, nodes are selected randomly only from the functions set until a node is at maximum depth. Then it chooses only terminals. (See right-hand draw in Figure 2.1 for a graphical example)

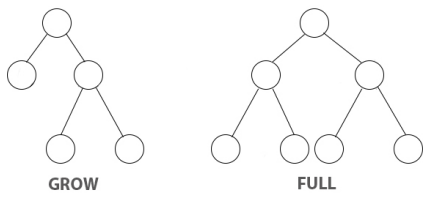


Figure 2.1: Tree structure initialization examples.

It should be noticed that having a function set too large enlarges the search space and can, sometimes, make the search for a solution harder. In order to have a good function set it should include arithmetic and logic operators, and, all of them, must be able to handle all elements in the terminal set.

GP is very creative at taking simple functions and combining them creating what it needs and it often ignores the more sophisticated functions in favor of the primitives during the evolution.

Diversity is valuable in GP populations. By itself, the above methods could result in a uniform set of structures in the initial population because the routine is the same for all individuals. To prevent this, the "half-and-half" technique has been devised [Koza, 1992].

Half-and-half method works as follow. Suppose the maximum depth parameter is $depth_{max}$. The population is divided equally among individuals to be initialized with trees having depths $1, \ldots, depth_{max}$. For each depth group, half of the trees are initialized with the full technique and half with the grow technique.

2.2.2 Tree Structure's Crossover and Mutation

Genetic programming is a particular kind of evolutionary algorithm. For that reason, the general idea of its genetic operators is the same than the one explained in the previous section. Crossover operator combines the genetic material of two parents and the mutation operator alters a part of a single individual's genetic material. However, these operators are implemented differently than in the other case.

The crossover operator for tree-based structures works by selecting a random node in two graphs and then swapping the two subgraphs having these nodes as roots. Figure 2.2 illustrates this idea.

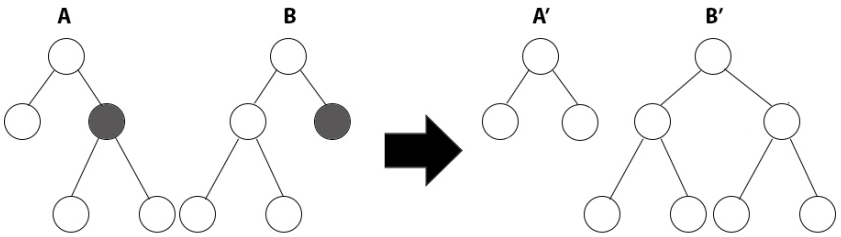


Figure 2.2: Tree structure crossover example.

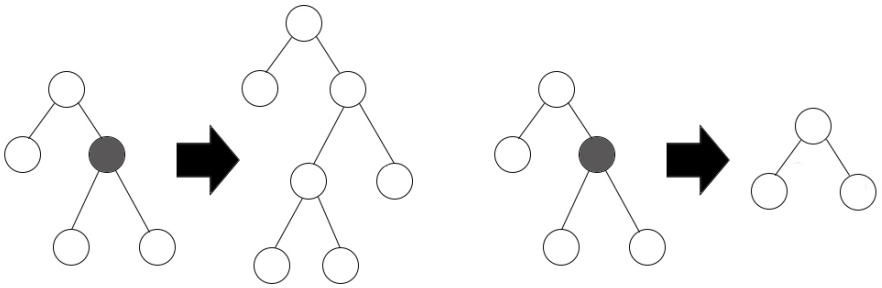
In the case of the mutation operator for tree-based structures there exist several approaches to choose from.

Point mutation is the simplest one and works by randomly selecting a node in a tree and changing it, also randomly, for another node of the same class. That is, if the node is a functional node, its contained function is replaced by another different function. On the other hand, if the node is a terminal node, its contained value is replaced by another terminal value. It should be noticed that this method does not change the tree structure but, at the same time, it prevents to explore other tree structures that might be better.

Hoist mutation creates a new offspring individual which is copy of a randomly

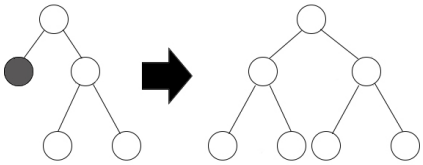
chosen subtree of the parent. Thus, the offspring will be smaller than the parent and will have a different root node

Subtree mutation is the most widely used in GP and it works by randomly selecting a node in a tree and it is changed by a new random subtree. It should be noticed that the newly created random subtree must fulfill the restriction of not exceeding the maximum individual depth when it is added to the mutated tree (See Figure 2.3 for a graphical example).



(a) Subtree mutation

(b) Collapse mutation



(c) Expansion mutation

Figure 2.3: Tree structure mutations example.

Collapse and Expansion mutations are two particular cases of the subtree mutation. The first one changes a randomly selected subgraph by a randomly selected terminal node. On the other hand, the second one does the inverse, that is, replaces a randomly selected terminal node by a random subgraph (See Figure 2.3 for a graphical example).

2.3 Microdata Protection Methods

This section is a review of the state-of-the-art masking methods for categorical data. These masking methods are splitted in two main groups: the perturbative methods, and the non-perturbative methods.

Perturbative masking methods are the ones that distort the data before being published. In this way, unique combinations of scores in the original dataset may

disappear and new unique combinations may appear in the perturbed dataset obtaining a confusion that is beneficial for preserving statistical confidentiality. The perturbation method should provide computed statistics in the perturbed dataset that do not differ significantly from the statistics that would be obtained on the original dataset.

In the non-perturbative masking methods case the data is not perturbed. Instead, they produce partial suppressions or reductions of detail (generalizations) on the original dataset values.

In this section we review the most relevant protection methods for this thesis. For the perturbative case we review the Microaggregation, Post Randomization Method, Mondrian, and Rank Swapping methods. On the other hand, for the non-perturbative case we review the Global Recoding, Top Coding, and Bottom Coding methods.

2.3.1 Post Randomization Method

The Post Randomization method (PRAM) was introduced in [Gouweleeuw et al., 1998] and [Kooiman et al., 1997] as a method for masking categorical variables in microdata files. In [De Wolf and Van Gelder, 2004] and [Wolf et al., 1998] the method and some of its implications were discussed in more detail. However, the PRAM method is still one of the least used for protecting microdata because of the difficulty in obtaining an optimal transition matrix to perform safe protections whilst maintaining data utility. This was demonstrated in the experiments carried out in [Domingo-Ferrer and Torra, 2001b] where the PRAM method was shown to have the worst utility and protection scores.

The PRAM method is as follows: Let t be the vector of frequencies and t/n the vector of relative frequencies of a categorical variable having L categories and n is the number of records in the microdata. Let P be a $L \times L$ probability transition matrix containing conditional probabilities: $p_{ij} = p(value_{perturbed} = j | value_{original} = i)$. In each record of the data, the category of the variable is changed or not changed according to the prescribed transition probabilities in the matrix P and the result of a draw of a random multinomial variate u with parameters p_{ij} $(j = 1, ..., L)$. If the j-th category is selected, category i is moved to category j. When $i = j$, no change occurs.

There are different ways to define the Markov matrices in the literature. We discuss here two of the approaches, which are the most commonly used. In the discussion we understand p_{kl} as the probability of changing a value k to a value l. Then, $\sum_{l=1}^{n} p_{kl} = 1$, where n is the number of categories. We choosed two types of matrices design to work. The first type is a fully-filled matrix with the off-diagonal elements depending on the corresponding frequencies in the original microdata file. This approach has been used in [De Wolf and Van Gelder, 2004]. Formally, the probability p_{kl} for $k \neq l$ is defined by

$$p_{kl} = \frac{(1 - p_{kk})(\sum_{i=1}^{n} T_\xi(i) - T_\xi(k) - T_\xi(l)}{(n-2)(\sum_{i=1}^{n} T_\xi(i) - T_\xi(k))} \tag{2.3}$$

where $T_\xi(i)$ is the frequency of the category i inside the original dataset for the actual variable. In this approach p_{kk} is left as constant, that is, $p_{kk} = p$ for all k. The key point of this equation is that it assigns the higher exchange probabilities to the categories with less frequency. In this way, the resultant dataset has more confusion.

The second type is a fully-filled matrix with the diagonal elements depending on the corresponding frequencies in the original microdata file. This approach has been used in [Domingo-Ferrer and Torra, 2001b]. In this case the row values are determined by the following expressions:

$$p_{kk} = 1 - (\theta T_\xi(K)/T_\xi(k)) \tag{2.4}$$

for $k = 1, \ldots, n$ and, then,

$$p_{kl} = \frac{1 - p_{kk}}{n - 1} \tag{2.5}$$

for $k \neq l$, where $T_\xi(K)$ is the lower value frequency higher than zero, and θ is a parameter in $[0, 1]$.

2.3.2 Microaggregation

In the Microaggregation method the records are clustered into small aggregates of groups of size k. Then, instead of publishing an original variable V_i for a given record, the average of the values of V_i over the cluster to which the records belongs to is published. This kind of protection introduced what afterwards was called the k-anonymity concept. It says that a dataset is k-anonymised if for each combination of quasi-identifiers values there are at least k registers with the same combination. In this way, if an intruder try to link the protected data to a external data, there will be always k equally probable matches making impossible to know exactly which is the correct one. But to perform this protection without loosing a big quantity of information, clusters should be as homogeneous as possible.

In [Oganian and Domingo-Ferrer, 2001] it was shown that solving the multivariate Microaggregation problem is NP-Hard and it has been also proved that in the univariate case the problem is polynomial [Hansen and Mukherjee, 2003]. So methods in the literature are heuristic and can be univariate or multivariate:

- Univariate methods deal with multivariate microdata sets by protecting one variable at a time. This approach is known as individual ranking or blurring [Defays and Nanopoulos, 1993] and, although it generates low information loss, it can cause high disclosure risk.

- Multivariate methods either rank multivariate data by projecting them onto a single axis [Defays and Nanopoulos, 1993] or dealing directly with unprojected data [Domingo-Ferrer and Mateo-Sanz, 2002]. When working with unprojected data, variables can be microaggregated by groups of n.

There exist different heuristic Microaggregation implementations in the literature [Domingo-Ferrer and Mateo-Sanz, 2002]. However, the one used in this thesis is the MDAV-generic (Maximum Distance to Average Vector) [Domingo-Ferrer and Torra, 2005] implementation shown in Algorithm 4. A good point for this algorithm is that it can work with any type of attribute, aggregation operator and distance (contiuous, ordinal or nominal).

Implementation of MDAV-generic for a particular attribute type requires specifying how the average record is computed and what distance is used. In our case, we deal with categorical data (nominal and ordinal types of data). We discuss below the ones used in this thesis.

The distance between two ordinal categories a and b of an attribute V_i, with $a < b$ is

$$dist_{ordinal}(a, b) = \frac{|\{i|a \leq i \leq b\}|}{|D(V_i)|} \tag{2.6}$$

that is, the number of categories separating a and b divided by the number of categories in the range of the attribute (the division is used to standardize the distance between 0 and 1). Regarding the average operator for ordinal attributes, we used the median. In terms of frequency, the median of a categories set S is the one such that its predecessors and successors in the ordered S have equal frequency.

Nominal attributes distance is defined using the equality predicate. Thus, the distance between two values of a nominal attribute is 0 if they are equal and 1 if they are not (See Equation 2.7). The plurality rule (or mode) is used as the average operator. This is, for a set $S = a_1, a_2, \ldots, a_n$, the most frequent value is selected as the average.

$$dist_{nominal}(a, b) = \begin{cases} 0, & if \quad a = b \\ 1, & if \quad a \neq b \end{cases} \tag{2.7}$$

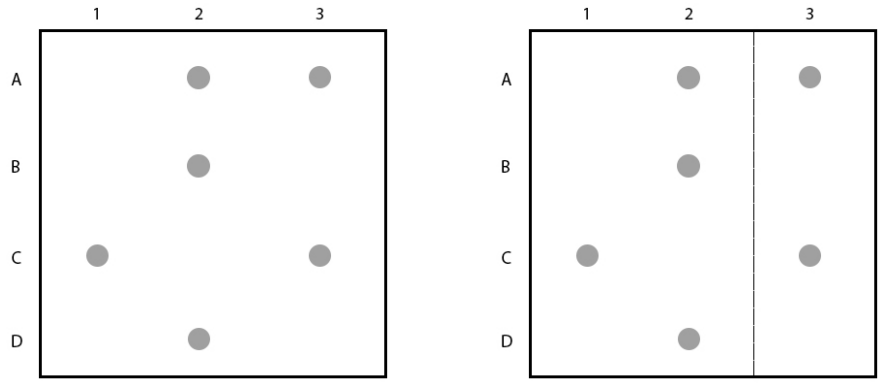
2.3.3 Mondrian

Mondrian is a greedy multidimensional recoding algorithm [LeFevre et al., 2006] for categorical variables. Like in the Microaggregation case, this method relies on the principle of k-anonymity, generating multidimensional partitions over the whole dataset dimensions until each region have the minimum number of elements $\geq k$. Then, the protected value of each attribute in a group is the interval of values taken by all the individuals of the group.

To generate these partition, this method relies on checking the existence of allowable cuts in any dimension. Consider a multiset P of points in d-dimensional space. A cut perpendicular to axis X_a at x_i is allowable if and only if $Count(P.X_a > x_i) \geq k$ and $Count(P.X_a \leq x_i) \geq k$. This is, a cut is allowable if there exist a value x_i in the axis X_a that separates the actual region in two new ones where each one of them contains at least k points.

Algorithm 4 MDAV-generic Algorithm

Input: X original dataset, k level of k-anonymity.
Output: X' partition of the original dataset records.
$i \Leftarrow 0$
$n \Leftarrow |X|$
$C \Leftarrow []$
while $n \geq 3k$ **do**
  $\bar{x} \Leftarrow$ compute centroid of remaining records in X
  $x_r \Leftarrow$ find the most distant record from $\bar{x}$
  $x_s \Leftarrow$ find the most distant record from x_r
  $Y_r \Leftarrow$ find $(k-1)$-nearest neighbours of x_r
  $Y_s \Leftarrow$ find $(k-1)$-nearest neighbours of x_r
  $c_i \Leftarrow$ form cluster with the elements Y_r and x_r
  $c_(i+1) \Leftarrow$ form cluster with the elements Y_s and x_s
  $C \Leftarrow$ add c_i
  $C \Leftarrow$ add $c_(i+1)$
  remove records in c_i from dataset X
  remove records in c_j from dataset X
  $n \Leftarrow n - 2k$
  $i \Leftarrow i + 2$
end while
if $n \geq 2k$ **then**
  $\bar{x} \Leftarrow$ compute centroid of remaining records in X
  $x_r \Leftarrow$ find the most distant record from $\bar{x}$
  $Y_r \Leftarrow$ find $(k-1)$-nearest neighbours of x_r
  $c_i \Leftarrow$ form cluster with the elements Y_r and x_r
  $C \Leftarrow$ add c_i
  remove records in c_i from dataset X
  $n \Leftarrow n - k$
  $i \Leftarrow i + 1$
end if
if $n > 0$ **then**
  $c_i \Leftarrow$ form cluster with the remaining elements in X
  $C \Leftarrow$ add c_i
  $n \Leftarrow n - n$
  $i \Leftarrow i + 1$
end if
$A \Leftarrow []$
for c_i in C **do**
  $a_i \Leftarrow$ aggregate attribute values of records in c_i
end for
$X' \Leftarrow$ substitue attribute value in X by the aggregated ones in A
return X'



(a) Initial space

(b) First cut

(c) Second cut

Figure 2.4: Visual example of Mondrian masking method with $k = 2$.

To illustrate the idea Figure 2.4 shows an example of achievent 2-anonymity ($k = 2$). On the left-hand side we have the spatial representation of two attributes (axis X and Y) with 3 and 4 categories in their domain. The points represent the records in the dataset with certain combination of values for those two attributes. The idea of the Mondrian method is to keep cutting all dimensions space (one at a time) until there are no more allowable cuts remaining in any dimension. In the middle of Figure 2.4 we show a spatial cut on the Y dimension. It results in two regions each one containing at least k points. Then, each region is treated independently and we try to cut both regions on the other (X) dimension. However, there is only one allowable cut shown in the right-hand side of Figure 2.4. With these two cuts we obtained three regions, and all of them contain 2 points. Now the attribute values of each point in the same region

are changed by the interval of values of all points in the region. For example, in the case of the bottom point, its original values are $\{X = 2, Y = D\}$ and the masked values would be $\{X = [1, 2], Y = [C, D]\}$.

2.3.4 Rank Swapping

The Rank Swapping method is a perturbative masking method which was originally described for ordinal variables [Moore Jr., 1996] but it can be used for any numerical variable.

Rank swapping with parameter p and with respect to an attribute V_j (i.e., the jth column of the original dataset X) can be defined as follows: first, the records of X are sorted in increasing order of the values x_{ij} of the considered attribute V_j. For simplicity, assume that the records are already sorted, that is $x_{ij} \leq x_{lj}$ for all $1 \leq i < l \leq n$. Then, each value x_{ij} is swapped with another value x_{lj}, randomly and uniformly chosen in the set of still unswapped values in the limited range $i < l \leq i + p$. Finally, the sorting step is undone.

Usually, when rank swapping is applied to a dataset, the algorithm explained above is run for each attribute to be protected, in a sequential way. The parameter p is used to control the swap range. Normally, p is defined as a percent of the total number n of records in X. Therefore, when p increases the difference between x_{ij} and x_{lj} may increase. This fact makes record linkage more difficult, but of course the information loss of the protected data is higher, decreasing in this way its statistical utility.

Original Dataset			Masked Dataset		
V_1	V_2	V_3	V_1'	V_2'	V_3'
8	9	1	10	10	3
6	7	10	5	5	8
10	3	4	8	4	2
7	1	2	9	2	4
9	4	6	7	3	5
2	2	8	4	1	10
1	10	3	3	9	1
4	8	7	2	6	9
5	5	5	6	7	6
3	6	9	1	8	7

Table 2.1: Rank Swapping example with $p = 2$

Table 2.1 shows an example of using the Rank Swapping method. We take the original dataset and apply Rank Swapping with $p = 2$ to the three attributes independently obtaining the right-hand side masked dataset. For the sake of simplicity records in original and masked dataset are equally sorted. However, they could be sorted differently in order to add even more uncertainty.

2.3.5 Global Recoding and Top/Bottom Coding

Global Recoding [Hundepool et al., 1998] is a non-perturbative masking method that can be seen as a function F over a categorical variable V_i which is transformed into a variable V_i' with $|D(V_i')| > |D(V_i)|$, where $D(V_i)$ is the domain of the variable V_i and $||$ the cardinality operator. In other words, this method combines several categories in order to form other new (less specific) categories.

This thesis uses a specific case of Global Recoding where only the p least frequent categories of a variable V_i are combined into a new category.

Top and Bottom Coding [Hundepool et al., 1998] are special cases of Global Recoding. In this case of Top Coding only the p first categories allowed for a variable V_i are combined to form a new one. Similarlly, in Bottom Coding only the p last categories allowed for an attribute V_i are combined to form a new one.

It is easy to see that Top and Bottom Coding methods are only suitable for ordinal attributes as they need to stablish an ordering of the categories to know which ones are the first and which ones the last.

2.4 Microdata Protections Evaluation Measures

To determine the protection and quality of a microdata dataset there are two main measures used in the microdata protection field: the information loss and the disclosure risk.

Information loss is known as the quantity of harm that is inflicted to the data by a given masking method. This measure is small when the analytic structure of the masked dataset is very similar to the structure of the original dataset. Then, the motivation for preserving the structure of the dataset is to ensure that the masked dataset will be analytically valid and interesting. In this work we used the *contingency table-based information loss* (CTBIL)[Domingo-Ferrer and Torra, 2001b], the *distance-based information loss* (DBIL)[Domingo-Ferrer and Torra, 2001b], and the *entropy-based information loss* (EBIL)[Kooiman et al., 1997].

Assessment of the quality of a protection method cannot be limited to information loss because disclosure risk has also to be measured. Disclosure risk measures the information that can be obtained about the individuals from the protected data set. This measure is small when the masked dataset values are very different to the original values. In this work we used the *interval disclosure* (ID)[Domingo-Ferrer and Torra, 2001a], the *distance-based record linkage* (DBRL)[Domingo-Ferrer and Torra, 2002], and the *probabilistic record linkage* (PRL)[Domingo-Ferrer and Torra, 2002].

The problem here is that both measures are inversely related so the higher information loss the lower disclosure risk, and the inverse. In order to perform a good protection there must be a minimized and balanced combination of both measures.

2.4.1 Information Loss

When evaluating a masking method for being valid for statistical studies the main thing that we want to mantain is the information stored in the original dataset in order to be able to perform studies over the masked dataset obtaining real original statistical information. So this property is evaluated with information loss measures.

Information loss occurs when the data is distorted and is more difficult to extract useful information from them, that is, some information is lost. However, this measure depends on how the protected data is going to be used, but there exist several kind of usages and it can be difficult to indentify all of them at the moment to publish the data.

This measure can lead to a bad evaluation of the dataset because if a protected dataset has a high amount of information loss, this dataset will be useless for any kind of statistical studies because the dataset has lost all its original properties.

In order to assess the preservation of the original dataset's structure we can try several things:

- Compare the data in the original and the masked datasets. The more similar the masking method to the identity function, the less impact (but the higher disclosure risk).
- Compare some statistics computed on the original and the masked datasets. Little information loss should translate to little differences between the statistics.
- Analyze the behavior of the particular masking method used to measure its impact on the structure of the original dataset.

In this chapter we present a review of the general measures used in this thesis to evaluate the information loss of the protection methods in our experiments. Those measures are the distance-based information loss (DBIL), the contingency table-based information loss (CTBIL), and the entropy-based information loss (EBIL).

Distance-Based Information Loss (DBIL)

The first way to evaluate the information distortion inflected by a data protection method to a certain dataset is measuring the difference between the initial values in the original dataset and the final values in the masked dataset [Torra and Domingo-Ferrer, 2001].

Then, the difference between the original dataset X and the masked dataset X' can be defined as the sum of the distances between each original register r_X and its correspondent protected register $r_{X'}$ (see Equation 2.8), where the distance between two registers is the sum of the distances between each original value and its correpondent masked value for all variables V_i (see Equation 2.9).

$$DIST(X, X') = \sum_{r \in X} d(r_X, r_{X'}) \tag{2.8}$$

$$d(r_1, r_2) = \sum_{V_i \in V} distValues_{V_i}(V_i(r_1), V_i(r_2)) \tag{2.9}$$

It should be noticed that it is needed to define a distance function for the two kind of variables when dealing with categorical data: ordinal variables and nominal variables. In order to compute these distances we used the distances described in Equations 2.6 and 2.7 of Section 2.3.

Contingency Table-Based Information Loss (CTBIL)

The second way to evaluate the information loss in a protected dataset is by comparing all the contingency tables from 1 to K dimensions of the original dataset and the protected dataset [Torra and Domingo-Ferrer, 2001]. In other words, given two datasets X and X' (original and protected respectively) and a set of variables V, there will be generated as much contingency tables as different possible combinations with up to K elements are in V.

Then, the contingency table-based information loss can be defined as the sum of the differences between the same cell in both original and masked contingency tables. Equation 2.10 formalizes this concept where X is the original dataset, X' is the masked dataset, V is the set of variables, K is the maximum number of dimensions for the tables, and a_i^t is the value in table t accessible by the indices i .

$$CTBIL(X, X'; V, K) = \sum_{\substack{\{V_{j1} \dots V_{jt}\} \subseteq V \\ |\{V_{j1} \dots V_{jt}\}| \leq K}} \sum_{i_1 \dots i_t} |a^X_{i_1 \dots i_t} - a^{X'}_{i_1 \dots i_t}| \tag{2.10}$$

The main problem for this method is that the number of contingency tables depends on the number of variables $|V|$, the number of categories for each variable, and the dimension K. Because of this, it could be very difficult to compare the results in tables with large number of categories with tables with only few categories, and the same would happen with the dimension of these tables. To solve this, it was proposed a normalized version dividing the CTBIL result by the total number of cells in all considered tables as shown in Equation 2.11.

$$ACTBIL(X, X'; V, K) = \frac{CTBIL(X, X'; V, K)}{\sum_{\substack{\{V_{j1} \dots V_{jt}\} \subseteq V \\ |\{V_{j1} \dots V_{jt}\}| \leq K}} |D(V_{j1})| \dots |D(V_{jt})|} \tag{2.11}$$

In this thesis we used this normalized version of the contigency table-based information loss to evaluate the method in the experiments part.

Entropy-Based Information Loss (EBIL)

The last measure used to evaluate the information loss is based on measuring the entropy of the protected data respect to the original data [Torra and Domingo-Ferrer, 2001]. This approach interprets the masking method as a noisy channel through where information is transmitted. With this interpretation, the loss of information can be calculated as the quantity of noise or disorder between the original dataset and the masked dataset.

Let V be a variable in the original dataset and V' be the corresponding variable in the masked dataset, let $P_{V,V'} = \{p(V = i|V' = j)\}$ be the conditional probabilities matrix, and let S be the set of categories for the variable V. Then the information loss for the variable V in a register r would be

$$H(V|V' = j) = -\sum_{i,j \in S} p(V = i|V' = j) log p(V = i|V' = j) \tag{2.12}$$

Finally, the entropy-based information loss (EBIL) is obtained by accumulating the results of expression 2.12 for all registers r in the masked dataset X' as follows

$$EBIL(P_{V,V'}, X') = \sum_{r \in X'} H(V|V' = j_r) \tag{2.13}$$

It should be noticed that Expression 2.13 only calculates the information loss for a single variable. In order to obtain the global information loss for a set of variables, the result of each variable have to be added together.

2.4.2 Disclosure Risk

We presented above different ways of measuring he information loss caused by microdada protection methods. However, the assessment of the quality of a protection method cannot be limited to information loss; disclosure risk must also be measured.

Disclosure risk is the measure to evaluate the protection degree of a masked dataset, that is, it measures the quantity of original sensitive data that can be obtained from the masked dataset.

Like in the case of information loss, this measure can also lead to a bad evaluation of the dataset because if a protected dataset has a high amount of disclosure risk means that there is almost no protection and most of the sensitive information about the individuals can be either infered or linked to the users, loosing then their privacy. However, it is necessary to find the optimum trade-off between information loss and disclosure risk. To understand the trade-off, consider the two extreme cases between which masking methods lie:

- If a masking method alterates so much the original dataset, then most of the original information is lost but no sensitive information is disclosed.

- If a masking method do not alterate the original dataset, then the original information is completely mantained but all sensitive information is disclosed.

Then, in this chapter we present a review of the disclosure risk used in this thesis to evaluate the disclosure risk of the protection methods in our experiments. The reviewed methods are interval disclosure (ID), distance-based record linkage (DBRL), probabilistic record linkage (PRL), and rank swapping record linkage (RSRL).

Interval Disclosure

This first approach was described in [Domingo-Ferrer and Torra, 2001b] and it only checks the quantity of individual original values that can be discovered from the masked dataset.

Each variable is independently ranked and a rank interval is defined around the value the variable takes on each record. The ranks of values within the interval for a variable around record r should differ less than p percent of the total number of records and the rank in the center of the interval should correspond to the value of the variable in record r. If so, the proportion of original values that fall into the interval centered around their corresponding masked value is a measure of disclosure risk.

Distance-Based Record Linkage (DBRL)

This approach was described first in [Domingo-Ferrer and Torra, 2002] for the specific case of microaggregation masking method with continuous data using the Euclidean distance. It can be generalized, however, for any method provided that a distance between the original and the masked value can be defined.

It is assumed that an intruder has an external dataset containing as key variables the same variables present in the released masked dataset. It is also assumed that the intruder is trying to link the masked dataset with the external dataset using the key variables.

The main idea of this measure is to compute the distance between a protected record and all the original records, getting the closest original record. A record in the masked dataset is labeled as linked when this nearest record in the original dataset turns out to be the corresponding original record. Then, the percentage of linked records is a measure of disclosure risk.

Because of this thesis is focused on categorical microdata files it was needed to define the distance between categorical values in this method. As it has been used in the distance-based information loss, these distances have been splitted in two types: one distance for ordinal variables and one distance for nominal variables. In order to compute these distances we used the distances described in Equations 2.7 and 2.6 of Section 2.3.

Probabilistic Record Linkage (PRL)

This method was described in [Domingo-Ferrer and Torra, 2002] having a

matching algorithm which uses the linear sum assignment model to pair records in the two files to be matched, original and masked datasets.

Although, this method is less simple than the explained in the previous section, this approach is attractive because it requires the user to provide only two probabilities as input: an upper bound of the probability of a false match and an upper bound of the probabiltiy of a false non-match. Then, a probabilistic index is computed for each pair or registers and having two bounds it is decided if the pair of records is linked or not.

Finally, the percentage of correctly paired records is a measure of disclosure risk.

2.5 Clustering Partitions Similarity Measures

There exist several techniques to deal with the problem of clustering partitions comparison. In this thesis we use three well known indices which are focused on checking whether two partitions contain the same number of clusters and each cluster contains the same elements.

We consider that $\Pi = \{\pi_1, \ldots, \pi_m\}$ and $\Pi' = \{\pi'_1, \ldots, \pi'_m\}$ are the resulting partitions for the original and masked datasets. Then, we consider r, s, t, u and $np(\Pi)$ as follows:

r is the number of pairs (a, b) where a and b are in the same cluster in Π and in Π'.

s is the number of pairs (a, b) where a and b are in the same cluster in Π but not in Π'.

t is the number of pairs (a, b) where a and b are in the same cluster in Π' but not in Π.

u is the number of pairs (a, b) where a and b are in different clusters in Π and in Π'

$np(\Pi)$ is the number of pairs within clusters in the partition Π

The measures we use are the Rand index [Rand, 1971], the Jaccard index [Tan et al., 2005] and the Wallace index [Wallace, 1983], which using the above parameters are defined as follows:

Rand index

$$RI(\Pi, \Pi') = \frac{r + u}{r + s + t + u}$$

Jaccard index

$$JI(\Pi, \Pi') = \frac{r}{r + s + t}$$

Wallace index

$$WI(\Pi, \Pi') = \frac{r}{\sqrt{np(\Pi)np(\Pi')}}$$

2.6 Social Networks Users Privacy Protection Methods

Social Networks can be thought as graphs where each of the nodes represents a unique user and the links that join two nodes represents a relationship between two users. In a scenario like that there are two main privacy issues to be concerned about. The first one is the information contained by each node (user) of the graph, and the second one is the relationships that each node has with other nodes. Those two issues are the ones that are able to disclose the identity of the people or institutions contained in the social graph. Figure 2.5 shows an example of a tiny social network graph. The relations between nodes (people) represents friendships, then, for example, it can be seen that Alice is friend of everybody but Harry has only Alice as a friend.

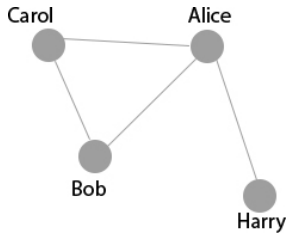


Figure 2.5: Social Graph Example

In the literature there exist several protection methods to protect relationships avoiding the disclosure of nodes identity. The idea here is that nodes that look structurally similar may be indistinguishable to an adversary, in spite of external information. A strong form of structural similarity between two nodes is automorphic equivalence. Two nodes $x, y \in V$ are atomorphically equivalent if there exist an isomorphism from the graph onto itself that maps x to y. Most graphs have small automorphism classes, likely to be insufficient for protection against re-identification. However, in order to distinguish two nodes in different automorphic equivalence classes it may be necessary to use complete information about their position in the graph but adversaries are unlikely to have access to such complete information. For example if a weaker adversary only knows the degree (number of relationships) of targeted nodes (users), *Carol* and *Bob* nodes of Figure 2.5 are indistinguishable.

The most important family of privacy methods dealing with graph structure are the ones based on the k-anonymity principle. These methods try to add, remove, or generalize edges in order to have at least k equal nodes or structures in the graph.

In [Stokes and Torra, 2012] we find a theoretical definition of the classical $k-anonymity$ adapted to graphs and they propose to create clusters of edges and nodes in order to create superedges and supernodes (a single edge/node representing all the elements in the cluster it belongs to). Another example can be found in [Campan and Truta, 2009] where the authors describe a practical protection method which takes into account the information loss when creating clusters (using the nodes information) and creates also superedges and supernodes. In [Zhou and Pei, 2008] it can be found another more sophisticated anonymization method which firstly generalizes vertex labels and secondly adds edges with the precept of creating local topologies which are isomorphic with other local topologies. Finally, in [Liu and Terzi, 2008] the authors propose a defense method which produces k-degree anonymized degree sequences.

There is another family of methods wich rely on randomization when adding and removing edges like in [Hay et al., 2007] where the authors describe a protection method which relies on adding and removing edges at random (first removing n edges at random and afterwards adding n edges at random).

2.6.1 Privacy Analysis Measures

In order to assess the quality of the protections here there also exist a modified approaches of the two kind of measures described in Section 2.4: information loss and disclosure risk.

As information loss there are several different approaches used in the literature. In the case of structural information loss the most widely used measures are common graph metrics such as clustering coefficient, average path length and degree distribution [Nettleton et al., 2011]. Comparing these measures of the protected graph with the ones in the original graph gives us a hint of the amount of structural information lost during the protection.

For the disclosure risk case almost all existing approaches are based on the number of correct matches between nodes or structures in the original graph and the ones in the protected graph. In [Hay et al., 2007] the authors also consider the risk measure as the percentage of nodes whose equivalent candidate set falls into one of a given set of buckets (1 node, 2-4 nodes, 5- 10 nodes, ...).

In [Shetty and Adibi, 2005] the authors present some concepts related to graph entropy and the identification of important or interesting nodes. The basic idea is to measure the effect of removing a node from a graph, as the difference between the entropy of the graph before and after removing the given node

2.7 Datasets

In this section we introduce the main datasets we used through most of the experiments in this thesis.

The 1993 U.S. Housing Dataset

The first dataset is the 1993 U.S. Housing Survey dataset. This dataset was

extracted from the U.S. Census Bureau [U.S. Census Bureau, 1993] and it contains information about the size and the composition of the U.S. houses inventory on 1993. This dataset consists on 1000 registers represented in terms of 11 categorical attributes and 3 continuous attributes. Table 2.2 show the attributes in this dataset as well as the number of valid categories for each one.

The German Credit Dataset

The second dataset is the 1994 German Credit Data dataset. It was extracted from the UCI Machine Learning Respository [Bache and Lichman, 2013] and it describes german people financial aspects. This dataset has 1000 registers with 7 numerical attributes and 13 categorical attributes. Table 2.3 show the attributes in this dataset as well as the number of valid categories for each one.

The Solar Flare Dataset

The third dataset is the Solar Flare dataset and it was also extracted from the UCI Machine Learning Repository [Bache and Lichman, 2013]. This dataset contains information about 1389 different solar flares described with 10 categorical attributes. Table 2.4 show the attributes in this dataset as well as the number of valid categories for each one.

The Adult Dataset

The last dataset is the Adult dataset from the UCI Machine Learning Repository [Bache and Lichman, 2013]. This dataset is also known as "Census Income" dataset because it contains personal information about people such as education, occupation and marital status. In addition it also contains information about the income of this people. The dataset consists on 48842 registers with 14 attributes each one (6 continuous and 8 categorical). Table 2.5 show the attributes in this dataset as well as the number of valid categories for each one.

Attribute Name	Number of Categories
UNKNOWN	Continuous
AGE	Continuous
BUILT	24
DEGREE	7
GRADE1	20
METRO	9
SCH	6
SHP	6
SMSA	Continuous
TRAN1	Continuous
WEIGHT	Continuous
WFUEL	10
WHYMOVE	19
WHYTOH	13
WHYTON	13

Table 2.2: 1993 U.S. Housing Dataset Attributes

Attribute Name	Number of Categories
EXISTACC	5
DURATION	Continuous
CREDHIST	6
PURPOSE	12
CREDAMOUNT	Continuous
SAVINGS	6
PRESEMPLOY	6
INSTALRATE	Continuous
STATSEX	6
OTHER	4
RESIDENCE	Continuous
PROPERTY	5
AGE	Continuous
OTHERPLANS	4
HOUSING	4
NUMCREDITS	Continuous
JOB	5
PEOPLE	Continuous
TELEPHONE	3
FOREIGN	3

Table 2.3: German Credit Dataset Attributes

Attribute Name	Number of Categories
CLASS	8
LARGSPOT	7
SPOTDIST	5
ACTIVITY	3
EVOLUTION	4
PREVFLARE	4
HISTOCOMP	3
SUNDISK	3
AREA	3
AREALARGSPOT	3
C-CLASS	10
M-CLASS	10
X-CLASS	10

Table 2.4: Solar Flare Dataset Attributes

Attribute Name	Number of Categories
AGE	Continuous
WORKCLASS	9
FNLWGT	Continuous
EDUCATION	17
EDUCATION-NUM	Continuous
MARITAL-STATUS	8
OCCUPATION	15
RELATIONSHIP	7
RACE	6
SEX	3
CAPITAL-GAIN	Continuous
CAPITAL-LOSS	Continuous
HOURS-PER-WEEK	Continuous
NATIVE-COUNTRY	42
INCOME	Continuous

Table 2.5: Adult Dataset Attributes

Chapter 3

Evolutionary Approach for Better Microdata Protections

This chapter explains how to see as an optimization problem the fact of seeking for better microdata protections, and introduces some approaches to solve it using evolutionary algorithms.

Seeking a good protection that has low information loss and low disclosure risk is a difficult task. There exist many protection methods, each of them having different parameters to tweak. However, the protection process can be thought as a function that takes the original data set and generates a new data set which should be optimized with respect to the concept of good protection (low information loss and low disclosure risk). Then, the protection process can be modelized as an optimization problem and can be solved using state-of-the-art optimization methods but, as the protection process is a very unknown and difficult task and the search space is large (the product of the domains size of all atributes to protect), this optimization problem is not suitable to be solved using analytical methods.

Evolutionary algorithms are a good choice in this case because, as explained in Chapter 2, they work well in this kind of situations. In the following sections we introduce a way to use an evolutionary algorithm to get better protections. Section 3.1 describes the general algorithm. In Section 3.2 we show how to perform the genotype encoding. Section 3.3 introduces the genetic operators. Section 3.4 describes the fitness function used to guide the algorithm to evolve towards a better protectisons. Finally, in Section 3.5 experimental results are presented to show the performance of our approach.

3.1 General Evolutionary Protection

In this section we present how to use an evolutionary algorithm as the protection method. As said above, we are going to look at the protection problem as an optimization problem using an evolutionary algorithm.

The main idea of this approach is to have an initial population of different protections of the same original data set (using different state-of-the-art protection methods) and then keep optimizing them based on a fitness function containing the integration of the information loss and the disclosure risk measures. Although in traditional evolutionary algorithms all genetic operators have chances to be executed in each generation, in our case we decided to apply either mutation or crossover in a single generation, not both. The reason for this is that we do not want to harm the data too much in each generation and also want the evolution to be smoother.

By doing this we are combining different parts of the same data but protected with different methods. In addition, individuals also experience mutations which tries to find new perturbations on the data that provides better fitness than the ones done by the state-of-the-art protection methods.

The only open feature in this algorithm is the stopping criteria. This criteria is up to the user and, for example, it could be to reach a maximum number of generation, to achieve a certain level of improvement (in terms of fitness) for the best individual... Then, after executing the algorithm, it returns the best individual (protection) in the population based on the fitness of all individuals (protections).

Pseudo-code summarizing the algorithm is provided in Algorithm 5 below.

Algorithm 5 Evolutionary Algorithm to Enhance Privacy.

Input: $P(0) = \{X'_i\}$ initial population of protections for X.
Output: $P(t) = \{X'_j\}$ generation t.
$t \Leftarrow 0$
evaluate($P(0)$)
while stopping($P(t)$) $\neq$ `true`; **do**
  alter $\Leftarrow$ randomly choose between mutation and cross
  if alter by mutation **then**
    $i \Leftarrow$ select(N)
    $X'_j \Leftarrow$ mutate(X'_i) evaluate(X'_i, X'_j)
  else
    $\{i_1, i_2\} \Leftarrow$ select(N_b, N)
    $X'_{j_1}, X'_{j_2} \Leftarrow$ cross(X'_{i_1}, X'_{i_2}) evaluate($X'_{i_1}, X'_{i_2}, X'_{j_1}, X'_{j_2}$)
  end if
  $t \Leftarrow t + 1$
end while
return *best*($P(t)$)

In the following sections we describe the key aspects of this algorithm such

x'	x	graycode(x')	graycode(x)	genome($x'\|x$)
1500	1000	111 0011 0010	10 0001 1100	101 0010 1110
1500	1900	111 0011 0010	100 1101 1010	11 1110 1000
12	7	1010	100	1110
7	12	100	1010	1110

Table 3.1: Some Examples of Genome Representations.

as the genotype econding, the genetic operators, the fitness function, and the selection criteria.

It should be noticed that this approach is suitable for both continuous and categorical data. The only differences are how the genotype is encoded and how the mutation genetic operator is performed. Both cases are explained in each of the following sections too.

3.2 Genotype Encoding

Our search of good individuals is in direct correspondence with the search of perturbative masking protections offering a good tradeoff between information loss and disclosure risk.

In the case of continuous data encoding of an individual X' is done value by value, relative to the corresponding values at the original data set X. For each value x' in the masked data set X', let x be its associated value in the original data set X. Consider that graycode(x) and graycode(x') represent the Gray codes of x and x' as in [Caruana and Schaffer, 1988]. Then the genomic representation of x' *given* x is taken as the *bitwise xor* between the Gray codes, graycode(x) and graycode(x').

$$\text{genome}(x'|x) = \text{xor}(\text{graycode}(x'), \text{graycode}(x)) \tag{3.1}$$

We have chosen to work with Gray-coded values because its hability to obtain fast and more accurate solutions than regular binary representations as explained in Section 2.1.

Recall that the representation of a value x' is computed with respect to its original version x, so what is encoded is the *perturbation* that changes each initial value into its masked version. Interestingly enough, if there were no perturbation at all for that specific value, so $x' = x$, then the genomic representation of x' given x (defined by Equation 3.1) equals zero. In general, the length of *genome*($x'|x$) is directly proportional to the strength of the perturbation, so weaker perturbations have lower *lengths*. Here length is understood as the position of the most significant bit. Figure 3.1 shows some examples of genome representations.

An important property of this representation is that all decoded values are feasible. When an evolutionary algorithm produces *unfeasible solutions*, an ad-

ditional *repairing* step is usually added to the algorithm to obtain a feasible version [Back et al., 2000]. All representations being feasible generally imply a significant advantage of not requiring the repair algorithm step.

A complete file encoding example is shown in Figure 3.1. The example includes the original and the protected file as well as the genome file. All files needed to construct the genome file are also included.

In the categorical data case the attribute values only have meaning in the form of a string and, in addition, its meaning can only be modified by changing the entire string, so partial modifications of the string can generate categories out of our domain. For that reason we decided to deal with the original categories directly without any type of encoding, this is, the chromosomes of method's population are just the protected data-files read and loaded into memory, where the genes are the string values.

Regarding the space complexity of our approach, it will be determined by the number of registers n in the data, the number of attributes a, and the number of files loaded into memory f, obtaining a space complexity of $O(n \times a \times f)$.

3.3 Genetic Operators

Our proposed algorithm uses two basic operators: crossover and mutation [Holland, 1975].

Crossover operation is identical in both continuous and categorical cases. The crossover of two masked data sets X' and Y' is performed by a 2-point crossing as follows. Take a value position at random s as the first point, and consider that the two values at this position are $x'_s \in X'$ and $y'_s \in Y'$. Take another value position at random between the value taken before and the last value position of the data set. Set r as this second point, and consider the two values at this position to be $x'_r \in X'$ and $y'_r \in Y'$. When $s = r$ there is only one value selected, so only this value will be swapped obtaining two new offsprings $Z'_1 = \{x'_1, \ldots, x'_{s-1}, y'_s, x'_{s+1}, \ldots, x'_n\}$ and $Z'_2 = \{y'_1, \ldots, y'_{s-1}, x'_s, y'_{s+1}, \ldots, y'_n\}$. When $s \neq r$ all values between the two positions have to be swapped obtaining two new offsprings $Z'_1 = \{x'_1, \ldots, x'_{s-1}, y'_s, y'_{s+1}, \ldots, y'_r, x'_{r+1}, \ldots, x'_n\}$ and $Z'_2 = \{y'_1, \ldots, y'_{s-1}, x'_s, x'_{s+1}, \ldots, x'_r, y'_{r+1}, \ldots, y'_n\}$.

Mutation consists on altering a single value in an individual and, as the representation of continuous data is different than the one of categorical data, we defined a specific mutation operator for each case.

For continuous data, the mutation operation is performed as follows: given an individual X' (i.e. a protected data set), take a value position at random, and consider that the value at this position is x', with genome$(x') = b_s b_{s-1} \ldots b_1$. Choose a bit position k at random, such that $1 \leq k \leq s$. Then a new offspring is obtained just by replacing the bit b_k by its negation counterpart, $b'_k = not(b_k)$.

In the categorical data case, values cannot be modified internally but we also want to obtain a new offspring. In addition, we have also to deal with the constraint of that each variable have a limited number of categories admitted as a valid values, and we need to take it into account when altering them. So,

1	3	1	5	9
5	2	2	6	7
4	8	3	4	9
1	6	5	8	2
4	7	2	1	2

Original file.

1	2	1	5	7
9	1	2	3	5
4	1	9	8	9
1	6	7	9	1
3	1	2	7	2

Protected file.

0001	0011	0001	0101	1001
0101	0010	0010	0110	0111
0100	1000	0011	0100	1001
0001	0110	0101	1000	0010
0100	0111	0010	0001	0010

Binary-coded original file.

0001	0010	0001	0101	0111
1001	0001	0010	0011	0101
0100	0001	1001	1000	1001
0001	0110	0111	1001	0001
0011	0001	0010	0111	0010

Binary-coded protected file.

0001	0010	0001	0111	1101
0111	0011	0011	0101	0100
0110	1100	0010	0110	1101
0001	0101	0111	1100	0011
0110	0100	0011	0001	0011

Gray-coded original file.

0001	0011	0001	0111	0100
1101	0001	0011	0010	0111
0110	0001	1101	1100	1101
0001	0101	0100	1101	0001
0010	0001	0011	0100	0011

Gray-coded protected file.

0000	0001	0000	0000	1001
1010	0010	0000	0111	0011
0000	1101	1111	1010	0000
0000	0000	0011	0001	0010
0100	0101	0000	0101	0000

Genomes file.

Figure 3.1: Example of genotype encoding.

we decided to define the mutation operation as follows. Given an individual X' (i.e. a protected data set), it is mutated by randomly selecting a value x_i and changing it by a randomly selected value among all valid values for the specific attribute v_i.

3.4 Fitness Function and Selection

As explained in the previous chapter, the fitness function is the responsible of guiding the evolutionary algorithm to keep optimizing the individuals in the population. In our case we are dealing with a population of protections so there are two measures to take into account when determining the protection quality: the information loss (IL) and the disclosure risk (DR).

As we are dealing with two objective measures, the general protection problem is a multi-objective optimization problem. Then, fitness evaluation function is based on the multi-objective optimization method of Objective Weighting with $\frac{1}{2}$ as the weight for disclosure risk (DR) evaluation function and also for information loss (IL) evaluation function, so the individual score can be obtained as follows:

$$\text{Score}(X') = \frac{\text{DR}(X') + \text{IL}(X')}{2} \tag{3.2}$$

Here $\text{DR}(X')$ represents the disclosure risk of X' with respect to X as defined by [Domingo-Ferrer and Torra, 2001b], and $\text{IL}(X')$ represents the information loss of X' with respect to X as defined by [Mateo-Sanz et al., 2005]. The closer the fitness $\text{Score}(X')$ is to zero, the better the protection provided by the masking X'.

Consider that the current population $P(t) = \{X'_1, X'_2, \ldots\}$ is sorted by the score function, with $Score(X'_i) \leq Score(X'_j)$ whenever $i \leq j$. Given a fixed parameter N corresponding to the population size, the selection method of Algorithm 5 filters the best N_b individuals in terms of its score value, and selects an individual X'_i with probability $p(X'_i)$ as given by Equation 3.3.

$$p(X'_i) = \frac{Score(X'_i)}{\sum_{j=1}^{N} Score(X'_j)} \tag{3.3}$$

Expression 3.2 has been used in several papers [Domingo-Ferrer and Torra, 2001b][Marés and Torra, 2010] . According to this expression, the best protection is achieved with a minimum value for each measure (i.e., IL=0 and DR=0). Nevertheless, given a particular score we prefer to have the same value in both scores. That is, for a score of 20%, we prefer IL=20 and DR=20 than IL=0 and DR=40. Expression 3.2 works fine in the case of continuous data because the altered values in the individuals have less impact on the information loss and disclosure risk as there is a large (sometimes infinite) range of possible values to take by a continuous attribute. However, in the categorical case the changes have much more impact because of the small

quantity of valid categories in each attribute. Because of this, the changes tend to go from a big disclosure risk and low information loss to a low disclosure risk and big information loss so, a pair of values with $score = 20$ are much likely to have $IL = 30$ and $DR = 10$ than to have $IL = DR = 20$ what would be desirable.

To better represent our choices in the categorical case, we present an alternative function that penalizes such unbalanced trade-offs between information loss and disclosure risk. The expression is the maximum of information loss and disclosure risk as follow

$$Score(X) = max(IL(X), DR(X)) \tag{3.4}$$

This second function penalizes a protected dataset that has a large unbalance between disclosure risk and information loss. Note that just one bad value of IL or DR leads to a bad score.

Regarding the individual(s) selection criteria, Equation 3.3 is characteristic of a proportional selection strategy, where individuals are evaluated and selected based on this fitness function [Back et al., 2000, Holland, 1975]. With proportional selection, better individuals have a greater probability of being selected.

In the mutation case, an individual X'_i is chosen from the current population, from which a potentially new individual is obtained (as described in Section 3.3 above). During the evaluation, an elitism replacement strategy is followed, which means that the two individuals are compared and only the individual with the best fitness value survives. The use of the elitism replacement strategy is to guarantee that the next generation individual will be at least not worse than the actual, then, it can prevent a loss of the best solution found.

In the crossover case, two individuals X'_{i_1} and X'_{i_2} are chosen. X'_{i_1} is selected from a leader group with the N_b best scores. The second individual X'_{i_2} is chosen from the population. A recombination of these two individuals produce two new individuals, X'_{j_1} and X'_{j_2} (as described in Section 2.1 above). In our case, each newcomer X'_{j_k} maintains a proximity relation with its parent X'_{i_k}. During the evaluation, an elitist niching method — known as Deterministic Crowding (DC) [Dick, 2005, Mahfoud, 1992] — is followed such that only individuals with the best fitness value in each pair (X'_{i_k}, X'_{j_k}) survive. Should be noticed that this method is effective in maintaining the diversity of the population in terms of genotypic search space, but it does not necessarily guarantee the diversity of maskings.

3.5 Experimental Results

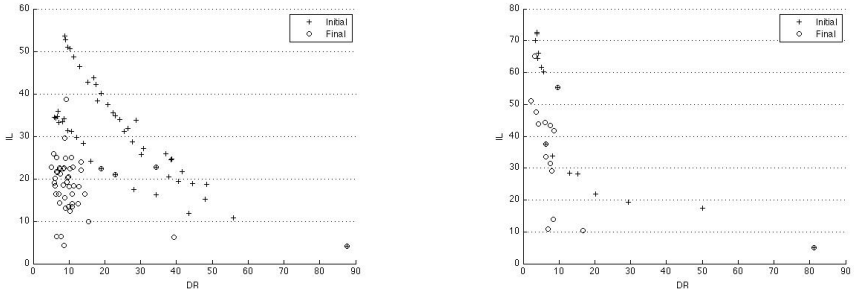
In this section we show some experimental results in order to test the performance of our approach. Although we are mainly focused on the protection of categorical data, here we wanted to extend it to continuous data too. Then, we present two different scenarios having in first place the experimental results for the continuous data case and then presenting the ones for the categorical data case.

Continuous Data Case Results

To illustrate and empirically evaluate our proposed method we have used four different data sets. The first data set is a Census [Brand et al., 2002] extracted from the U.S. Current Population Survey corresponding to 1995, and consists of 1080 records with 13 continuous attributes. This particular data set is commonly used to empirically evaluate different privacy-protecting methods [Nin et al., 2008b, Domingo-Ferrer and Torra, 2001b, Domingo-Ferrer and Torra, 2004, Domingo-Ferrer and Torra, 2005, Yancey et al., 2002]. The second data set is Tarragona which contains 834 records with 13 numerical attributes corresponding to financial information on 834 companies located in the area of Tarragona, Catalonia. This data set is also very common used in this field like in [Domingo-Ferrer and Mateo-Sanz, 2002], and [Laszlo and Mukherjee, 2005]. The third data set is EIA [Brand et al., 2002] which is an electric utility data file that includes utility level retail sales of electricity and associated revenues by end-use sector, state, and reporting month, and consists of 4092 records with 10 continuous attributes, but we have only used 2000 records. Finally, the fouth data set is Diabetes (UCI Machine Learning Repository, [Bache and Lichman, 2013]), which contains information from diabetes patients, and consists of 768 records with 8 continuous attributes.

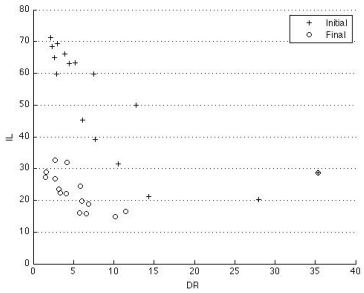
Our proposed method *a priori* applies to *any* particular set of protection masks, from which it starts searching new protected datasets by evolution of the initial set. But of course it is better to start from a *good* set of protections.

In our experiment we consider a leader group size $N_b = 2$. For the Census data set we take as initial population 50 masks including rank swapping (RS) and microagregation protection methods. This particular set of masks was obtained by means of the QJ algorithm [Jiménez and Torra, 2009a, Jiménez and Torra, 2009b]. Details about the QJ algorithm are not relevant for our experiment, except that it provides a diversity of masks with different levels of protection, from which we take the best 50 masks — in terms of its score — as initial individuals. Ranking protections of the Census data by its score, RS protections achieved the highest position with a score equal to 20.68546, followed in the score ranking by QJ and Microaggregation, that obtained similar results [Domingo-Ferrer et al., 2001, Jiménez and Torra, 2009a]. But these scores have to be considered with caution because, recently, specific attacks to RS and to Microaggregation were discovered that allow to establish more links than with the standard record-linkage techniques used to compute disclosure risk [Nin et al., 2008a, Nin et al., 2008b]. In the case of the Diabetes data set we take 16 masks including rank swapping and microagregation methods. For the EIA data set we take 16 masks including rank swapping and microagregation methods. Finally for Tarragona data set we take 16 masks including rank swapping and microagregation

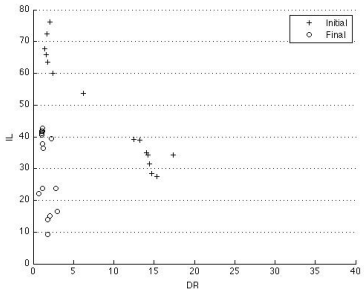


(a) Census dataset

(b) Diabetes dataset



(c) EIA dataset



(d) Tarragona dataset

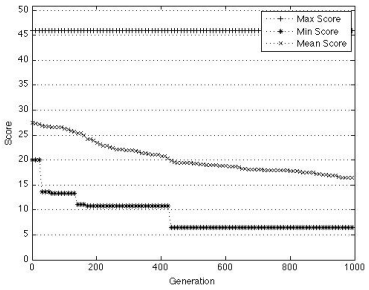
Figure 3.2: Initial and Final Populations of all datasets for the First Experiment.

methods as well. For Diabetes, EIA and Tarragona data sets we have used protections with parameters similar to the Census masks.

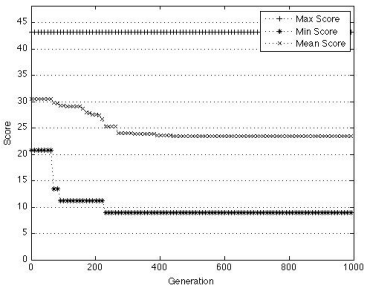
Figures 3.2a to 3.2d are R–U confidentiality maps [Duncan et al., 2001a, Duncan et al., 2001b] of the evolution of protections for each data set. A R-U map is nothing more than a two-dimensional Cartesian system where the abscissa is the disclosure risk DR and the ordinate is the information loss IL provided by each protection. Recall that the origin at the lower-left corner corresponds to an ideal but unfeasible protection with score zero, where there is no distortion (IL is zero) but also there is no disclosure risk (DR is zero). For our experiment, the closer a point is to the origin, the better is the protection it represents.

Initial and final population masks are represented by crosses and circles respectively. It can be observed, first, that the overall set of masks improves in terms of its score. Second, we can also visually appreciate that diversity is maintained, while protections improve both in terms of information loss (IL) and disclosure risk (DR). This was intentional because, as we said in Section 3.4, our algorithm is based on Deterministic Crowding (DC), which is basically a diversity maintenance technique. This gives

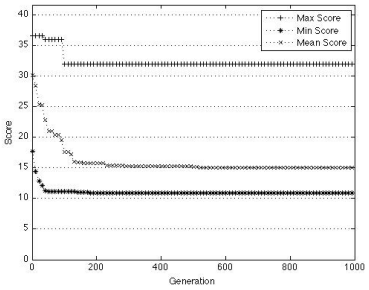
the flexibility of choosing between different protections with approximately the same score but different levels of protection. Let us develop this point further with an example. Consider three hypothetical protections with DR-IL coordinates $(30, 10)$, $(20, 20)$ and $(2, 38)$. These three protections all have the same score 20 but different values of information loss (IL) and disclosure risk (DR). The first point has the highest disclosure risk, that might be excessive. The third has the highest information loss, that might also be unacceptable for some users. Points in between represent a balance between both DR and IL measures, which may be preferable for some users. Finally, we observe at Figure 3.2a, Figure 3.2b and Figure 3.2d masks that have achieved scores very close to zero, which is a very good improvement because it means that those masks have low information loss and low disclosure risk, that is almost the ideal result.



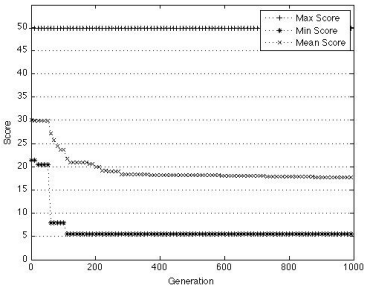
(a) Census dataset



(b) Diabetes dataset



(c) EIA dataset



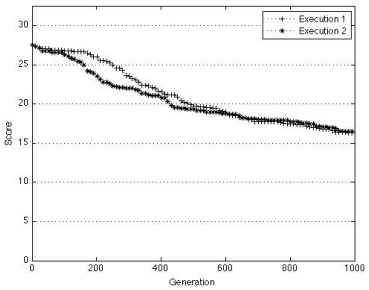
(d) Tarragona dataset

Figure 3.3: Evolution of the population of the First Experiment for all datasets.

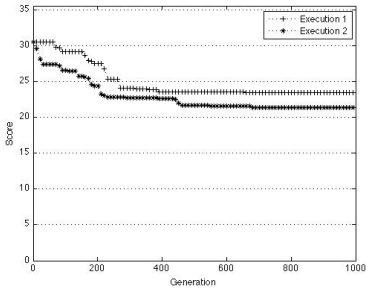
Figure 3.3a summarizes the evolution of scores for the experiment with the Census data set, Figure 3.3b represents it for the Diabetes data set, Figure 3.3c for the EIA data set, and Figure 3.3d for the Tarragona data set. The three curves from top to bottom show, respectively, the maximum, average and minimum scores, as population evolved from generation

to generation. The upper line shows the evolution of the maximum score, which decreased at a very low pace. This is explained by the selection process itself, that gives preference to best individuals, so worst protections in terms of score have lowest probability to be selected. The line in the middle shows how the score average decreased, which means that many individual protections were enhanced in terms of information loss and disclosure risk. Then, in the Census data set, the lower line shows how the minimum score drops from 20.0802 to 6.4298, decreasing by more than three times the best score of the initial population after 1000 generations. In the case of the Diabetes data set the minimum score evolution drops from 20.7190 to 8.8885, which is a decrease of more than half. For EIA data set minimum score evolution drops from 17.7906 to 10.8979 which is a decrement of 38.74% of the initial minimum score, representing nearly the ideal result. Finally, for Tarragona data set, the minimum score evolution drops from 21.4104 to 5.5862 which is a decrement of $73,91\%$.

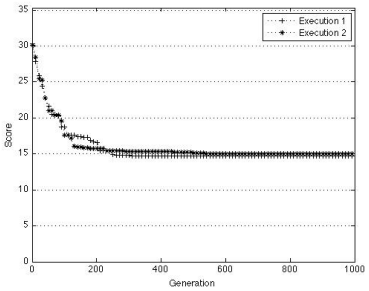
It is important to recall here that a low information loss means that the properties represented in the IL expression are kept. They are statistics of the data as means and correlations. Other statistics or properties not included in the expression for IL might be corrupted.



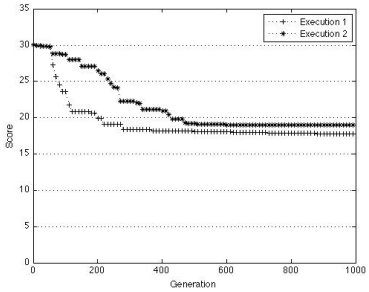
(a) Census dataset



(b) Diabetes dataset



(c) EIA dataset



(d) Tarragona dataset

Figure 3.4: Evolution of the score for two different executions with each dataset.

Dataset		Min Score	Mean Score	Max Score
Census	Execution 1	6.1238	16.2848	45.9179
	Execution 2	6.4298	16.4577	45.9179
Diabetes	Execution 1	8.8885	23.4609	43.1845
	Execution 2	6.1106	21.3981	43.1845
EIA	Execution 1	11.4812	14.7233	31.9693
	Execution 2	10.8979	15.0532	31.9693
Tarragona	Execution 1	5.5862	17.7956	49.8576
	Execution 2	7.1513	19.0316	49.8576

Table 3.2: Final Scores all data sets for the Experiment.

Recall also that our algorithm is stochastic in nature, so there is implicit some randomness into the process. To observe differences between two executions, two runs of the algorithm were considered. Evolutions of scores through 1000 generations are represented for the Census data set in Figure 3.4a, for the Diabetes data set in Figure 3.4b, for EIA data set in Figure 3.4c, and for the Tarragona data set in Figure 3.4d. So we see that the two curves for each figure initiate from the same score. Final scores are summarized in Table 3.2.

In this framework, our approach constitutes a valid alternative in two ways. On the one hand, lower scores can be reached, as shown above. On the other hand, the resulting data will overcome the specific attacks developed in an ad-hoc way for RS and Microaggregation.

In order to be sure that our parameters value and our approach of performing only one operation (mutation or crossover) in a single generation were good choices we executed two tests. These tests were designed using parameter values from the work done in [Schaffer et al., 1989] which are considered the optimals for genetic programming. Furthermore, the execution of both mutation and crossover in each generation (depending on their corresponding probability) was allowed. The difference between both tests was in the mutation operator which in one experiment (test 1) was applied to all values inside the matrix (also depending on its probability), while in the other experiment (test 2) it was only applied to one single value (depending on the probability too). Figure 3.5 shows the tests mean score for the individuals at each generation. It is easy to see that during all 1000 generations the results obtained with our approach (see Figure 3.4d) were better than those results obtained in the tests.

Our last experiment was conducted to estimate the relative time spent at each generation computing DR and IL values. Table 3.3 summarizes the time spent in CPU seconds (CPU sec) in a Dell Precision T7400 Desktop, with an Intel Xeon Processor at 2.66 GHz and 4 GB of RAM, and oper-

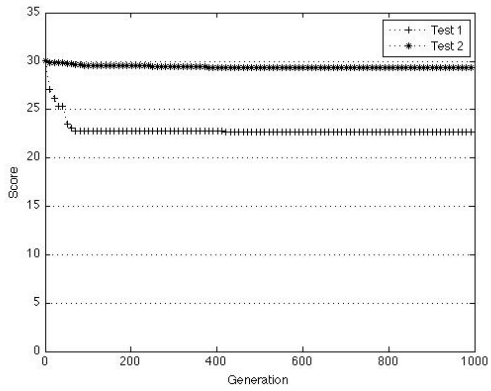


Figure 3.5: Evolution of the mean score for the tests with Tarragona data set.

ative system Microsoft Windows 7 x64. Results are segregated by genetic operation type (mutation or crossover). First row corresponds to the time for computing DR and IL values during the fitness evaluation of new individuals. Second row summarizes the remaining time spent to complete an iteration.

As it can be seen, fitness evaluation consumes most of the time of the algorithm in all cases. That is because it is the place where most of the work is done. It needs to compute several measures to determine the quality of the protection while the rest of the evolutionary algorithm consists only of executing simple (and fast) instructions to select individuals, change a single or few values in an individual and replace an individual inside the population collection. In addition it can be seen that the execution time when crossover is performed doubles the time obtained when a mutation is performed. This is expected because in crossover the algorithm generates two new offspring and both of them must be evaluated in the fitness function, while in the mutation the algorithm only produces one offspring. The absolute times shown in Table 3.3 vary depending on the dataset size. However, using the numbers obtained for our tested datasets we can say that execution times go from 145.92 to 861.81 seconds for an iteration where a crossover is performed, and from 75.07 to 443.55 seconds for an iteration where a mutation is performed.

With those times it is easy to know how much time will take to find a good individual. In our experiment we used 1000 generations, so in the case of the Diabetes data set it has spent about 30 hours and 41 minutes, for EIA data set 181 hours and 17 minutes, for Census data set 57 hours and 40 minutes, and for Tarragona data set 35 hours and 4 minutes.

Categorical Data Case Results

Dataset		crossover				mutation			
		mean	min	max	sd	mean	min	max	sd
Diabetes	Fitness	144.8	141.9	151.1	2.96	74.50	71.99	77.22	1.74
	Other	1.12	1.03	1.54	0.19	0.57	0.52	1.05	0.14
EIA	Fitness	858.7	836.5	868.9	11.0	442.0	434.5	450.8	5.19
	Other	3.11	3.04	3.18	0.05	1.55	1.48	1.59	0.03
Census	Fitness	271.6	267.7	273.7	2.24	140.4	137.3	142.6	2.43
	Other	2.18	2.14	2.215	0.02	1.09	1.08	1.11	0.01
Tarragona	Fitness	167.4	163.1	171.7	2.33	82.85	78.85	85.84	2.07
	Other	1.83	1.70	2.29	0.09	0.94	0.85	1,82	0.13

Table 3.3: Execution Time (CPU sec) for all data sets.

In the experiments for the case of categorical data we used the U.S. Housing Survey of 1993, German Credit, Solar Flare and Adult datasets presented in Section 2.7.

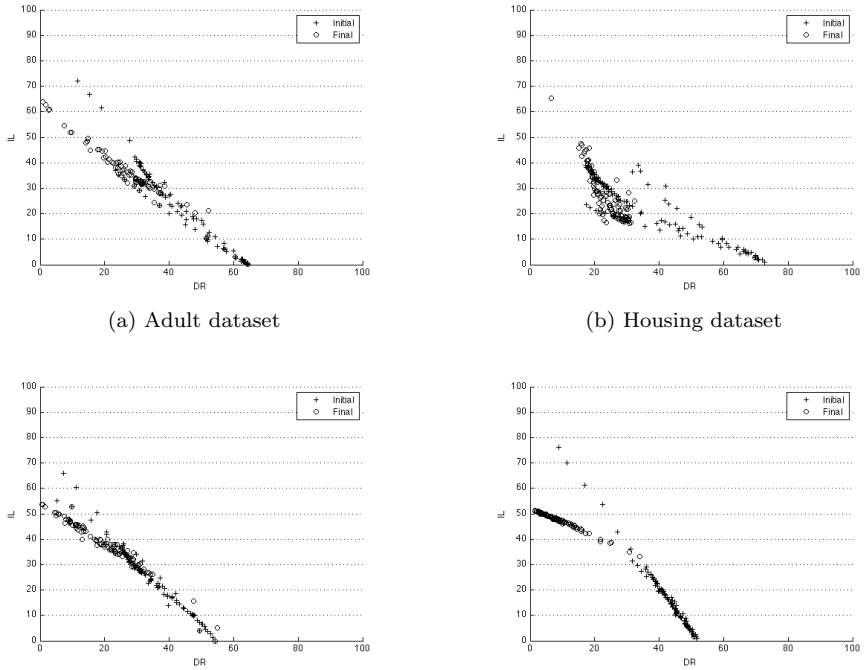
For each dataset we constructed a population of protections using the state-of-the-art protection techniques: Microaggregation, Bottom Coding, Top Coding, Global Recoding, Rank Swapping and, Post Randomization Method (PRAM). For the first dataset we had a population of 110 protections (72 of Microaggregation, 6 of Bottom Coding, 6 of Top Coding, 6 of Global Recoding, 11 of Rank Swapping and, 9 of PRAM). For the second and third datasets we had a population with a 104 protections for each one (72 of Microaggregation, 4 of Bottom Coding, 4 of Top Coding, 4 of Global Recoding, 11 of Rank Swapping and, 9 of PRAM). The last dataset had a population of 86 protections (48 of Microaggregation, 6 of Bottom Coding, 6 of Top Coding, 6 of Global Recoding, 11 of Rank Swapping and, 9 of PRAM).

Regarding the attributes selected to protect in each dataset are as follows. For the Housing dataset we protected three attributes: BUILT with 25 categories, DEGREE with 8 categories and, GRADE1 with 21 categories. In the case of German dataset: EXISTACC with 5 categories, SAVINGS with 6 categories, and PRESEMPLOY with 6 categories. Flare dataset attributes are: CLASS with 8 categories, LARGSPOT with 7 categories, and SPOTDIST with 5 categories. Finally, Adult dataset protected attributes are: EDUCATION with 16 categories, MARITAL-STATUS with 7 categories, and OCCUPATION with 14 categories.

To test the performance of our approach we performed three different kind of experiments. The first one consists of using the mean of information loss and disclosure risk as a score. In our second experiment we use the max value of both measures as a score. Finally, we present a final experiment to prove the robustness of our approach when the best initial individuals

are missing therefore, it is is supposed that it will be more difficult for the algorithm to reach the same level of protection quality because it starts working with worse individuals than in the other experiments.

In the first experiment we applied our evolutionary algorithm using the fitness function that uses the mean values of both information loss and disclosure risk as a score shown in Equation 3.2 to all four dataset populations independently.



(a) Adult dataset

(b) Housing dataset

(c) German dataset

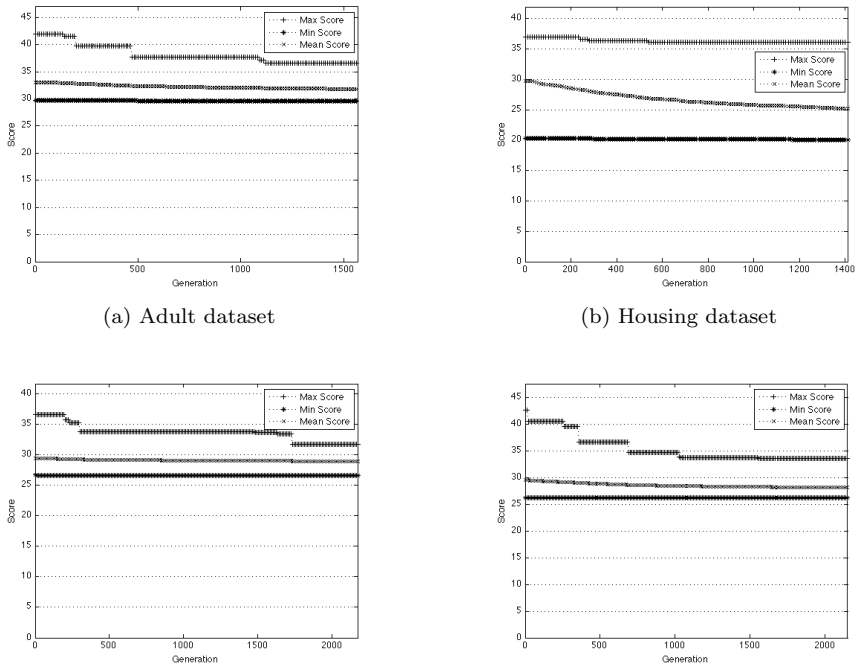
(d) Flare dataset

Figure 3.6: Dispersion plots of initial and final population information loss and disclosure risk for each dataset using fitness Equation 3.2.

In order to evaluate the results of our experiment we splitted our analysis into two parts. The first part of our analysis is focused on the initial and final pairs of values (IL,DR) for all datasets shown in Figure 3.6a for the Adult dataset, Figure 3.6b for the Housing dataset, Figure 3.6c for the German dataset and, Figure 3.6d for the Flare dataset.

It can be noticed that, in all the cases, final population is more optimized than the initial population because of the reduction of the values in the tuples (IL,DR). However, there also exist individuals in the final population that have reduced their score value but obtaining an individual with very unbalanced measures. Recall that, according to our preferences, given

a certain score, we prefer balanced information loss and disclosure risk. Furthermore, this effect does not appear in the same degree to all datasets. It can be seen that the Flare and German datasets have more unbalanced final individuals than the Housing and Adult datasets.



(a) Adult dataset

(b) Housing dataset

(c) German dataset

(d) Flare dataset

Figure 3.7: Evolution of the information loss and disclosure risk during the execution of the evolutionary algorithm for each dataset using fitness Equation 3.2.

The second part of our analysis focuses on the evolution of the max, mean and min score values of the population during all the generations shown in Figure 3.7a for the Adult dataset, Figure 3.7b for the Housing dataset, Figure 3.7c for the German dataset and, Figure 3.7d for the Flare dataset.

In these figures it can be seen that max score has few decrements but most of them are quite abrupt, and this is because our selection policy gives few opportunities to the individuals with bad score to be selected, and when they are selected they almost always have a considerable improvement of their score value using parts of other better individuals. The improvements obtained for the max score are the following: in the case of the Adult dataset we had a decrement from 41.95 to 36.6 (12.75% of improvement), for the Housing dataset we obtained a decrement from 36.96 to

36.14 (2.22% of improvement), for the German dataset it was from 36.59 to 31.74 (13.25% of improvement) and, for the Flare dataset this max score decreased from 42.53 to 33.56 (21.09% of improvement).

Looking at the evolution of the mean score it can be seen that it has more or less continuous decrement and this is what we expected because in almost every iteration there is a score improvement for an individual, so the mean score of the entire population is also improved. Concretely, the improvement obtained for the mean score in all datasets during this first experiment is as follows: in the case of the Adult dataset we had a decrement from 33.05 to 31.78 (3.84% of improvement), for the Housing dataset the decrement was from 29.79 to 25.25 (15.24% of improvement), for the German dataset it was from 29.37 to 28.91 (1.57% of improvement) and, for the Flare dataset it was from 29.57 to 28.13 (4.87% of improvement).

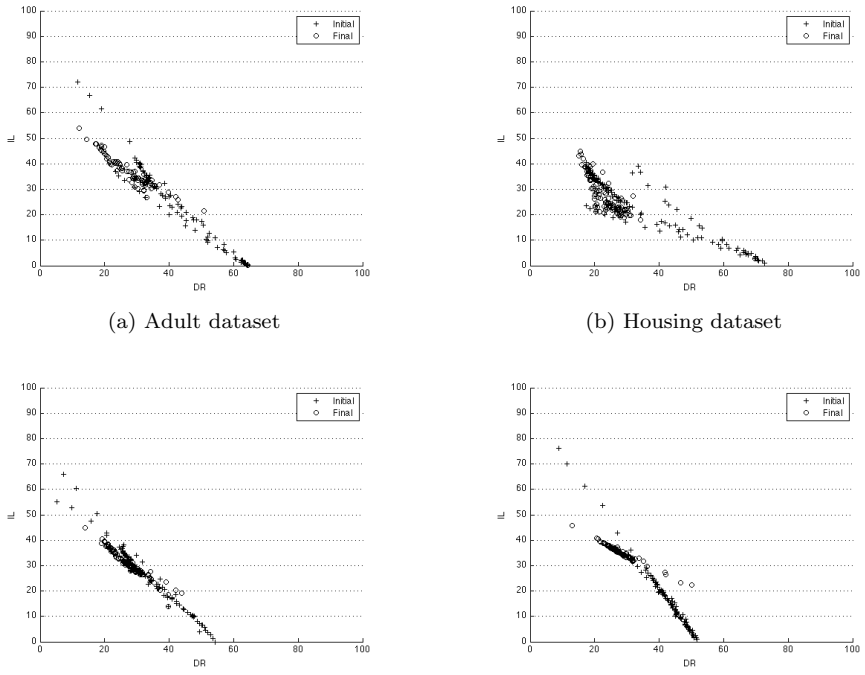
The last score to analyze is the min score evolution. In this case it can be noticed that the improvement is very small and the reason for this is that it is very difficult to improve a protected dataset that already has a good score (in terms of the fitness function used) using other protected files with a worse score. The improvements obtained for this min score are as follows: for the Adult dataset we obtained a decrement from 29.68 to 29.61 (0.24% of improvement), in the case of the Housing dataset there is a decrement from 20.36 to 20.12 (1.18% of improvement), for the German dataset we obtained a decrement from 26.68 to 26.54 (0.52% of improvement) and, for the Flare dataset did not obtain any decrement.

To summarize the results found after the first experiment, we can say that the fitness function shown in Equation 3.2 is not very appropriate for categorical data because it does not permit to discriminate individuals with a high unbalance and those with a low score in both measures. In addition, unfortunately, the alteration of values in categorical datasets produces quite high modifications in information loss and disclosure risk values because of the limited number of available categories to use.

In this second experiment we wanted to try to improve the results obtained in the first experiment by applying the same evolutionary algorithm but using the fitness function shown in Equation 3.4 which takes as a score the maximum value between information loss and disclosure risk.

In this experiment we also splitted our analysis into two parts. The first part of our analysis is focused on the initial and final pairs of values (IL,DR) for all datasets shown in Figure 3.8a for the Adult dataset, Figure 3.8b for the Housing dataset, Figure 3.8c for the German dataset and, Figure 3.8d for the Flare dataset.

It can be seen that final population is more concentrated (in general) to pairs of (IL,DR) with more equal values than the original population (compare with Figures 3.6a, 3.6b, 3.6c, and 3.6d of the first experiment). This was the expected behavior because the fitness function requires to have



(a) Adult dataset

(b) Housing dataset

(c) German dataset

(d) Flare dataset

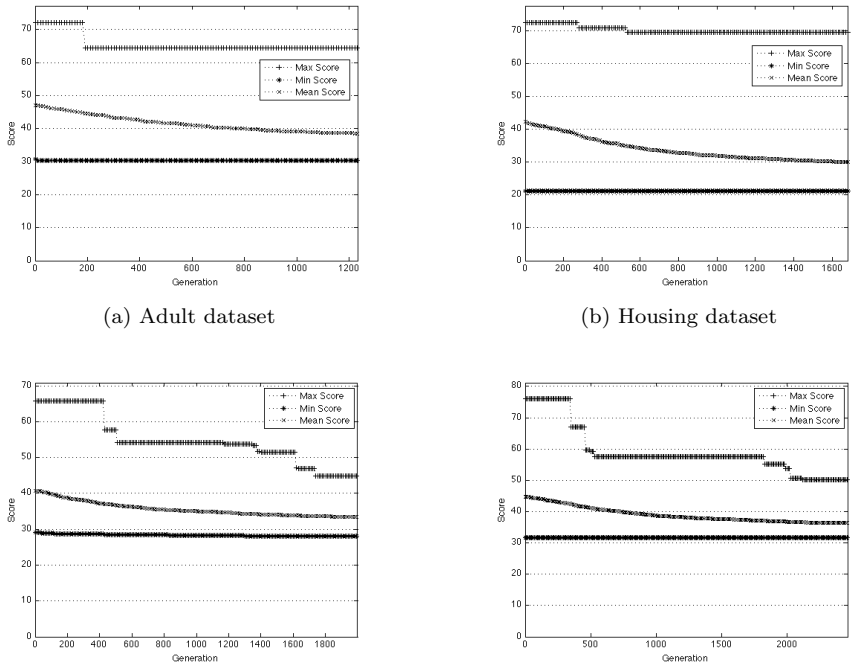
Figure 3.8: Dispersion plots of initial and final population information loss and disclosure risk for each dataset using fitness Equation 3.4.

low values in both measures in order to declare a new individual better than the parent.

The second part of this second experiment analysis focuses on the evolution of the max, mean and min score values of the population during all the generations. This is shown in Figure 3.9a for the Adult dataset, Figure 3.9b for the Housing dataset, Figure 3.9c for the German dataset and, Figure 3.9d for the Flare dataset.

It can be seen that max score decreases quite abruptly in some points for all datasets and remain stable between those points because our selection method gives more chances to the *best* individuals. The improvements obtained for this max score are as follows: for the Adult dataset we obtained a decrement from 72.19 to 64.38 (10.82% of improvement), in the case of the Housing dataset there is a decrement from 72.65 to 69.63 (4.16% of improvement), for the German dataset we obtained a decrement from 65.87 to 44.85 (31.91% of improvement) and, for the Flare dataset it went from 76.17 to 50.22 (34.07% of improvement).

For the mean score evolution we have that it decreases at almost every



(a) Adult dataset

(b) Housing dataset

(c) German dataset

(d) Flare dataset

Figure 3.9: Evolution of the information loss and disclosure risk during the execution of the evolutionary algorithm for each dataset using fitness Equation 3.4.

generation in all cases and its value evolves towards the value of the min score because most of the times the individuals are improved using the one with minimum score, so they go close to this min score value. In this case, the improvements obtained are as follows: for the Adult dataset we obtained a decrement from 47.05 to 38.57 (18.02% of improvement), in the case of the Housing dataset there is a decrement from 42.32 to 30.12 (28.83% of improvement), for the German dataset we obtained a decrement from 40.76 to 33.42 (18.01% of improvement) and, for the Flare dataset it went from 44.83 to 36.36 (18.89% of improvement).

Finally, the min score has little decrement in all the datasets because it is difficult to get a big improvement in this value using individuals with worse score. The improvements obtained for the min score are as follows: for the Adult dataset we obtained a decrement from 30.70 to 30.28 (1.34% of improvement), in the case of the Housing dataset there is no decrement for this score, for the German dataset we obtained a decrement from 29.18 to 28.05 (3.87% of improvement) and, for the Flare dataset it went from

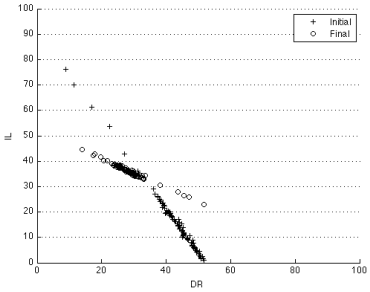


Figure 3.10: Dispersion plot of initial and final population information loss and disclosure risk for the Flare dataset using fitness Equation 3.4 without the 5% best initial individuals.

31.77 to 31.63 (0.44% of improvement).

After the first two experiments, we can see an interesting fact. We have seen that, in this second experiment, in all the cases the final population is grouped around the pairs of values (IL,DR) with more balanced values than the ones in the first experiment. However, this is achieved in a different way in the four datasets. An analysis of the total number of valid categories in the attributes show that the larger the number of categories, the better the equilibrium of values in both measures. Note that few categories supply a small number of possible different registers. Then, altering some categories increase one of the measures (information loss or disclosure risk) quite abruptly and reduce the other one, and this makes difficult to find an equilibrium between both values.

In addition, we have seen that, using maximum in the fitness function (Equation 3.4) performs better in the optimization than using mean in the fitness function (Equation 3.2) because the final information loss and disclosure risk measures are more balanced.

It also should be noticed that in all our experiments we obtained an average computation time of $120.34s$ for each entire generation with mutation operation, and $242.48s$ for each entire generation with crossover operation. However, most of the time is consumed by the fitness function ($120.32s$ in mutation generation and $242.46s$ in crossover generation) and a very small amount of time is consumed by the rest of each generation ($0.02s$ in both cases).

Finally, to conclude our study, we applied to the Flare dataset our approach using Equation 3.4 (the maximum value of the two measures is taken as the score value) but in this case not including in the population the best 5% and 10% individuals in terms of the fitness function score. This experiment assesses the robustness of our method trying to achieve the best solutions

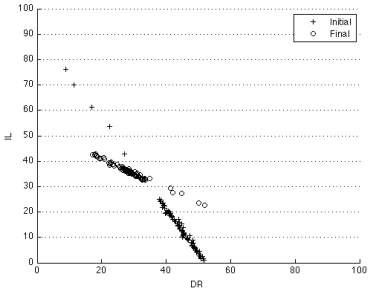


Figure 3.11: Dispersion plot of initial and final population information loss and disclosure risk for the Flare dataset using fitness Equation 3.4 without the 10% best initial individuals.

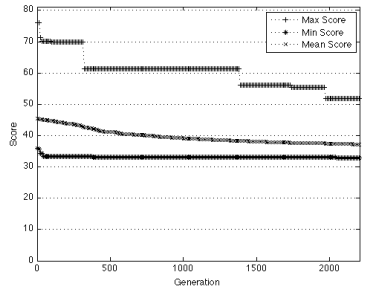


Figure 3.12: Evolution of the information loss and disclosure risk during the execution of the evolutionary algorithm for the Flare dataset fitness Equation 3.4 without the best 5% initial individuals.

starting from worse solutions.

After several generations we could see that initial and final populations follow the same behavior than in the case with the entire population. In addition, it can be seen that, compared with Figure 3.8d, the initial population has a *hole* in the region with more balanced pairs of (IL,DR) which where the ones removed from the population (Figures 3.10 and 3.11).

However, looking at the evolution of max, mean and min scores in Figures 3.12 and 3.13 we see that we almost reached the best min score obtained without removing these solutions. In the case of removing the 5% of the best initial protections we reached a minimum score of 32.96 what represents a difference of 1.33 points from the minimum value obtained using the entire population, and in the case of removing the 10% of the best initial protections we reached a minimum score of 32.71 what represents a

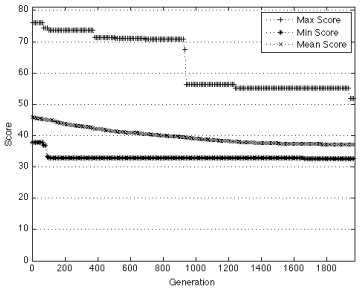


Figure 3.13: Evolution of the information loss and disclosure risk during the execution of the evolutionary algorithm for the Flare dataset fitness Equation 3.4 without the best 10% initial individuals.

difference of 1.08 points.

It should be noticed that the fact of having better results in the case of removing the 10% of the best individuals than in the case of removing just the 5% is produced because of the stochasticity of evolutionary algorithms.

So, looking at this behavior we can assess that our evolutionary approach is robust enough to achieve good protections even when the best ones are missing.

Chapter 4

Evolutionary Seek for Better PRAM Matrices

In the previous chapter we introduced a way to use evolutionary algorithms to directly deal with a dataset in order to find better protections. In that case the optimization problem was stated thinking with a protection process as a function that takes an original dataset and generates a new protected dataset which better preserves the privacy of its contenders.

In this chapter we go one step further and we introduce a new way to use evolutionary algorithms to improve the privacy protection in a dataset but, instead of dealing directly with the dataset, it optimizes a PRAM matrix in order to use it inside the PRAM protection method allowing it to perform better protections. Then, in this case, optimization process is thought as a function that takes a certain PRAM matrix and returns an optimized one, which, using it inside the PRAM, optimizes the privacy quality of the method and results in a better protected dataset.

Two different approaches are described in this chapter. The first one is introduced in Section 4.1 and it is based on a generic evolutionary algorithm dealing with the PRAM matrix probability values. The second one, shown in Section 4.2, is based on a genetic programming algorithm which deals with analytical equations to generate better PRAM matrices.

4.1 General Evolutionary Approach for Better PRAM Matrices

In this section we continue with the idea of thinking about the protection process as an optimization problem. However, in this case, instead of focusing on optimizing the data protection itself by dealing directly with the data set in the evolutionary algorithm, we focus on optimizing a state-of-the art protection method to perform better protections.

In particular we focus on the Post Randomization Method (PRAM) which we introduced in Section 2.3. As we explained, this method is able to perform the same protections provided by many other state-of-the-art methods only by selecting the appropiate PRAM transition matrix. However, the difficulty of getting a good matrix to perform good protections is also the reason why this method is not widely used. Then, we can think that this matrix is the key point of the PRAM and it has to be the thing to be optimized.

The proposed method's algorithm is shown in Algorithm 6. In this case we are dealing with an initial population P_0 of PRAM matrices (one per attribute to protect) where each one is treated independently, this is, data is not changed across individuals, only inside the same individual. The reason for this is that matrices are of different sizes because it depends on the size of each attribute's domain and it would not be possible to exchange rows (or ranges of values) between a small matrix and a big one properly.

The different initial PRAM matrices X_{i_0} are being optimized through the iterations of the evolutionary algorithm where at generation t we produce a modified PRAM transition matrix represented by X_{i_t}. To produce the $X_{i_{t+1}}$ PRAM transition matrix at generation $t+1$, we generate an intermediate matrix X'_{i_t} resulting from applying a genetic operator to the current matrix X_{i_t}. The PRAM transition matrix $X_{i_{t+1}}$ at generation $t+1$ will be the one with better fitness (either X_{i_t} or X'_{i_t}) and the other discarded. This process is repeated at each generation.

Algorithm 6 Proposed Evolutionary Algorithm to Enhance PRAM Transition Matrices

Input: $P_0 = \{X_{0_0}...X_{n_0}\}$, initial population of PRAM matrices
Output: $P'_t = \{X_{0_t}...X_{n_t}\}$, optimized PRAM matrices after t generations
$t \Leftarrow 0$
$fitness\_eval(P_0)$
while $stopping(X_t) \neq true$; **do**
  $alter \Leftarrow$randomly choose between mutation and cross
  if $alter$ by mutation **then**
    $X'_{i_t} \Leftarrow mutate(X_{i_t})$
  else
    $X'_{i_t} \Leftarrow cross(X_{i_t})$
  end if
  if $fitness\_eval(X'_{i_t}) < fitness\_eval(X_{i_t})$ **then**
    $X_{i_{t+1}} \Leftarrow X'_{i_t}$
  else
    $X_{i_{t+1}} \Leftarrow X_{i_t}$
  end if
  $t \Leftarrow t + 1$
end while
return $\{X_{0_t}...X_{n_t}\}$

0.185	0.283	0.532
0.609	0.089	0.302
0.002	0.277	0.721

PRAM transition matrix.

185	283	532
609	89	302
2	277	721

PRAM transition matrix in integers mode.

0010111001	0100011011	1000010100
1001100001	0001011001	0100101110
0000000010	0100010101	1011010001

Binary matrix.

0011100101	0110010110	1100011110
1101010001	0001110101	0110111001
0000000011	0110011111	1110111001

Gray-coded genome matrix.

Figure 4.1: Example of genotype encoding

The following subsections are mainly devoted to a description of the algorithm that explains how an individual, i.e. the PRAM transition matrix is represented, how the genetic operations are defined, and how the adaptability of the individual through time is evaluated. Other parameters, such as the initial population, i.e. the original PRAM transition matrix, population size, or crossover and mutation rates, are exemplified with a practical experiment.

4.1.1 Genotype Encoding

The initial probability transition matrix that we are trying to optimize contains probabilities with several decimals so in order to simplify, all the probabilities are multiplied by 1000 and only the integer part of the value is kept for the encoding.

Encoding of the individual X is done value by value transforming them into its Gray code representation. Recall that Gray-coded representation allows to obtain fast and more accurate solutions than regular binary representations as explained in Section 2.1.

A complete file encoding example is shown in Figure 4.1. The example includes all the steps required during the entire encoding process.

0011100101	0110010110	1100011110
110101**0**001	0001110101	0110111001
0000000011	0110011111	1110111001

Gray-coded matrix.

0011100101	0110010110	1100011110
110101**1**001	0001110101	0110111001
0000000011	0110011111	1110111001

Mutated gray-coded matrix.

Figure 4.2: Example of mutating a transition matrix

4.1.2 Genetic Operators

The most common genetic operators in an evolutionary algorithm are mutation and crossover. These operators are executed according to a specified rate. This is done to simulate the species evolution where species evolve by being crossed with another species or by being mutated during generations.

In this approach we decided (based on empirical tests) to use the value 0.5 for both the crossover rate and mutation rate in order to have approximately the same number of operations performed by each operator. A random value (alter) between 0 and 1 decides the operation to perform, using 0.5 as a delimiter.

Mutation

Recall that the main idea of mutation is to apply a slight random change in an individual of the population (see Section 2.1). In our case, the population consists only of a single individual (a gray-coded matrix) so we will perform mutation by altering a single bit from a single gray-coded number inside the transition matrix. To do that, both the bit and the number are chosen randomly.

The mutation used in this approach is performed as follows: Take a random value of the individual X and consider that the value at this position is x_i with $genome(x_i) = b_j b_{j-1} \dots b_1$. Choose a bit position k at random, such that $1 \leq k \leq j$. Then a new individual is obtained by replacing the bit b_k by its negation counterpart, $b'_k = not(b_k)$.

An example of the effect of the mutation operator is shown in Figure 4.2.

Crossover

Recall that in crossover the general idea is to select two individuals from the population and generate two new individuals by concatenating a part of each one that is delimited by one or two crossing points chosen at random (see Section 2.1). In our approach, we deal with a population with only one individual so we modified this operator to pick two ranges of values inside the individual, i.e. the PRAM transition matrix, delimited by two

0011100101	0110010110	1100011110
1101010001	**0001110101**	0110111001
0000000011	**0110011111**	**1110111001**

Gray-coded matrix.

0011100101	0110010110	1100011110
0110011111	**1110111001**	0110111001
0000000011	**1101010001**	**0001110101**

Crossed gray-coded matrix.

Figure 4.3: Example of crossover in the transition matrix

crossing points selected at random, and then swapping those two ranges inside the matrix to create a new matrix.

Formally, we define the crossover of the individual X is performed by swapping two ranges of values within the individual as follows: Take two value positions $\{s, r\}$ at random, and consider that the two values at this position are $x_s \in X$ and $x_r \in X$. Generate a random number m to indicate the length of the ranges. This number must be in the range $[0, min(length(X) - s, length(X) - r, |s - r|)]$, where length($X$) is the total number of values inside the individual X, and $||$ is the absolute value operator. Then the ranges $[x_s, x_{s+m}]$ and $[x_r, x_{r+m}]$ are swapped obtaining a new individual. For example, having $s < r$ and $X = \{x_1, \ldots, x_n\}$ the new individual will be $X' = \{x_1, \ldots, x_r, \ldots, x_{r+m}, \ldots, x_s, \ldots, x_{s+m}, \ldots, x_n\}$.

An example of the effect of mutation operator is shown in Figure 4.3.

4.1.3 Fitness Function and Selection

As explained in Chapter 2, fitness function is the most important part of an evolutionary algorithm. It controls the convergence of the algorithm to the desired optimal solution, similar to the objective function in mathematical programming. The main idea in our fitness function is to use the new probability transition matrix obtained after mutation and crossover to perturb the original data according to the PRAM method described in Section 2. The fitness function for the new transition matrix is calculated on the perturbed data and compared with the fitness of the perturbed data based on the current transition matrix. The transition matrix resulting in the lowest score of the fitness function (i.e. the one that provides better utility) is retained as the probability transition matrix to use in the next generation.

In our case, the evaluation of PRAM matrices need several steps before checking their protection quality. First of all, these PRAM matrices values are in Gray code representation so it is needed to restore them to floating point values. Then, it is not possible to check the quality of the matrices just by taking a look at

them so, as a second step, we use these matrices to perform the multivariate PRAM protection on the original data obtaining a certain protected dataset. After this second step we are finally able to check the protection quality using the two measures described in Section 2.4: information loss and disclosure risk.

We used two different kind of fitness functions because we performed executions based on different aspects. The first case is based on general purposes information loss and disclosure risk measures. Recall that it can be considered as a multi-objective optimization problem. To solve this we used the same approach than in Section 3.4 giving the same importance to both Disclosure Risk (DR) and Information Loss (IL) measures, so both have $\frac{1}{2}$ as a weight value.

If F is the original file and $PRAM_{multivariate}(F, \{X_1, \ldots, X_n\})$ is the function that performs multivariate PRAM protection in F with the set of PRAM matrices $\{X'_1, \ldots, X'_n\}$, then, the score of the set of matrices $\{X_1, \ldots, X_n\}$ is computed as follows

$$\{X'_1, \ldots, X'_n\} = restore(\{X_1, \ldots, X_n\}) \tag{4.1}$$

$$F' = PRAM_{multivariate}(F, \{X'_1, \ldots, X'_n\}) \tag{4.2}$$

$$Score(\{X'_1, \ldots, X'_n\}) = \frac{DR(F') + IL(F')}{2} \tag{4.3}$$

where $DR()$ is the disclosure risk evaluation function and $IL()$ is the information loss evaluation function.

Because the PRAM method takes random decisions in the protection step, the method can generate different protected files for the same Markov matrix, and they will also have different scores. In order to have more robust results, we compute 5 protected files for each candidate to be evaluated (i.e. each Markov matrix) and the average of their scores is taken as the candidate's final score. It should be noticed that the number of executions to perform is not fixed and it can be changed by the user. We used 5 executions because with it we obtained enough robust results without penalizing too much the execution time. More formally:

$$FinalScore(\{X_1, \ldots, X_n\}) = \frac{\sum_{i=1}^{5} Score(\{X'_1, \ldots, X'_n\})}{5} \tag{4.4}$$

The second kind of fitness function has been used to test the information gain when adding the invariance property to the matrices. In this case to compute the fitness function of a certain PRAM transition matrix generated by the evolutionary algorithm we propose to use the difference in bivariate counts of two cross-classified categorical variables between the original data and the perturbed data where one of the categorical variables is perturbed with PRAM and the other categorical variable is not perturbed.

Formally, the fitness function is defined as follows

$$Fitness(R) = \frac{\sum_{ij} |counts_{original}(x_i, z_j) - counts_{perturbed}(x_i, z_j)|}{2 * \#records} \tag{4.5}$$

where x_i refers to the category i of attribute x, z_j refers to category j of attribute z, and $||$ is the absolute value operator. It should be noted that only one of the attributes (x or z) is protected, the other one must be unprotected.

The use of this fitness function shows the optimization of the transition matrix in preserving the frequency distribution of two cross-classified categorical variables in the perturbed data and whether it is similar to the distribution in the original data given that one of the categorical variables has been perturbed.

Before calculating the fitness function on each new generation of the transition matrix, we first need to carry out a pre-processing stage to ensure that the property of a probability transition matrix is fulfilled, i.e. each row of the matrix must add to one. This property can easily be lost when altering values with mutation and crossover. This is achieved normalizing the row by dividing each element by the sum of the entire row.

4.1.4 Adding Invariance and Controlling Diagonal Values

A technique to boost the performance of probability transition matrices used for PRAM is to include the property of invariance. This property ensures that the sufficient statistics of the protected attributes are preserved in expectation in the perturbed data and that the perturbed data is an unbiased moment estimator of the original data. In addition, controlling for the diagonal probabilities of the transition matrices ensures the desired level of perturbation according to the standards and thresholds set by data providers and also guarantees that the matrices can be inverted.

Placing the condition of invariance on the transition matrix P, i.e. $tP = t$ releases the users of the protected file of the extra effort to obtain an unbiased estimate of the original data, since t^* itself will be an unbiased estimate of t. This is an important property to instill in the protected data since data providers will generally not release the transition matrix P that is used to perturb the data. The property of invariance means that the marginal distribution of the variable being perturbed is preserved in expectation.

In this work, the invariance is computed by following the two stage algorithm proposed in [Willenborg and Waal, 2000]. Let P be the PRAM matrix with $p_{jk} = p(c' = k|c = j)$ the probability of changing the value of category c equal to j to a new category c' equal to k. Now calculate the matrix Q using Bayes formula by $Q_{kj} = p(c = j|c' = k) = \frac{p_{jk}p(c=j)}{\sum_l p_{lk}p(c=l)}$. We estimate the entries of this matrix by $\frac{p_{jk}v_j}{\sum_l p_{lk}v_l}$, where v_j is the relative frequency of the category value j. For $R = PQ$ we obtain an invariant matrix where $vR = vPQ = v$ since $r_{ij} = \sum_k \frac{v_j p_{ik} p_{jk}}{\sum_l p_{lk} v_l}$ and $\sum_i v_i r_{ij} = \sum_k v_j p_{jk} = v_j$.

However, before making the transition matrix invariant, its diagonal dominance must be checked, that is, the diagonal probability of each row must be higher than the sum of all off-diagonal probabilities. That property is required to ensure that we are able to invert the transition matrix.

With respect to the property of diagonal dominance in the transition matrix, we also want to control the range of values that the diagonal of the matrix can

have. This is because we do not want to have a very high probability of preserving the same category since then the related attribute will not be protected enough. On the other hand, we do not want to have a very low probability because it would mean the information contained on that attribute would be totally lost. For that reason we decided to force the diagonal values to be between 0.55 and 0.75. In addition, by preserving this range of perturbation through the diagonal probabilities, we obtain transition matrices with similar levels of disclosure risk and therefore can focus on maximizing the utility of the matrices towards the optimization of the evolutionary algorithm generations.

If a transition matrix contains a diagonal element below 0.55 we apply the same approach shown in Equation 4.6 to increase the value. On the other hand, if a matrix contains a diagonal element over 0.75 we use the approach shown in Equation 4.7 to reduce the value. However, the diagonal dominance could be lost when reducing values. For that reason, after every execution of reducing values we test again for diagonal dominance.

Then, in order to ensure that a matrix is diagonal dominant we use the approach shown in [Shlomo and Young, 2008] where the diagonal values are increased (and off diagonal are decreased proportionally) according to a parameter α. Equation 4.6 shows this approach where R' is the new PRAM matrix, I is the identity matrix and α is the control parameter. It should be noted that the higher the value of α, the smaller the increment in the diagonal values.

$$R' = \alpha R + (1 - \alpha)I \tag{4.6}$$

$$\beta = 0.75/max(p_{kk})$$

$$p_{ij} = \begin{cases} \beta * p_{ij}, & \text{if i = j} \\ (\beta * p_{ij}) + \frac{1-\beta}{length(row_i)}, & \text{if i} \neq \text{j} \end{cases} \tag{4.7}$$

The invariance property is applied every time a new matrix is evaluated in the Fitness function.

4.1.5 Experimental Results

In this section we present the results of the experiments done to test the performance of our approach. These experiments were splitted in two parts. The first part shows the results regarding the general information loss and disclosure risk measures while in the second part shows the results regarding the addition of the invariance property to the PRAM matrices.

General Measures Testing Results

In order to illustrate and empirically evaluate our proposed method we used three different datasets to perform some experiments. The ones used in in this experiments are the U.S. Housing Survey of 1993 from the U.S. Census Bureau, the German Credit and the Solar Flare datasets.

0.500	0.067	0.060	0.065	0.067	0.076	0.080	0.083
0.073	**0.500**	0.058	0.064	0.066	0.075	0.080	0.083
0.072	0.064	**0.500**	0.062	0.064	0.074	0.080	0.083
0.073	0.065	0.057	**0.500**	0.066	0.075	0.080	0.083
0.073	0.066	0.058	0.064	**0.500**	0.075	0.080	0.083
0.074	0.068	0.061	0.066	0.068	**0.500**	0.080	0.083
0.075	0.068	0.062	0.067	0.069	0.076	**0.500**	0.083
0.075	0.069	0.062	0.067	0.069	0.077	0.081	**0.500**

Table 4.1: Initial PRAM matrix with p=0.5 for the attribute DEGREE in the U.S. Housing Survey dataset

Categories	'1'	'2'	'3'	'4'	'5'	'6'	'9'	'-'
Frequency	98	173	251	195	170	80	33	0

Table 4.2: Frequencies of the DEGREE attribute in the U.S. Housing Survey dataset

In these experiments we chose p=0.5 as a parameter value to create the initial Markov matrices. This value represents the quantity of original values that are wanted to be kept after perturbation (in this case we want to keep only 50% of the original values). Its value is up to the data user and, in this work, we used this value to demonstrate the ability to find good matrices from a bad one. As an example, Table 4.1 shows the initial Markov matrix for the DEGREE attribute corresponding to the U.S. Housing Survey data set with 8 categories, and Table 4.2 shows the frequencies of the categories corresponding to this attribute in the original data set. Note that the max values in each row are highlighted.

It is easy to see that the higher off-diagonal values corresponds to the attributes that have less frequency inside the original data set. This effect makes that, after the protection process, the frequencies of all categories are more balanced in order to increase the uncertainty inside the data set. Then, the problem here is to obtain a good PRAM matrix in order to achieve a better protection minimizing the Information Loss and the Disclosure Risk.

Our proposed method *a priori* applies to any particular PRAM matrix, but of course it is generally better to start from a *good* matrix because in this way it starts from matrices with higher fitness value.

Each experiment is divided in two phases. In the first phase, some attributes are going to be protected independently checking the performance of our method when only one attribute is protected, while in the second phase all the previous attributes are going to be protected together at the same time checking the performance of our method when multi-attribute protection is performed.

In our first experiment we used the U.S. Housing Survey dataset considering 3 attributes to protect and their respective PRAM matrices. The first attribute is named DEGREE and it has 8 different ordinal categories, the second one is the BUILT attribute with 25 different ordinal categories, and finally, the third

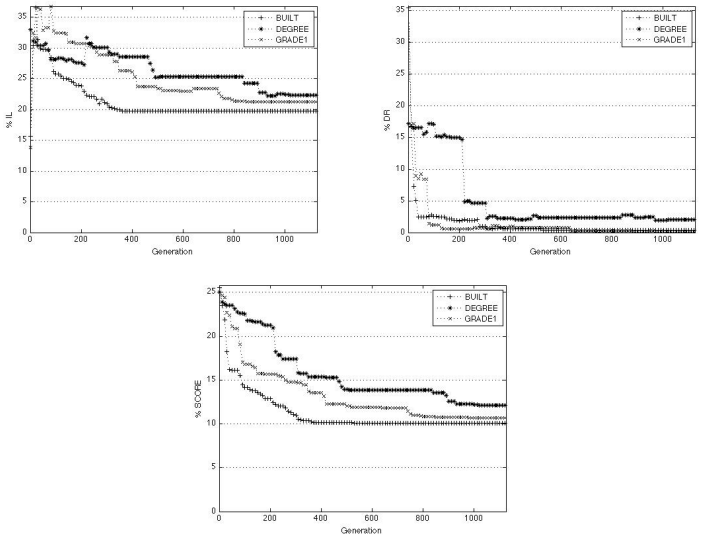


Figure 4.4: Evolution of the measures for the individual protection of the three attributes BUILT, DEGREE and GRADE1 in U.S. Housing dataset

attribute is named GRADE1 with 21 different ordinal categories.

Starting with the first experiment, Figure 4.4 shows the results for the measures evolution for the individual protections of the three attributes (BUILT, DEGREE, and GRADE1) during all the 1125 generations.

As it can be seen, Information Loss and Disclosure Risk are being adjusted in order to reduce the Score value. It does not mean that all measures are decreasing all the time (e.g. there is an increment of the Information Loss at generation 200). Score is the only measure that is strictly decreasing its value and never increases, so this implies that the result at any generation will be at least as good as the previous one.

Looking at the decrement of the Score we can see that, during the optimization approach that we propose, the measure has been reduced quite significantly in all the cases. It means that the new PRAM matrix performs a much better protection than the original one.

Once we know that our approach is working quite good when protecting isolated attributes, a multi-attribute protection of these three attributes can also be performed.

In Figure 4.5 we can find the evolution of the Information Loss, Disclosure Risk and Score for this multi-attribute protection. During the evolutionary process it can be seen that there is a progressive decrement until around generation 1100 of the Score value during all the process. Moreover, Disclosure Risk has suffered a big decrement, so the combination of this decrement with the little reduction of Information Loss has forced the Score value to be reduced. In this

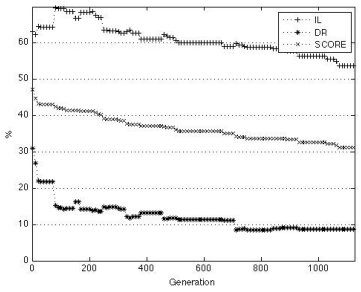


Figure 4.5: Results for the protection of all three attributes at the same time in the U.S. Housing dataset

	IL	DR	Score
Initial	63.14	31.08	47.11
Final	53.77	8.61	31.20

Table 4.3: Initial and final Scores for the protection of the three attributes DEGREE, BUILT, GRADE1 in U.S. Housing dataset at the same time.

case, Score measure started with a value of 47.11% and, after the evolutionary process, it ended with a value of 31.20%. This represents a decrement of a 33.77%. Table 4.3 shows the initial and final results of the three measures.

In the second experiment we used the German dataset considering 3 attributes to protect and their respective PRAM matrices. This dataset has attributes with less categories than the dataset used in the first experiment. So, with this second experiment we are going to proof that our approach works well also with this kind of attributes. The first attribute is named EXISTACC with 5 different ordinal categories, the second one is PRESEMPLOY with 6 different ordinal categories, and finally, the third attribute is named SAVINGS with 6 different ordinal categories.

The evolution of its Information Loss, Disclosure Risk and Score measures of all the individual protections for the attributes are shown in Figure 4.6. It can be seen that Disclosure Risk measures have been reduced so much (specially in the case of PRESEMPLOY) with a big decrement in the first generations. The decrement of the Information Loss measures is smaller but it also exhibits the main decrement in the first generations like in the case of Disclosure Risk. Moreover, the measures of Information Loss and Disclosure Risk are being adjusted in an irregular way (there are some increments and decrements of their values while Score is being reduced).

It has been proved that our approach is working well in protecting one ordinal attribute with only few categories. Now in the last part of this second

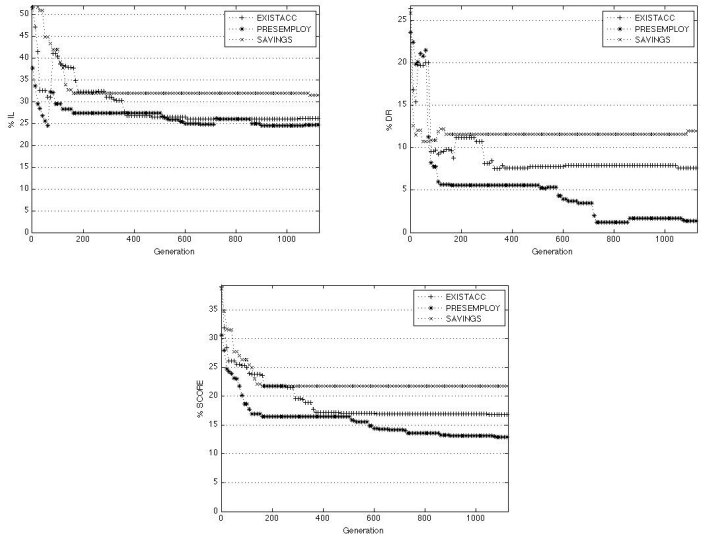


Figure 4.6: Evolution of the measures for individual protection of the three attributes EXISTACC, PRESEMPLOY, and SAVINGS in the German dataset

	IL	DR	Score
Initial	80.36	25.63	52.99
Final	38.39	27.30	32.84

Table 4.4: Initial and final Scores for the protection of the three attributes EXISTACC, PRESEMPLOY, and SAVINGS in the German dataset at the same time

experiment all three attributes are going to be protected together in order to test our approach in a multi-attribute protection for this kind of attributes.

Figure 4.7 shows the evolution of the Information Loss, Disclosure Risk and Score for this multi-attribute protection. During the evolutionary process it can be seen that there is a quite progressive decrement of the Score value during all the process. Moreover the Disclosure Risk has increased instead of decreased, but its final value is quite close to the initial one while the Information Loss has suffered a big decrement, so the combination of the two measures forced to reduce the Score value. In this case, the Score measure started with a value of 52.99% and after the evolutionary process it ended with a value of 32.84%. This represents a decrement of a 38.03%. Table 4.4 shows the initial and final results of the three measures.

Finally in our third experiment we used the Solar Flare dataset considering 3 attributes to protect with their respective PRAM matrices. The difference

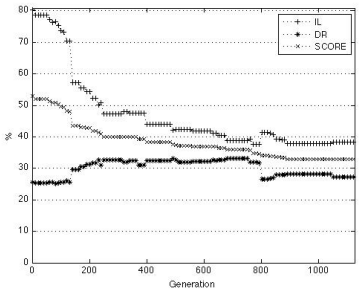


Figure 4.7: Results for the protection of all three attributes EXISTACC, PRESEMPLOY, and SAVINGS at the same time in the German dataset

	IL	DR	Score
Initial	81.18	26.49	53.83
Final	34.91	31.41	33.16

Table 4.5: Initial and final Scores for the protection of the three attributes CLASS, LARGSPOT, and SPOTDIST at the same time in the Solar Flare dataset

of this dataset is that we are going to protect nominal attributes instead of ordinal attributes like in the previous experiments in order to prove that our approach works also well with this kind of attributes. The first attribute is named CLASS with 8 different nominal categories, the second one is LARGSPOT with 7 different nominal categories, and finally, the third attribute is named SPOTDIST with 5 different nominal categories.

First we protected each attribute independently obtaining the results shown in Figure 4.8. In this case there is a slow decrement for the Information Loss measures while Disclosure Risk is having a very big decrement in the first generations. Looking at the evolution of the Score measure, we see that it has a quite regular and important decrement during all the process.

Next step is to protect all three attributes together in order to test our approach in a multi-attribute protection for this kind of attributes.

Figure 4.9 shows the behavior of Information Loss, Disclosure Risk and Score for this multi-attribute protection in this Solar Flare dataset. In this figure it can be seen that all three measures have a fast stabilization around generation 600. Disclosure risk has increased a little but Information Loss has been suffered a big decrement which causes an important reduction to the values of the Score measure. In Table 4.5 we see that the Score measure has started the evolutionary process with a value of 53.83% and after the evolutionary process it ended with a value of 33.16%. This represents a decrement of a 38.40%.

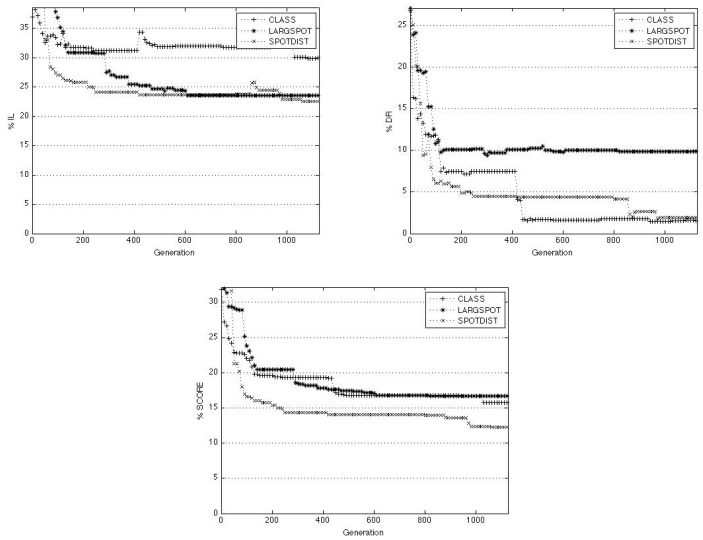


Figure 4.8: Evolution of the measures for the individual protection of the three attributes CLASS, LARGSPOT, and SPOTDIST in Solar Flare dataset

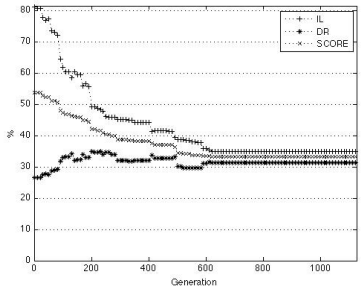


Figure 4.9: Results for the protection of all three attributes CLASS, LARGSPOT, and SPOTDIST at the same time in Solar Flare dataset

0.011	0.009	0.011	0.009	**0.926**	0.009	0.011	0.014
0.000	0.004	**0.971**	0.005	0.005	0.005	0.004	0.004
0.027	**0.625**	0.009	0.027	0.205	0.036	0.036	0.036
0.049	0.037	**0.432**	0.074	0.062	0.259	0.049	0.037
0.032	0.005	0.006	**0.928**	0.006	0.008	0.006	0.008
0.892	0.008	0.033	0.008	0.011	0.017	0.017	0.014
0.003	0.030	0.004	0.004	0.466	**0.487**	0.003	0.003
0.430	0.416	0.005	0.006	0.019	0.006	0.006	0.111

Table 4.6: Final PRAM matrix with p=0.5 for DEGREE attribute corresponding to U.S. Housing Survey dataset

Categories	'1'	'2'	'3'	'4'	'5'	'6'	'9'	'-'
Initial Freq.	98	173	251	195	170	80	33	0
Final Freq.	96	171	250	181	181	81	17	23

Table 4.7: Initial and final categories frequencies in DEGREE attribute corresponding to U.S. Housing Survey dataset

There is also another interesting point to discuss regarding the changes that have been performed between the original matrices and the final ones. Table 4.6 shows the final matrix of the DEGREE attribute (initial matrix is shown in Table 4.1). Note that, as in the initial matrix, the maximum values in each row are highlighted.

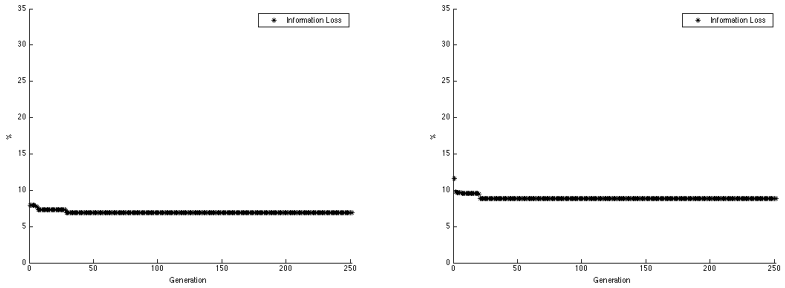
It is easy to see that after the evolutionary process the new matrix has a lot of changes, but the most remarkable one is that, in general, all the highest row values (which are the ones corresponding to the category with highest probability to substitute the original one) are outside of the diagonal, whereas in the initial matrices, largest values were all in the diagonal.

Nevertheless, it is very difficult to find a model of matrix or a general pattern that is the best for all problems. That is so because the best matrices obtained in the different evolutions have different structures and different probability distributions over the rows. To find such a model is an open problem.

Another fact that can be seen is that, in the final protected dataset, the frequencies of the categories are more balanced than in the original dataset, and the matrix also introduces more uncertainty to the protected dataset (i.e. the final distribution entropy is bigger than the one of the initial distribution). Table 4.7 shows the differences between initial and final frequencies inside the dataset after the protection.

Invariance Information Gain Testing Results

In order to test the performance of our proposed approach, we show in this section analytical results obtained from a set of experiments made on the U.S. Housing Survey of 1993 dataset from the U.S. Census Bureau [U.S. Census Bureau, 1993]. We perturb the variable DEGREE (long-term aver-



Invariant applying diagonal values control.

Non invariant applying diagonal values control.

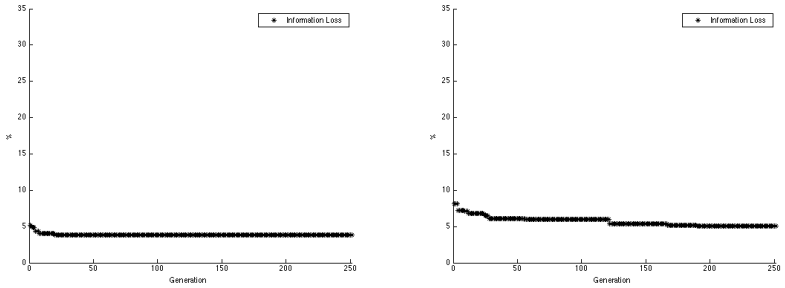
Figure 4.10: DEGREE - METRO fitness function evolution.

age degree days) and calculated the fitness function under two scenarios: crossing DEGREE with the unperturbed categorical variable SCH (schools adequate); and crossing DEGREE with the unperturbed categorical variable METRO (metropolitan areas). We chose these two scenarios because crossing DEGREE with the variable SCH represents the case where the bivariate counts distribution is skewed and crossing DEGREE with the variable METRO represents the case where the bivariate counts distribution is more uniform. This will test the performance of our approach under two extreme scenarios.

For the initial probability transition matrix we decided to use a maximum entropy matrix with a 0.6 value in the diagonal. The selection of the diagonal value is dependent on the amount of information the data provider wants to perturb. In our case, we wanted to show the ability of our algorithm to find more optimal transition matrices starting from a matrix that perturbs a large amount of data. Maximum entropy matrices ensure the maximum protection as each value has the same probability to be changed to a different value regardless of the number of individuals having that attribute.

Figures 4.10 and 4.11 show the evolution of the fitness function across 250 generations of protecting the categorical variable DEGREE and using the categorical variables METRO and SCH, respectively, to produce the bivariate counts in the fitness function under each scenario. In each figure we show the evolution of the fitness function for the case where we control the diagonal and the property of invariance and the case where we control the diagonal but do not include the property of invariance. It can be seen that for both cases on each of the figures according to METRO and SCH we experienced a decrement in the difference of bivariate frequency counts between the original and the perturbed data based on the probability transition matrix that is computed in each generation. This demonstrates that the evolutionary algorithm is optimizing the transition matrix in terms of the fitness function.

In general, adding the property of invariance to the evolutionary algorithm



Invariant applying diagonal values control.

Non invariant applying diagonal values control.

Figure 4.11: DEGREE - SCH fitness function evolution.

makes it converge faster and we obtain better results with respect to the fitness function, i.e. smaller differences between bivariate counts. We think that this is because the property of invariance is already an optimization method so, in this case, adding this property to another optimization algorithm, such as the evolutionary algorithm, we have a much faster rate of convergence and achieve better results. Moreover, as we are dealing with an algorithm that is trying to improve at each generation, we could achieve the optimization in terms of the fitness function without the property of invariance if we let the algorithm run for a longer time.

To evaluate the final optimal probability transition matrix obtained from the evolutionary algorithm after 250 generations, we assess the data utility of the perturbed data using two measures: the difference in bivariate counts (which was also the measure used as the fitness function to guide the evolutionary algorithm) as shown in Equation 4.5 and the relative absolute difference between the χ^2 test statistic calculated on the original and perturbed bivariate counts as shown in Equation 4.8. The χ^2 statistic tests the association between categorical variables (see [DeGroot and Schervish, 2012]).

$$\chi^2{}_{AbsDiff}(perturbed, original) = 100 * abs(\frac{\chi^2_{perturbed} - \chi^2_{original}}{\chi^2_{original}}) \qquad (4.8)$$

The absolute relative difference in the χ^2 statistic provides an indication of attenuation of the association between the perturbed variable and the non-perturbed variable and whether we are moving towards the assumption of independence as a result of the perturbation.

Figure 4.12 shows the results of the percent difference in the bivariate counts as defined in Equation 3 between the original dataset and the dataset that was perturbed using the final probability transition matrix for the categorical variable DEGREE. Smaller percent differences show better results for data utility.

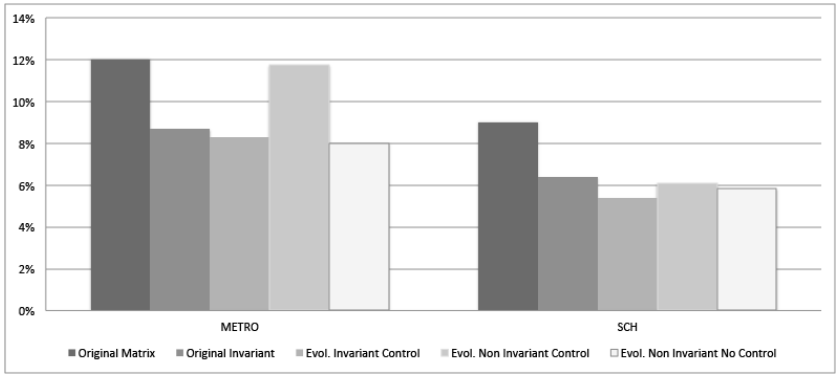


Figure 4.12: Bivariates difference comparison.

The bivariate counts are produced by crossing the categorical variable DEGREE with the non-perturbed categorical variables METRO and SCH, respectively. We compare five perturbations, four of them with controls on the diagonal probabilities and one without this control: the original probability transition matrix, the original probability transition matrix with the property of invariance, the evolved probability transition matrix with the property of invariance, the evolved probability transition matrix without the property of invariance and the evolved probability transition matrix representing the approach shown in the previous section (without both invariance and diagonal values control). It can be seen that in all cases the percent difference in bivariate counts has been reduced compared to using the original transition matrix to perturb DEGREE prior to the evolutionary algorithm. The transition matrix obtained by the evolutionary algorithm with the property of invariance is slightly outperforming the case where the property of invariance is applied directly on the original transition matrix. This shows that the optimization of the evolutionary algorithm is improving an already optimizing feature embedded in the original transition matrix. The percent difference in bivariate counts varies more for the transition matrix obtained by the evolutionary algorithm without the property of invariance. For the more uniform bivariate counts obtained by crossing DEGREE and METRO, there is only a slight improvement compared to perturbing DEGREE by the original transition matrix, but there is a greater reduction in the percent difference of bivariate counts when crossing DEGREE and SCH which is a more skewed distribution.

In general, the transition matrix obtained by the evolutionary algorithm without the property of invariance does not perform as well as adding in the property of invariance. This is because we are dealing with a stochastic evolutionary algorithm and therefore it is more difficult to continually improve when we need to control the diagonal probabilities since the algorithm sometimes needs to go through a non-valid state, i.e. a matrix with diagonal values out of range, to reach a more optimal valid state afterwards.

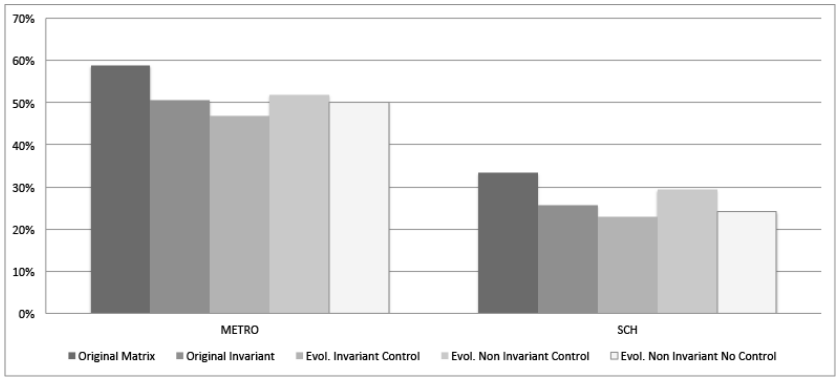


Figure 4.13: Relative absolute difference in χ^2 statistics comparison.

The results of the relative absolute difference in the χ^2 statistic are shown in Figure 4.13. In this case, the height of each bar represents the percent absolute difference between the χ^2 test statistic calculated on the original data compared to the perturbed data according to the same transition matrices evaluated above. The lower the bar means that there is less difference, i.e. the categories frequencies are more similar in both datasets. It can be seen that perturbing the variable DEGREE using the probability transition matrix obtained by the evolutionary algorithm with the property of invariance outperforms the other transition matrices, even in the case where the property of invariance is applied directly on the original transition matrix. There is more of a difference in the χ^2 test statistic based on the uniform bivariate counts of DEGREE crossed with METRO compared to the skewed bivariate counts of DEGREE crossed with SCH.

Based on the data utility measures, for the two cases of the evolutionary algorithm, we see that executing the algorithm with invariance provides, in general, probability transition matrices with better utility. In addition, our evolutionary algorithm is able to reach close results even without using the property of invariance. This is because of the use of the fitness function we have chosen. As we have used the differences between bivariate counts as the fitness function, this implies an indirect control over the marginal frequencies of the perturbed variable DEGREE, so ultimately we will have a similar effect to the case of applying the property of invariance. This fact proves that our evolutionary algorithm is able to learn this behavior and to reach transition matrices with similar effects compared to using the original transition matrix with the property of invariance.

We next move to assessing disclosure risk. We use the disclosure risk measure described in [Gouweleeuw et al., 1998] which is defined for each value (attribute) of the categorical variable. This measure computes the ratio of the expected number of records in the perturbed file with a value k equal to its value in the original dataset divided by the expected number of records in the perturbed file with a value k that arises from a different value in the original dataset. Hence, the smaller the value of the expectation ratio, the more likely it is that a record in

DEGREE values / Protections	1	2	3	4	5	6
Initial Matrix	0.846	1.634	2.629	1.894	1.599	0.676
Only Invariance	0.981	2.356	4.838	2.911	2.287	0.740
Evol. Invariant Control	0.926	1.859	4.490	1.834	1.813	0.691
Evol. Non-Invariant Control	1.019	3.268	5.198	1.536	1.567	1.200
Evol. Non-Invariant No Control	0.586	4.028	4.885	5.226	4.244	1.850

Table 4.8: Disclosure risk analysis results in the case of attributes DEGREE-SCH.

DEGREE values / Protections	1	2	3	4	5	6
Initial Matrix	0.846	1.634	2.629	1.894	1.599	0.676
Only Invariance	0.981	2.356	4.838	2.911	2.287	0.740
Evol. Invariant Control	0.982	2.356	4.837	2.911	2.283	0.739
Evol. Non-Invariant Control	1.596	3.113	2.985	3.609	1.213	1.388
Evol. Non-Invariant No Control	0.943	1.327	10.205	5.181	1.240	0.866

Table 4.9: Disclosure risk analysis results in the case of attributes DEGREE-METRO.

the perturbed file with value k did not originally belong to this category, and thus the more protection in the perturbed file. Tables 4.8 and 4.9 show the results of the disclosure risk analysis based on the two scenarios: crossing DEGREE and SCH and crossing DEGREE and METRO, under the four probability transition matrices described above.

The first two rows of 4.8 and 4.9 in the two tables are the same since we have only one original probability transition matrix with and without the property of invariance. The tables show that the resulting transition matrix from our evolutionary algorithm with the property of invariance has lower disclosure risk under the case of DEGREE-SCH and equal disclosure risk under the case of DEGREE-METRO compared to the case of applying the property of invariance on the original transition matrix. The disclosure risks vary for each category of DEGREE for the cases of the probability transition matrix generated from the evolutionary algorithm without the property of invariance.

To provide a disclosure risk-data utility summary and final assessment of the different probability transition matrices used to perturb the variable DEGREE in this experiment, Tables 4.10 and 4.11 show data utility and disclosure risk measures. The data utility measure is the difference in bivariate counts as shown in Figure 4.12. To obtain the single disclosure risk measure, we calculated a weighted average of the disclosure risk measures in Tables 4.8 and 4.9 by taking into account the frequency of each possible value of DEGREE. The weighted average is the ratio of original values in the perturbed data divided by the

Matrix type	Data Utility	Weighted Avg Risk
Evol. Invariant Control	5.38	2.338
Evol. Non-Invariant Control	6.10	2.722
Only Invariance	6.41	2.827
Evol. Non Invariant No Control	5.84	4.001
Initial Matrix	8.99	1.780

Table 4.10: Summary of disclosure risk/data utility measures in the case of DEGREE-SCH

number of values that were changed in the perturbed data across all categories of DEGREE. As exemplified by the disclosure risk paradigm, higher data utility is associated with higher disclosure risks whilst lower data utility is associated with lower disclosure risks. The data provider needs to decide on the tolerable disclosure risk threshold and release the data which has the highest data utility.

For the case of the skewed bivariate counts of DEGREE crossed with SCH in Table 4.10, we can see that the transition matrix with the highest data utility (selected with the smallest percent difference in the bivariate counts) is the matrix generated by the evolutionary algorithm with the property of invariance. This increased the disclosure risk from 1.780 based on the original transition matrix to 2.338, meaning that there are more values on average in the perturbed data that were not changed. The original transition matrix with the property of invariance had higher disclosure risk and lower data utility compared to the matrix resulting from the evolutionary algorithm with the property of invariance. In addition, the matrix obtained with the evolutionary approach without the property of invariance nor the diagonal values control is the one with the worst results (before the initial matrix).

For the case of the more uniform bivariate counts of DEGREE crossed with METRO in Table 4.11, we again see that the transition matrix with the highest data utility is the matrix generated by the evolutionary algorithm with the property of invariance, but at higher disclosure risk compared to the bivariate counts of DEGREE and SCH. From Table 4.11, the original matrix with the property of invariance and the evolutionary algorithm with the property of invariance had similar results with respect to disclosure risk and data utility. In this case, using the matrix obtained with the evolutionary approach without the property of invariance nor the diagonal values control, we obtained a similar behavior than in the skewed bivariate counts showing that applying invariance property boosts the performance of the evolutionary approach.

In conclusion from this experiment, if the data provider is willing to set the tolerable risk threshold to an average of 1 changed value for every 2.8 original values in the perturbed dataset (which is an acceptable threshold at statistical agencies), it is clear that the evolutionary algorithm with the property of invariance (and controlled diagonals) provides an optimal probability transition matrix in terms of data utility based on the difference in bivariate counts as well as the relative absolute difference in the χ^2 statistic for both the uniform

Matrix type	Data Utility	Weighted Avg Risk
Evol. Invariant Control	8.27	2.826
Only Invariance	8.69	2.827
Evol. Non Invariant No Control	8.01	4.316
Evol. Non-Invariant Control	11.74	2.549
Initial Matrix	12.00	1.780

Table 4.11: Summary of disclosure risk/data utility measures in the case of DEGREE-METRO

and skewed bivariate distributions. In the case of the transition matrices obtained when no invariance property is applied, we see that we need to rely on the control of the diagonal probabilities in order to obtain disclosure risk results similar to the transition matrices with the invariance property. In addition, we can conclude that with only 250 generations of the evolutionary algorithm we have been able to achieve transition matrices that behave similarly with the invariant transition matrices simply by choosing a good fitness function and that it is possible to reach the levels of data utility if we had let the algorithm run for many more generations.

4.2 Genetic Programming Approach for Better PRAM Matrices Generator Equations

In this section we present our genetic programming algorithm used to seek for analytical PRAM matrices. This algorithm is based on the steady-state approach because it ensures that a good solution in the actual population will be kept in the next generation. Algorithm 7 shows our proposed approach.

It can be seen that it works with a population of pop_{max} individuals where each one is an equation to build a PRAM matrix. Then, in each generation, a random subset of individuals is selected from the population and the selected programs are evaluated. Using these fitness results, selected individuals are splitted between winners (the half with higher fitness) and loosers (the other half with lower fitness). Winners are the ones that are going to be crossed or mutated in order to try to further improve their fitness, and loosers are the ones that might be substituted in the population for the next generation. The election between to cross or to mutate the winners is done randomly by drawing a random number from a uniform distribution and comparing its value with the crossover rate ($cross_{rate}$) given by the user. Finally, after that, we compute the fitness of the new offspring and then we check whether they are going to replace the loosers in the next generation. When the algorithm finishes, the best individual in the population in terms of fitness value is returned.

In the following sections we describe how to represent and initialize the population and how to compute the fitness of the individuals.

Algorithm 7 GP Steady-State Algorithm for Seeking PRAM Matrices Analytically.

Input: X original dataset, pop_{max} maximum number of programs in the population, gen_{max} maximum number of generatons, $cross_{rate}$ crossover rate
Output: $best_p$ best PRAM matrix equation in final population
$P \Leftarrow$ initializePopulation(pop_{max})
$t \Leftarrow 0$
while $t < gen_{max}$ **do**
  $S \Leftarrow$ selectSubset(P_t,sel_{max})
  $F_S \Leftarrow$ computeFitness(S,X)
  $[Winners, Loosers] \Leftarrow$ selectWinners(F_S,S)
  $a \Leftarrow$ randomNumberBetween(0,1)
  if $a < cross_{rate}$ **then**
    $newInds \Leftarrow$ cross($Winners$)
  else
    $newInds \Leftarrow$ mutate($Winners$)
  end if
  $F_{newInds} \Leftarrow$ computeFitness($newInds$,X)
  $P_{t+1} \Leftarrow$ replace($Winners$,$Loosers$,P_t)
  $t \Leftarrow t + 1$
end while
$best_p \Leftarrow$ selectBestProgram(P_t)
return $best_p$

4.2.1 Population Representation and Initialization

In the problem of finding analytical PRAM matrices we have to deal with equations that are going to be used to build the transition matrices. Then, in our genetic programming approach we have to initialize population with equations. These equations are represented as tree structures following the approach described in Section 2.2 based on a set of terminals and a set of functions.

In our case we decided to define the terminals set as $T = \{N, freq_{max}, freq_{min}, freq_i, freq_j\}$. N represents the total number of records in the dataset, $freq_{max}$ is the maximum from all catgories frequencies, $freq_{min}$ is the minimum of all categories frequencies, $freq_i$ is the frequency of the ith original category, $freq_j$ is the frequency of the jth category that the ith category can be changed to.

Regarding the functions set we decided to define it as $F = \{sum, sub, mul, div\}$ representing the sumation, the substraction, the multiplication and the division arithmetic functions. This selection of functions was driven by the fact described in Section 2.2, which states that having a function set too large can make the search for a solution harder and that to have a good function set it should only include arithmetic and logic operators. In our case, logic operators do not make sense, so we decided to use only the basic arithmetic operators.

Once we had defined the terminals and functions sets, the population was built using the half-and-half approach to ensure the diversity of individuals (See Section 2.2.1).

4.2.2 Mutation and Crossover Operators

Like in the regular evolutionary approaches presented previously in this thesis, the main genetic operators for genetic programming approaches are the mutation and the crossover. Both are executed at a certain rate given as a parameter. In our approach we decided (based on empirical tests) to use the value 0.5 for both the crossover rate and mutation rate in order to have approximately the same number of operations performed by each operator. A random value between 0 and 1 decides the operation to perform, using 0.5 as a delimiter.

In Section 2.2.2 we presented different ways to perform mutation on tree structures. In this case we decided to use the subtree mutation method. The election was based on the freedom this method gives to create new shapes for the mutated trees, this is, mutating a tree at a node n_i with a certain level l_{n_i} can produce a new subgraph with any level in the range $[1, level_{max} - l_{n_i}]$ and each branch can have different length as well. Therefore, by applying subtree mutation we allow the genetic programming algorithm be more creative when altering known solutions to find new and better ones.

In the case of the crossover we used the same tree crossover approach presented in Section 2.2.2. To do that, we only added the following constraints based on the selected nodes levels

$$level_{n1_i} + subtreeDepth_{n2_j} \leq level_{max}$$

$$level_{n2_i} + subtreeDepth_{n1_j} \leq level_{max}$$

where $level_{n2_i}$ is the level of the selected node in the second tree, $level_{n1_j}$ is the level of the selected node in the first tree, $subtreeDepth_x$ is the maximum depth of the subtree starting in node x and $level_{max}$ is the maximum number of levels allowed in a tree.

4.2.3 Fitness and Replacement

In order to guide the improvement of the programs in the population we have to define a fitness function to be applied on them. In our case, this function has several steps to follow:

1. The program (equation) to evaluate has to be executed to form a real PRAM matrix.

2. The obtained PRAM matrix has to be used to protect the original dataset.

3. Information loss and disclosure risk measures have to be applied on the masked dataset.

4. The fitness value for the given equation will be the maximum value between both mesures.

In order to execute the programs we have to transform the tree structures into a real executable programs. In our case, we have to add the tools to be able to execute the equations to get their associated PRAM matrix. To do that, we followed the approach described in Algorithm 8.

Algorithm 8 Tree Structure Equation Execution Algorithm.

INPUT: P tree-structured program, i, j PRAM matrix cell to compute indices , F functions set, T terminals set
OUTPUT: v value after executing the program for the given cell
$elements \Leftarrow$ postOrdenTraversal(P)
$S \Leftarrow$ new Stack()
$z \Leftarrow 0$
while $z < size(elements)$ **do**
  $next \Leftarrow elements.get(z)$
  if $next \in T$ **then**
    $s.push(next)$
  else
    $operand1 \Leftarrow$ getValue($s.pop()$,i,j)
    $operand2 \Leftarrow$ getValue($s.pop()$,i,j)
    $temp \Leftarrow$ performOperation($operand1$,$operand2$,$next$)
    $s.push(temp)$
  end if
  $z \Leftarrow z + 1$
end while
return $s.pop()$

In this approach we take the tree-structure equation and it is traversed using the postorden approach. By doing that we have that any function will be preceded in the final list by its operands (See Figure 4.14). Then, we simply go through this list of operands and operations saving terminal values in a stack and when a function is found we take the two first elements from the stack to use them in the function and the results is saved again in the stack. At the end, we end up with only one element in the stack which is the final result of the execution and it is returned. This approach is shown in Figure 4.15.

Next important point from the four steps to follow in the fitness function shown above is the election of information loss and disclosure risk measures to be used when evaluating the protected dataset. This point is a key one because these measures are the ones that will guide our approach to evolve towards a solution with better and minimized trade-off between them.

In the case of information loss measures we decided to use the average of the contingency table-based information loss (CTBIL), distance-based information loss (DBIL) and entropy-based information loss (EBIL) introduced in Section

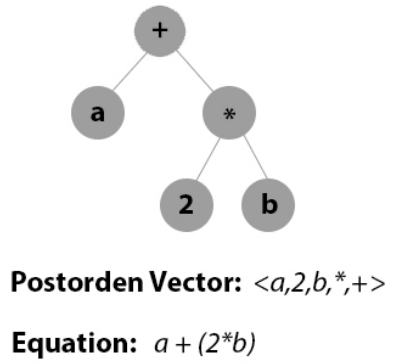


Figure 4.14: Postorden equation traversal example.

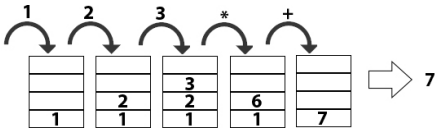


Figure 4.15: Stack evolution when executing a postorden equation vector with parameters $a = 1$ and $b = 3$.

2.4 (See Equation 4.9). On the other hand, as disclosure risk measure we used the average of interval disclosure (ID) and the maximum between distance-based record linkage (DBRL) and probabilistic record linkage (PRL) also introduced in Section 2.4 (See Equation 4.10). Finally, we need to aggregate both measures in order to have a single measure to be used as a single indicator of the quality of each matrix. To aggregate the measures we used the two different approaches also used in previous chapters in order to compare their behaviors. The first one is to take the maximum of the two values as shown in Equation 4.11, and the second one is to take the average of both values as shown in Equation 4.12. A discussion on the difference between Equations 4.11 and 4.12 has been presented in Chapter 3.4;

$$IL(X) = \frac{CTBIL(X) + DBIL(X) + EBIL(X)}{3} \tag{4.9}$$

$$DR(X) = \frac{ID(X) + max(DBRL(X), PRL(X)}{2} \tag{4.10}$$

$$Score(X) = max(IL(X), DR(X)) \tag{4.11}$$

$$Score(X) = \frac{IL(X) + DR(X)}{2} \tag{4.12}$$

At the end of the evaluation, we have to decide whether the new offspring will be part of the population for the next generation. In our case, the replacement is done by comparing the fitness score of the new offspring and the fitness of the tournament loosers in the selection process. We keep taking the program inside the loosers set which has the lowest fitness score and we replace it with the new offspring with the highest score. This process continues with the other programs until we have checked all new offspring programs.

4.2.4 Experimental Results

In this section we present the experimental results to show the performance of our approach. To do that we used the U.S. Housing, German Credit and Solar Flare datasets introduced in Section 2.7. In this case we performed a multivariate protection of three attributes in each dataset. For the U.S. Housing dataset we protected the BUILT, DEGREE and GRADE1 attributes. For the German Credit dataset we protected the EXISTACC, SAVINGS and PRESEMPLOY attributes. Finally, for the Solar Flare dataset we protected the CLASS, LARGSPOT and SPOTDIST attributes.

The experiments consisted on running 1500 generations of the genetic programming approach for each dataset several times with different configurations in order to compare them. These configurations were the different combinations of maximum depth of the tree-based equations in the population and the two fitness measures we wanted to test. Regarding the maximum depth of the tree-based equations we used trees of 5 levels at maximum in order to be able to test simple and also complex generated equations. To have this variety of levels in the population we initialized it using the half-and-half approach with a population size of 6 equations.

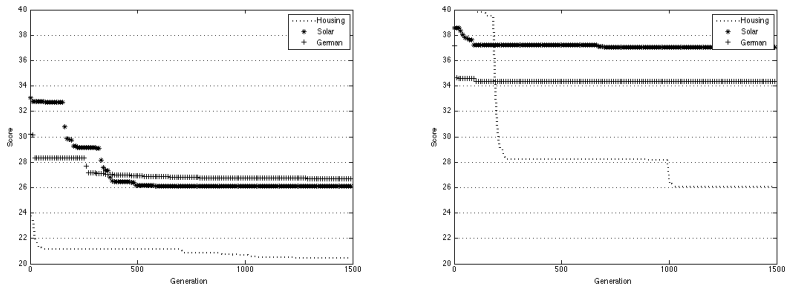


Figure 4.16: Best Equation's Scores Evolution Using the Mean (left) and Max (right) Fitness Function.

Figure 4.16 shows the evolution of the best solution's score during the 1500 generations in all three datasets using the two different (max and mean) proposed fitness functions. It can be seen that in all datasets we achieved an improvement of the best solution's score.

In the case of using the mean fitness function we can see that we got a significant improvement for each of the datasets. The U.S. Housing dataset's best solution went from a score of 23.77 to a score of 20.45, the German Credit dataset's one went from 30.23 to 26.72 and the Solar Flare's one from 33.09 to 26.10.

On the other hand, if we take a look at the results from the executions using the max fitness function we can see that we had been able to improve all datasets protections again. The U.S. Housing dataset's best solution went from a score of 39.96 to a score of 26.03, the German Credit dataset's one went from 37.18 to 34.33 and the Solar Flare's one from 38.57 to 37.03. However, in the cases of German Credit and Solar Flare datasets, we experimented much less improvement than in the U.S. Housing one. The reason for that is that this second fitness function is much more strict than the first one as it will only improve if the maximum value between IL and DR is decreased, while in the mean fitness function case it will improve when any change of their values makes the average decrease (for example, if we have $IL = 20$ and $DR = 30$, this function will think the individual improves if it goes to $IL = 5$, $DR = 40$). This behavior then makes it more difficult to improve the datasets with a small number of avaliable categories per attribute like these two because changing a category in these datasets causes more abrupt impact on IL and DR.

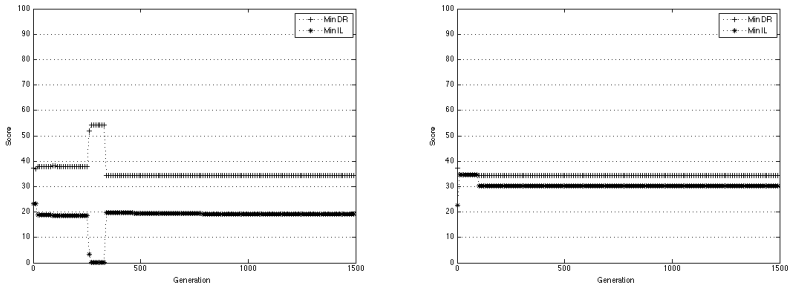


Figure 4.17: Best Equation's IL and DR Evolution for the German Credit Dataset using the Mean (left) and Max (right) Fitness Function.

It may seem that, after seeing the results of the score evolution, the mean fitness function is the one that performs better protections. Figures 4.17, 4.18 and 4.19 show the evolution of the same executions presented above but decomposing the score with the information loss and disclosure risk evolutions. There, it can be seen that in all cases when using the mean fitness function we obtained a very irregular behavior with a very distant values of information loss

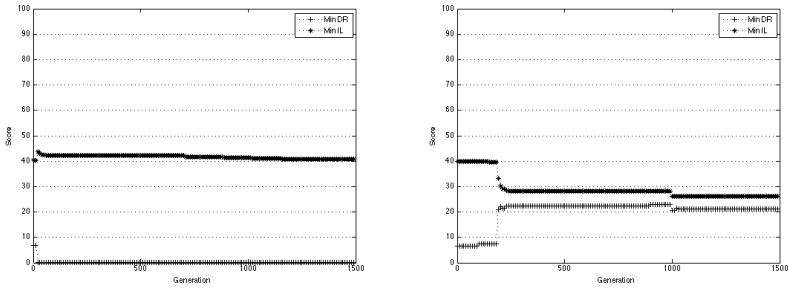


Figure 4.18: Best Equation's IL and DR Evolution for the U.S. Housing Dataset using the Mean (left) and Max (right) Fitness Function.

and disclosure risk. However, when using the max fitness function we got a very different behavior having a more controlled evolution and a kind of converging behavior of the two measures. Then, having into account that we want to achieve protections with low and balanced values for both measures, we can say that the max fitness function performs better protections than the mean fitness function.

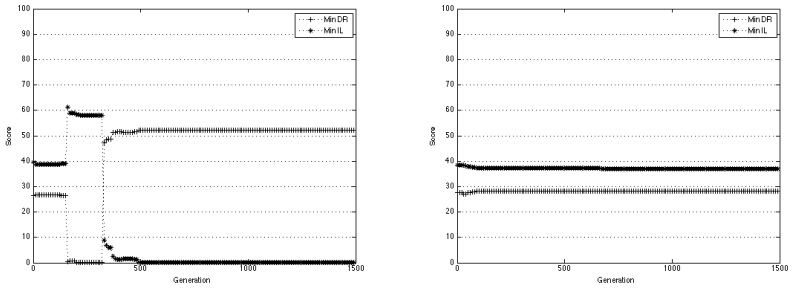


Figure 4.19: Best Equation's IL and DR Evolution for the Solar Flare Dataset using the Mean (left) and Max (right) Fitness Function.

This fact of having better protections using the max fitness function can be seen more detailed in Tables 4.12, 4.13 and 4.14 which show, for each dataset, the best protections using the two most used state-of-the-art equations to build PRAM matrices, and our genetic programming approach with the two different fitness functions.

In the case of the German Credit dataset (Table 4.12) it can be seen that using the uniform PRAM matrix results in a very bad protected dataset with very unbalanced information loss and disclosure risk measures and that using the frequency-based PRAM matrix results in a quite balanced measures values. However, our genetic programming approaches using max and mean fitness

Protection	IL	DR	$SCORE_{max}$	$SCORE_{mean}$
Freq PRAM	37.88	31.03	37.88	34.45
Unif PRAM	13.60	46.12	46.12	29.86
GP Max	30.39	34.33	34.33	32.36
GP Mean	19.11	34.33	34.33	26.72

Table 4.12: Measures comparison between standard PRAM and the output of our approach for the German Credit dataset.

functions beated the performance of the state-of-the-art measures. This is, we obtained better balanced and lower values in the genetic programming using max fitness function than in the frequency-based state-of-the-art matrix case and we also obtained more balanced and better trade-off using the genetic programming with mean fitness function than using the state-of-the-art uniform PRAM matrix.

Protection	IL	DR	$SCORE_{max}$	$SCORE_{mean}$
Freq PRAM	29.26	36.09	36.09	32.67
Unif PRAM	16.06	42.68	42.68	29.38
GP Max	37.03	28.25	37.03	32.64
GP Mean	0.03	52.19	52.19	26.10

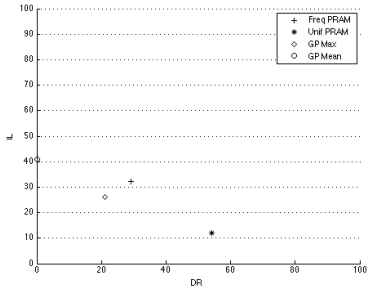
Table 4.13: Measures comparison between standard PRAM and the output of our approach for the Solar Flare dataset.

Table 4.13 shows the results of the Solar Flare dataset and we can observe a similar behavior than in the previous dataset. However, in this case we obtained very bad results in the case of using the mean fitness function in our genetic programming approach because it generated a protection with a very unbalanced measures values. This fact shows that, for this dataset, the mean fitness function is not useful.

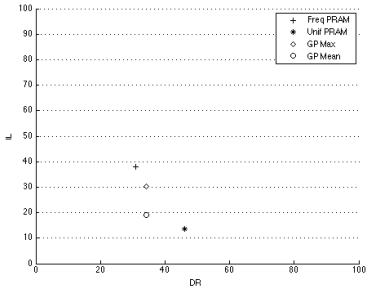
Protection	IL	DR	$SCORE_{max}$	$SCORE_{mean}$
Freq PRAM	32.31	29.18	32.31	30.74
Unif PRAM	11.92	54.15	54.15	33.04
GP Max	26.03	21.10	26.03	23.57
GP Mean	40.84	0.09	40.84	20.45

Table 4.14: Measures comparison between standard PRAM and the output of our approach for the U.S. Housing dataset.

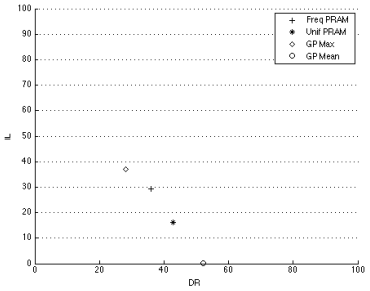
The results of the U.S. Housing dataset are shown in Table 4.14. For this last case, the results follow the same pattern. We obtained a significant improvement using the genetic programming approach with the max fitness function, and the results show again that using the mean function is a bad idea because it leads again to protections with bad trade-off between measures.



(a) U.S. Housing dataset



(b) German Credit dataset



(c) Solar Flare dataset

Figure 4.20: Scatter plot with the best protections for each PRAM method.

To wrap it up, Figure 4.20 shows, for each dataset, the position of the best protections for each approach in the space of values for information loss and disclosure risk. As said before, our goal is to obtain good protections and those protections will be the ones with balanced and low pair of values for information loss and disclosure risk. In these scatter plots, a good protection will be placed close to the diagonal and close to the ideal point (but impossible) $(0, 0)$ where we would not have any information loss neither any disclosure risk. It can be seen that in all cases the protections made by our genetic programming approach using the max fitness function are located in this area of good protections and it also beats the best state-of-the-art matrix: the frequency-based PRAM matrix. Here it can be seen again that the protections generated by the genetic programming using the mean fitness function are bad as they fall too far away from the good protections region.

Dataset		IL	DR	Score
German	Depth = 3	34.32	34.52	34.52
	Depth = 4	30.97	30.17	30.97
	Depth = 5	30.39	34.33	34.33
Housing	Depth = 3	25.98	21.11	25.98
	Depth = 4	19.22	34.57	34.57
	Depth = 5	26.03	21.10	26.03
Flare	Depth = 3	38.27	27.55	38.27
	Depth = 4	38.08	27.79	38.08
	Depth = 5	40.84	0.09	40.84

Table 4.15: Results with different limits on the depth of the equations tree structures for all three datasets using Max(IL,DR) as fitness function.

Finally, we wanted to perform one last experiment by checking the quality of the protections when we limit the depth of the equation tree to different sizes. Table 4.15 shows the results for this experiment.

Allowing the tree to have more space to adapt the equation would lead us to a better equation than if we limit the tree to be shorter which could prevent the equation to find a more larger one that better fits the problem. Having this into account, we expected to obtain better score results as we allow the tree to be deeper. However, we can see that this is not happening in our case. We think this is because of our population is too small and there is not enough individuals variety as the program keeps evolving. This reflects that this problem needs further research to solve this issue.

Chapter 5

Enhancing Protected Microdata Utility based on Pre-Clustering approach

In this chapter we present a new approach to protect categorical attributes using clustering techniques to create clusters before the protection and then protect each cluster independently. In this way we achieve protections with lower and more balanced data utility and identity disclosure.

There exist some approaches based on clustering like in [Domingo-Ferrer and Úrsula González-Nicolás, 2010] where the authors create clusters in order to generate a certain amount of synthetic data in each cluster and get an hybrid masked dataset with original and synthetic data. The difference between this approach and the method proposed here is that we generate perturbed (modified) data instead of generating whole new synthetic data.

This chapter is organized as follows. Section 5.1 gives a brief description of the whole protection algorithm. Section 5.2 presents the clustering technique used in this work. Section 5.3 describes the two different protection methods used to protect each cluster. Finally, the experimental results proving the performance of our approach are shown in Section 5.4.

5.1 Algorithm Outline

In this section we introduce the outline of the algorithm to perform the clustering-based protection. Details are given in Algorithm 9.

This algorithm consists on three different parts. The first part loads the original data file into memory and applies a clustering technique. In this work we used the c-median clustering method which we describe in Section 5.2. This clustering algorithm requires the number of clusters as input. We denote this pa-

Algorithm 9 Clustering-Based Protection Algorithm

Input: X Original data file, k Level of k-anonymity, c Number of clusters.
Output: X' Protected data file.
$Y \Leftarrow copy(X)$
clustered $\Leftarrow applyClustering$(Y,c)
for all cluster $\in$ clustered **do**
  regs $\Leftarrow getRecords$(cluster)
  regs $\Leftarrow protect$(regs, k)
  cluster$\Leftarrow setRecords$(cluster,regs)
end for
$X' \Leftarrow rebuldDataFile(Y$,clustered)
return X'

rameter by c, and, because of that, this parameter will be one of the parameters of the protection method.

The c-median clustering method is stochastic because it depends on the choice of the initial centroids. To deal with that we perform 20 clustering partitions of the original dataset and we keep the one that minimizes the global distance between each record and its related centroid.

Once we have the clustering result, the second part is started. In this case the algorithm goes through all the clusters and protect each cluster using a given parameter k as the level of k-anonymity to achieve.

Recall that, the k-anonymity concept says that a dataset is k-anonymous if it has the property that each record is indistinguishable from at least $k-1$ other records within the dataset.

In our work we used two different protection methods suitable for categorical data: microaggregation, and Global median-based k-anonymity method. Those methods are described in Section 5.3.

Finally, when all clusters are protected, the data file is rebuilt using the new data, having as a result a cluster-based protected file.

5.2 The c-Median Clustering Method

The c-median is a variation of c-means [Jain and Dubes, 1988] where instead of using the mean for each cluster when determining its centroid, it uses the median. The goal is to find c centers such that the clusters formed by them are the most compact. Formally, given a set of data points x_j, the c centers c_i are to be chosen so as to minimize the sum of the distances from each x_j to the nearest c_i. Algorithm 10 shows the outline of this method algorithm.

The categories distance measures are different when we are dealing with nominal attributes than when we are dealing with ordinal attributes. Here we use the distances introduced in Section 2.3;

The weak point of the c-median clustering method is the way to do the initialization of the centroids. This is a very important issue when doing clus-

Algorithm 10 Outline of c-median algorithm

Input: c number of clusters
Output: *part* partition of the data containing c clusters
Initialize the centroids (medians) $m_1, m_2, \ldots, m_c$
while There is any change in centroids **do**
  $part \Leftarrow$ Assign each sample to the group that has the closest centroid
  Recalculate the positions of the c centroids
end while

tering because different executions of c-median can lead to different solutions that depend on the initial centroids guess. In this work we use the dissimilarity algorithm described in [Kennard and Stone, 1969] to initialize the centroids. The main idea of this algorithm is to select a random sample from the dataset as a first centroid and then keep picking samples that maximize the dissimilarity with the set of centroids already taken. Algorithm 11 shows the outline of this initialization method.

Algorithm 11 Outline of maximum dissimilarity algorithm

Input: c number of centroids to compute
Output: S centroids set
Select the first sample at random
Add the sample to the centroids set
while $size$(centroids) $< c$ **do**
  Calculate $dissimilarity$(remaining samples,centroids set)
  Select the sample that is most dissimilar and add it to the centroids set S
end while

In order to be able to achieve k-anonymity in each cluster we need to end up with all clusters containing m records, where $m \geq k$. To ensure this constraint, if we obtain clusters with less than k records at the end of the partition we iterate merging the two smallest clusters until all remaining clusters satisfy the constraint.

5.3 The Protection Methods

In this section we present the two different protection methods we have used to protect the clusters in our approach: The microaggregation, which we described in Section 2.3, and the global median-based k-anonymity.

5.3.1 Global Median-based k-Anonimity

This method is based on the Mondrian protection method that was developed in [LeFevre et al., 2006] and reviewd in Section 2.3 of this thesis. Recall that

the Mondrian method is a greedy multidimensional recoding algorithm for categorical variables that relies on the principle of k-anonymity, generating multidimensional partitions over the whole dataset dimensions until each region have the minimum number of elements $\geq k$.

In this work we want the attribute values to be a single category instead of an interval. To have the values of this form, the median category between the two interval end point categories is taken as a value for the protected attribute.

The median category selection is done by taking all the valid categories in the attribute's domain between the two end categories of an interval and then picking the one in the middle.

It should be noticed that in order to be able to perform this protection the attribute must be ordinal.

5.4 Experimental Results

In this section we present the results of the experiments we made in order to assess the correct behavior of our approach.

The experiments were run using U.S. Housing and the German Credit datasets described in Section 2.7. Regarding the attributes selected to protect in each dataset are as follows. For the Housing dataset we protected three attributes: BUILT with 25 categories, DEGREE with 8 categories and, GRADE1 with 21 categories. In the case of German dataset the protected attributes are: EXISTACC with 5 categories, SAVINGS with 6 categories, and PRESEMPLOY with 6 categories.

Experiments shown here are based in two aspects. The first two subsections present the analysis of our approach when data is used in typical statistical studies. Then, the last subsection presents the analysis of our approach when data is used for clustering studies.

To evaluate the experiments in each kind of analysis we used two different information loss measures. However, disclosure risk is the same in both cases.

In the experiments for the general statistical microdata case the information loss has been evaluated using the CTBIL, DBIL, and EBIL measures, and the disclosure risk using the ID, DBRL, and PRL measures (See Section 2.4). However, in order to obtain a single information loss measure and a single disclosure risk measure we have aggregated them using the approaches shown in Equations 5.1 and 5.2.

$$IL(X) = \frac{CTBIL(X) + DBIL(X) + EBIL(X)}{3} \tag{5.1}$$

$$DR(X) = \frac{ID(X) + max(DBRL(X), PRL(X))}{2} \tag{5.2}$$

In the experiments for the clustering data case, we defined a clustering-based information loss by using the Rand, Wallace, and Jaccard indices introduced in Section 2.5 and aggregating them as shown in Equation 5.3. We have to take into account that information loss is expected to be an increasing function

with respect to the noise inflicted to the data. That is, information loss is a distance function between the results obtained with the original data and the ones obtained with the protected data. Nevertheless, the above mentioned indices are similarity functions. Due to this, we use the complementaries of the indices as such complementaries define distances (see Equation 5.3). Then, we use the average of the complementaries as our overall information loss. Disclosure risk is computed as in the previous case.

$$IL(p_1, p_2) = \frac{(1 - RI(p_1, p_2)) + (1 - JI(p_1, p_2)) + (1 - WI(p_1, p_2))}{3} \tag{5.3}$$

Finally, information loss and disclosure risk have also been aggregated into a score measure in our usual way:

$$Score(X) = \frac{IL(X) + DR(X)}{2} \tag{5.4}$$

5.4.1 U.S. Housing Dataset Results

Figures 5.1 and 5.2 show the results for the protections on the U.S. housing dataset using both clustered microaggregation and clustered global-median protection methods. In addition, the results for the original microaggregation protection method are shown in Figure 5.3.

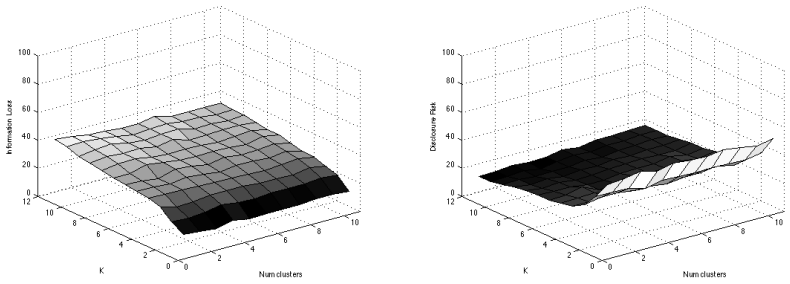


Figure 5.1: Results of clustered microaggregation protections on U.S. housing dataset.

In the case of the clustered microaggregation protection it can be seen that information loss decreases when the number of clusters increases and also when the k decreases. This is the expected behavior because the more clusters we use, the more specific the protection is and the less data is harmed. In addition, the lower k, the weaker protection and the less data is harmed too.

In the case of disclosure risk it can be observed the inverse behavior. As we introduced in Section 2.4, disclosure risk and information loss are inverse measures. So, this is the behavior we expected. Then, disclosure risk decreases when the number of clusters is small and when k is big.

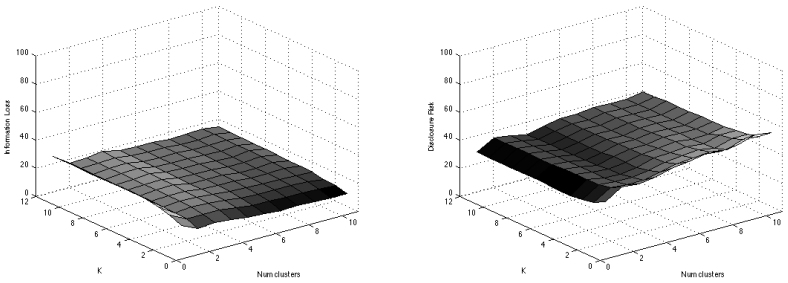


Figure 5.2: Results of clustered global-median protections on U.S. housing dataset.

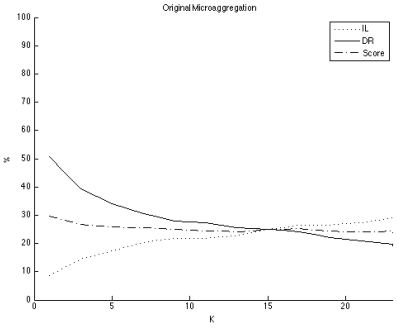


Figure 5.3: Original microaggregation results for the U.S. housing dataset.

In the case of the clustered global-median protection the obtained behavior follows the same pattern we have observed in the clustered microaggregation case but with a different speed of decrement in both k and number of clusters axis. That is, in the clustered microaggregation case we obtained a fast change of values on the number of clusters axis and a slow change of values on the k value axis while in the clustered global-median case we obtain inverse results.

This behavior is because in the clustered global-median case we are introducing some points that are not part of the original dataset and this makes the information loss higher when we want to achieve a big k-anonymity, while in the clustered microaggregation case we are generating points that are already in the original dataset what makes the information loss to be low. Furthermore this can be extended to the disclosure risk but in the inverse case.

Figure 5.4 shows the dispersion plot for the clustered microaggregation, clustered global-median, and original microaggregation protections of the U.S. housing dataset. Recall that, as we want to obtain good protections, and the quality of these protections is described by two inverse related measures (information loss and disclosure risk), the best protections will be the ones that are closer to

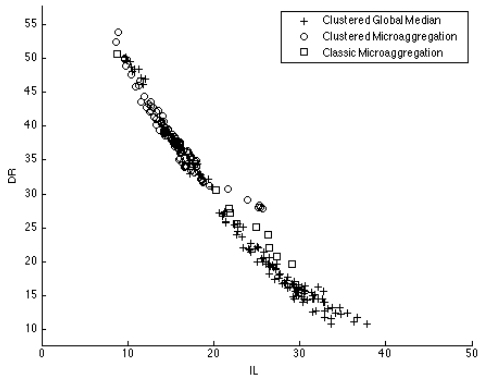


Figure 5.4: Dispersion plot of the protections on U.S. housing dataset.

Method	Information Loss	Disclosure Risk
Original Microaggregation	24.95	25.13
Clustered Microaggregation	25.63	27.92
Clustered Global-median	23.48	22.10

Table 5.1: Comparison of the best protections for the U.S. housing dataset

the (0,0) point and that are also closer to the dotted line indicating the perfect balance of the measures. Having this into account, it can be seen that the clustered global-median protection method can achieve some of the best protections, improving also the original microaggregation protections.

Finally, Table 5.1 shows the best protection in all three cases: original microaggregation, clustered microaggregation and clustered global-median. It can be seen that the best protection is the one using clustered global-median as the values are more balanced and lower than in the other cases.

5.4.2 German Credit Dataset Results

The results for the protections using both clustered microaggregation and clustered global-median methods on the german credit dataset are shown in Figures 5.5 and 5.6. In addition, the original microaggregation results are shown in Figure 5.7.

In the clustered microaggregation case we obtained, as expected, the same behavior as in the U.S. housing dataset case. However it can be seen that the information loss and disclosure risk values change much faster. This fact happens because of the reduced number of categories in this dataset attributes. Having small number of available categories makes that altering a small part of a large group of records lead to a higher information loss and a lower disclosure risk.

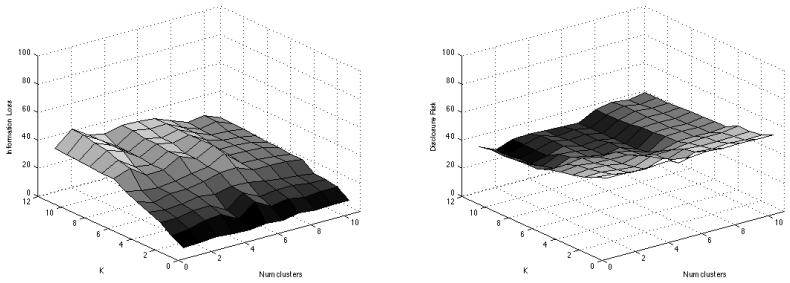


Figure 5.5: Results of clustered microaggregation protections on german credit dataset.

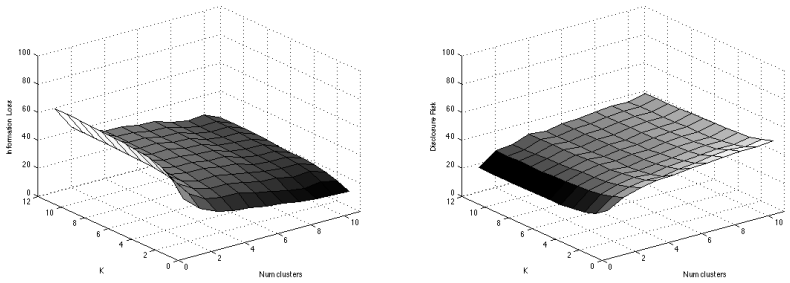


Figure 5.6: Results of clustered global-median protections on german credit dataset.

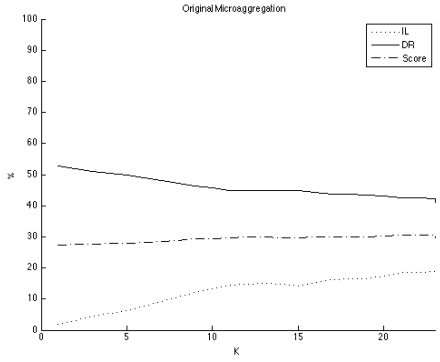


Figure 5.7: Original microaggregation results for the German dataset.

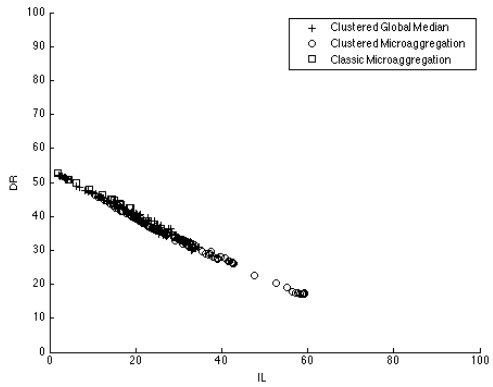


Figure 5.8: Dispersion plot of the protections on the german credit dataset.

Method	Information Loss	Disclosure Risk
Original Microaggregation	18.90	42.32
Clustered Microaggregation	31.90	31.67
Clustered Global-median	31.62	31.42

Table 5.2: Comparison of the best protections for the german credit dataset

Looking at the clustered global-median case it can be seen that information loss is lower than in the clustered microaggregation case but disclosure risk is a little bit higher. This fact occurs because of the behavior of the Mondrian algorithm which is the base of the global-median method. In Mondrian, all similar points are treated as a single point in the space, so all similar points will always be modified in the same way, so having a small number of available categories there will be a big number of similar records (i.e. there will be few points in the space). Then protecting clusters with a large number of records will alter less records than in the clustered microaggregation case, but this case is still the worst case in terms of information loss.

Figure 5.8 shows the dispersion plot for the clustered microaggregation, clustered global-median, and original microaggregation protections of the german credit dataset. As in the case of the U.S. housing dataset, the best protections will be the ones that are closer to the (0,0) point and that are also closer to the dotted line indicating the perfect balance of the measures. Having this into account, it can be seen that both clustered global-median and clustered microaggregation protection methods can achieve good protections, improving also the original non-clustered microaggregation protections.

Table 5.2 shows the best protection in all three cases: original microaggregation, clustered microaggregation and clustered global-median. In this case, it can be seen that the best protection is the one using the clustered microaggregation

as the values are more balanced and lower than in the other cases.

5.4.3 Clustering Information Loss Analysis

After the analysis based on general information loss and disclosure risk measures we wanted to perform one last analysis to test whether our new protection approaches would work better than the typical microaggregation when data is used for clustering purposes.

The process is simply based on performing 20 clustring partitions of the original dataset and take the one that minimizes the distance of each element with its assigned centroid. Then, using the same initial centroids we perform a clustering partition in the masked dataset. Finally we compare the two partitions as explained above.

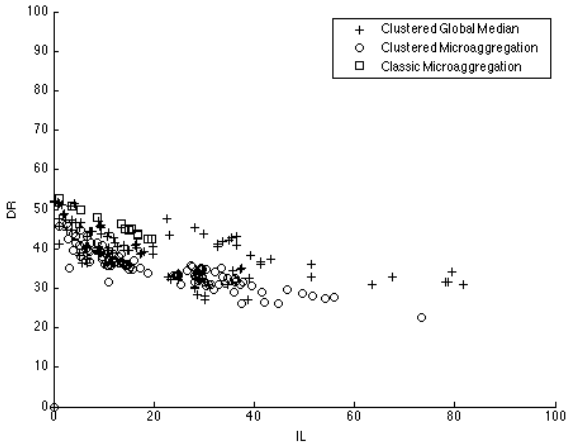


Figure 5.9: Dispersion plot with clustering-based information loss for the german credit dataset.

Figure 5.9 shows the results obtained in the case of the german credit dataset where each point represents a single parametereization of each protection method. Recall that a good protection needs to have balanced pair of values for informarion loss and disclosure risk, and that both should be low values. In this case, classic microaggregation is far from obtaining balanced measures but both protection methods we presented are able to obtain better ones. Then, in this dataset we can say that both of our pre-clustering approaches perform better than original microaggregation when data is used in clustering studies.

The results for the U.S. housing dataset are shown in Figure 5.10. In this case, it can be seen that the performance boost provided by our pre-clustering approaches is even higher. Classic microaggregation provides protections which are too far from the optimal point (0,0), but our approaches are able to obtain protections with much lower and balanced measures values. Specially in the case

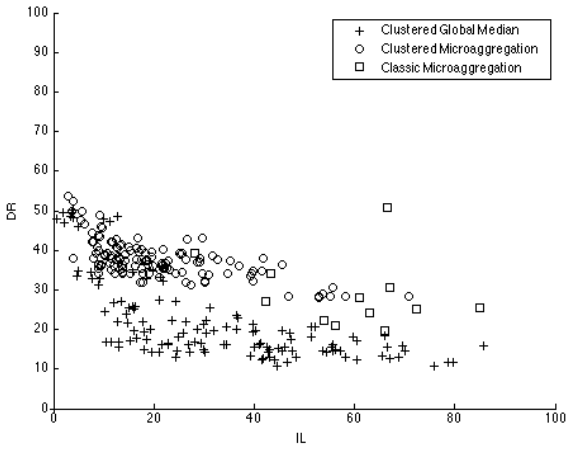


Figure 5.10: Dispersion plot with clustering-based information loss for the U.S. Housing dataset.

of the clustered global median which obtains really good protections. Then, our approaches have outperformed again the classical microaggregation in the case of when data is used in clustering studies.

Dataset		Orig Micro	Clust Micro	Clust Global Median
German credit	IL	19.62	30.51	28.71
	DR	42.55	30.63	28.50
U.S. Housing	IL	28.20	27.35	12.99
	DR	39.28	31.15	15.70

Table 5.3: Best Scores for all methods in the clustering-based experiment.

Finally, Table 5.3 presents the best solutions values for all methods in each dataset. Again, it shows that the solutions obtained with our approaches outperform the ones obtained with the original microaggregation.

Chapter 6

Social Network-extracted Graph Semantical Protection

In this chapter we present an approach wherein the extraction of implicit and explicit information is derived from a small sample of a social social network (Twitter) that seeks also to preserve user's privacy whilst maintaining information utility.

This chapter is splitted in four sections. Section 6.1 deals with the information extraction from Twitter and the construction of the social graph with both implicit and explicit information in the nodes. Section 6.2 describes the social graph protection approach presenting also how to aggregate information in the nodes. We also present new analytical measures to check the information loss and disclosure risk based on the information in the nodes in Sections 6.3 and 6.4. Finally, Section 6.5 shows some experimental results.

6.1 Social Network-Extracted Graph Generation

The first step to take is to build a crawler [Herrera-Joancomartí and Pérez-Solà, 2011] in order to get information about connected users in the social network. Algorithm 12 shows the steps followed by our crawler.

The algorithm is started with a given initial user id as the starting node in the social network, a maximum number of users we want to get information from, and a number of tweets we want to get from each user. Then, we use the Twitter API [Twitter, 2013] to get user data such as location, hashtags, urls, following users, and tweets posted by the user. Three lists are used: *unvisited* contains the ids of the not yet crawled users connected to the already crawled

Algorithm 12 Twitter Profiles Crawling Algorithm

Input: uID Initial user id, $numUsers$ Maximum number of user to crawl, $numTweets$ Number of tweets to get from each user.
Output: Y List of public available data for each user.
id $\Leftarrow$ uID
actualUser $\Leftarrow$ $getDataFromUser$(id,numTweets)
unvisited$\Leftarrow$ $getFollowingUsers$(actualUser)
visited $\Leftarrow$ [id]
$Y \Leftarrow$ [actualUser]
while ($|unvisited| > 0$) and ($|visited| < numUsers$) **do**
  id $\Leftarrow$ $getRandomId$(unvisited)
  actualUser $\Leftarrow$ $getDataFromUser$(id,numTweets)
  unvisited.$remove$(id)
  newRemaining $\Leftarrow$ $getFollowingUsers$(actualUser)
  unvisited.add(newRemaining)
  visited.add(id)
  Y.add(actualUser)
end while
return Y

ones, *visited* contains the ids of the already crawled users, and Y contains the data structures containing all the information about each crawled user.

This is executed in a loop until we reach the maximum number of users we wanted to crawl or until we have no more users in the *unvisited* list.

After this step we have a collection of structures containing information about each user

The second step to do is to use the data structures collected by the crawler in order to get a profile for each user containing his location, his connected users and his three most relevant topics of interest. In order to do this it should be noticed that information is not always explicitly given in the social networks. That is, using the Twitter API we can get the location but it is not possible to get the topics that a user is interested about because they are not specified nor described anywhere. However, these topics can be extracted using natural language processing techniques on the text of the tweets shared by the user.

In order to process the information contained in the tweets we used Web services provided by OpenCalais [Thomson Reuters, 2013], which allow for the extraction of entities such as people, organizations or events and moreover assign topics to a piece of text. In this work we only used the topic categorization capacities of OpenCalais. The 18 possible topic output values are: Business_Finance, Disaster_Accident, Education, Entertainment_Culture, Environment, Health_Medical_Pharma, Hospitality_Recreation, Human Interest, Labor, Law_Crime, Politics, Religion_Belief, Social Issues, Sports, Technology_Internet, Weather, War_Conflict and, Other.

Our first approach was to apply directly the OpenCalais Web services to

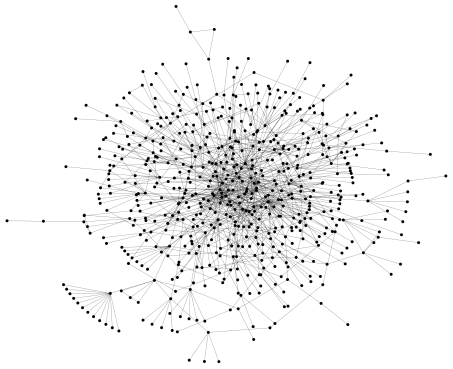


Figure 6.1: Sample graph generated from the crawled users profiles. This graph contains users (nodes) and connections between them representing "connections" between them.

the tweets text. However, as tweets are very short pieces of text (maximum of 140 characters) it was very difficult to extract topics and we got a very high percentage of users without any topic of interest found. Then, as a second approach, we used the urls within the tweets texts to enhance their semantics following the approach described in [Abel et al., 2011].

In this work, we do not use the hashtags because most of the times they are written in a useless form such as $\#ToMyFutureKids$. This forms do not provide any information to us and therefore we decided to not use hashtags but use the web pages shared in the tweets, which are much more rich semantically.

To do this, we executed two times the OpenCalais Web service to check the topics found in the tweet text and also in the text of the website shared inside the tweet. Then, the topics found in both executions were merged. At the end of processing all the tweets from a given user, the three most frequent topics were the ones taken as a result. This list of three topics is an ordered list according to that frequency.

At the end of this profiles generation step we have a set of user profiles containing the location of a user, the users who is connected with, and the three major topics of interest. So, as a result we obtained profiles combining explicit information given by the Twitter API calls and implicit information extracted from the tweets shared by the user using natural language processing tools.

As a third step, after generating the users profiles, we generated the social graph connecting all the users with the ones they are following in the real social network. Figure 6.1 shows the resulting graph representing the relations between users.

It can be seen that there are more density of edges in the center of the graph than in the borders. This is because when we crawled the social network we kept a list of remaining users to crawl which are connected to already crawled users. This fact gives higher probabilities to the first crawled users to expand

more their neighbors than to the last crawled users.

Then, as the initial user we crawled is represented in the center of the figure, all the users near to him had much more attempts to expand their neighbors than the users in the borders which are the newest ones.

6.2 Social Graph's Semantical Protection

In order to obtain a protected graph able to be published, we developed a protection method that is based on a modified version of k-Anonymity for graphs taking into account the information inside the nodes using two aggregation techniques for locations and topics of interests.

Next subsections describe each part of this approach, as well as the evaluation measures used to check the quality of the protections. The section ends up showing some experimental results.

6.2.1 Protection Algorithm Based on k-anonymity

In order to protect the social graph we adapted the principle of k-anonymity to the graphs taking into account the information contained in the nodes (which represent users profiles).

The original principle of k-anonymity says that any record of a dataset must be equal to other $k-1$ records. By doing that we have k indistinguishable records and then, the identity of the user/entity represented by these records is preserved.

In the case of the social graph we have a set of linked nodes where each node represents a user profile containing his location and his three preferred topics of interest. Then, in order to protect the graph using k-anonymity, we propose the Algorithm 13 which aggregates the information of at least k nodes and then substitute the aggregated values into the original nodes maintaining the structure of the whole social graph.

Algorithm 13 Social graph protection algorithm

Input: G social graph, k level of anonymity.
Output: G' protected social graph.
G' = copy(G)
$clusters$ = createClusters(G',k)
agg_{values} = aggregateAttributes($clusters$,G')
G' = substituteValues($clusters$,agg_{values},G')
return G'

There is some previous work on the field of protecting graphs by aggregating nodes [Campan and Truta, 2009]. However, most of the methods alter the graph structure. This makes the graph to loose its structure and in social graphs terms, it looses the communities structure and other possible interesting social properties. For that reason we decided to just substitute the aggregated values

Algorithm 14 Clustering algorithm

```
Input: G social graph, k level of anonymity.
Output: clusters set of clusters of size ≥ k.
toVisit ⇐ copy(G.nodes())
if number of nodes is multiple of k then
  numClusters ⇐ size(toVisit) / k
  lastCluster ⇐ 0
else
  numClusters ⇐ truncate(size(toVisit) / k) - 1
  lastCluster ⇐ 1
end if
allDists ⇐ computeDistancesBetweenAllNodes(toVisit,G)
visited ⇐ []
for i in [0..numClusters] do
  actualNode ⇐ random(toVisit - visited)
  visited.append(actualNode)
  index ⇐ toVisit.getIndex(actualNode)
  cluster ⇐ getClosestNodes(allDists[index],index,toVisit,k,visited)
  visited.append(cluster.nodes)
  clusters.append(cluster)
end for
if lastCluster == 1 then
  cluster ⇐ []
  for node in (toVisit - visited) do
    cluster.append(node)
  end for
  clusters.append(cluster)
end if
return Y
```

in each of the corresponding original nodes, having at least k copies of the same node. However, if the data owner wants to achieve a higher level of privacy, our method can be easily combined with any existing method dealing with edges. In addition, our approach deals with the case where the attacker know the attributes of the users whereas in the case of protecting the graph structure deal with atackers that know the relations between users.

Algorithm 14 shows the algorithm used to cluster the graph nodes. It just recalculates the number of clusters to perform and then keep taking a random unvisited node together with its $k-1$ closest nodes to form a cluster. Then all those nodes are marked as visited and the process iterates until all clusters are fulfilled.

In the next subsections we present the approaches used to aggregate locations and topics of interest, as well as the respective distances used to compare their values.

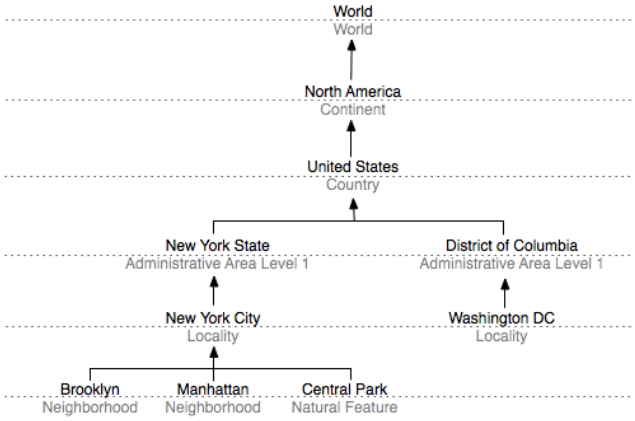


Figure 6.2: Hierarchy of location values

User's Location Values Treatment

In this work we treated the possible locations values as a hierarchical tree where the leafs of the tree contain the most specific values like neighborhoods, natural parks, or cities. Then, going up in the tree these values can be generalized to their respective provinces, states, countries, ... ending up to the value *world* which is the maximum possible generalization for any location value. Figure 6.2 shows an example of the generalization tree.

In order to build the generalization tree for all the locations in the dataset we used the Google Maps Geocoding API [Google Inc., 2013] which, given a location name, returns a structure containing all the generalized locations from the original value to its country. Then, the *continent* and *world* values are added afterwards. Algorithm 15 shows the steps taken to generate the global hierarchy tree for the location values in the graph to protect.

This algorithm shows that first of all we take all the distinct locations in the graph nodes to minimize the Google Maps Geocoding API calls. Then for each location we compute the hierarchy from the original node to the *world* value and we add this to the final hierarchy structure H_{loc}. In addition, in order to have a complete hierarchy, we check if all the values in the computed hierarchy are already in the *locationsList* hierarchy to find its own hierarchical set of values. If not, the missing values are inserted into the *locationsList*. At the end of the algorithm we have a structure having all possible generalizations for all the values in the generalization tree.

It is worth mentioning that location values also have an associated type indicating which kind of location is. In the example shown in the Figure 6.2 they can be seen below each location value. This will help us to compare location values.

Having the locations hierarchy structure, now we can go through the val-

Algorithm 15 Algorithm to build the locations hierarchy

```
Input: G social graph.
Output: H_loc locations hierarchy.
locationsList ⇐ []
H_loc ⇐ {}
for each node n in G do
  if n.location not in locationsList then
    locationsList.append(n.location)
  end if
end for
for loc in locationsList do
  hie ⇐ getFullHierarchy(loc)
  for loc_value in hie do
    if loc_value not in locationsList then
      locationsList.append(loc_value)
    end if
    H_loc[loc] ⇐ hie
  end for
end for
return H_loc
```

ues aggregation. To do this, we take as the aggregated value the closest common parent of all location values. For example, in the hierarchy shown in Figure 6.2 if we would like to aggregate $Brooklyn$ and $WashingtonDC$ we would take $UnitedStates$ as the aggregated value, and, in the same way, if we would like to aggregate $Manhattan$ and $CentralPark$ we would take $NewYorkCity$ as the aggregated value.

The last thing to discuss related to the location values is how to compute the distance between two values. In this work we used the types of the locations to compute this. Table 6.1 shows the different location types used as well as its associated level of abstraction:

Then, the distance between two locations is computed as shown in Equation 6.1 where loc_i represents a location and $||$ represents the absolute value operator. Locations have a type, and each type of location has an associated hierarchy level value (Table 6.1 show all possible types and levels), so $loc_i.type.level$ represents the associated level of the ith location's type. We also use the maximum level over all possible location types to normalize the distance (it is represented by $maxLevel(Locationtypes)$). Finally, $branch(loc_i)$ represents the path of going from the ith location to the top level location in the hierachical locations tree like the one in Figure 6.2.

Type	Level
Route	0
Sublocality	0
Natural feature	0
Point of interest	0
Neighborhood	0
Locality	1
Colloquial area	1
Administrative area level 3	2
Administrative area level 2	3
Administrative area level 1	4
Country	5
Political	5
Continent	6
World	7

Table 6.1: Location types with their related abstraction level

$$d_{loc}(loc1, loc2) = \begin{cases} 1 & \text{if } loc2 \notin branch(loc1) \\ \frac{|loc1.type.level - loc2.type.level|}{maxLevel(Locationtypes)} & \text{if } loc2 \in branch(loc1) \end{cases} \tag{6.1}$$

It can be seen that there are two different cases, one for the case when the two locations are in different branches of the location hierarchy tree (i.e. they have different paths from the location node to the root node), and one when the two locations are in the same branch.

The distance for the first case is the maximum, which is one (distances are normalized between zero and one). This is obvious because if the two locations have no relation, the distance between them must be maximum. Regarding the second case, we compute the distance between two locations of the same branch in the hierarchy tree as the difference of abstraction levels of their types, divided by the maximum abstraction level in the locations type list.

As an example, taking again the hierarchy shown in Figure 6.2 the distance between *New York City* and *Washington DC* will be 1, and the distance between *Brooklyn* and *United States* will be 0.714.

User's Topics of Interest Treatment

In this case we consider that all possible *topics of interest* values are independent one from another, so, they are not related. Having this into account, the distance between them is computed just by comparing the two values checking if they are equal or different as shown in Equation

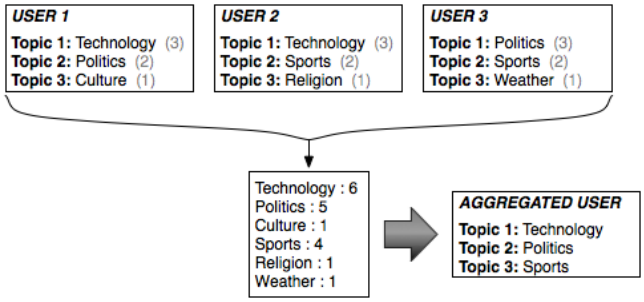


Figure 6.3: Overview of *topics of interest* values aggregation

6.2. Then, in the case that both values are equal, the distance will be the minimum (i.e. zero), otherwise the distance will be the maximum (i.e. one).

$$d_{topics}(val1, val2) = \begin{cases} 1 & \text{if } val1 \neq val2 \\ 0 & \text{if } val1 = val2 \end{cases} \tag{6.2}$$

It should be noticed that each user profile contains three attributes with topics of interest values representing the most preferred topic of interest, the second most preferred, and the third most preferred. Then, when aggregating the information of several user profiles, we would like to aggregate the values of the three attributes at the same time like when doing a voting. To do this we used the Borda's scoring rule [Borda, 1781] (which was previously defined by Nicholas of Cusa on the 15th century) giving weights to the values of the three attributes, then adding the weights of each value taking into account all the user profiles to aggregate, and finally taking the three values with highest weight as the values for the three aggregated attributes.

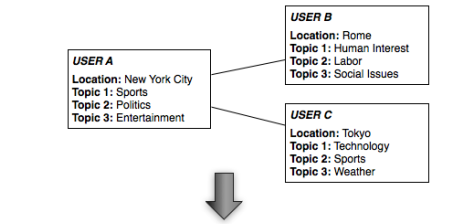
Figure 6.3 shows an example of topics of interest values aggregation where we want to aggregate three user profiles. The weights given to the topics of interest values are shown in gray color besides each value. The addition of all weights for each value results in a table where each value has a related global weight. Using this table we can then generate the aggregated profile with the three most relevant values (in this case, *technology*, *Politics* and, *Sports*).

6.2.2 Microdata Dataset Extraction

Once we have a protected social graph where each node has information about a single user, and each user is connected to his real Twitter neighbors, it is possible to extract all this information from the nodes and generate a microdata dataset.

In order to do this we extracted the information of each node placing it in a single row of the dataset. Then, the resulting dataset file has one row per user

and one column per attribute. In our case we used five attributes per user: the degree of the user node, the location of the user, the main topic of interest, the second main topic of interest, and the third main topic of interest. Figure 6.4 shows an example of microdata dataset construction from a graph with three user profiles.



DEGREE	LOCATION	TOPIC 1	TOPIC 2	TOPIC 3
2	New York City	Sports	Politics	Entertainment
1	Rome	Human Interest	Labor	Social Issues
1	Tokyo	Technology	Sports	Weather

Figure 6.4: Schematic diagram of microdata extraction process

At the end of this step we have a real social network-extracted microdata dataset with either explicit and implicit information about the users. Although this kind of datasets would be very interesting for research purposes, they must be protected before publishing. It may seem that microdata set carries less information than the graph but adding the degree to the other node attributes makes the microdata set to not fulfill the k-anonymity requirement (each record must be equal to at least k-1 other records). Then, making the microdata set k-anonymous provides better protection.

6.3 Semantic Information Loss Measures

In this section we present the measures developed in this work to assess the performance of our protection approach. This performance has two different aspects to take into account. The first one is that a protection method is considered a good method if after protecting the social graph it still maintains the original analytical information. However, on the other hand, a protection method also needs to minimize the disclosure risk of the identity of the individuals contained in the social graph.

It is easy to see that the two aspects are inversely related. That is, the more aggressive protection method, the less disclosure risk but the more analytically valid information is lost. Analogously, if we perform no protection we do not loose any analytically valid information but the disclosure risk is very high. For that reason we want to achieve a combination of low and balanced values for both aspects.

In addition, we also describe the specific information loss and disclosure risk measures used for evaluating the protection quality on the case of releasing

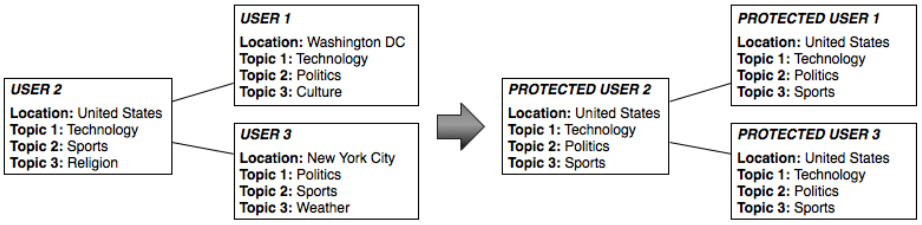


Figure 6.5: Example of nodes information aggregation

microdata extracted from the social graph.

To calculate the loose of analytically valid information in the social graph we used three different approaches that check different aspects: the attributes adjacency matrices information loss (*AAMIL*), the categories distribution information loss (*CDIL*) and, the distance-based information loss (*DBIL*).

6.3.1 Attributes Adjacency Matrices Information Loss (AAMIL)

This measure checks whether the original relations between each node attribute and the same attribute of its neighbors is maintained. For each attribute an adjacency matrix is built containing the information about the relations of each node attribute value and the values of its neighbors for the same attribute. Then, the same adjacency matrix of the original and masked graphs are compared adding the differences in all cells as the resulting score for each attribute. Finally, the average of all scores is taken as the final score for this measure. Figure 6.6 shows an example of computing this measure using the aggregated nodes shown in Figure 6.5.

This distance can be described more formally like

$$AAMIL_{graph} = \frac{\sum_{i \in Attributes} AAMIL_{attr_i}}{\#Attributes} \tag{6.3}$$

$$AAMIL_{attr_i} = \frac{\sum_{i,j \in Dom(attr_i)} abs(x_{ij} - x'_{ij})}{2 * \sum_{i,j \in Dom(attr_i)} x_{ij}} \tag{6.4}$$

where x_{ij} is the number of links from the category i to the category j in the current attribute on the original graph and, x'_{ij} is the same in the case of the masked graph.

6.3.2 Categories Distribution Information Loss (CDIL)

The *CDIL* measure focuses on the maintenance of the topics values distribution on the entire social graph. In this case, two distributions are computed for each attribute, one using the original graph and one using the masked graph. Then,

	United States	Washington DC	New York City
United States	0	1	1
Washington DC	1	0	0
New York City	1	0	0

Original location attribute adjacency matrix.

	United States	Washington DC	New York City
United States	2	0	0
Washington DC	1	0	0
New York City	1	0	0

Masked location attribute adjacency matrix.

$$AAMIL_{location} = \frac{2+1+1+0+0+0+0+0+0}{2*4} = 0.5$$

Figure 6.6: AAMIL calculation for *Location* attribute

	Technology	Politics	Sports	Religion	Culture	Weather
Original Topic 1	2	1	0	0	0	0
Masked Topic 1	3	0	0	0	0	0

$$CDIL_{topic1} = \frac{1+1+0+0+0+0}{2*3} = 0.333$$

Figure 6.7: CDIL calculation for *Topic 1* attribute

the two distributions are compared and the normalized sum of all the values frequency differences is taken as a score for each attribute. Finally, the average of all attributes score is taken as the final *CDIL* result. Figure 6.7 shows an example of computing this measure using the aggregated nodes shown in Figure 6.5.

This distance can be described more formally like

$$CDIL_{graph} = \frac{\sum_{i \in Attributes} CDIL_{attr_i}}{\#Attributes} \tag{6.5}$$

$$CDIL_{attr_i} = \frac{\sum_{j \in Dom(attr_i)} abs(freq(j) - freq(j'))}{2 * \sum_{z \in Dom(attr_i)} freq(z)} \tag{6.6}$$

where $freq(j)$ is the frequency of the value j in the original graph and, $freq(j')$ is the frequency of the value j in the masked graph.

	Location	Topic 1	Topic 2	Topic 3
User 2	United States	Technology	Sports	Religion
Protected User 2	United States	Technology	Politics	Sports

$$DBIL_{user2} = \frac{0+0+1+1}{4} = 0.5$$

Figure 6.8: DBIL calculation for User 2

6.3.3 Distance-based Information Loss (DBIL)

The last information loss measure, *DBIL*, focuses on how much the attributes values are modified in each node. To do this, for each node in the social graph, we calculate the distance between each original value and its corresponding masked value. The average of all normalized sums of the attributes differences for all nodes is taken as this measure result. In order to compute the distances, we used the Equation 6.1 for the location values, and the Equation 6.2 for the topics of interest values. Figure 6.8 shows an example of computing this measure using the aggregated nodes shown in Figure 6.5.

This distance can be described more formally as

$$DBIL = \frac{\sum_{i \in Attributes, j \in Nodes} dist(x_{ij}, x'_{ij})}{\#Attributes * \#Nodes} \tag{6.7}$$

where # is the cardinality operator, x_{ij} is the original value and, x'_{ij} is the masked value. Regarding the $dist()$ function it depends on the attribute type. To compute the distance in $location$ and $topicsofinterest$ attributes we used the approaches described in Section 6.2.

6.3.4 Average Semantic Information Loss

In order to aggregate the three information loss measures into a single one we used the mean of all three results as the average information loss. Equation 6.8 shows it more formally.

$$IL(X, X') = \frac{AAMIL(X, X') + CDIL(X, X') + DBIL(X, X')}{3} \tag{6.8}$$

6.4 Semantic Disclosure Risk Measure

This second measure is needed in order to assess the level of privacy obtained applying the protection method to the social graph.

Social network user's identity is given by the information contained in its own profile and the users they are connected with. However, taking into account that during the protection process we are producing a protected graph that for each

node there will be at least other $k-1$ indistinguishable nodes, the first fact is already solved. This is, there are no users with a unique combination of values in their attributes, so it is not possible to uniquely identify a user by using only the single node information because there are k nodes with the same combination of values.

Then, the only way to identify a user is taking into account the information about the neighbors surrounding him (only the ones directly connected to the user). To do this, we decided to take the original subgraph S_{orig} around the original node n_{root} and try to perform a matching in the entire masked graph. If we find a masked subgraph S_{mask} that has the same topology and the neighbors have the same information than in the original graph we set the node n_{root} as matched.

Finally, the quantity of matched nodes normalized with the number of nodes in the graph is taken as the disclosure risk final measure

$$DisclosureRisk(G, G') = \frac{\sum_{i \in G.nodes} match(G', subgraph(i))}{\#G.nodes} \tag{6.9}$$

where

$$match(G, S_i) = \begin{cases} 1 & \text{if } S_i \in G \\ 0 & \text{otherwise} \end{cases} \tag{6.10}$$

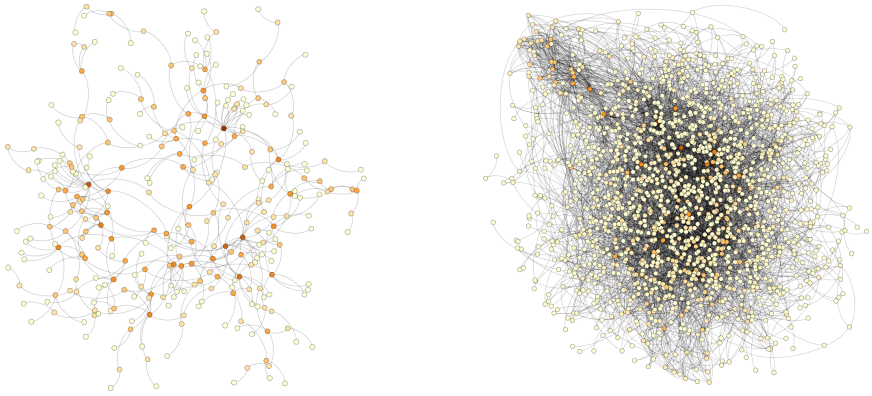
6.5 Experimental Results

In this section we present some experimental results in order to show the performance of our approach. To do that we used two different graphs. The first graph consists of 306 user profiles, with 342 edges, a diameter of 17 hops, a characteristic path length of 8 hops, an average shortest path length of 7.597 hops and, an average degree of 2. The second graph is much more complex. It has 1638 user profiles, 4622 edges between profiles, a diameter of 11 hops, a characteristic path length of 4 hops, an average shortest path of 4.2595 hops and, an average degree of 5. Both graphs have been extracted from the real-life social network Twitter using the approach described in Section 6.1.

Figure 6.9 shows the layout of these graphs. It can be seen the difference between the two graphs in the number of links between nodes. That difference is also reflected in that the second graph has a reduced average shortest path, diameter and, characteristic path length. In addition, it makes the nodes of the second graph to have a higher average degree even though there are many more nodes than in the first graph.

The aim of using these two graphs is to test the performance of our approach in the case of these two very different cases: one graph with few nodes and edges, and one with plenty of both.

Another fact that can be observed is that following our approach to build the social graph we are able to maintain the "small world" phenomenon which is present in all social networks. This fact was introduced in [Milgram, 1967] and



(a) First graph (b) Second graph

Figure 6.9: Layout of the graphs extracted from Twitter

it says that any two nodes in a network can be reached with a reduced number of steps. It can be observed than in the second graph, even having 1638 nodes, it has an average shortest path of 4.2595 hops, that is, any two nodes in the graph are connected through 4.2595 nodes as average.

6.5.1 Social Graph Protection Results

In the case of protecting the social graph we performed several experiments with different values for the parameter K (the level of k-anonymity). In addition, in order to obtain more robust results, we performed five executions for each K value and then we took the average of all executions as the final result for this value of K.

Figure 6.10 shows the information loss results of the protection for the first graph. The general behavior obtained is what was expected as the information loss increases as the K value increases too. This fact is because as K increases, the groups of records to aggregate are bigger and then the aggregated value will be less similar to all records in the group than when there are only two records to aggregate.

It can be seen that the categories distribution information loss (CDIL) is the one that grows faster. This fact is caused by the low number of nodes in the first graph because, as the value of K increases, the amount of registers to aggregate represent a quite large percentage of the entire number of nodes and, then, all their values are aggregated in the way that minimizes the distance of the original and protected values, but this has a big impact on the entire categories distribution of the graph. However, the other measures grow in a much lower rate. This makes the average information loss to moderate its growing.

Taking a look at Figure 6.11 we can see the information loss results for the

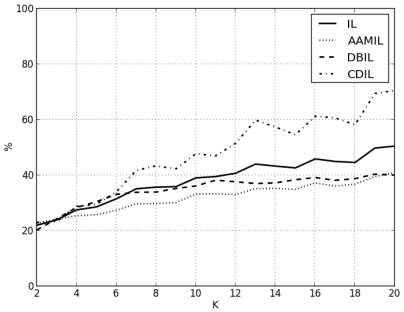


Figure 6.10: Information loss results for the protection of the first graph

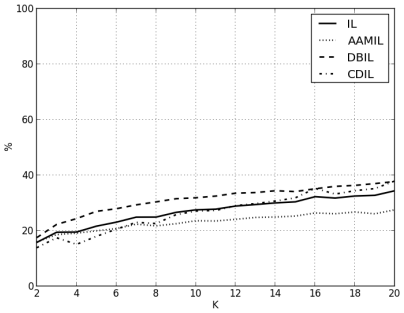


Figure 6.11: Information loss results for the protection of the second graph

second graph. In this case we obtained better results because the information loss in all values of K is much lower. This occurs because this graph has a large amount of nodes to work with, and their values aggregations are less aggressive than in the case of the first graph.

It should also be noticed that our approach was able to preserve the neighbors values for each node in order to preserve the information of the graph's social relations. This can be observed looking at the AAMIL results, which is the one with the lowest values in almost all K values.

Regarding the disclosure risk results for the first graph, Figure 6.12 shows what we obtained. The results are splitted into different cases to show the protection quality in the following scenarios: only focused on each single attribute, focused on all the topics of interest but not on the location attribute, and taking into account all attributes. It can be seen that, as expected, disclosure risk decreases in all cases as the value of K increases. The explanation for this is that, as the K value increases, we have bigger groups of records to aggregate and, then, there are more chances to have more dissimilar values to aggregate. This makes the information to go far from the original and then the disclosure

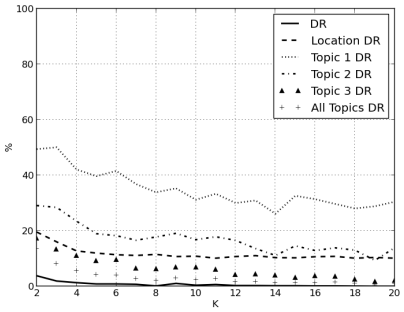


Figure 6.12: Disclosure risk results for the protection of the first graph

risk of the original information decreases.

It can also be seen that the attribute with the higher disclosure risk is the first topic of interest, which represents the preferred topic for each user. This is because when we aggregate values by doing voting aggregation the attributes that have more chances to be preserved are the first topic of each user because are the ones that obtain higher score. This is also the explanation of why the second topic of interest is the next topic with more disclosure risk, and the last one is the third topic of interest. Finally, in the case of the location attribute we obtained a low disclosure risk results. This is because this attribute is almost always generalized because there are a big number of possible categories and, even when aggregating groups of two nodes, they are forced to be generalized in order to be aggregated. These generalizations make the original locations hard to be discovered from the protected graph.

However, when we take all the topics of interest attributes together, the disclosure risk decreases much more and, when we take all attributes together, it decreases even more.

The problem of having a high disclosure risk in the first topic of interest only matters if we want to preserve the values disclosure of this specific attribute. However, the identity of a user can only be determined by all the values of the entire group of attributes, and all the values of their neighbors, so we can say that the users identity is preserved.

Figure 6.13 shows the results for the second graph. In this case it can be observed that we obtained the same behavior in all attributes and groups of attributes than in the first graph case. However, all the disclosure risk values are lower for this second graph. The reason for that is like in the case of the information loss: the number of nodes in the graph. In this case we have a much larger number of nodes and then, there are more chances to have the same configuration for several nodes, so the disclosure risk is reduced.

Finally, looking at the results shown above, we can say that our method provides a good trade-off between data utility and privacy. However, the performance varies depending on the amount of data in the social graph. For small

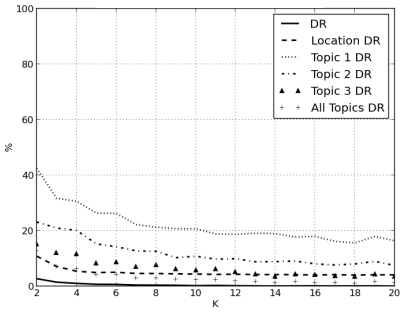


Figure 6.13: Disclosure Risk results for the protection of the second graph

graphs such as the first graph, our approach works well for low values of k because the data utility is lost rapidly as k increases. In the case of large graphs such as the second graph our approach works better and it allow to have very good results for larger values of k.

6.5.2 Extracted Microdata Protection Results

This section presents the results of testing the privacy in the case of a user wants to extract a microdata file from the protected graph. In this case we should provide enough privacy to preserve the identity of each record (i.e. user) as well as provide enough data utility in order to allow the use of this microdata file in statistical studies.

Figure 6.14 shows the results of the average information loss and disclosure risk for the first graph. As in the case of the social graph protection results, we obtained the same behavior of decreasing disclosure risk and increasing information loss as the value of K increases, as expected, for the same reasons than in the other cases.

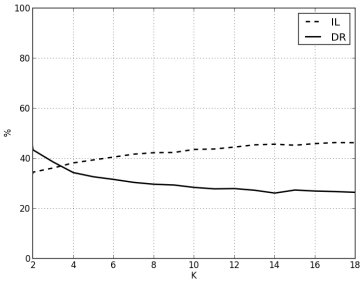


Figure 6.14: Microdata protection results for the first dataset

Apart from that, it can be seen that information loss does not experiment

a very big increase, while disclosure risk decreases faster. This is good because we are able to get more private microdata without loosing too much more valid information contained in it.

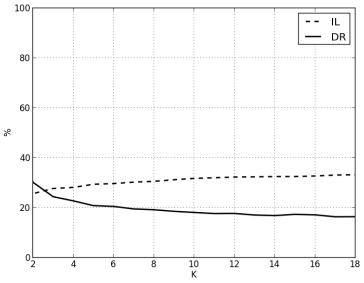


Figure 6.15: Microdata protection results for the second dataset

For the second graph's microdata, we show the obtained results in Figure 6.15. In this case, we obtained the same behavior than in the first graph microdata but getting a much lower values of information loss and disclosure risk. This is, as in the case of social graph protection, because of the amount of nodes contained in the graph. These nodes are transferred as records inside the microdata and that means that the second graph's microdata contains much more records than the first graph's microdata. Because of this, there is much more probability to obtain records having the same configuration of attributes values and it reduces the change to discover the identity of a user (i.e. record in the microdata) what makes the disclosure risk to be lower. In addition, it also makes the microdata to be more similar to the original one because the records are aggregated as small portions of the entire dataset and then the aggregated values are much more similar to the original ones.

Chapter 7

Conclusions and Future Directions

Several approaches to deal with data privacy in statistical microdata datasets and in social networks have been presented in the preceeding chapters. In this last chapter we start by reviewig all the contributions made in this thesis and then we present some conclusions to wrap up all the results obtained in this thesis. Finally, in the last section of this chapter, we introduce some future lines of research.

7.1 Summary of Contributions

In this thesis we proposed different kind of evolutionary approaches to perform better protections when dealing with microdata datasets. We also proposed a way to get better protections following a pre-clustering approach. In addition, a new way to extract and protect explicit and implicit information about the users in social networks has been proposed as well as some new measures to test the protection quality. Here we review each contribution shortly summarizing its relevance.

Evolutionary Microdata Protections Contributions.
We started proposing an evolutionary algorithm that deals directly with microdata datasets combining and altering them in order to generate better protections. Inside this approach we defined how to represent data in the population, we defined specific genetic operators, and we also introduced how to integrate information loss and disclosure risk measures inside the evolutionary algorithm to make it improve with respect to these measures. Finally, we presented experimental results showing the effectiveness of this approach.

Evolutionary PRAM Matrices Enhancement Contributions.
We have proposed a variation of the previous evolutionary algorithm but

this time it is used to optimize the Post-Randomization Method (PRAM) Markov matrices in order to make the method more effective when protecting. We presented how to represent PRAM matrices in the population and how to operate on them using genetic operators. Regarding the fitness testing we have shown how to integrate all the protection process itself in the algorithm to guide its evolution towards obtaining better protections in terms of information loss and disclosure risk. We also presented experimental results showing the gain of performance of the PRAM method when using our approach.

In addition, we also proposed a genetic programming approach to deal with PRAM matrices but working with the analytical equations that are used to generate these matrices. We defined how to represent and manipulate equations as individuals in the algorithm's population. Furthermore, we compared the use of two different fitness functions to check which one is the most suitable for this problem. Finally, we presented experimental results showing the performance of our approach.

Pre-clustering Microdata Protections Contributions.
We introduced an approach based on clustering to get better protections (i.e. protections with more balanced and low information loss and disclosure risk measures) when dealing with categorical data. We have shown that by doing a clustering and then protecting each cluster independently provides better protections than protecting the whole dataset at once. In addition we presented a new protection method based on the Mondrian method which performs better for this kind of protections. Finally, we presented experimental results showing the performance of our approach in the cases when data is used for typical statistical uses and when the data is used for clustering studies.

Social Networks Users' Information Protection Contributions.
We have shown that it is possible to extract both explicit and implicit information about users' profiles in social networks and build a social graph with this information embedded in the nodes. Then, we presented an approach to protect the social graph's nodes information, based on the k-anonymity principle. We also described how to aggregate location data and how to aggregate rankings of topics of interest in a way that the resulting aggregation has low information loss. In addition, we introduced new measures to test the masked nodes information loss and disclosure risk. We finally provided experimental results showing the performance of our approach based on data extracted from the real social network Twitter.

7.2 Conclusions

In this thesis we have covered the fields of statistical microdata protection and privacy of users' information in social networks graphs. Although we have been

focused more in the first one we provided a set of new methods and measures for both cases in order to improve the data protection quality.

The work presented for the field of statistical microdata protection was divided in two parts: different kind of evolutive protection methods and a pre-clustering protection method.

From the work presented for the first part it can be concluded that evolutive algorithms can be very helpful in improving the protection quality because it is a difficult task to obtain a good analytical method that works well in any case/dataset. That protection's quality improvement then, can be automatized using an evolutionary algorithm, which guides the solutions in the algorithm's population towards the optimum one respect to the provided fitness function. In addition we have seen that it is possible to use evoutionary algorithms to optimize transition matrices for the PRAM protection method obtaining matrices that perform better protections. It is difficult to obtain good matrices analytically so using the evolutionary algorithm it makes the task easier for us. Finally, we have shown how to add additional properties to these PRAM matrices like invariance. This property makes the evolutionary algorithm produce matrices that perform protections with better data utility.

We have also shown a way to optimize the PRAM matrices by embedding the analytical equations used to create these matrices in a genetic programming algorithm. Although there is much work still to be done in this aspect, it can be concluded that genetic programming can be a good approach to find new and enhanced PRAM matrix equations. We compared two different aggregation functions to compute the fitness (max and mean functions) and the best one has been the max function as it generated equations that performed protections with much better balance between information loss and disclosure risk, and with lower values in these measures. It has also been proven that, in almost all cases, our genetic programming approach has beated the performance of the two most used state-of-the-art PRAM matrix equations.

In the case of the pre-clustering protection method we have seen that doing a clustering partition and then protecting each cluster independently results in protections with better trade-off between information loss and disclosure risk. Then, with this pre-clustering technique records are better aggregated together in terms of similarity so, when they are protected, the loss of information is lower. In addition, using our proposed global-median protection method we outperformed the microaggregation method. This is becasue taking the aggregated value from the attribute's domain produces less information loss than taking it from the selected records' values. We also presented an experiment showing the improvement that our approach provides when data is used in clustering studies.

In the work on the topic of users' information in social networks we have proven that not all information is explicitly given in the user's profile. There is also implicit information hidden in the user's posts such as the user's preferred topics of interest. We extracted from Twitter the location and the ranking of preferred topics of interest for several users generating two different social graphs with different properties. In order to protect this information in the social graph

we have seen that locations can be aggregated by using a hierarchical approach and the ranking of topics of interest by using a voting approach. Following these approaches, the data preserves more utility.

7.3 Future Directions

In all topics described in this dissertation there are some open lines of research for future work. In this section we provide some ideas.

Evolutionary Optimization of PRAM Matrices.
In this thesis we presented two approaches that seeks for PRAM matrices, one using an evolutionary algorithm also adding the interesting property of invariance, and another using a genetic programming algorithm dealing directly with the PRAM matrix equations.

In the case of general evolutionary algorithm for PRAM matrices there are other interesting properties to be considered. One of them would be to limit the range of categories where a certain attribute's values can be changed to. This property could be very useful in cases where there is no sense to exchange a certain category to another which is too far. For example, if the attribute "age" has value 22, it would have sense to change it in a range of +/-10 years, however it would not have any sense to change it to 90 years. This kind of problems could be important when we want to preserve original statistical information. We would like to add this kind of properties in our evolutionary approach to get more robust protections.

The genetic programming approach has also several things to be improved. In our approach we normalize the Markov matrices once they are created using the equations found by the genetic programming algorithm. However, it would be very interesting to add analytical constraints in the generated equations such as creating them already normalized, that is, equations which creates Markov matrices where each row sum is 1. Finally, there is also some work to do in order to fix the behavior of our approach when allowing to have larger equation trees to achieve better equations.

Clustering-based Microdata Protection Methods.
We have presented a method to protect categorical microdata based on a pre-clustering approach which performs better protections based on statistical and clustering-based information loss and disclosure risk measures. However, it would also be interesting to test the performance using other clustering techniques, that would make our method more robust.

Social Networks Users' Information Privacy.
In the case of privacy of users' information in social graphs we presented a way to protect the graph taking into account only the information in the nodes, and skipping the links between nodes. This could also be a privacy issue. Then, we would like to add links protection (using any of the several

already existent methods for this kind of protection) in the same approach in order to obtain a protected graph in both cases.

Our Contributions

[1a] J. Jiménez, J. Marés, V. Torra, An Evolutionary Approach to Enhance Data Privacy. *Soft Computing*, volume 15, number 7, pages 1301-1311. Springer-Verlag, 2011. **(SCI Index 2011: 1.880)**

[2a] J. Marés, V. Torra, PRAM Optimization Using an Evolutionary Algorithm. In *Privacy in Statistical Databases (PSD) 2010*, J. Domingo-Ferrer and E. Magkos (Eds.), Volume 6344 of *Lecture Notes in Computer Science*, pages 97-106. Springer-Verlag, 2010.

[3a] J. Marés, V. Torra, An Evolutionary Optimization Approach for Categorical Data Protection. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops (EDBT-ICDT '12)*, D. Srivastava and I. Ari (Eds.), pages 148-157. ACM, 2012.

[4a] J. Marés, V. Torra, Clustering-Based Categorical Data Protection. In *Privacy in Statistical Databases (PSD) 2012*, J. Domingo-Ferrer and I. Tinnirello (Eds.), Volume 7556 of *Lecture Notes in Computer Science*, pages 78-89. Springer-Verlag, 2012.

[5a] J. Marés, V. Torra, On the Protection of Social Network-Extracted Categorical Microdata. In *Citizen in Sensor Networks 2012* J. Nin and D. Villatoro (Eds.), Volume 7685 of *Lecture Notes in Computer Science*, pages 33-42. Springer-Verlag, 2013.

[6a] J. Marés, V. Torra, On the Protection of Social Networks User's Information. *Knowledge-Based Systems*, volume 49, number 0, pages 134-144. Elsevier, 2013. **(SCI Index 2013: 1.519)**

[7a] J. Marés, N. Shlomo, Data Privacy Using an Evolutionary Algorithm for Invariant PRAM Matrices. *Computational Statistics*, Elsevier. **(under review)**

Other References

[Abel et al., 2011] Abel, F., Gao, Q., Houben, G., and Tao, K. (2011). Semantic enrichment of twitter posts for user profile construction on the social web. In *Proceedings of the 8th extended semantic web conference on The semanic web: research and applications - Volume II*, ESWC'11, pages 375–389, Berlin, Heidelberg. Springer-Verlag.

[Bache and Lichman, 2013] Bache, K. and Lichman, M. (2013). UCI machine learning repository. `http://archive.ics.uci.edu/ml`.

[Back et al., 2000] Back, T., Fogel, D., and Michalewicz, Z., editors (2000). *Evolutionary Computation Vol. 2: Advanced Algorithms and Operations*. Institute of Physics Publishing.

[Borda, 1781] Borda, J. (1781). Mèmoire sur les élections au scrutin. Historie de l'Académie Royale des Sciences, Paris.

[Brand et al., 2002] Brand, R., Domingo-Ferrer, J., and Mateo-Sanz, J. (2002). Reference data sets to test and compare SDC methods for protection of numerical microdata. Unscheduled Deliverable, European Project IST–2000–25069 CASC.

[Bremermann et al., 1966] Bremermann, H., Rogson, M., and Salaff, S. (1966). Global Properties of Evolution Processes. In Pattee, H. H., Edlsack, E. A., Fein, L., and Callahan, A. B., editors, *Natural Automata and Useful Simulations*, pages 3–41. Spartan Books, Washington, DC.

[Campan and Truta, 2009] Campan, A. and Truta, T. (2009). Data and structural k-anonymity in social networks. In Bonchi, F., Ferrari, E., Jiang, W., and Malin, B., editors, *Privacy, Security, and Trust in KDD*, volume 5456 of *Lecture Notes in Computer Science*, pages 33–54. Springer Berlin Heidelberg.

[Caruana and Schaffer, 1988] Caruana, R. and Schaffer, J. (1988). Representation and hidden bias: Gray vs. binary coding for genetic algorithms. In *Proc. of the 5th Int. Conf. on Machine Learning*, pages 153–161, Los Altos, CA. Morgan Kaufmann.

[Davis and Mitchell, 1991] Davis, L. and Mitchell, M. (1991). Handbook of Genetic Algorithms. *Van Nostrand Reinhold*.

[De Jong and Kenneth, 1975] De Jong, K. and Kenneth, A. (1975). *An analysis of the behavior of a class of genetic adaptive systems.* University of Michigan, Ann Arbor, MI, USA. AAI7609381.

[De Jong and Kenneth, 1993] De Jong, K. and Kenneth, A. (1993). Genetic algorithms are NOT function optimizers. In Whitley, D. L., editor, *Foundations of Genetic Algorithms 2*, pages 5–17. San Francisco, CA: Morgan Kaufmann.

[De Jong and Spears, 1992] De Jong, K. and Spears, W. (1992). A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 5(1):1–26.

[De Wolf and Van Gelder, 2004] De Wolf, P. and Van Gelder, I. (2004). An empirical evaluation of PRAM. Discussion Paper No. 04012, Statistics Netherlands, Voorburg/Heerlen.

[Defays and Nanopoulos, 1993] Defays, D. and Nanopoulos, P. (1993). Panels of enterprises and confidentiality: The small aggregates method. in Proceedings of the 1992 Symposium on Design and Analysis of Longitudinal Suerveys, Ottawa: Statistics Canada, pp.195-204.

[DeGroot and Schervish, 2012] DeGroot, M. and Schervish, M. (2012). *Probability and Statistics (4th Edition).* Addison-Wesley series in statistics. Addison-Wesley.

[Dick, 2005] Dick, G. (2005). A comparison of localised and global niching methods. In *Proc. of the 17th Annual Colloquium of the Spatial Information Research Centre*, pages 91–101.

[Domingo-Ferrer and Mateo-Sanz, 2002] Domingo-Ferrer, J. and Mateo-Sanz, J. (2002). Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering*, 14:189–201.

[Domingo-Ferrer et al., 2001] Domingo-Ferrer, J., Mateo-Sanz, J., and Torra, V. (2001). Comparing SDC methods for microdata on the basis of information loss and disclosure risk. In *New Techniques and Technologies for Statistics: Exchange of Technology and Know-How, ETK-NTTS'2001*, pages 807–826, Creta, Hersonissos.

[Domingo-Ferrer and Torra, 2001a] Domingo-Ferrer, J. and Torra, V. (2001a). Disclosure control methods and information loss for microdata. In Doyle, P., Lane, J. I., Theuwes, J. J. M., and Vatz, L., editors, *Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies*, chapter 5, pages 91–110. Elsevier.

[Domingo-Ferrer and Torra, 2001b] Domingo-Ferrer, J. and Torra, V. (2001b). A quantitative comparison of disclosure control methods for microdata. In Doyle, P., Lane, J. I., Theuwes, J. J. M., and Zayatz, L., editors, *Confidentiality, disclosure, and data access : Theory and practical applications for statistical agencies*, pages 111–133. Elsevier.

[Domingo-Ferrer and Torra, 2002] Domingo-Ferrer, J. and Torra, V. (2002). Distance-based and probabilistic record linkage for re-identification of records with categorical variables. *Butlletí de lACIA*, 28:243–250.

[Domingo-Ferrer and Torra, 2004] Domingo-Ferrer, J. and Torra, V. (2004). Disclosure risk assessment in statistical data protection. *Journal of Computational and Applied Mathematics*, 164:285–293.

[Domingo-Ferrer and Torra, 2005] Domingo-Ferrer, J. and Torra, V. (2005). Ordinal, continuous and heterogeneous k-anonymity through microaggregation. *Data Mininig Knowledge Discovery*, 11(2):195–212.

[Domingo-Ferrer and Úrsula González-Nicolás, 2010] Domingo-Ferrer, J. and Úrsula González-Nicolás (2010). Hybrid microdata using microaggregation. *Information Sciences*, 180(15):2834–2844.

[Duncan et al., 2001a] Duncan, G., Fienberg, S., Krishnan, R., Padman, R., and Roehrig, S. (2001a). Disclosure limitation methods and information loss for tabular data. In Doyle, P., Lane, J. I., Theuwes, J. J. M., and Vatz, L., editors, *Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies*, chapter 7, pages 135–166. Elsevier.

[Duncan et al., 2001b] Duncan, G., Keller-McNulty, S., and Stokes, S. (2001b). Disclosure risk vs. data utility: The R-U Confidentiality Map. Technical Report 121, National Institute of Statistical Sciences, NISS, North Carolina, USA.

[Eshelman et al., 1989] Eshelman, L., Caruana, R., and Schaffer, J. (1989). Biases in the crossover landscape. In *Proceedings of the third international conference on Genetic algorithms*, pages 10–19, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Goldberg, 1989] Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

[Google Inc., 2013] Google Inc. (2013). Google Maps geocoding api. `https://developers.google.com/maps`.

[Gouweleeuw et al., 1998] Gouweleeuw, J., Kooiman, P., Willenborg, L., and de Wolf, P. (1998). Post randomization for statistical disclosure control: Theory and implementation. *Journal of Official Statistics*, 14(4):463–478.

[Greiner et al., 2005] Greiner, D., Winter, G., Emperador, J. M., and Galván, B. (2005). Gray coding in evolutionary multicriteria optimization: application in frame structural optimum design. In *Proceedings of the Third international conference on Evolutionary Multi-Criterion Optimization*, EMO'05, pages 576–591, Berlin, Heidelberg. Springer-Verlag.

[Hansen and Mukherjee, 2003] Hansen, S. and Mukherjee, S. (2003). A polynomial algorithm for optimal univariate microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):1043–1044.

[Hay et al., 2007] Hay, M., Miklau, G., Jensen, D., Weis, P., and Srivastava, S. (2007). Anonymizing social networks. Technical report, SCIENCE.

[Herrera-Joancomartí and Pérez-Solà, 2011] Herrera-Joancomartí, J. and Pérez-Solà, C. (2011). Online social honeynets: trapping web crawlers in osn. In *Proceedings of the 8th international conference on Modeling decisions for artificial intelligence*, MDAI'11, pages 1–16, Berlin, Heidelberg. Springer-Verlag.

[Holland, 1975] Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press. Second edition: MIT Press, 1992.

[Hundepool et al., 1998] Hundepool, A., Willenborg, A., Wessels, A., Van Germerden, L., Tiourine, S., and Hurkens, C. (1998). μ-argus users manual version 2.5. Proceedings of Joint UNECE-Eurostat Work Session on Statistical Data Condifentiality, pp.87-98, Luxembourg.

[Jain and Dubes, 1988] Jain, A. and Dubes, R. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Jiménez and Torra, 2009a] Jiménez, J. and Torra, V. (2009a). JPEG-based microdata protection methods. Technical Report IIIA–TR–2009–06, IIIA - CSIC.

[Jiménez and Torra, 2009b] Jiménez, J. and Torra, V. (2009b). Utility and risk of JPEG–based continuous microdata protection methods. In *IEEE Proc. of the 4th Int. Conf. on Availability, Reliability and Security, ARES*.

[Kennard and Stone, 1969] Kennard, R. and Stone, L. (1969). Computer Aided Design of Experiments. *Technometrics*, 11:137–148.

[Kooiman et al., 1997] Kooiman, P., Willenborg, L., and Gouweleeuw, J. (1997). PRAM: A method for disclosure limitation of microdata. Research Paper No. 9705, Voorburg: Statistics Netherlands.

[Koza, 1992] Koza, J. (1992). *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, USA.

[Laszlo and Mukherjee, 2005] Laszlo, M. and Mukherjee, S. (2005). Minimum spanning tree partitioning algorithm for microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, 17(7):902–911.

[LeFevre et al., 2006] LeFevre, K., DeWitt, D., and Ramakrishnan, R. (2006). Mondrian multidimensional k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 25–, Washington, DC, USA. IEEE Computer Society.

[Lin, 1965] Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269.

[Liu and Terzi, 2008] Liu, K. and Terzi, E. (2008). Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 93–106, New York, NY, USA. ACM.

[Mahfoud, 1992] Mahfoud, S. (1992). Crowding and preselection revisited. Technical Report 92004, Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois. Also in Parallel Problem Solving From Nature, PPSN, 2:27-36, 1992.

[Marés and Torra, 2010] Marés, J. and Torra, V. (2010). PRAM optimization using an evolutionary algorithm. In *Proceedings of the 2010 international conference on privacy in statistical databases*, PSD'10, pages 97–106, Berlin, Heidelberg. Springer-Verlag.

[Mateo-Sanz et al., 2005] Mateo-Sanz, J., Domingo-Ferrer, J., and Sebé, F. (2005). Probabilistic information loss measures in confidentiality protection of continuous microdata. *Data Mining and Knowledge Discovery*, 11(2):181–193.

[Milgram, 1967] Milgram, S. (1967). The small world problem. *Psychology today*, 2(1):60–67.

[Moore Jr., 1996] Moore Jr., R. (1996). Controlled data-swapping techniques for masking public use microdata sets. Research report, RR 96-04, Statistical Research Division Report Series, U.S. Bureau of the Census.

[Nettleton et al., 2011] Nettleton, D., Sáez-Trumper, D., and Torra, V. (2011). A comparison of two different types of online social network from a data privacy perspective. In *Proceedings of Modeling Decision for Artificial Intelligence - 8th International Conference*, pages 223–234. Springer-Verlag.

[Nin et al., 2008a] Nin, J., Herranz, J., and Torra, V. (2008a). On the disclosure risk of multivariate microaggregation. *Data and Knowledge Engineering*, 67(3):399–412.

[Nin et al., 2008b] Nin, J., Herranz, J., and Torra, V. (2008b). Rethinking rank swapping to decrease disclosure risk. *Data and Knowledge Engineering*, 64(1):346–364.

[Oganian and Domingo-Ferrer, 2001] Oganian, A. and Domingo-Ferrer, J. (2001). On the complexity of microaggregation. in Second Joint UNECE-Eurostat Work Session on Statistical Data Confidentiality, Skopje.

[Rand, 1971] Rand, W. M. (1971). Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850.

[Rechenberg, 1973] Rechenberg, I. (1973). *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog.

[Schaffer et al., 1989] Schaffer, J., Caruana, R., Eshelman, L., and Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. In *ICGA*, pages 51–60.

[Schwefel, 1977] Schwefel, H. (1977). *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, volume 26 of *ISR*. Birkhaeuser, Basel/Stuttgart.

[Shetty and Adibi, 2005] Shetty, J. and Adibi, J. (2005). Discovering important nodes through graph entropy the case of enron email database. In *Proceedings of the 3rd international workshop on Link discovery*, LinkKDD '05, pages 74–81, New York, NY, USA. ACM.

[Shlomo and Young, 2008] Shlomo, N. and Young, C. (2008). Invariant post-tabular protection of census frequency counts. In Domingo-Ferrer, J. and Saygin, Y., editors, *Privacy in Statistical Databases*, volume 5262 of *Lecture Notes in Computer Science*, pages 77–89. Springer.

[Stokes and Torra, 2012] Stokes, K. and Torra, V. (2012). Reidentification and k-anonymity: a model for disclosure risk in graphs. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 16(10):1657–1670.

[Sywerda, 1989] Sywerda, G. (1989). Uniform crossover in genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, pages 2–9, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Tan et al., 2005] Tan, P.-N., Steinbach, M., and Kumar, V. (2005). Introduction to data mining. ISBN 0-321-32136-7.

[Thomson Reuters, 2013] Thomson Reuters (2013). OpenCalais web services. `http://www.opencalais.com/calaisAPI/`.

[Torra and Domingo-Ferrer, 2001] Torra, V. and Domingo-Ferrer, J. (2001). Disclosure control methods and information loss for microdata. in Confidentiality, disclosure, and data access : Theory and practical applications for statistical agencies, pp 91-110, Elsevier.

[Turing, 1948] Turing, A. (1948). Intelligent machinery. Report for National Physical Laboratory.

[Turing, 1950] Turing, A. (1950). Computing Machinery and Intelligence. *Mind*, LIX:433–460.

[Twitter, 2013] Twitter (2013). Twitter API. `http://api.twitter.com/`.

[U.S. Census Bureau, 1993] U.S. Census Bureau (1993). U.S. Housing Survey of 1993. `http://quickfacts.census.gov`.

[Wallace, 1983] Wallace, D. (1983). "comment". in Journal of the American Statistical Association, vol. 78, pp 56976, (on the preceding paper, "A method for comparing two hierarchical clusterings" by E. B. Fowlkes and C. L. Mallows).

[Willenborg and Waal, 2000] Willenborg, L. and Waal, T. D. (2000). Elements of statistical disclosure control. Lecture Notes in Statistics, Springer.

[Wolf et al., 1998] Wolf, P. D., Gouweleeuw, J., Kooiman, P., and Willenborg, L. (1998). Reflections on PRAM. In: Statistical Data Protection, Luxembourg, Office for Official Publications of the European Communities, pp. 337-349.

[Yancey et al., 2002] Yancey, W., Winkler, W., and Creecy, R. (2002). Disclosure risk assessment in perturbative microdata protection. In *Inference Control in Statistical Databases: From Theory to Practice*, volume 2316 of *LNCS*, pages 135–152. Springer.

[Zhou and Pei, 2008] Zhou, B. and Pei, J. (2008). Preserving privacy in social networks against neighborhood attacks. In *Proc. IEEE 24th International Conference on Data Engineering 2008 (ICDE 2008*, pages 506–515. IEEE.