A
Dissertation On

WATERMARKING RELATIONAL DATABASES USING OPTIMIZATION BASED TECHNIQUES

Submitted in the partial fulfillment of the requirements
For the award of degree of
Master of Engineering
In
Computer Technology And Applications
Delhi University
Delhi

Submitted By
Vidhi Khanduja (19/ME/CTA/08)

Under the Guidance of
Prof. O. P. Verma
Head of Department,
Department of Information Technology

Department of Computer Engineering
Delhi College of Engineering
Delhi University

CERTIFICATE

Date: _____

This is certified that the work contained in this dissertation entitled "Watermarking Relational Databases using Optimization Based Techniques**."** by Vidhi Khanduja (19/ME/CTA/08) is the requirement of the partial fulfilment for the award of degree of Master of Engineering in Computer Technology and Application at Delhi College of Engineering.

The work is a bonafide piece of work carried out and completed under my supervision and guidance. She has completed her work with utmost sincerity and diligence. The work embodied in this major project has not been submitted for the award of any other degree to the best of my knowledge.

(Project Mentor)

Prof. O. P. Verma
Head of Department
Dept. of Information Technology
Delhi Technological University
(Formerly Delhi College of Engineering)

(Head of Department)

Dr. Daya Gupta
Head of Department
Dept. of Computer Engineering
Delhi Technological University
(Formerly Delhi College of Engg.)

ACKNOWLEDGEMENT

It gives me a pleasure to express my profound gratitude to my project mentor Prof.O.P.Verma, Head of Department, Information Technology, Delhi Technological University for his valuable and inspiring guidance throughout the progress of this project. His rich experience, cooperation and valuable support have been a great motivation for me to complete this dissertation.

I would like to extend my heartfelt thanks to Dr. (Mrs.) Daya Gupta, Head of Department, Department of Computer Engineering, Delhi Technological University for keeping the spirits high and clearing the visions to work on the project.

I humbly extend my words of gratitude to other faculty members of this department for providing their valuable help and time whenever it was required.

Vidhi Khanduja
Roll No. 19/ME/CTA/08
Master of Engineering
(Computer Technology and Applications)

ABSTRACT

Proving ownership rights on outsourced relational databases is a crucial issue in today's internet-based application environments and in many content distribution applications. In this dissertation, a mechanism is presented for proof of ownership and ownership identification based on the secure embedding of a robust imperceptible watermark in relational data. Watermarking problem of relational databases is formulated as a constrained optimization problem and Genetic and Bacterial Foraging Optimization algorithms are implemented to solve the optimization problem and to handle the constraints. Proposed watermarking technique is resilient to watermark synchronization errors because it uses a partitioning approach that does not require marker tuples. Watermark decoding is based on a threshold-based technique characterized by an optimal threshold that minimizes the probability of decoding errors. Experimental results have shown that technique proposed in this dissertation is resilient to tuple deletion, alteration, and insertion attacks.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Ant Colony Optimization	ACO
Bacterial Foraging Optimization Algorithm	BFOA
Cryptographic Pseudorandom Sequence Generators	CPSG
Discrete Wavelet Transform	DWT
Evolutionary Programming	EP
Evolutionary Strategies	ES
Escherichia Coli	E.Coli
Genetic Algorithm	GA
Least Significant Bit	LSB
Message Authentication Code	MAC
Message Digest	MD5
Most Significant Bit	MSB
Pattern Search	PS
Particle Swarm Optimization	PSO
Random Access Memory	RAM
World Trade Organization	WTO

LIST OF SYMBOLS

Items	**Symbols**
Dataset	D
Hash function	H
Primary Key	P
Concatenation operator	||
Usability Constraints	G
Partition Set	S_i
Manipulation Vector	Δ
Mean	μ
Threshold	T
Objective Function	J, θ_c
Population	S
Watermarked Dataset	D_W
Reference point	ref
Probability of Error	P_{err}

CHAPTER 1
INTRODUCTION

Internet is an excellent distribution system for digital media because it is inexpensive, eliminates warehousing and stock and delivery is almost instantaneous. Copying and distributing digital assets have become layman's task. The Internet is exerting tremendous pressure on data providers to create services that allow users to search and access databases remotely. Although this trend is a boon to end users, it exposes the data providers to the threat of data theft. Providers are therefore demanding technology for identifying pirated copies of their databases as they are concerned about the copyright of their products

Digital watermarking is an approach to solve such problems related to ownership issues, tamper detection, etc. The watermarking technique introduces small errors into the object being watermarked. These intentional errors are called marks, and all the marks together constitute the watermark. The marks are chosen so as to have an insignificant impact on the usefulness of the data [1] and are placed in such a way that a malicious user cannot destroy them without making the data significantly less useful.

Although watermarking does not prevent illegal copying, it deters such copying by providing a means for establishing the original ownership of a redistributed copy. Unlike encryption and hash description, typical watermarking techniques modify ordinal data as a modulation of the watermark information and inevitably cause permanent distortion to the original data and therefore cannot meet the integrity requirement of the data as required in some applications

1.1 DATABASE COPYRIGHT

Legislations for Protection of Intellectual Property

The intellectual property is protected and governed by appropriate national legislations and international treaties [Gupta][wiki copyrights]. The main national legislations are The Indian Patents Act, 1970, The Trade and Merchandise Act, 1958, The Copyright Act, 1957 with amendments to the Act in 1994, The Designs Act, 1911. For the protection of databases, it is the Copyright Act that is most important. The key International Treaties on copyright are WTO Agreement on Trade Related Aspects of Intellectual Property Rights, Berne Convention for the Protection of Literary and Artistic Works, Universal Copyright Act.

Key Issues in Copyright Protection of Databases

In theory, databases may be protected *per se* under copyright within a national law. Key issues in copyright protection of databases are concerned with:

i. Individual records within the database are recognized as literary works and are thus individually and separately have their own proprietary value.

ii. Mode of compilation of database may be protected solely as compilations because skill and effort were extended in making the collection.

Recently, the Delhi High Court in a case held that even if only labour had gone into the making of a database, it is good enough and a copyright subsists in the compilation. A US court in a similar case involving copying of telephone directory refused copyright protection on the grounds that it was not an intellectual task as it did not involve special skills.

However, what is important is that compilation of databases does need classification systems to organise data. The intellectual skills of `data organisation' facilities quick retrieval and a variety of analyses of data.

Databases - whether in online form, CD-ROM form, or any other form _ are thus treated as a standard copyright work. Such compilations should not be downloaded or copied in any other way without prior permission (except for small portions only for fair use such as research or private study). They should not be distributed around local or wide area networks to multiple key stations without prior permission.

1.2 MOTIVATION

The rapid growth of the Internet and related technologies has offered an unprecedented ability to access and redistribute digital contents. In such a context, enforcing data ownership is an important requirement, which requires articulated solutions, encompassing technical, organizational, and legal aspects [Vaas]. Although we are still far from such comprehensive solutions, in the last years, watermarking techniques have emerged as an important building block that plays a crucial role in addressing the ownership problem. Such techniques allow the owner of the data to embed an imperceptible watermark into the data. A watermark describes information that can be used to prove the ownership of data such as the owner, origin, or recipient of the content. The problem of watermarking relational data has not been given appropriate attention. There are, however, many application contexts for which data represent an important asset, the ownership of which must thus be carefully enforced. This is

the case, for example, of weather data, stock market data, power consumption, consumer behaviour data, and medical and scientific data. Watermark embedding for relational data is made possible by the fact that real data can very often tolerate a small amount of error without any significant degradation with respect to their usability. For example, when dealing with weather data, changing some daily temperatures of 1 or 2 degrees is a modification that leaves the data still usable. To date, only a few approaches to the problem of watermarking relational data have been proposed [Agrawal][Al-Haj][Sion]. These techniques, however, are not very resilient to watermark attacks. In this dissertation, a watermarking technique is presented for relational data that is highly resilient compared to these techniques. In particular, our proposed technique is resilient to tuple deletion, alteration, and insertion attacks.

1.3 ORGANIZATION OF THESIS

The thesis first introduces the concept of digital watermarking, along with its significance in copyright issues in respect to relational databases. It proceeds to elaborate the Optimization Algorithms that have been implemented in technique proposed in this dissertation to arrive at the experimental observations. The next chapter gives a detailed outline of the existing techniques related to proposed technique which is followed by detailed discussion on proposed algorithm, Chapter 7 goes on to state the experimental results and inferences of consequence. From here on, the thesis concludes the study, with a mention of the references. The chapter wise description is as follows:

Chapter 1: Introduction about digital Watermarking and Copyright Issues.

Chapter 2: Literature Survey includes references related to watermarking relational databases, optimization techniques based on Genetic algorithms and Bacterial Foraging.

Chapter 3: Watermarking in Relational Databases deals with details of embedding watermarks into relational databases discussing about basic requirements of watermarking, technical challenges faced and various attacks possible in context of relational databases.

Chapter 4: Optimization Algorithm deals with detailed discussion of two evolutionary algorithms, Genetic Algorithms and bacterial foraging. Workings of both algorithms are discussed along with their algorithms.

Chapter 5: Identification and Proof of Ownership by Watermarking Relational Databases discusses about the robust secure and imperceptible procedure of watermarking numeric attributes such that it provides ownership identification and proof of ownership. The watermarking process is divided into two phases i.e. watermark encoder and watermark decoder.

Chapter 6: Proposed Technique is discussed. The technique proposed divides watermark encoder into 4 phases i.e. watermark preparator, watermark partitioner, watermark embedder and threshold evaluator. Each of these phases are explained in this chapter. Similarly, watermark decoder is divided into 4 phases i.e. watermark preparator, watermark partitioner, threshold decoder and majority voter. The working of these phases are discussed.

Chapter 7: Experiments and Analysis goes on to state the experimental results and inferences of consequence. Experiments are conducted that shows proposed algorithm is robust against various attacks.

Chapter 8: In Conclusion, the thesis concludes the study along with suggestions for future work.

CHAPTER 2
LITERATURE SURVEY

2.1 INTRODUCTION

Relational databases watermarking is used to protect the intellectual or property in today internet-based application environments and in many content distribution applications. Much research has done on the watermarking relational database. Many algorithms are proposed that watermark both numeric and non-numeric attributes using various techniques. However, very few works focus on the optimization of watermark signal of relational databases.

2.2 RELATED WORK

The main application of watermarking is copyright protection, proof of ownership and ownership identification. Zhi-Hao Zhang *et al* [Zhang] proposed image-based novel watermarking method for the numeric data. In their method an identification image is embedded into relational data for representing copyright information. Several other image-based watermarking mechanisms [Haj][Sun][Hu][Odeh] are proposed in literature for watermarking numeric and non-numeric attributes.

However [Deshpande][Bhattacharya] proposed different mechanism for watermarking relational databases based on partitioning the databases and then embedding watermarks into them. Chuanxian Jiang *et al* [Jiang] proposed the watermarking algorithm, which can embed the watermark into relational database in DWT domain**.** Damien Hanyurwimfura *et al* [Hanyurwimfura] watermarks non-numeric multi words data based on lavenshtein distance. Haiting Cui *et al* [Cui] proposed a public key cryptography based algorithm for watermarking relational databases.

In general [Hanyurwimfura], any watermarking scheme (algorithm) needs three components such as original data, secret key and watermark as shown in Fig.2.1[Hanyurwimfura]

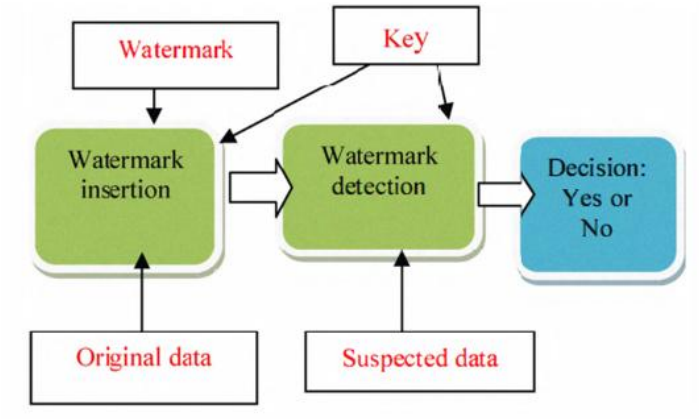

Fig. 2.1 Block Diagram of digital Watermarking Scheme

The watermarking algorithm for relational databases proposed in [Agrawal] assume that database relations can be watermarked in some attributes, such that changes in few values do not affect their applications. This algorithm embeds watermarks only in one attribute out of several candidates attributes in a tuple. This technique ensures that some bit positions of some of the attributes of some of the tuples contain specific values. The specific bit locations and values are algorithmically determined under the control of a secret key known only to the owner of the data. This bit pattern constitutes the watermark. Only if one has access to the secret key can the watermark be detected with high probability.

However, Vidhi Khanduja and O.P.Verma [Khanduja] has proposed new robust secure and imperceptible embedding mechanism that securely and randomly select any number of attributes out of selected candidate attributes for embedding watermarks in varying number of least significant bits and resolves the two important concerns namely; owner identification and proof of ownership. This algorithm embeds the identity of a work's copyright holder as a watermark and this watermark can be used to provide evidence in ownership disputes.

Mohamed Shehab *et al* [Shehab] formulate the watermarking of relational databases as a constrained optimization problem and discuss efficient techniques to solve the optimization problem and to handle the constraints. This watermarking technique is resilient to watermark synchronization errors because it uses a partitioning approach that does not require marker tuples. Genetic Algorithm and Pattern Search techniques were employed to solve the proposed optimization problem and to handle the constraints. Genetic Algorithms are used widely to solve optimization problems [Mathew][Goldberg]. A genetic algorithm (GA) is a search technique that is based on the principles of natural selection or survival of the fittest. Genetic algorithms are one of the best ways to solve a problem for which little is known objective function directly in the search. It searches the referred to solution space by

maintaining a population of potential solutions. Then, by using evolving operations such as crossover, mutation, and selection, the GA creates successive generations of solutions that evolve and inherit the positive characteristics of their parents and thus gradually approach optimal or near optimal solutions. By using the objective function directly in the search, GA's can be effectively applied in nonconvex, highly nonlinear, complex problems.

Bacterial Foraging Optimization (BFO) is a recently developed nature-inspired optimization algorithm, which is based on the foraging behaviour of *E. coli* bacteria. Up to now, BFO has been applied successfully to some engineering problems due to its simplicity and ease of implementation [Passino][Chen][Das]. [Passino] gives brief discussion on the potential uses of biomimicry of social foraging to develop adaptive controllers and cooperative control strategies for autonomous vehicles. Several BFO variants have been developed to improve its optimization performance. [Tripathy], proposed an improved BFO algorithm using two approaches: (1) in order to speed up the convergence, the average value is replaced by the minimum value of all the chemotactic cost functions for deciding the bacterium's health; (2) for swarming, the distances of all the bacteria in a new chemotactic step are evaluated from the globally optimal bacterium to these points and not the distances of each bacterium from the rest of the others. [Mishra] proposed a fuzzy bacterial foraging (FBF) algorithm using Takagi-Sugeno-type fuzzy inference scheme to select the optimal chemotactic step size in BFO.

2.3 CONCLUSION

The [Agrawal] has proposed novel watermarking technique for numeric data which is referenced widely in this area and forms the basics of work presented in this thesis. Further, [Shehab] has presented the watermarking problem as a constraint optimization problem and proposed GA and PS to solve the problem. Work presented in this dissertation solves the problem of watermarking relational databases using Bacterial Foraging. Bacterial Foraging technique is described in [Passino] which is followed.

CHAPTER 3
WATERMARKING IN RELATIONAL DATABASES

3.1 INTRODUCTION

Digital images, video and audio are examples of digital assets which have become easily accessible by ordinary people around the world. However, the owners of such digital assets[Haj] have long been concerned with the copyright of their digital products, since copying and distributing digital assets across the Internet was never easier and possible as its nowadays. Therefore, researchers have been looking for ways to protect the ownership of digital assets for a long time.

Digital watermarking technology was suggested lately as an effective solution for protecting the copyright of digital assets. This technology provides ownership verification of a digital product by inserting imperceptive information into the digital product. Such 'right witness' information is called the watermark and it is inserted in such a way that the usefulness of the product remains, in addition to providing it with robustness against attempts to remove the watermark.

Most watermarking research concentrated on watermarking multimedia data objects such as still images and video and audio. However, watermarking of database systems started to receive attention because of the increasing use of database systems in many real-life applications. Examples where database watermarking might be of a crucial importance include protecting rights of outsourced relational databases and allowing the creation of secured and copyright-protected web-based services that enable users to search and access databases remotely.

With the development of Internet and databases application techniques, the demand that lots of databases in the Internet are permitted to remote query and access for authorized users has become common. But in this case the copyright of the data may not be protected effectively, the data providers are worried about the data being burgled, illegal copy etc. Embedding digital watermark into the relational databases solves these problems by indicating the invasion and piracy of the databases.

3.2 WATERMARKING LIFE-CYCLE PHASES

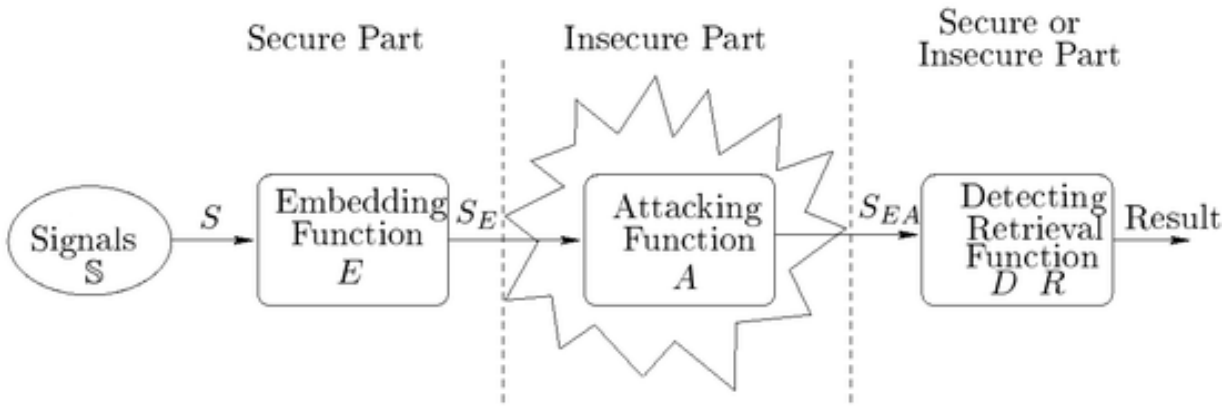

Fig. 3.1 Watermarking Life-cycle

General watermark life-cycle phases as shown in fig 3.1 [wiki] with embedding-, attacking- and detection/retrieval functions [wiki]. The information to be embedded is called a digital watermark, although in some contexts the phrase digital watermark means the difference between the watermarked signal and the cover signal. The signal where the watermark is to be embedded is called the host signal. A watermarking system is usually divided into three distinct steps, embedding, attack and detection.

In embedding, an algorithm accepts the host and the data to be embedded and produces a watermarked signal. The watermarked signal is then transmitted or stored, usually transmitted to another person. If this person makes a modification, this is called an attack. While the modification may not be malicious, the term attack arises from copyright protection application, where pirates attempt to remove the digital watermark through modification. There are many possible modifications, for example, lossy compression of the data, cropping an image or video or intentionally adding noise.

Detection (often called extraction) is an algorithm which is applied to the attacked signal to attempt to extract the watermark from it. If the signal was unmodified during transmission, then the watermark is still present and it can be extracted.

In robust watermarking applications, the extraction algorithm should be able to correctly produce the watermark, even if the modifications were strong. In fragile watermarking, the extraction algorithm should fail if any change is made to the signal.

Applications of watermarking are following:

i. *Broadcast monitoring*: Identifying when and where works are broadcast by recognizing watermarks embedded in them.
ii. *Ownership Identification*: Embedding the identity of a work's copyright holder as a watermark.
iii. *Proof of ownership*: Using watermarks to provide evidence in ownership disputes.
iv. *Transaction tracking*: Using watermarks to identify people who obtain content legally but illegally redistribute it.
v. *Content authentication*: Embedding signature information in content that can be checked to verify it has not been tampered with.

3.3 COPYRIGHT ISSUES

Digital watermarking technology was suggested as an effective solution for protecting the copyright of digital assets [Gupta].

- Copyright is a set of exclusive rights granted to the author or creator of an original work, including the right to copy, distribute and adapt the work. Copyright does not protect ideas, only their expression.
- In most jurisdictions copyright arises upon fixation and does not need to be registered. Copyright owners have the exclusive statutory right to exercise control over copying and other exploitation of the works for a specific period of time, after which the work is said to enter the public domain [wiki copyright].
- Uses covered under limitations and exceptions to copyright, such as fair use, do not require permission from the copyright owner. All other uses require permission.
- Copyright owners can license or permanently transfer or assign their exclusive rights to others.

Database Copyright

The salient aspects concerning database protection under copyright are:-

- Under the copyright laws, databases are protected as collections or compilations of literary and artistic works. The essential requirement is that a database should be

the result of its creator's own intellectual effort and that it achieves a sufficient level of originality.

- The intellectual skill involved in copyright protection is the conceptual approach to classification and data organisation, which facilitates quick retrieval and various analyses of the data.

Importance of Database Watermarking as a part of Database Copyright

- Although watermarking does not prevent illegal copying, it deters such copying by providing a means for establishing the original ownership of a redistributed copy
- The Internet is exerting tremendous pressure on data providers to create services that allow users to search and access databases remotely. Although this trend is a boon to end users, it exposes the data providers to the threat of data theft. Providers are therefore demanding technology for identifying pirated copies of their databases.
- Also People pay much attention to the technology of data mining recently and more and more research institutions begin to buy the databases to analyze.
- If it doesn't concern customer's secrets the enterprises would also like to sell their data warehouse to do the research. Therefore, it becomes an important subject to prove the integrity of the database.
- The concept of digital watermarking is from Information Hiding. If people argued the copyright of protected information, we can extract the embedded watermarks to prove the copyright.
- Digital watermarking technique mainly applies to copyright protected and integrity of information content authenticated. In the copyright protected, the watermarks must have robustness. Having robustness is even if the data is altered maliciously and it could be extracted watermarks back easily.
- In the integrity of information content authenticated, the watermarks have to make sure whether the data is attacked. In the past, digital watermarking technique is widely used on image process. At present, it used on databases because the markets of databases is rising.

3.4 RELATIONAL DATABASES

A *relation* is defined as a set of tuples that have the same attributes. A tuple usually represents an object and information about that object. Objects are typically physical objects

or concepts. A relation is usually described as a table, which is organized into rows and columns. All the data referenced by an attribute are in the same domain and conform to the same constraints.

The relational model specifies that the tuples of a relation have no specific order and that the tuples, in turn, impose no order on the attributes. Applications access data by specifying queries, which use operations such as *select* to identify tuples, *project* to identify attributes, and *join* to combine relations. Relations can be modified using the *insert*, *delete*, and *update* operators. New tuples can supply explicit values or be derived from a query. Similarly, queries identify tuples for updating or deleting. It is necessary for each tuple of a relation to be uniquely identifiable by some combination (one or more) of its attribute values. This combination is referred to as the primary key.

As relational database consists of tuples, each of which represents a separate object. The watermark needs to be spread over these separate objects. The tuples of a relation constitute a set, and there is no implied ordering between them. Insertions, deletions, and updates can be done easily on data stored. Due to the different characteristics between images or audio and relational data, there exists no image or audio watermarking method suitable for watermarking relational databases. Therefore, relational database watermarking is, in fact, a process challenged by many factors such as data redundancy fewness, relational data out-of-order and frequent updating.

Moreover, database systems watermarking have unique and sometimes complex, requirements that differ from those required for watermarking digital audio-visual products. Due to such unique requirements and challenges, literature on watermarking relational databases is very limited and has focused mainly on embedding short strings of binary bits in randomly selected locations in numerical databases. Most proposed algorithms lack robustness against bit-level attacks such as bit-setting, bit resetting and bit-flipping. There are many application contexts for which data represent an important asset, ownership of which must be carefully enforced.

3.5 TECHNICAL CHALLENGES OF DATABASE WATERMARKING

There are many differences between the structures of multimedia data and relational databases. Therefore, the watermarking process on relational database is challenged [Zhang] by the following factors:

i. **Few redundant data**: A relational database is made up of tuples, each indicating an independent object. Therefore, watermarks basically have no places to hide whereas multimedia object consists of a large number of bits with considerable redundancy. Thus, the watermark has a large cover in which to hide.

ii. **Out-of-order relational data**: Tuples of a relational database have no fixed location. This makes building a corresponding relative is very difficult in relational databases. However relative spatial/temporal positioning of various pieces of a multimedia object typically does not change.

iii. **Frequent updating**: Insertion, dropping, updating of operation of relational database is very frequent. Without malicious intention, users often casually drop some tuples or attributes. On the other hand, the pirate can add or substitute the tuples and attributes whereas, multimedia objects typically remain intact; portions of an object cannot be dropped or replaced arbitrarily without causing perceptual changes in the object.

Because of these differences, techniques developed for multimedia data cannot be directly used for watermarking relations. Work presented in this dissertation meets all the above challenges to watermark relational databases.

3.6 REQUIREMENTS OF DATABASE WATERMARKING

Watermarking database systems has unique requirements that differ from those required for watermarking digital image and audio systems. The watermarked database must maintain the following properties [Agrawal]:

i. **Usability**: The amount of change in the database caused by the watermarking process should not result in degrading the database and making it useless. The amount of allowable change differs from one database to another, depending on the nature of stored records.

ii. **Robustness**: Watermarks embedded in the database should be robust against attacks to erase them. That is, the database watermarking algorithm must be developed in such a way to make it difficult for an adversary to remove or alter the watermark beyond detection without destroying usability of the database.

iii. **Blindness**: Watermark extraction should neither require the knowledge of the original un-watermarked database nor the watermark itself. This property is critical as it allows the

watermark to be detected in a copy of the database relation, irrespective of later updates to the original relation.

iv. **Structure**: A database is made of inter-related tuples. The tuples that are joined before the watermarking process should not be altered during watermarking. Moreover, scale and classification must be considered during the watermarking process since they have impact on the semantics of the database.

v. **Security**: Choice of the watermarked tuples, attributes, and bit positions should be secret and be only known through the knowledge of a secret-key. Owner of the database should be the only one who has knowledge of a secret-key. A watermark is secure if knowing the algorithms for embedding and extracting does not help unauthorised party to detect or remove the watermark.

3.7 TYPES OF ATTACKS ON WATERMARKED DATA

There are many ways in which a watermark can potentially be damaged, erased or compromised. A watermarking system must be resistant to both intentional and unintentional assaults, while not hindering ordinary data-processing operations.

Some common attacks [Hanyurwimfura] in databases are following:

i. Subset deletion attack: In this type of attack, the attacker may take a subset of the tuples of the watermarked database and hope that the watermark will be removed.

ii. Subset addition attack: In this type of attack, the attacker adds a set of tuples to the original database. This is one of the most difficult attacks to defeat. The attacker may add some tuples to the watermarked table.

iii. Subset alteration attack: In this type of attack, the attacker alters the tuples of the database through operations such as linear transformation. The attacker hopes by doing so to erase the watermark from the database.

iv. Subset selection attack: In this type of attack, the attacker randomly selects and uses a subset of the original database that might still provide value for its intended purpose. The attacker hopes by doing so that the selected subset will not contain the watermark.

3.8 CONCLUSION

As an embranchment of information hiding, the digital watermark techniques have been attracting more and more interests in both research and industrial fields. As a tool for storing and managing data, relational database is widely used in many information systems. It is a crucial issue to protect the copyright of relational data. Watermarking techniques developed for multimedia data cannot be directly used for watermarking relations because of lot of differences as discussed above in. Thus, watermarking techniques for relational databases should be devised such that it should follow all the necessary properties defined above i.e. watermarking system must be blind, robust, secure and usable and system must be resistant against various possible attacks.

CHAPTER 4
OPTIMIZATION ALGORITHMS

4.1 INTRODUCTION

An optimization algorithm is an algorithm for finding a value *x* such that *f(x)* is as small (or as large) as possible, for a given function *f*, possibly with some constraints on *x*. Here, *x* can be a scalar or vector of continuous or discrete values. An algorithm terminates in a finite number of steps with a solution. In the past several decades, research on optimization has attracted more attention. The general unconstrained optimization problem can be defined as:

$$Minimize\ (X),\ X=[x1, x2, \ldots, xD] \qquad (4.1)$$

Where, *D* is the number of the parameters to be optimized.

There are different optimization methods and algorithms that can be grouped into deterministic and stochastic [Floudas][Spall]. Deterministic techniques depend on the mathematical nature of the problem. Weaknesses of this technique are dependent on gradient, local optimums, and inefficiency in large-scale search space. Stochastic techniques are considered to be more users friendly because they do not depend on the mathematical properties of a given function and are hence more appropriate for finding the global optimal solutions for any type of objective function. As many real-world optimization problems become increasingly complex, using stochastic methods is inevitable.

Nature ecosystems have always been the rich source of mechanisms for designing artificial computational systems to solve difficult engineering and computer science problems [Chen, 2011]. In the optimization domain, researchers have been inspired by biological processes to develop some effective stochastic techniques that mimic the specific structures or behaviours of certain creatures. For examples, genetic algorithms (GA), originally conceived by [Holland], represent a fairly abstract model of Darwinian evolution and biological genetics; ant colony optimization (ACO), proposed by [Dorigo, 1996][Dorigo, 1999], is developed based on the foraging behaviours of real ant colonies; particle swarm optimization (PSO), proposed by Eberchart and Kennedy[Eberchart] and Chen et al. [Chen, 2010], glean ideas from social behaviour of bird flocking and fish schooling. These algorithms have been found to perform better than the classical heuristic or gradient-based methods, especially for

optimizing the nondifferentiable, multimodal, and discrete complex functions. Currently, these nature-inspired paradigms have already come to be widely used in many areas.

4.2 GENETIC ALGORITHMS

Genetic Algorithms [Goldberg] are a family of computational models inspired by evolution. It is a global optimization method that manipulates a string of numbers in a manner similar to how chromosomes are changed in biological evolution. Genetic algorithms search by starting from an initial set of solutions or hypotheses, and generating successive "generations" of solutions. There is an initial set of population from which the most optimum solution is obtained. Genetic algorithms are basically used in maximization and minimization problems. The concept of GA is based on Darwin's theory of survival of fittest. In GA a binary string which works as a solution set and the perspective problem to work upon is used. An initial population made up of strings of numbers is chosen at random or is specified by the user. Each string of numbers is called a "chromosome" or an "Individual and each number slot are called a "gene." A set of chromosomes forms a population. Each chromosome represents a given number of traits which are the actual parameters that are being varied to optimize the "fitness function". The fitness function is a performance index that we seek to maximize.

WORKING PRINCIPLE

A GA uses the following in its evaluation [Mathew]:

- Population
- Objective function
- Fitness function
- Genetic operators

A GA uses a series of steps to reach the optimum solution. The first step is to select and initialize the population i.e. from where the solution is obtained and preceded to what is collectively known as generation. Then the Objective function for the problem is evaluated and the fitness function corresponding to that objective function is found. After that a set of genetic operators comprising of reproduction, mutation, and crossover are applied. These steps are continued until a desired criterion is reached and the optimum solution is obtained. The steps for the GA can be generalised as:

1. Select a population to work on

2. Initialize that selected population
3. Repeat these steps until a desired stopping criteria is reached
    a) We evaluate the Objective function for the problem
    b) find the fitness function corresponding to that objective function
    c) Apply a set of genetic operators comprising of reproduction, mutation, and crossover.

GA OPERATORS

There are three operators in Genetic Algorithm [Mathew]

- Reproduction
- Crossover
- Mutation

The main purpose of these operators is to create new solution vectors by selection, combination and alteration to solution vectors with favourable solutions.

The operation of the GA proceeds in steps. Beginning with the initial population, "selection" is used to choose which chromosomes should survive to form a "mating pool." Chromosomes are chosen based on how fit they are (as computed by the fitness function) relative to the other members of the population. More fit individuals end up with more copies of themselves in the mating pool so that they will more significantly affect the formation of the next Generation. Next, several operations are taken on the mating pool. First, "crossover" (which represents mating, the exchange of genetic material) occurs between parents. To perform crossover, a random spot is picked in the chromosome, and the genes after this spot are switched with the corresponding genes of the other parent. Following this, "mutation" occurs. This is where some genes are randomly changed to other values. After the crossover and mutation operations occur, the resulting strings form the next generation and the process is repeated. A termination criterion is used to specify when the GA should end (e.g., the maximum number of generations or until the fitness stops increasing).

REPRODUCTION

It makes more than one copies of better strings in the new solution. This results individuals with better encoded structures to produce copies more frequently. There are 2 types of reproduction operator:

i. Roulette wheel selection

Here the string is selected with the probability proportional to its fitness value. Hence the probability of selecting i^{th} string is

$$p_i = \frac{F_i}{\sum_{i=1}^{n} F_i} \qquad (4.2)$$

Where, F_i is the fitness value of i^{th} string

ii. Stochastic remainder function

The string is removed or copied based on their reproduction counts. First the probability is calculated (same as above), then the expected value of each string is calculated by

$$e_i = p_i * P \qquad (4.3)$$

where, P is the population size of i^{th} string

The fractional parts of e_i are treated as probabilities with which strings are selected for reproduction. For example if e_i=1.5 then the count will be 1 and another with a probability of 0.5. In the end the strings with count 0 are eliminated and non zero counts get copies equal to the value of their counts.

CROSSOVER

In this newer individuals are produced by recombining the material from two individuals of the previous generation. The two strings participating are the parent strings and the one produced is the child string. The child string may be stronger or weaker than the parent string. All strings in the population are not used for crossover in order to preserve some of the good strings. When a crossover probability say p_c, is used only 100 p_c per cent strings are used in crossover and the remaining 100(1-p_c) per cent of the strings remain in the population as they are in current population.

MUTATION

In this we randomly add new information in the existing population. This helps us to avoid in getting trapped in local optima. They operate at the bit level; when the bits are being copied from the current string to the new string, there is a probability that the bit might be mutated. This probability is called mutation probability p_m. A coin toss mechanism is employed; if the random number is less than p_m then the bit is inverted. For example the population have 3 eight bit strings :

00100111 , 00101100, 00111001

It can be seen that all have 0 in their left most bit, if true optimum requires 1 in that position then only mutation can help. The inclusion of mutation introduces probability p_m of turning 0 to 1.

4.3 BACTERIAL FORAGING

Bacterial Foraging Optimization (BFO) proposed by [Passino] is a recently developed nature-inspired optimization algorithm, which is based on the foraging behaviour of *E. coli* bacteria. For over the last five decades, optimization algorithms like Genetic Algorithms (GAs) [Mathew], Evolutionary Programming (EP), Evolutionary Strategies (ES), which draw their inspiration from evolution and natural genetics, have been dominating the realm of optimization algorithms. Recently natural swarm inspired algorithms like Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) [Dorigo, 1996] have found their way into this domain and proved their effectiveness. Following the same trend of swarm-based algorithms, [Passino] proposed the BFOA. Application of group foraging strategy of a swarm of *E.coli* bacteria in multi-optimal function optimization is the key idea of the new algorithm [Das]. Bacteria search for nutrients in a manner to maximize energy obtained per unit time. Individual bacterium also communicates with others by sending signals. A bacterium takes foraging decisions after considering two previous factors. The process, in which a bacterium moves by taking small steps while searching for nutrients, is called chemotaxis and key idea of BFOA is mimicking chemotactic movement of virtual bacteria in the problem search space.

THE E.COLI BACTERIUM

The E. coli bacterium has a plasma membrane, cell wall, and capsule that contains the cytoplasm and nucleoid [Passino]. The pili (singular, pilus) are used for a type of gene transfer to other E. coli bacteria, and flagella (singular, flagellum) are used for locomotion. The cell is about 1 μmin diameter and 2 μm in length. The E. coli cell only weighs about 1 picogram and is about 70% water. Salmonella typhimurium is a similar type of bacterium. The E. coli bacterium is probably the best understood microorganism. Its entire genome has been sequenced; it contains 4,639,221 of the A, C, G, and T "letters"—adenosine, cytosine, guanine, and thymine—arranged into a total of 4,288 genes. Mutations in E. coli occur at a rate of about 10−7 per gene, per generation, and can affect its physiological aspects (e.g.,

reproductive efficiency at different temperatures). E. coli bacteria occasionally engage in a type of "sex" called "conjugation" where small gene sequences are unidirectionally transferred from one bacterium to another via an extended pilus. When E. coli grows, it gets longer, and then divides in the middle into two "daughters." Given sufficient food and held at the temperature of the human gut (one place where they live) of 37 ° C, E. coli can synthesize and replicate everything it needs to make a copy of itself in about 20 min; hence growth of a population of bacteria is exponential with a relatively short time to double. The E. coli bacterium has a control system (guidance system) that enables it to search for food and try to avoid noxious substances. For instance, it swims away from alkaline and acidic environments and toward more neutral ones.

SWIMMING AND TUMBLING VIA FLAGELLA [PASSINO]

Locomotion is achieved via a set of relatively rigid flagella that enable the bacterium to swim via each of them rotating in the same direction at about 100-200 revolutions per second. Each flagellum is a left-handed helix configured so that as the base of the flagellum (i.e., where it is connected to the cell) rotates counter clockwise, as viewed from the free end of the flagellum looking toward the cell, it produces a force against the bacterium so it pushes the cell. If a flagellum rotates clockwise, it will pull at the cell. An E. coli bacterium can move in two different ways; it can run (swim for a period of time) or it can tumble, and it alternates between these two modes of operation its entire lifetime (i.e., it is rare that the flagella will stop rotating).

In the algorithm the bacteria undergoes chemotaxis, where they like to move towards a nutrient gradient and avoid noxious environment. Generally the bacteria move for a longer distance in a friendly environment. Figure 4.1[Passino] depicts how clockwise and counter clockwise movement of a bacterium take place in a nutrient solution.

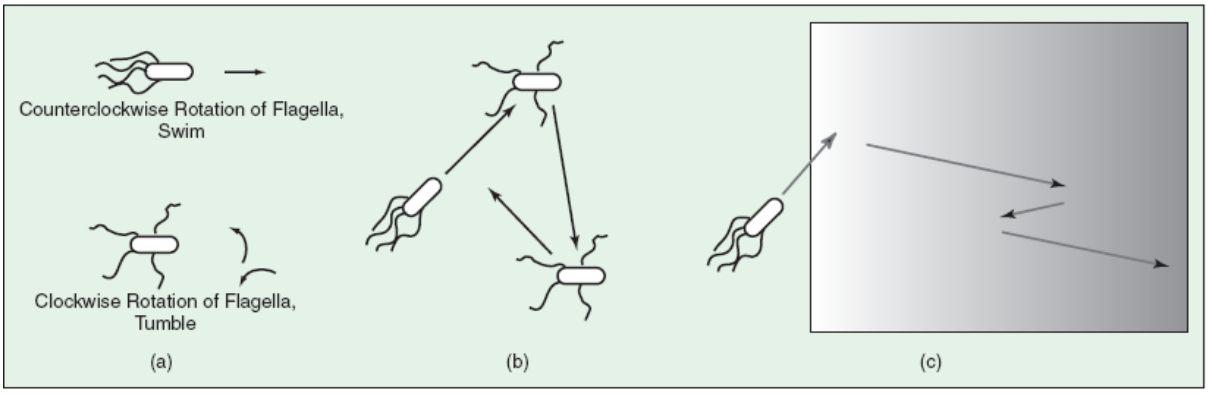

Fig. 4.1. Swimming, tumbling, and chemotactic behavior of E. coli.

When they get food in sufficient, they are increased in length and in presence of suitable temperature they break in the middle to form an exact replica of it. This phenomenon inspired Passino to introduce an event of reproduction in BFOA. Due to the occurrence of sudden environmental changes or attack, the chemotactic progress may be destroyed and a group of bacteria may move to some other places or some other may be introduced in the swarm of concern. This constitutes the event of elimination-dispersal in the real bacterial population, where all the bacteria in a region are killed or a group is dispersed into a new part of the environment.

It is interesting to note that the "decision-making" system in the E. coli bacterium must have some ability to sense a derivative, and hence it has a type of memory. Experiments have shown that it performs a type of sampling, and it remembers the concentration a moment ago, compares it with a current one, and makes decisions based on the difference.

BACTERIAL FORAGING OPTIMIZATION ALGORITHM

BFOA mimics the four principal mechanisms observed in a real bacterial system: chemotaxis, swarming, reproduction, and elimination-dispersal to solve this non-gradient optimization problem. Let us define a chemotactic step to be a tumble followed by a tumble or a tumble followed by a run.

Let j be the index for the chemotactic step. Let k be the index for the reproduction step. Let l be the index of the elimination-dispersal event.

Let p: Dimension of the search space,

S: Total number of bacteria in the population,

Nc : The number of chemotactic steps,

Ns: The swimming length.

Nre : The number of reproduction steps,

Ned : The number of elimination-dispersal events,

Ped : Elimination-dispersal probability,

C (i): The size of the step taken in the random direction specified by the tumble.

$P(j,k,l) = \{\theta^i(j,k,l) | i = 1,2,K,S\}$ represent the position of each member in the population of the *S* bacteria at the *j*-th chemotactic step, *k*-th reproduction step, and *l*-th elimination-dispersal event.

Here, let $J(i, j, k, l)$ denote the cost at the location of the *i*-th bacterium $\theta^i(j, k, l)$ (sometimes the indices are dropped and referred to the *i*-th bacterium position as θ^i). Note that *J* is interchangeably referred as being a "cost" (using terminology from optimization theory) and

as being a nutrient surface (in reference to the biological connections). For actual bacterial populations, S can be very large (e.g., S =109), but p = 3. BFOA, however, allows $p > 3$ so that the method can be applied to higher dimensional optimization problems.

Let Nc be the length of the lifetime of the bacteria as measured by the number of chemotactic steps they take during their life.

Let$C(i) > 0, i = 1,2,K,S$, denote a basic chemotactic step size that we will use to define the lengths of steps during runs.

To represent a tumble, a unit length random direction, say $\varphi(j)$, is generated; this will be used to define the direction of movement after a tumble.

In particular, we let $\theta i(j+1,k,l) = \theta i(j,k,l) + C(i)\varphi(j)$,so that$C(i)$ is the size of the step taken in the random direction specified by the tumble. If at $\theta i(j+1,k,l)$ the cost $J(i, j+1,k,l)$ is better (lower) than at $\theta i(j,k,l)$, then another step of size $C(i)$ in this same direction will be taken, and again, if that step resulted in a position with a better cost value than at the previous step, another step is taken. This swim is continued as long as it continues to reduce the cost, but only up to a maximum number of steps, Ns. This represents that the cell will tend to keep moving if it is headed in the direction of increasingly favourable environments.

The above discussion was for the case where no cell-released attractants are used to signal other cells that they should swarm together. E.coli have cell-to-cell signalling via an attractant and represented with $J_{cc}^{i}(\theta,\theta^{i}(i,j,k))$, $i = 1,2,K,S$, for the ith bacterium. Work presented in this dissertation using BFOA does not include Cell-to-cell signalling.

Bacterial Foraging Optimization Algorithm [Passino]:

p, S, Nc , Ns , Nre , Ned , ped ,and the C(i), i = 1,2,K,S parameters are initialised. If swarming is used, the parameters of the cell-to-cell attractant functions are also chosen.

Also, initial values for the θ^{i} , i = 1,2,K, S, must be chosen. Choosing these to be in areas where an optimum value is likely to exist is a good choice. Alternatively, simply randomly distribute them across the domain of the optimization problem. The algorithm that models bacterial population chemotaxis, swarming, reproduction, elimination, and dispersal is given here (initially, j = k = l = 0). For the algorithm, note that updates to the θ^{i} automatically result in updates to P. Termination test is simply specifying a maximum number of iterations.

1) Elimination-dispersal loop: l = l + 1

2) Reproduction loop: k = k + 1

3) Chemotaxis loop: j = j + 1

a) For i = 1, 2, K, S, take a chemotactic step for bacterium i as follows.

b) Compute J (i, j, k, l). (i.e., add on the cell-to-cell attractant effect to the nutrient concentration). Let,

$$J(i, j, k, l) = J(i, j, k, l) + J_{cc}(\theta^i(j, k, l), P(j, k, l)) \quad (4.4)$$

c) Let J_{last} =J(i ,j, k, l) to save this value since a better cost via a run can be found.

d) Tumble: Generate a random vector $\Delta(i) \in R^p$ with each element $\Delta_m(i)$, m = 1,2,K,p, a random number on [−1,1].

e) Move: Let

$$\theta^i(j,k,l) = \theta^i(j,k,l) + C(i) \cdot \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \quad (4.5)$$

This results in a step of size C(i) in the direction of the tumble for bacterium i.

f) Compute J(i, j+1, k, l), and then Let,

$$J(i, j+1, k, l) = J(i, j+1, k, l) + J_{cc}(\theta^i(j+1, k, l), P(j+1, k, l)) \quad (4.6)$$

g) Swim (note that an approximation is used since swimming behaviour of each cell is decided as if the bacteria numbered {1,2,K,i} have moved and {i + 1,i + 2,K, S} have not; this is much simpler to simulate than simultaneous decisions about swimming and tumbling by all bacteria at the same time):

i) Let m = 0 (counter for swim length).

ii) While m <Ns (if have not climbed down too long)

• Let m = m+ 1.

• If J(i, j+1, k, l) < J_{last} (if doing better), let J_{last} =J(i, j+1, k, l) and let

$$\theta^i(j+1, k, l) = \theta^i(j+1, k, l) + C(i) \cdot \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \quad (4.7)$$

And use this θ^i(j + 1, k, l) to compute the new J(i, j + 1, k, l) as done in f).

• Else, let m =Ns . This is the end of the while statement.

h) Go to next bacterium (i + 1) if i ≠ S (i.e., go to b) to process the next bacterium).

4) If j< Nc , go to step 3. In this case, continue chemotaxis, since the life of the bacteria is not over.

5) Reproduction:

a) For the given k and l, and for each i = 1, 2, K, S, let

$$J^{i}_{healt\,h} = \sum_{j=1}^{Nc+1} J(i,j,k,l) \qquad (4.8)$$

be the health of bacterium i (a measure of how many nutrients it got over its lifetime and how successful it was at avoiding noxious substances). Sort bacteria and chemotactic parameters C(i) in order of ascending cost J_{health} (higher cost means lower health).

b) The Sr bacteria with the highest J_{health} values die and the other Sr bacteria with the best values split (and the copies that are made are placed at the same location as their parent).

6) If k<Nre, go to step 2. In this case, the numbers of specified reproduction steps have not reached, so the next generation in the chemotactic loop is started.

7) Elimination-dispersal: For i = 1,2,K, S, with probability ped , eliminate and disperse each bacterium (this keeps the number of bacteria in the population constant).

To do this, if a bacterium is eliminated, simply disperse one to a random location on the optimization domain.

8) If l< Ned, then go to step 1; otherwise end.

Below we briefly describe the four prime steps in BFOA.

i) **Chemotaxis**: This process simulates the movement of an *E.coli* cell through swimming and tumbling via flagella. Biologically an *E.coli* bacterium can move in two different ways. It can swim for a period of time in the same direction or it may tumble, and alternate between these two modes of operation for the entire lifetime.

ii) **Swarming**: Interesting group behaviour has been observed for several motile species of bacteria including *E.coli* and *S. typhimurium*, where intricate and stable spatio-temporal patterns (swarms) are formed in semisolid nutrient medium. A group of *E.coli* cells arrange themselves in a travelling ring by moving up the nutrient gradient when placed amidst a semisolid matrix with a single nutrient chemo-effecter. The cells when stimulated by a high level of *succinate*, release an attractant *aspertate*, which helps them to aggregate into groups and thus move as concentric patterns of swarms with high bacterial density.

iii) **Reproduction:** The least healthy bacteria eventually die while each of the healthier bacteria (those yielding lower value of the objective function) asexually split into two bacteria, which are then placed in the same location. This keeps the swarm size constant.

iv) **Elimination and Dispersal**: Gradual or sudden changes in the local environment where a bacterium population lives may occur due to various reasons e.g. a significant local rise of temperature may kill a group of bacteria that are currently in a region with a high concentration of nutrient gradients. Events can take place in such a fashion that all the

bacteria in a region are killed or a group is dispersed into a new location. To simulate this phenomenon in BFOA some bacteria are liquidated at random with a very small probability while the new replacements are randomly initialized over the search space.

4.4 ALGORITHMIC ANALOGIES AND DISTINGUISHED FEATURES OF BACTERIAL FORAGING AND GENETIC ALGORITHMS

There is the *algorithmic analogies* [Passino] between the fitness function and the nutrient concentration function, selection and bacterial reproduction (bacteria in the most favourable environments gain a selective advantage for reproduction), crossover and bacterial splitting (the children are at the same concentration, whereas with crossover they generally end up in a region around their parents on the fitness landscape), and mutation and elimination and dispersal. However, the algorithms are certainly not equivalent, and neither is a special case of the other. Each has its own distinguishing features. The fitness function and nutrient concentration functions are not the same (one represents likelihood of survival for given phenotypic characteristics, whereas the other represents nutrient/noxious substance concentrations or perhaps other environmental influences such as heat or light). Crossover represents mating and resulting differences in offspring, something we ignore in the bacterial foraging algorithm (we could, however, have made less than perfect copies of the bacteria to represent their splitting). Moreover, mutation represents gene mutation and the resulting phenotypical changes, not physical dispersal in a geographical area. From one perspective, note that all the typical features of genetic algorithms could augment the bacterial foraging algorithm by representing evolutionary characteristics of a forager in its environment. From another perspective, foraging algorithms can be integrated into evolutionary algorithms and thereby model some key survival activities that occur during the lifetime of the population that is evolving (i.e., foraging success can help define fitness, mating characteristics, etc.). For the bacteria studied here, foraging happens to entail hill-climbing via a type of biased random walk, and hence the foraging algorithm can be viewed as a method to integrate a type of approximate stochastic gradient search (where only an approximation to the gradient is used, not analytical gradient information) into evolutionary algorithms. Of course, standard gradient methods, quasi-Newton methods, etc., depend on the use of an explicit analytical representation of the gradient, something that is not needed by a foraging or genetic algorithm. Lack of dependence on analytical gradient information can be viewed as an

advantage (fewer assumptions) or a disadvantage (e.g., since if gradient information is available then the foraging or genetic algorithm may not exploit it properly).

4.5 CONCLUSION

An Evolutionary Algorithm uses some mechanisms inspired by biological evolution: reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the environment within which the solutions "live" (see also cost function). Evolution of the population then takes place after the repeated application of the above operators. Genetic Algorithms are widely used algorithms to solve optimization problems. The newly developed algorithm in this area is Bacterial Foraging which is based on the foraging behaviour of *E. coli* bacteria. The process, in which a bacterium moves by taking small steps while searching for nutrients, is called chemotaxis and key idea of BFOA is mimicking chemotactic movement of virtual bacteria in the problem search space.

CHAPTER 5
IDENTIFICATION AND PROOF OF OWNERSHIP BY WATERMARKING RELATIONAL DATABASES

5.1 INTRODUCTION

In [Khanduja], a new robust secure and imperceptible embedding mechanism was proposed to resolve the two important concerns namely; owner identification and proof of ownership. The steps of proposed mechanism for watermarking relational databases mainly involves encoding and decoding on numerical attribute of relational database in three phases:

1) Watermark preparator
2) Watermark position detector
3) Watermark Embedder or Detector.

The first phase resolves ownership identification issue as owner's identity is used to get watermark bits. In second phase position where watermarks are to be embedded are identified using secret key and pseudorandom generators. This phase marks multiple attributes with varying number of candidate bit positions within a single tuple. In the third phase watermarks are embedded in Encoder. While decoder extracts watermarks and detects database piracy.

5.2 PROPOSED ALGORITHM

Proposed watermarking system consists of two subsystems watermark encoder and respective decoder [Khanduja].

5.2.1 WATERMARK ENCODER

It embeds desired watermarks into relational database. This task is achieved using three steps as shown in Fig.5.1. Inputs fed to first block are Secret key K1 and text to be watermarked W. Using these data, watermark bits (Wp) are prepared which are used in third block i.e. Watermark Embedder. Inputs to second block are Relational database R and secret key K2 which help in identifying various watermarking positions (A_i and j) within database. In this step, bit positions where watermark bits Wp embedded are identified.

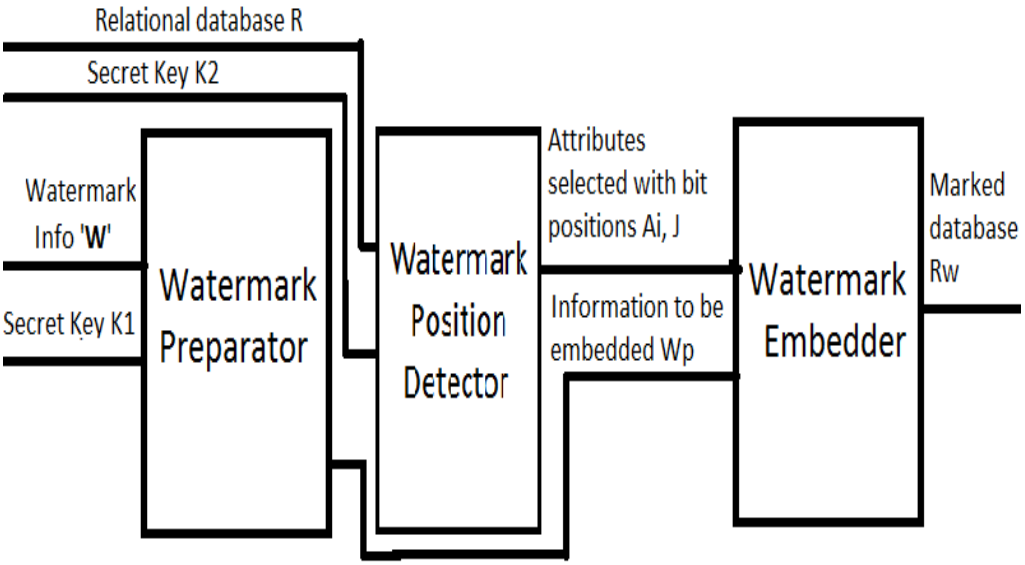

Fig. 5.1. Watermark Encoder

*A. **Watermark Preparator***

Watermark to be inserted is selected by owner of the database. The watermark must be chosen such that it reflects owner's identity. This step identifies the identity of database's copyright holder as watermark. Thus ensures owners' identification.

Owner selects the watermarking text 'W' and secret key 'K1' to create a watermark to be embedded.

The algorithm

1. Input the values of 'W' and K1
2. For each character C_i in W do
3. $W_b[i]=C_i + K1$

   [End of for loop]
4. W_P=binary (W_b) // binary (W_b) function converts number to binary.

Line 2 in the algorithm indicates that owner chosen text is read character by character and addition of each character with secret key is computed in line 3 to give integer value. These values are stored in W_b array. At line 4, binary of W_b is taken and finally stored in W_p array.

B. ***Watermarking position detector***

Suppose R is relation whose scheme is R (P, A_0, A1,....,A_{n-1}) where P is primary key attribute and R contains total *n* attributes. Let owner selects '*v*' number of numeric attributes that are candidates for marking. Each attribute A_i is numeric with values such that small changes in LB_{Ai} least significant bits are imperceptible. We consider that each attribute has varying number of candidate bit positions i.e. $LB_{Ai.}$ The gap γ [Agrawal] is a control parameter that determines the number *w* of tuples marked out of total *r* tuples via approximate relationship $w = r / \gamma$. The t.X represents the value of attribute X in tuple t ε R.

In this algorithm cryptographic pseudorandom sequence generators (CPSG) [Schneier] are used that generates computationally infeasible sequence of numbers which depends on initial seed.

Cryptographic pseudorandom sequences A cryptographically secure pseudorandom sequence generator *G* deterministically generates a sequence of numbers in which it is computationally infeasible to predict the next number in the sequence. Statistically, the numbers generated by *G* appear to be a realized sequence of independent and identically distributed random variables, in the sense that the numbers pass standard statistical tests for these properties. The values in the sequence are determined by the value of an initial seed. Given a fixed seed value, repeated executions of *G* generate the same fixed sequence of numbers every time. Here we have used BBS pseudo random number generator.

Table5.1 Notations Used

N	Number of attributes in the relation.
V	Attribute numbers to be marked
Γ	Fractions of tuples to be marked.
R	Number of tuples in the relation.
W	Number of tuples actually marked
K1, K2	Secret key selected by owner of database
W	Watermarking text selected by owner of database
W_P	Watermarking bits to be inserted
A	Significance level of test for detecting a watermark
T	Threshold parameter for detecting a watermark

The following functions are used in the algorithm

a) MAC: For each tuple 't' in relation R, secure Message Authentication Code[Sion] is computed using secret key K2 known only to owner of database and tuple's primary key t.P.

b) next(CPSG1*):* This generates next number in random sequence using CPSG1.

c) Selectattr(next(CPSG1)): An another pseudorandom sequence generator CPSG2 is created with initial seed as next(CPSG1) whose output is a vector with number of states equivalent to *v*. These states decide what all attributes in a tuple are selected for watermark. Since output of this depends on previous pseudorandom generator, this increases the level of security.
For erasing a watermark, the attacker needs to correctly guess the tuples that are marked and the selected attributes with their corresponding selected bit positions.

The algorithm

1. Input the value of secret key K2.
2. For each tuple t ϵ R do
3. Compute MAC = H(K2|| t.P || K2), where, H( ) is secure hash function and ' ||' is concatenation operator.
4. Seed CPSG1 with MAC of each tuple.
5. If (next(CPSG1) mod γ equals 0) then //mark the tuple
6. Attrindc[ ]= selectattr(next(CPSG1))
7. For each value in Attrindc[ ]
8. If (Attrindc[i] equals 1) // mark the attribute
9. Select A_i for marking
10. Bitindex j=next(CPSG1) mod LB_{Ai} // mark corresponding bit position

[end of if of line 8]
[end of for loop of line 7]
[end of if of line 5]
[end of for loop of line 2]

C. ***Embed Watermark***

For selected attribute A_i and corresponding selected bit position j, we embed watermark generated W_p in relational database R. If number of watermark bits in W_p are less then number of detected watermarked positions in step2 we repeat the watermark bits in W_p again.

5.2.2. WATERMARK DECODER

Fig. 5.2. shows watermark decoder which detects whether the database is pirated or not.

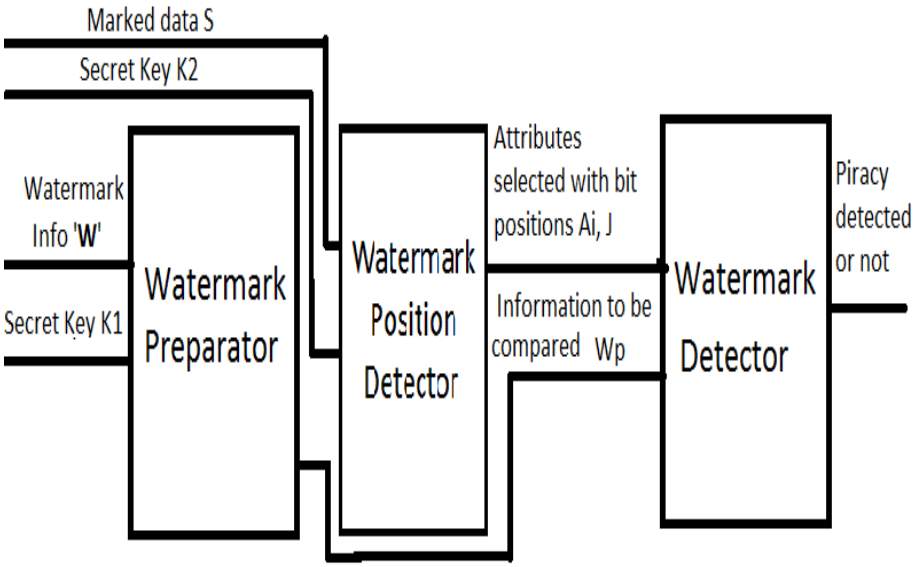

Fig. 5.2. Watermark Decoder

In detection process, the first two steps of watermark insertion are followed. Once attribute indices and bit positions are found we test whether or not the bits value matches the values that should have been assigned by insertion algorithm and count the number of matches matchcnt(m) against total number of watermarks totalcount(w). If there are very many matches or very few matches we suspect piracy [Agrawal]. We fix small value $\alpha \in (0, 1)$ and sets

$$\tau = max\{ t\epsilon \left[0, \frac{w}{2}\right] : \sum_{i=t}^{w-t} b\left(i; w, \frac{1}{2}\right) \geq 1 - \alpha\} \quad (5.1)$$

Where

$$b(i; n, p) = nkp^i(1-p)^{n-i} \quad (5.2)$$

We suspects piracy if either $m < \tau$ or $m > w - \tau$, as probability of so few or so many matches under null hypothesis is less than or equal to α. α is called significance level of the test.

***Functions used in watermark detector algorithm*:**

a) **match(s.A_i,j)**: This function test whether or not the bit value of attribute s.A_i at position j matches the values that is assigned by embedding algorithm i.e W_p and returns 1 if match found.

b) **threshold (totalcount,α)**: This function calculates threshold value τ using (1).

The algorithm

//*Watermark Preparation*

1. Input the values of watermark information 'W' and secret key K1
2. For each character C_i in W do
3. $W_b[i]=C_i + K1$

   [end of for loop]
4. W_P=binary (W_b) // binary(W_b) function converts number to binary.

//*Watermark Position Detection*

5. Input the value of secret key K2.
6. Totalcount=matchcnt=0
7. For each tuple t ϵ S do
8. Compute MAC = H(K2|| t.P || K2) where, H( ) is secure hash function and ' ||' is concatenation operator.
9. Seed CPSG1 with MAC of each tuple.
10. If (next(CPSG1) mod γ equals 0) then //mark the tuple
11. Attrindc[ ]= selectattr(next(CPSG1))
12. For each value in Attrindc[ ]
13. If (Attrindc[i] equals 1) // mark the attribute
14. Select A_i for marking
15. Bitindex j=next(CPSG1) mod LB_{Ai} // mark corresponding bit position
16. totalcount=totalcount+1

// Watermark Detector

17. matchcnt=matchcnt+match(s.A_i,j)
18. τ = threshold(totalcount, α)
19. If ((matchcnt< τ) or (matchcnt>totalcount-τ))

    then
20. Suspect piracy

    [end of if at line 19]

    [end of if at line 13]

[end of for loop at line 12]

[end of if of line 3 at line 10]

[end of for loop of line 7]

Detecting watermark is blind technique as it does not require original database and watermarks can be detected even in small subset of watermark relations as long as sample contains some of the marks

For ownership identity, the watermark bits are extracted from database S and reverse of the watermark preparation algorithm is followed to get watermarking text W` from which W is extracted.

5.3 CONCLUSION

Owner identification and proof of ownership issues are resolved in [Khanduja]. This paper proposes a secure robust and imperceptible algorithm. Embedding algorithm is divided into three phases: Watermark preparator, watermark position detector and watermark embedder. The ownership identification issue is resolved by embedding owner's identity as watermark in Preparator phase. Position detector phase securely identifies multiple attributes with varying number of candidate bit positions of the single table. Embedder inserts watermarks at identified bit positions of multiple attributes of relational database.

CHAPTER 6

PROPOSED METHOD

6.1 INTRODUCTION

A data set D is transformed into a watermarked version D_W by applying a watermark encoding function that also takes as inputs secret key Ks only known to the copyright owner and a watermark W. Watermarking modifies the data. However, these modifications are controlled by providing usability constraints referred to by the set G. These constraints limit the amount alterations that can be performed on the data, such constraints will be discussed in detail in the following sections. Technique proposed by [Shehab] is modified and implemented in proposed algorithm. In the proposed technique both GA and BFOA are implemented to optimize the result.

Proposed watermarking system consists of two subsystems watermark encoder and respective decoder.

6.2 WATERMARK ENCODER

It embeds desired watermarks into relational database. This task is achieved using four steps as shown in Fig 6.1

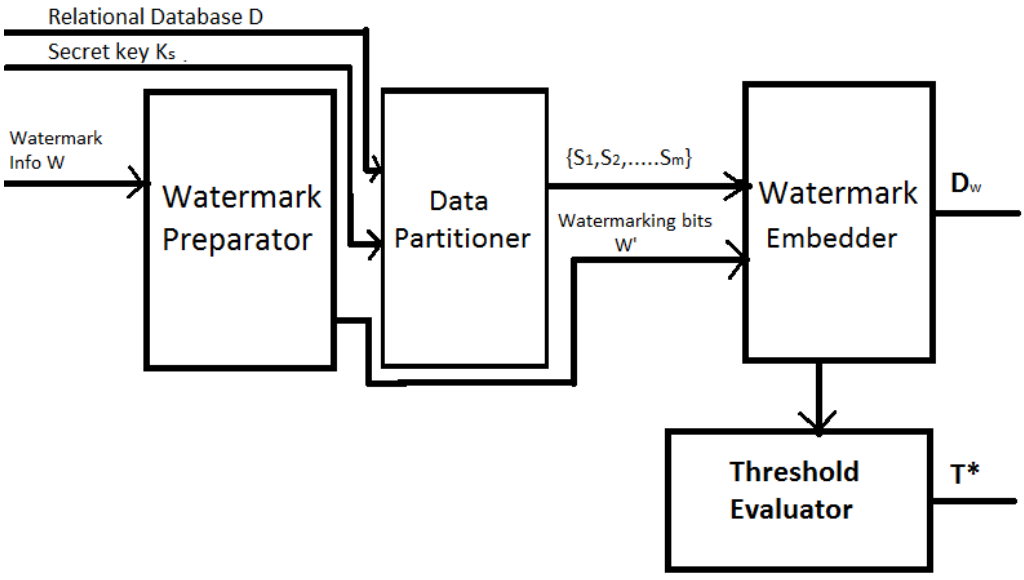

Fig. 6.1. Watermark Encoder

6.2.1 WATERMARK PREPARATOR

Watermark to be inserted is selected by owner of the database. The watermark must be chosen such that it reflects owner's identity. This step identifies the identity of database's copyright holder as watermark. Thus ensures owners' identification.

Owner selects the watermarking text W to create a watermark to be embedded (W') by applying hash function. A cryptographic hash function is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that an accidental or intentional change to the data will change the hash value. The hash values are called the message digest or simply digest. One of the most important properties of hash function is that it is infeasible to find a message that has a given hash.

Message-Digest 5 algorithm is a used as a cryptographic hash function with a 128-bit (16-byte) hash value. Specified in RFC 1321, MD5 has been employed in a wide variety of security applications, and is also commonly used to check the integrity of files. MD5 processes a variable-length message into a fixed-length output of 128 bits. The input message is broken up into chunks of 512-bit blocks (sixteen 32-bit little endian integers); the message is padded so that its length is divisible by 512. The padding works as follows: first a single bit, 1, is appended to the end of the message. This is followed by as many zeros as are required to bring the length of the message up to 64 bits less than a multiple of 512. The remaining bits are filled up with a 64-bit integer representing the length of the original message, in bits.

The main MD5 algorithm operates on a 128-bit state, divided into four 32-bit words, denoted *A*, *B*, *C* and *D*. These are initialized to certain fixed constants. The main algorithm then operates on each 512-bit message block in turn, each block modifying the state. The processing of a message block consists of four similar stages, termed *rounds*; each round is composed of 16 similar operations based on a non-linear function *F*, modular addition, and left rotation. Figure6.2. [wiki] illustrates one operation within a round. There are four possible functions *F*; a different one is used in each round:

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

$\oplus, \wedge, \vee, \neg$ denote the XOR, AND, OR and NOT operations respectively.

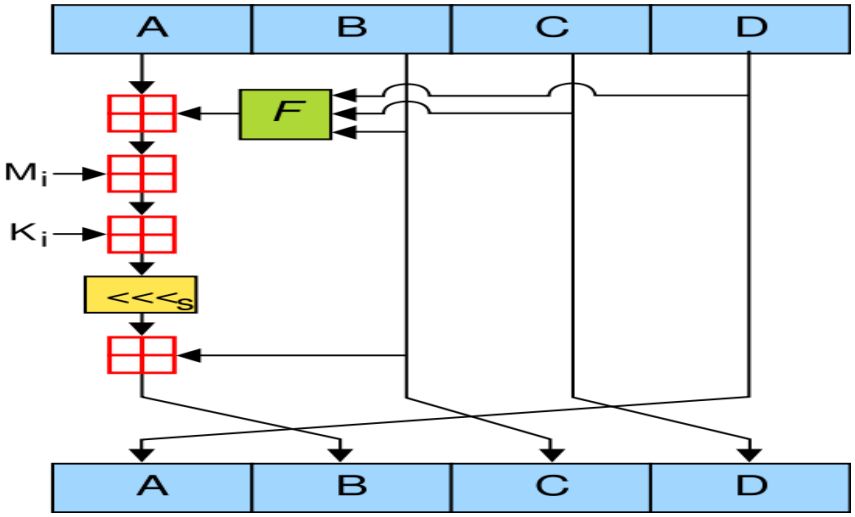

Fig. 6.2 MD5

6.2.2 DATA PARTITIONER

In this step, the data set D is partitioned into m non-overlapping partitions $\{S_0; \ldots; S_m\}$ using the secret key Ks as in [Shehab].

Algorithm name: get-partitions

Input: Data Set D, Secret key K_s, number of partitions m

Output:- Data partitions S_0, ..., S_{m-1}

1. $S_0,\ldots\ldots,S_{m-1} \leftarrow \{\}$
2. For each tuple r ϵ D
3. Partition(r) $\leftarrow H(K_s \| H(r.P \| K_s)) \bmod m$
4. Insert r into $S_{partition(r)}$
5. Return $S_0,\ldots\ldots,S_{m-1}$

The data partitioning algorithm partitions the data set based on a secret key Ks selected by owner of the database. The data set D is a database relation with scheme $D(P, A_0, \ldots, A_{v-1})$, where P is the primary key attribute, $A_0, \ldots, A_{v-1}$ are *v* attributes which are candidates for watermarking, and |D| is the number of tuples in relation D. The data set D is to be partitioned into m non-overlapping partitions, namely, $\{S_0, \ldots, S_{m-1}\}$, such that each partition S_i contains on the average |D|/m tuples from the data set D. At line 3, the data partitioning algorithm computes a MAC for each tuple r ϵ D, which is considered to be secure [Schneier] and is given by H(Ks||H(r.P||Ks), where , r.P is the primary key of the tuple r, H() is a secure hash function, and || is the concatenation operator.

Using the computed MAC, tuples are assigned to partitions. For a tuple r, its partition assignment is given by

$$Partition(r) = H(Ks||H(r.P||Ks)) \bmod m \quad (6.1)$$

Using the property that secure hash functions generate uniformly distributed message digests this partitioning technique, on average, places |D|/ m tuples in each partition.

A MAC algorithm, sometimes called a keyed (cryptographic) hash function, accepts as input a secret key and an arbitrary-length message to be authenticated, and outputs a MAC (sometimes known as a *tag*). The MAC value protects both a message's data integrity as well as its authenticity, by allowing verifiers (who also possess the secret key) to detect any changes to the message content.

Furthermore, an attacker cannot predict the tuples-to-partition assignment without the knowledge of the secret key Ks and the number of partitions m, which are kept secret. In case of a single attribute relation, the most significant χ bits (MSB) of the data could be used instead of the primary key [Shehab]. The use of the MSB assumes that the watermark embedding data alterations will unlikely alter the MSB χ bits. However, if too many tuples share the same MSB χ bits, this would enable the attacker to infer information about the partition distribution. The solution would be to select χ that minimizes the duplicates. Another technique, in case of a relation with multiple attributes, is to use identifying attributes instead of the primary key; for example, in medical data, the patient full name, patient address, and patient date of birth could be used.

6.2.3 WATERMARK EMBEDDER

The watermark embedding algorithm is explained by formalizing the bit encoding as a constrained optimization problem. A Genetic Algorithm or Bacteria Foraging technique can be used to efficiently solve such optimization problem. The selection of which optimization algorithm to use is decided according to the application time and processing requirements, as will be discussed further. It is assumed that the tuples in a partition S_i contain a single numeric attribute. In such a case each partition, S_i can be represented as a numeric data vector $S_i=[S_{i1},.....S_{in}]\epsilon R^n$.

Table 6.1 Notations used

Symbol	Description
m	Number of data partitions
ξ	Minimum partition size
W	Watermark bit sequence $\{b_{l-1}, \ldots, b_0\}$
l	Length of watermark bit sequence
X_{max}	Maximization embedding statistics
X_{min}	Minimization embedding statistics
S_i	Data partition, numeric data vector in R^n
$\|S_i\|, n$	Length of vector S_i
K_s	Secret Key
T^*	Optimal decoding threshold
G_i	Usability constraints
Δ_i	Manipulation vector in R^n

6.2.3.1 SINGLE BIT ENCODING ALGORITHM

Given a watermark bit b_i and a numeric data vector $S_i=[s_{i1}; \ldots ; s_{in}] \epsilon R^n$, the bit encoding algorithm maps the data vector S_i to a new data vector $S_i^W = S_i + \Delta_i$, where $\Delta_i = [\Delta_{i1}; \ldots ; \Delta_{in}] \epsilon R^n$ is referred to as the manipulation vector. The performed manipulations are bounded by the data usability constraints referred to by the set $G_i = [g_{i1}; \ldots ; g_{ip}]$. The encoding is based on optimizing encoding function referred to as the hiding function, which is defined as follows [Shehab]:

A hiding function $\theta_\gamma : R^n \rightarrow R$ (6.2)

where γ is the set of secret parameters decided by the data owner.

The set γ can be regarded as part of the secret key. Note that when the hiding function is applied to $S_i + \Delta_i$, the only variable is the manipulation vector Δ_i, whereas S_i and γ are constants. To encode bit b_i into set S_i, the bit encoding algorithm optimizes the hiding function $\theta_\gamma(S_i + \Delta_i)$. The objective of the optimization problem of maximizing or minimizing the hiding function is based on the bit b_i such that if the bit b_i is equal to 1, then the bit encoding algorithm solves the following maximization problem:

$$\max_{\Delta_i} \ \theta_\gamma(S_i + \Delta_i). \quad \text{subject to } G_i \qquad (6.3)$$

However, if the bit b_i is equal to 0, then the problem is simply changed into a minimization problem. The solution to the optimization problem generates the manipulation vector Δ_i *

at which $\theta_\gamma(S_i + \Delta_i^*)$ is optimal. The new data set S_i^W is computed as $Si + \Delta_i^*$.

At line 1 of the bit encoding algorithm, bit b_i is embedded in the partition S_i if $|S_i|$ is greater than €. The value of € represents the minimum partition size. The maximize and minimize in the bit encoding algorithm optimize the hiding function $\theta_\gamma (S_i + \Delta_i^*)$ subject to the constraints in G_i. The maximization and minimization solution statistics are recorded for each encoding step in X_{max}, X_{min}, respectively, as indicated in lines 4 and 7 of the encoding algorithm written below. These values are used by threshold evaluator to compute optimal decoding parameters.

The set of usability constraints G_i represents the bounds on the tolerated change that can be performed on the elements of S_i. These constraints describe the feasible space for the manipulation vector Δ_i for each bit encoding step. These constraints are application and data dependent.

In my problem, interval constraints are used to control the magnitude of the alteration for Δ_{ij}, that is,

$$\Delta_{ij}^{min} <= \Delta_{ij} <= \Delta_{ij}^{max} \quad (6.4)$$

The reference point is calculated as

$$\boldsymbol{ref = \mu + c * \sigma,} \quad (6.5)$$

Where, c $\epsilon(0,1)$; is a secret real number that is a part of the set γ , μ is mean of $S_i^W = Si + \Delta_i^*$ (i.e. $\mu_{(Si + \Delta i^*)}$) and σ is variance estimates of the set $S_i^W = Si + \Delta_i^*$ (i.e. $\sigma^2_{(Si + \Delta i^*)}$.)

The data points in $Si + \Delta_i$ that are above ref are referred to as the "tail" entries, as illustrated in Fig. 6.3 [Shehab].

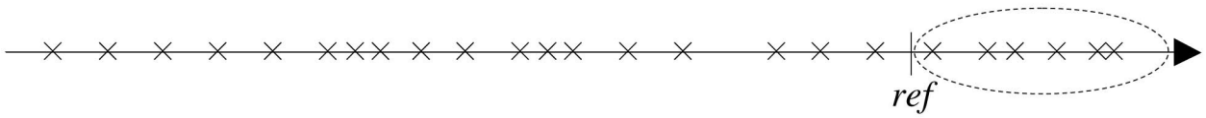

Fig6.3. The distribution of set $Si + \Delta_i^*$ on number line and tail entries circled.

The hiding function θ_c is defined as the number of tail entries normalized by the cardinality of S_i, also referred to as the normalized tail count. It is computed as follows:

$$\boldsymbol{\Theta_c(S_i + \Delta_i) = 1/n \sum_{j=1}^{n} 1 \{ Sij + \Delta ij >= ref \}} \quad (6.6)$$

Where, n is the cardinality of S_i, and $1_{\{\}}$ is the indicator function defined as follows:

$$1_{\{condition\}} = \begin{cases} 1, & if\ condition = TRUE \\ 0, & otherwise \end{cases}$$

The objective function $\Theta_c(S_i + \Delta_i)$ is nonlinear and nondifferentiable, which makes the optimization problem at hand a nonlinear constrained optimization problem. To solve this optimization problem, two techniques based on GA and BF is proposed respectively.

Algorithm:- encode_single_bit

Input:- Data Set S_i, bit b_i, constraint set G_i, secret parameters set γ,statistics X_{max}, X_{min}

Output:- data set $S_i+\Delta_i^*$

1. If ($|S_i|$<€), then return S_i
2. If (b_i==1) then,
3. Maximize ($\theta_\gamma(S_i+\Delta_i)$) subj to G_i (Call G.A or BFOA respectively)
4. Insert $\theta_\gamma(S_i+\Delta_i^*)$ into X_{max}
5. Else
6. Minimize ($\theta_\gamma(S_i+\Delta_i)$) subj to G_i (Call G.A or BFOA respectively)
7. Insert $\theta_\gamma(S_i+\Delta_i^*)$ into X_{min}
8. Return $S_i+\Delta_i^*$

To optimize the specific attribute value for all the tuples within a partition is done using Genetic Algorithm. At Line 3, Maximize ($\theta_\gamma(S_i+\Delta_i)$) and Line 6, Minimize ($\theta_\gamma(S_i+\Delta_i)$) is achieved by invoking genetic algorithm. The fitness value of function is calculated by calculating reference point for a partition first and then number of tail entries normalized by the cardinality of S_i is calculated. Similarly fitness value for every partition is calculated. In this approach the fitness function optimizes the vector consisting of number of elements in a particular partition. Thus, complexity of function increases as function has to optimize variable number of elements depending on number of elements in that partition. Fitness function changes the values of Δ_i such that ($S_i+\Delta_i$) lies within the specified range.

General Genetic Algorithm used for the implementation is as follows:

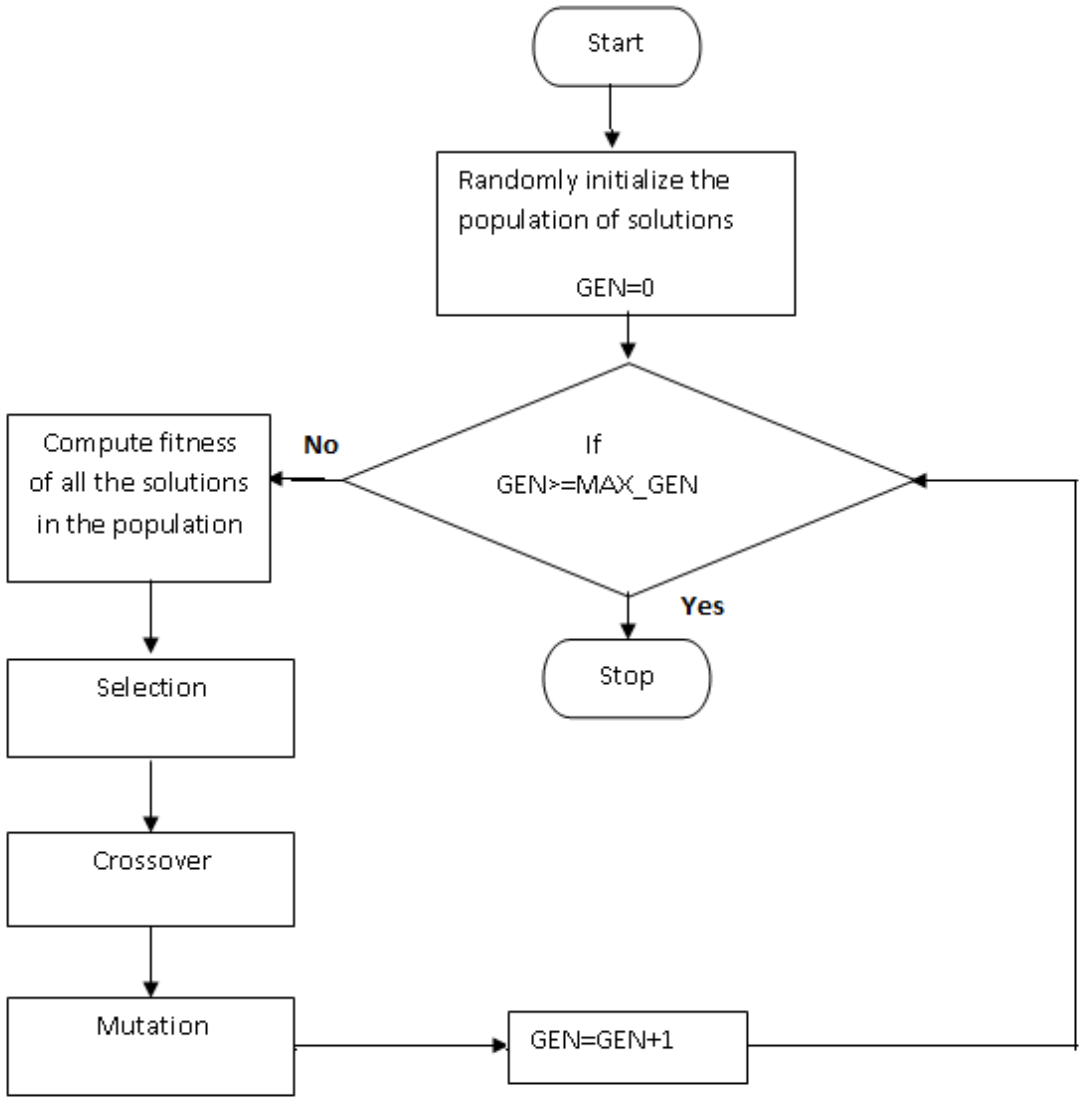

Fig. 6.4. Working Cycle of Each Generation Of Genetic Algorithm

Optimization can also be achieved using Bacterial foraging algorithm, details are already explained earlier. Objective function is calculated using same formula as for genetic algorithm. First reference point for a particular partition is calculated and then number of tail entries normalized by the cardinality of S_i is calculated to get objective function. Similarly for every partition is objective function is calculated. In this approach the objective function optimizes the vector consisting of number of elements in a particular partition. Flowchart of the steps followed in bacterial foraging for calculating optimal value of hiding function [Das]

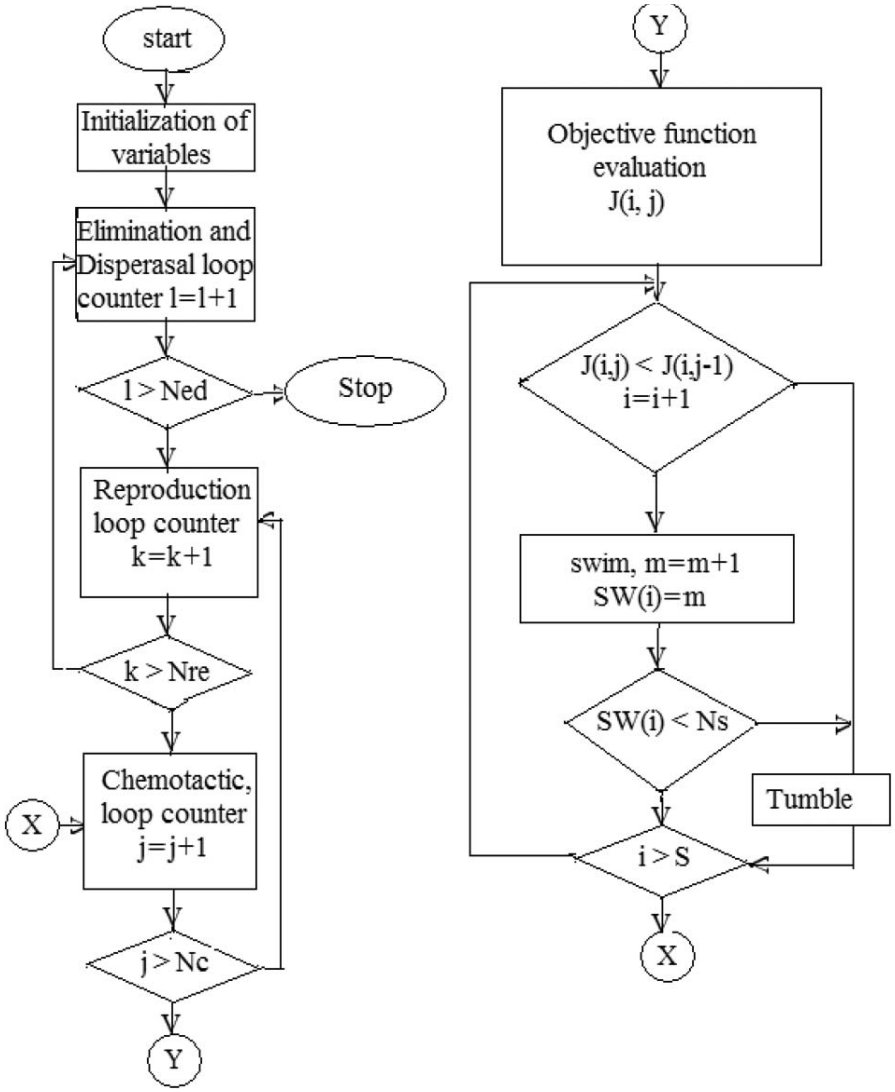

Fig. 6.5. Flowchart of Bacterial Foraging Algorithm

6.2.3.2 WATERMARK EMBEDDING ALGORITHM

Algorithm:- embed_watermark [Shehab]

Input:- Data set D, watermark W= $\{b_0,.....,b_{l-1}\}$, Secret key K_s, number of partitions m

Output:- Watermarked Data set D_w, optimal decoding threshold T^*

1. D_w, X_{max},X_{min}←{}
2. $S_0,.....,S_{m-1}$← get_partitions(D,K_s,m)
3. For each partition S_k
4. i←k mod l
5. S_k^w← encode_single_bit(b_i, S_k,c,X_{max},X_{min})
6. Insert S_k^w into D_w

7. $T^* \leftarrow$ get_optimal_threshold(X_{max}, X_{min})
8. Return D_w, T^*

A watermark is a set of l bits W = b_{l-1}; . . . ; b_0 that are to be embedded in the data partitions {S_0; . . . ; S_{m-1}}. To enable multiple embeddings of the watermark in the data set, the Watermark length l is selected such that l <<m. At line 4 of the watermark embedding algorithm bit b_i is embedded in partition S_k such that k mod l =i. This technique ensures that each watermark bit is embedded floor (m/ l) times in the data set D. The watermark embedding algorithm generates the partitions by calling get partitions at line 2, then for each partition S_k, a watermark bit b_i is encoded by using the single bit encoding algorithm (encode single bit). The generated altered partition S^W_k is inserted into watermarked data set D_W. Statistics (X_{max}, X_{min}) are collected after each bit embedding and are used by the get optimal threshold algorithm to compute the optimal decoding threshold.

6.2.4 THRESHOLD EVALUATOR

The value of the threshold T* is calculated so as to minimize the probability of bit decoding error. The probability of bit decoding error is defined as the probability of an embedded bit decoded incorrectly. The decoding threshold T* is selected such that it minimizes the probability of decoding error. The maximized hiding function values corresponding to b_0 is equal to 1 are stored in the set X_{max}. Similarly, the minimized hiding function values are stored in X_{min} Let P_{err}, P_0, and P_1 represent the probability of decoding error, the probability of encoding a bit = 0 and the probability of encoding a bit =1, respectively. Furthermore,

let b_e, b_d, and f(x) represent the encoded bit, decoded bit, and a probability density function, respectively. P_{err} is calculated as follows [Shehab]:

$$P_{err} = P(b_d = 0, b_e = 1) + P(b_d = 1, b_e = 0) \quad (6.7)$$

$$= P(b_d = 0 | b_e = 1)P_1 + P(b_d = 1 | b_e = 0)P_0 \quad (6.8)$$

$$= P(x < T | b_e = 1)P_1 + P(x > T | b_e = 0)P_0 \quad (6.9)$$

$$= P_1 \int_{-\infty}^{T} f(x|be = 1)dx + P_0 \int_{T}^{\infty} f(x|be = 0)dx \quad (6.10)$$

To minimize the probability of decoding error (P_{err}) with respect to the threshold T, we take the first order derivative of P_{err} with respect to T to locate the optimal threshold T*, as follows:

$$\frac{\partial P_{err}}{\partial T} = P1\frac{\partial}{\partial T}\int_{-\infty}^{T} f(x|be=1)dx + P0\frac{\partial}{\partial T}\int_{T}^{\infty} f(x|be=0)dx \quad (6.11)$$

$$= P_1 f(T|b_e=1) - P_0 f(T|b_e=0) \quad (6.12)$$

The distributions $f(x|b_e=0)$ and $f(x|b_e=1)$ are estimated from the statistics of the sets X_{min} and X_{max}, respectively. From our experimental observations of X_{min} and X_{max}, the distributions $f(x|b_e=0)$ and $f(x|b_c=1)$ pass the chi-square test of normality and thus can be estimated as Gaussian distributions $N(\mu_0,\sigma_0)$ and $N(\mu_1,\sigma_1)$ respectively. P_0 is estimated by $|X_{min}| / (|X_{max}|+|X_{min}|)$ and $P_1 = 1 - P_0$. Substituting the Gaussian expressions for $f(x|b_e= 0)$ and $f(x|b_e= 1)$, the first order derivative of P_{err} is as follows:

$$\frac{\partial P_{err}}{\partial T} = \frac{P_1}{\sigma_1\sqrt{2\pi}}\exp\left(-\frac{(T-\mu_1)^2}{2{\sigma_1}^2}\right) - \frac{P_0}{\sigma_0\sqrt{2\pi}}\exp\left(-\frac{(T-\mu_0)^2}{2{\sigma_0}^2}\right) \quad (6.13)$$

By equating the first order derivative of P_{err} to zero, we get the following quadratic equation, the roots of which include the optimal threshold T* that minimizes P_{err} . The second order derivative of P_{err} is evaluated at T* to ensure that the second order necessary condition $\frac{\partial^2 P_{err}(T*)}{\partial T^2} > 0$ is met.

$$\frac{\sigma_0^2-\sigma_1^2}{2\sigma_0^2\sigma_1^2}T^{*2} + \frac{\mu_0\sigma_1^2-\mu_1\sigma_0^2}{\sigma_0^2\sigma_1^2}T^* + \ln\left(\frac{P_0\sigma_1}{P_0\sigma_0}\right) + \frac{\mu_1^2\sigma_0^2-\mu_0^2\sigma_1^2}{2\sigma_0^2\sigma_1^2} = 0 \quad (6.14)$$

From the above analysis, the selection of the optimal T* is based on the collected output statistics of the watermark embedding algorithm. The optimal threshold T* minimizes the probability of decoding error and thus enhances the strength of the embedded watermark by increasing the chances of successful decoding.

6.3 WATERMARK DECODER

Watermark decoding is the process of extracting the embedded watermark using the watermarked data set D_W, the secret key Ks, and the optimal threshold T*. The decoding algorithm is blind as the original data set D is not required for the successful decoding of the embedded watermark.

The watermark decoding is divided into four main steps as shown in figure 6.6

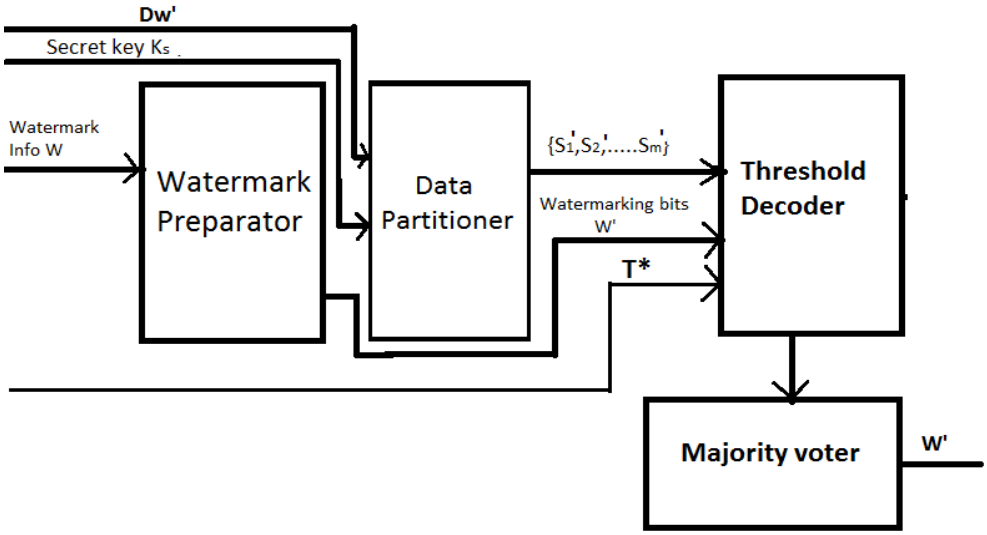

Fig. 6.6 Watermark Decoder

6.3.1 WATERMARK PREPARATOR

By following same algorithm of watermark preparatory of encoder watermarks are prepared. Watermark to be inserted is selected by owner of the database. The watermark must be chosen such that it reflects owner's identity.

Owner selects the watermarking text W to create a watermark to be embedded (W') by applying hash function.

6.3.2 DATA PARTITIONER

By using the data partitioning algorithm used in Encoder, the data partitions are generated.

Input to algorithm: Data Set D_W', Secret key K_s, number of partitions m.

Output of algorithm: Data partitions S_0', ..., S_{m-1}'.

6.3.3 THRESHOLD DECODER

The statistics of each partition are evaluated, and the embedded bit is decoded using a threshold-based scheme based on the optimal threshold T*. Presented with the data partition S^W_i , the bit decoding technique computes the hiding function ($\theta_\gamma S_i^w$) and compares it to the optimal decoding threshold T* to decode the embedded bit b_i. If ($\theta_\gamma S_i^w$) is greater than T*, then the decoded bit is 1; otherwise, the decoded bit is 0. The decoding technique computes the normalized tail count of S_i^w by computing the reference ref and by counting the number of entries in S_i^w that is greater than ref. Then, the computed normalized tail count is compared to T* see fig. 6.7[Shehab]

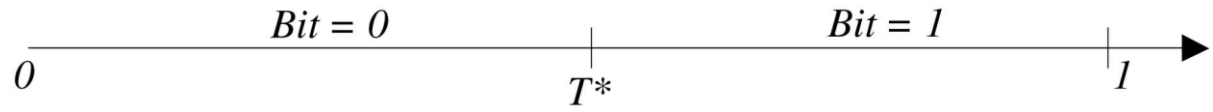

Fig.6.7. Threshold-based decoding scheme.

6.3.4 MAJORITY VOTER

The watermark bits are decoded using a majority voting technique. As the watermark W =b_{l-1}, . . ., b_0 is embedded several times in the data set, each watermark bit is extracted several times, where for a bit b_i, it is extracted from partition S_k, where k mod l= i. The extracted bits are decoded using the majority voting technique, which is used in the decoding of repetition error correcting codes. Each bit b_i is extracted m/ l times so it represents a floor (m/l)-fold repetition code [Shehab].

6.3.5 WATERMARK DETECTION ALGORITHM

Watermark detection algorithm combines all the above described steps to extract the embedded watermarking bits.

Algorithm: detect_watermark [Shehab]

Input: watermarked data set D_w, m,c,€, K_s, T^*, watermark length l

Output: detected watermark W_D

1. Set ones[0,.....,l-1]←0
2. Set zeros[0,.....,l-1]←0
3. S_0,.....,S_{m-1}← get_partitions(D_w,K_s,m)
4. For j=0 to m-1
5. If $|S_j|$>=€
6. I← j mod l
7. Value← θ(S_j, 0, c)
8. If value>=T^*
9. Ones[i]← Ones[i+1]
10. Else
11. Zeros[i]← Zeros[i+1]
12. For j=0,....., l-1
13. If ones[j]>zeros[j]
14. W_D[j]←1

15. Else If ones[j]<zeros[j]
16. $W_D[j] \leftarrow 0$
17. Else
18. $W_D[j] \leftarrow x$
19. Return W_D

6.4 CONCLUSION

Resilient watermarking technique for relational data that embeds watermark bits in the data statistics is proposed. The watermarking problem was formulated as a constrained optimization problem that maximizes or minimizes a hiding function based on the bit to be embedded. GA and BFOA techniques were employed to solve the proposed optimization problem and to handle the constraints. A data partitioning technique that does not depend on special marker tuples to locate the partitions and proved its resilience to watermark synchronization errors is proposed. I have developed an efficient threshold-based technique for watermark detection that is based on an optimal threshold that minimizes the probability of decoding error. The watermark resilience was improved by the repeated embedding of the watermark and using majority voting technique in the watermark decoding phase.

CHAPTER 7
EXPERIMENTS AND ANALYSIS

7.1 INTRODUCTION

In this section, we report the results of an extensive experimental study that analyzes the resilience of the proposed watermarking scheme to the attacks. All the experiments were performed on 2.13 GHz Intel Core i3 CPUs with 2GB of RAM. Experiment was performed to execute proposed method by using both Genetic Algorithms and Bacterial Foraging Optimization Algorithm. Various parameters of both the algorithms were optimized to get best results. A sample relational database is taken of 500 tuples and containing the record of salaries of various individual at the survey conducted by an organization. However, salary lies between a particular range and that range should not be violated while inserting watermarks. The usability constraint considered is interval constraints that are used to control the magnitude of the alteration for Δ_{ij}, that is,

$$\Delta_{ij}^{min} <= \Delta_{ij} <= \Delta_{ij}^{max}$$

Thus small changes in salary attribute are done by inserting watermarks in such a way that usability constraints are not violated.

7.2 GENETIC ALGORITHMS

Experiment is conducted by initializing c=5%, a 4-bit watermark is created using watermark preparatory algorithm; a minimum partition size of 20 and number of generation taken=1000, by using genetic algorithm was used. The optimal threshold value was computed using Matlab Genetic Algorithm Tool.

Matlab provides an optimization toolbox that includes a GA-based solver. The toolbox is started by typing ***optimtool*** in the Matlab's command line. As soon as the optimization window appears, the solver *ga – Genetic Algorithm* is selected and fitness function file is mentioned.

One of the important parameters that affects the diversity of the population (remember, it's vital to have good diversity in the population) is the Fitness Scaling (in Options). If the fitness values vary too widely, the individuals with the lowest values (for minimization) reproduce too rapidly, taking over the population pool too quickly and preventing the GA from searching other areas of the solution space. On the other hand, if the values vary only a

little, all individuals have approximately the same chance of reproduction and the search will progress very slowly. The Fitness Scaling adjusts the fitness values (scaled values) before the selection step of the GA. This is done without changing the ranking order, that is, the best individual based on the raw fitness value remains the best in the scaled rank, as well. Only the values are changed, and thus the probability of an individual to get selected for mating by the selection procedure. This prevents the GA from converging too fast which allows the algorithm to better search the solution space.

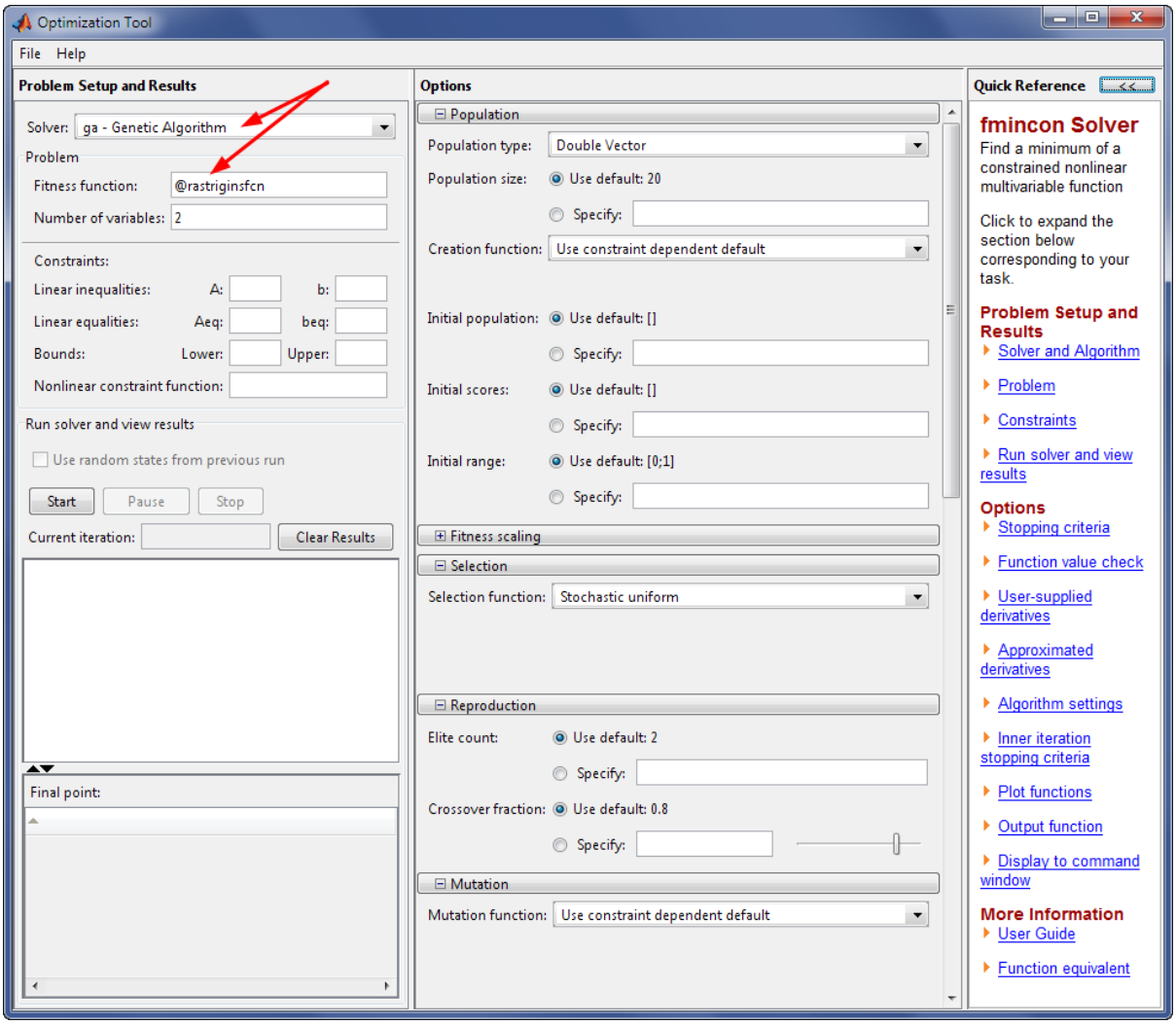

Fig.7.1. GA Toolbox in MATLAB

The following graphs show the results obtained by implementing GA.

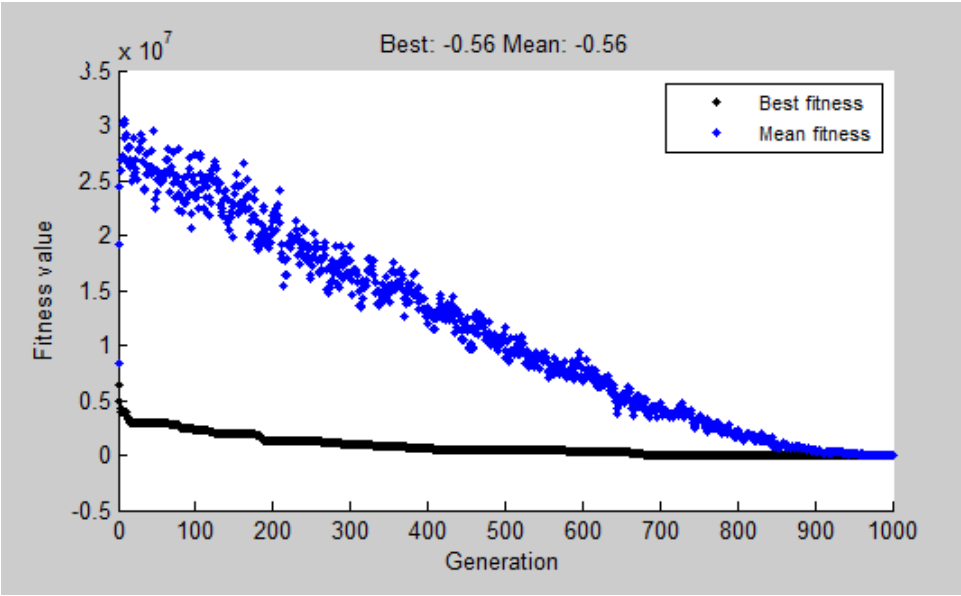

Figure7.2. Graph plots the value of fitness function with the generations.

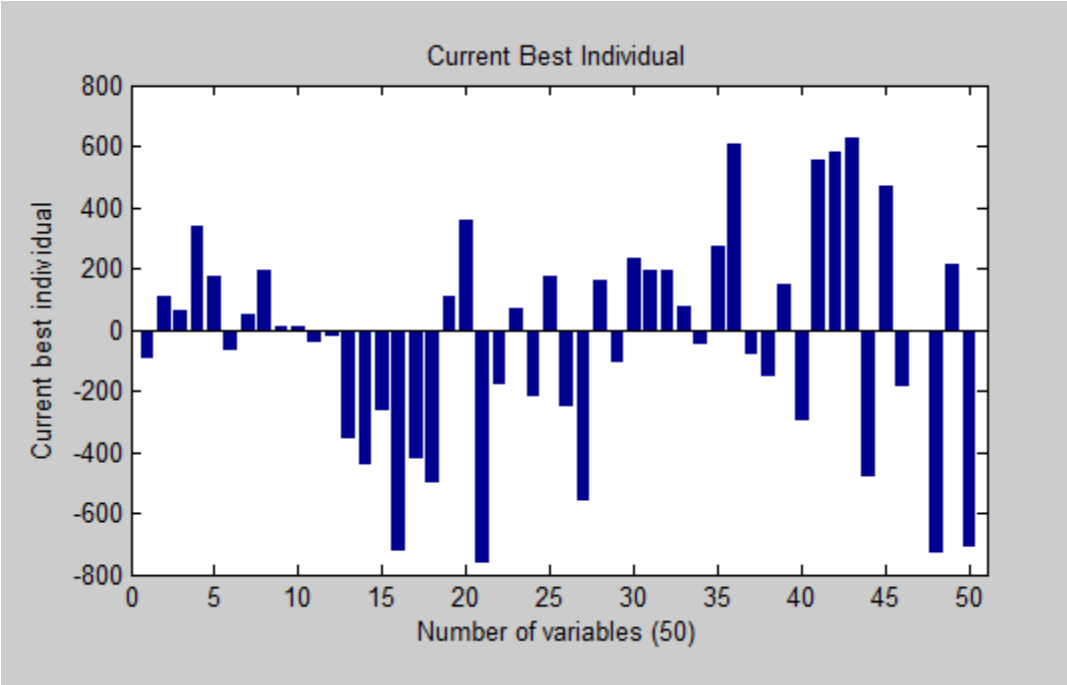

Figure 7.3 Graph calculates the current best value of delta of each individual in a partition.

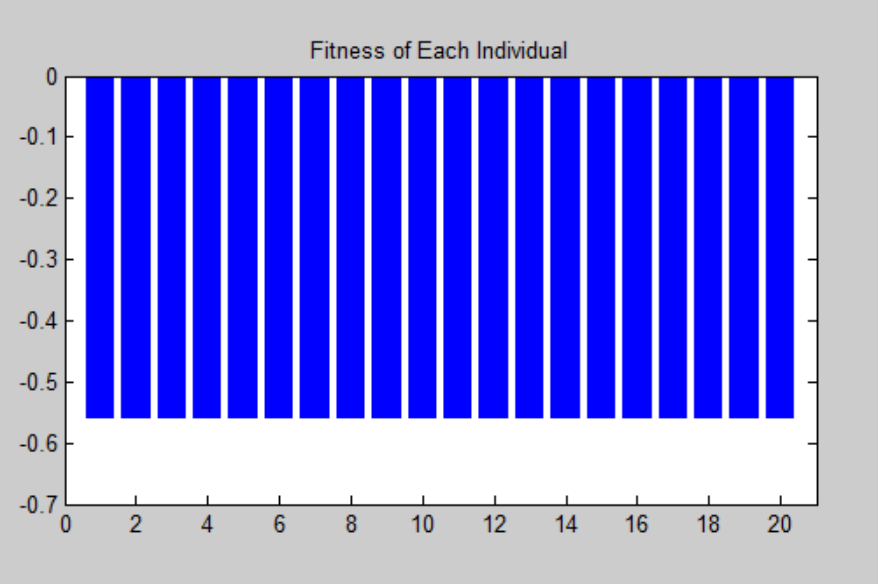

Figure7.4 Graph tells the fitness of each individual in a partition.

The GA will stop if any of the following 3 reaches 100%.

Generations (Generations) — specifies the maximum number of iterations for the genetic algorithm to perform.

Stall generations (StallGenLimit) — the algorithm stops if the weighted average change in the fitness function value over **Stall generations** is less than **Function tolerance**.

Stall time limit (StallTimeLimit) — the algorithm stops if there is no improvement in the best fitness value for an interval of time in seconds specified by **Stall time**.

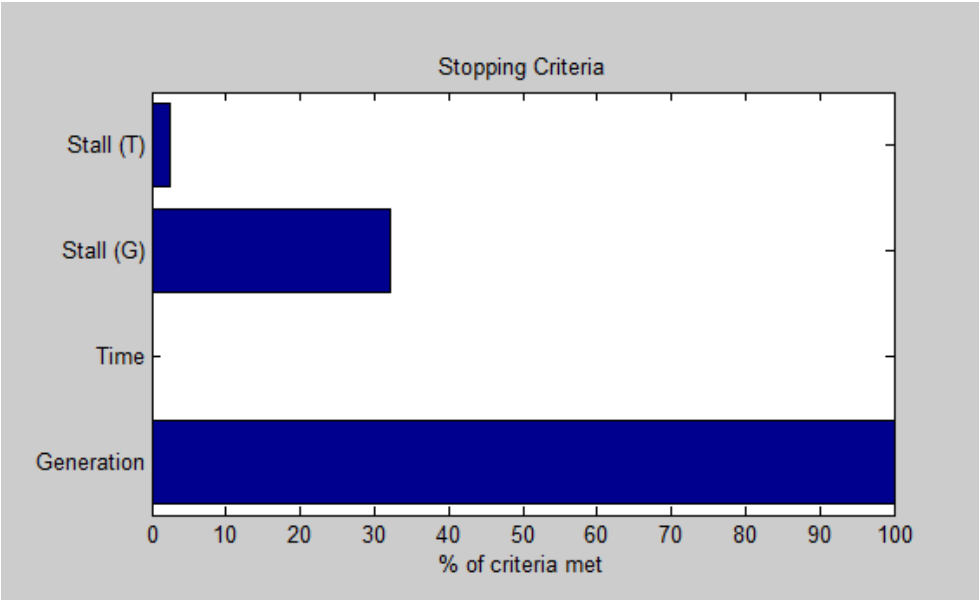

Figure 7.5. Graph plots the Stopping criteria with the % of criteria met.

7.3 BACTERIAL FORAGING

Bacterial Foraging Algorithm is executed to optimize the various partitions created of the sample relational database taken of 500 tuples. The database contains the record of salaries of various individual at the survey conducted by an organization. In this case, the exact value of attribute salary is not required. However, salary lies between a particular range and that range should not be violated while inserting watermarks. The usability constraint considered is interval constraints that are used to control the magnitude of the alteration for Δ_{ij}, that is,

$$\Delta_{ij}^{min} <= \Delta_{ij} <= \Delta_{ij}^{max}$$

Minimum and maximum value of interval constraint is decided and accordingly optimization is done using BFOA.

The reference point is calculated as ***ref =μ+ c* σ,*** where, c ϵ(0,1); is a secret real number that is a part of the set γ , μ is mean of S_i^W=Si +Δ_i* (i.e. $\mu_{(Si + \Delta i^*)}$) and σ is variance estimates of the set S_i^W=Si + Δ_i* (i.e. $\sigma^2_{(Si + \Delta i^*)}$.)

C=0.5 is chosen as it gives best results in this case.

Experiment is conducted by initializing following values to the parameters, the results obtained are shown in graph plotted (Fig.7.6)

S: Total number of bacteria in the population,

Nc =100: The number of chemotactic steps,

Ns=4: The swimming length.

Nre=4: The number of reproduction steps,

Ned =2: The number of elimination-dispersal events,

Ped=0.25: Elimination-dispersal probability,

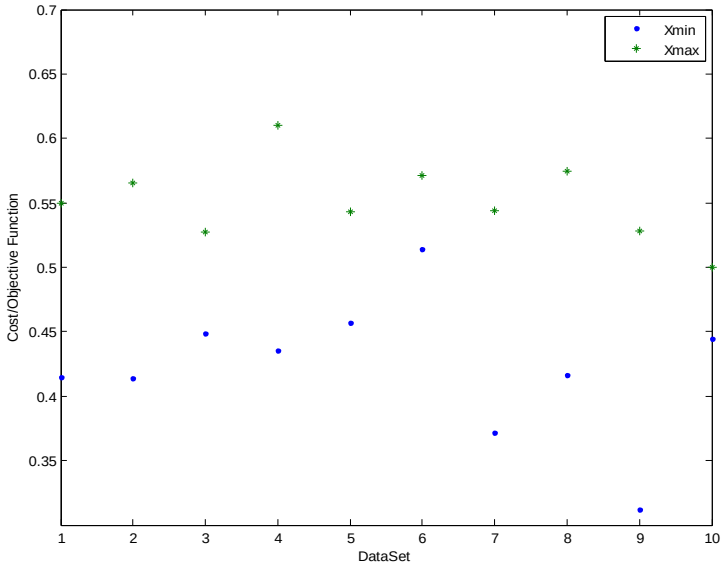

Fig.7.6. Optimal values attained at various datasets.

Experiment is conducted by varying values of S and keeping other variables at constant value reveals following result.

Considering following values of parameters the results obtained are shown in table 7.1. And graph plotted for single bacteria representing Cost at various chemotactic steps. (Fig.7.7)

S: Total number of bacteria in the population,

Nc =100: The number of chemotactic steps,

Ns=4: The swimming length.

Nre=4: The number of reproduction steps,

Ned =2: The number of elimination-dispersal events,

Ped=0.25: Elimination-dispersal probability,

Table 7.1 Cost and Execution time by varying S (for Minimization)

S	Cost(minimization)	Execution Time
50	0.3077	102.914887 seconds
40	0.3077	81.096739 seconds.
30	0.3077	61.379924 seconds
20	0.3077	40.704909 seconds.
10	0.3846	22.361223 seconds

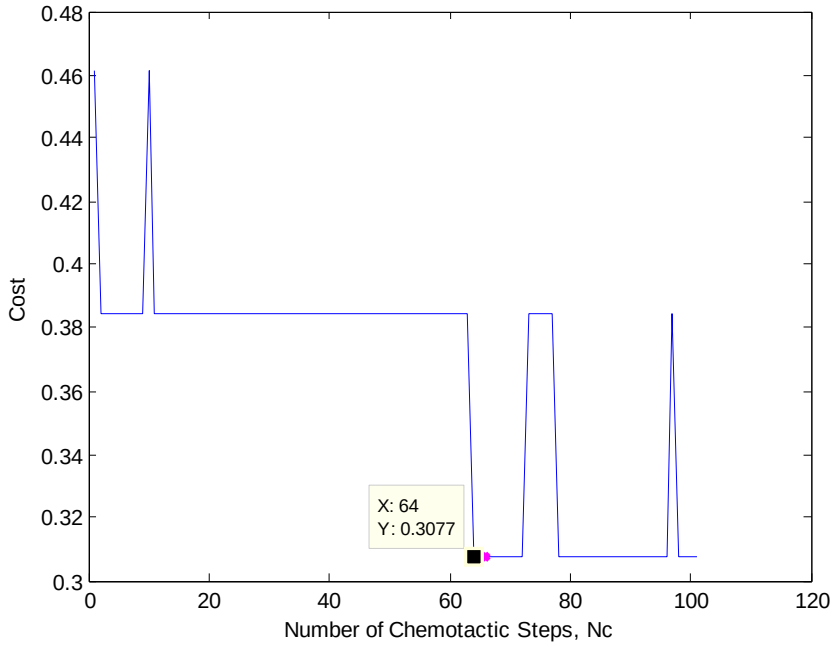

Fig. 7.7 Cost of Bacterium at various chemotactic steps for S=50 (Minimum value attained at Nc=64).

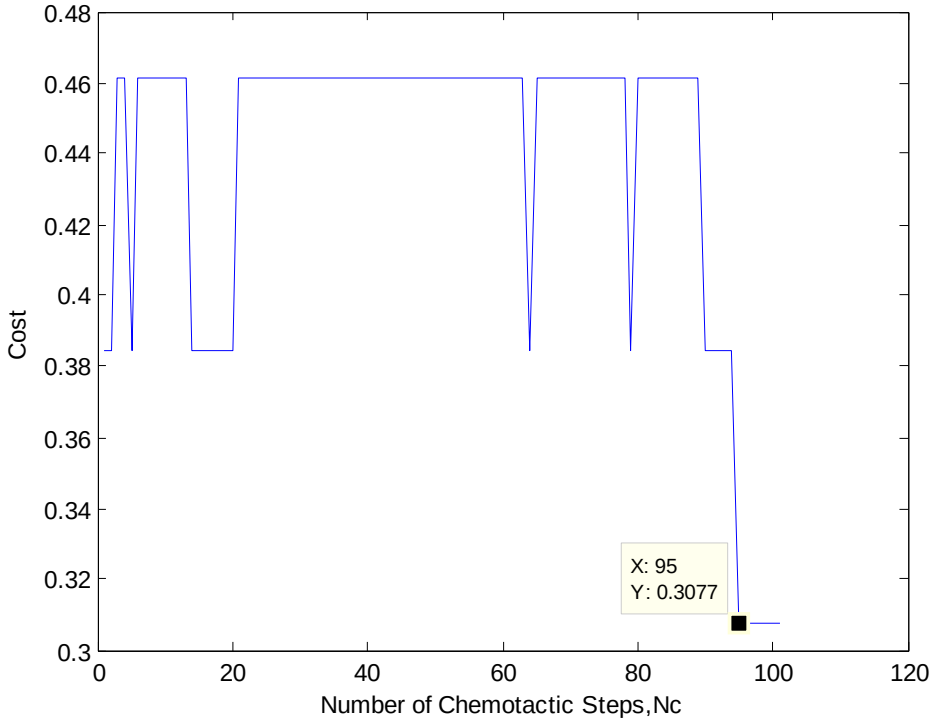

Fig. 7.8. Graph depicting optimal value attained at NC=95 for S=20.

Thus, the graphs reveals that for S=20 also, the results are obtained and execution time is reduced to 60%. However by reducing Nc will affect the optimal value as for S=20 at Nc=95 approximately, optimal value is obtained. For Maximization, same results are obtained and hence at S=20 optimal results can be obtained.

Thus, experiments are conducted by keeping value of Nc =100, S=20, Nre=4 while in case of GA number of generations taken=1000 to get desired results. Hence the choice of the technique to use depends on the application processing requirements. Solving the optimization problem does not necessarily require to find a global solution because finding such solution may require a large number of computations. GA could be used in order to determine optimal solutions when by trading processing time. GAs only when the processing time is not a strict requirement and watermarking is performed offline. For faster performance, the use of BFOA technique is recommended.

7.4 ATTACKS

7.4.1. Deletion Attack

In this attack, attacker randomly drops α tuples from the watermarked data set, the watermark is then decoded and watermark loss is measured for different α values. Fig. 7.9 shows the experimental results; they clearly show that proposed watermarking technique is resilient to the random deletion attack. Using proposed technique, the watermark was successfully extracted with 100 percent accuracy even when more than 80 percent of the tuples were deleted. Because our technique is highly resilient to tuple deletion attacks, the watermark can be retrieved from a small sample of the data. This important property combined with the high efficiency of proposed watermark detection algorithm makes it possible to develop tools able to effectively and efficiently search the Web to detect illegal copies of data.

Table 7.2 Resilience to Deletion Attack

Tuples Deleted (%)	**Watermarks Extracted (%)**
25%	100%
50%	100%
65%	100%
75%	80%
80%	80%
90%	50%

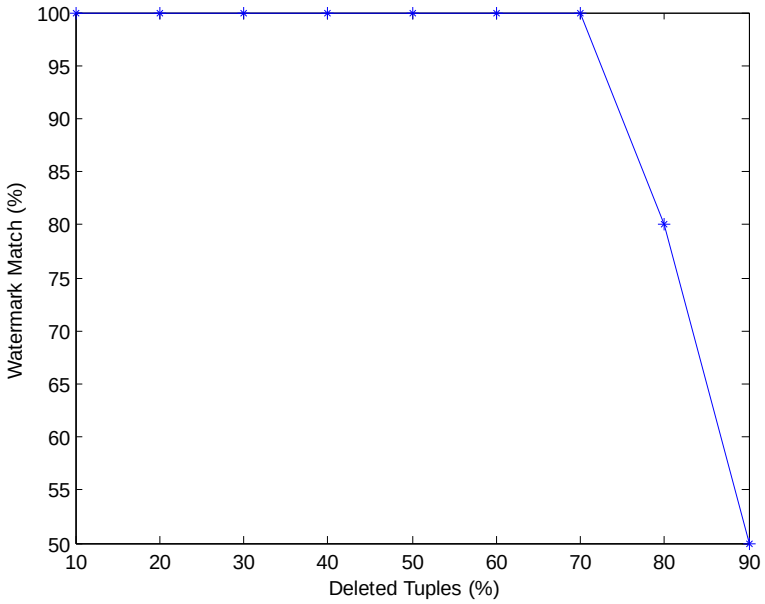

Fig. 7.9. Resilience to deletion attack

Alteration Attack

In this attack, attacker alters the data value of α tuples. Here, attacker is faced with the challenge that altering the data may disturb the watermark; however, at the same time, attacker does not have access to the original data set D and, thus, may easily violate the usability constraints and render the data useless. The alteration attack basically perturbs the data in hope of introducing errors in the embedded watermark bits. The attacker is trying to move the hiding function values from the left of the optimal threshold to the right and vice versa. However, using the conflicting objectives in encoding the watermark bits, that is the maximizing the tail count for b_i =1 and minimizing the tail count for b_i = 0, maximizes the distance between the hiding function values in both cases; thus, it makes it more difficult for the attacker to alter the embedded bit. In addition, by the repeated embedding of the watermark and the use of majority voting technique, this attack can easily be mitigated.

INSERTION ATTACK

Attacker decides to insert α tuples to the data set D_W hoping to perturb the embedded watermark. The insertion of new tuples acts as additive noise to the embedded watermark.

However, the watermark embedding is not based on a single tuple and is based on a cumulative hiding function that operates on all the tuples in the partition. Thus, the effect of

adding tuples is a minor perturbation to the value of the hiding function and thus to the embedded watermark bit. Marker-based watermarking techniques may suffer badly from this attack because the addition of tuples may introduce new markers in the data set and thus lead to the addition of new bits in the embedded watermark sequence. Consequently, this results in a watermark synchronization error. The watermark was recovered with 100 percent accuracy even when up to 75% percent of the data set size tuples were inserted.

Table 7.3 Resilience to Insertion Attack

Tuples Inserted (%)	**Watermarks Extracted (%)**
25%	100%
50%	100%
65%	100%
75%	100%
80%	80%
100%	80%

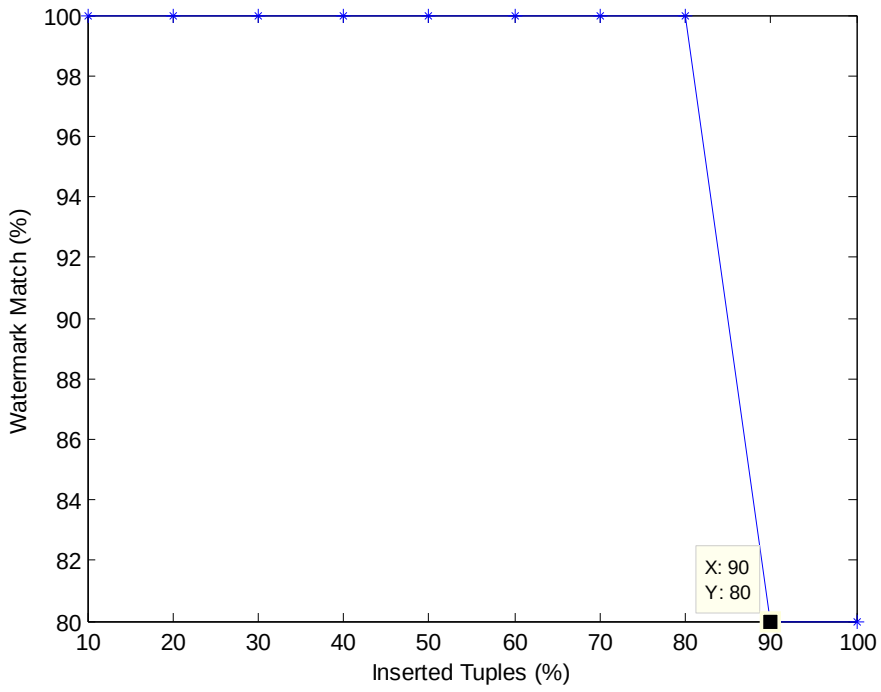

Fig. 7.10. Resilience to Insertion attack

Table 7.4. Comparison between our technique and techniques based on marker tuples

	Proposed Technique	Techniques based on marker tuples
Deletion Attack	Resilient to random tuple deletion attack; 100% watermark accuracy even when more than 80% of the tuples are deleted.	When the marker tuples are computed on the fly, the technique is not resilient to the random tuple deletion attack. The watermark accuracy deteriorates to 50% when only 10% of the tuples are deleted.
Insertion Attack	Resilient to random tuple insertion attacks; 100% watermark accuracy even when more than 100% of the original number of tuples is inserted in the α-insertion attack and similar watermark accuracy when subject to the (α, β)-insertion attacks.	When the marker tuples are computed on the fly, the technique is not resilient to random tuple insertion attacks; watermark accuracy deteriorates to 50% when only 10% of the tuples are inserted.
Alteration Attack	Resilient to random tuple alteration; 100% watermark accuracy even when 100% of the tuples are altered by $\beta > 1.0\%$ and $\beta > 3.0\%$ for the fixed-(α, β) and random-(α, β) attacks respectively. The watermark embedding technique exploits the feasible alteration space by solving an optimization problem to enforce the competing objectives based on the bit to be inserted. Furthermore, decoding threshold is computed based on the embedding statistics to maximize the probability of decoding error.	It is not clear how the bit embedding is performed; no systematic alteration scheme is defined that investigates the feasible embedding space. Thresholds are not based on embedding statistics and are chosen arbitrarily by the user without any optimality criteria.

7.5 CONCLUSION

An experiment is conducted to show resilience of our proposed technique to various attacks. A comparison our watermarking technique with previously posed marker tuples based techniques shows the superiority of our technique to deletion, alteration, and insertion attacks. Moreover optimization done by Bacterial Foraging gives better results than by genetic algorithm in terms of processing time and optimal results.

CHAPTER 8
CONCLUSION

The watermarking of relational databases is formulated as a constrained optimization problem and efficient techniques to handle the constraints are discussed. Two techniques are presented to solve the formulated optimization problem based on genetic algorithms (GAs) and Bacterial foraging (BF) techniques. .

In this dissertation a data partitioning technique is presented that does not depend on marker tuples to locate the partitions and, thus, it is resilient to watermark synchronization errors.

An efficient technique for watermark detection is proposed that is based on an optimal threshold. The optimal threshold is selected by minimizing the probability of decoding error

We have compared our watermarking technique with previous marker tuples based approaches and shown the superiority of our technique with respect to all types of attacks.

The watermark resilience was improved by the repeated embedding of the watermark and using majority voting technique in the watermark decoding phase.

Hence proposed watermarking technique is secure robust and imperceptible algorithm for numeric attributes that provides ownership identification as identity of owner is embedded as watermarks and proof of ownership as watermarks can be extracted back accurately from distorted or altered watermarked database. The robustness of the proposed algorithm is verified against number of database attacks.

FUTURE SCOPE

In this dissertation, watermarking technique for numeric attributes is proposed which can be extended for non-numeric attributes as well.

Further, in case of a relation with multiple attributes, the watermark resilience can be increased by embedding the watermark in multiple attributes. This is a simple extension to the presented encoding and decoding techniques in which the watermark is embedded in each attributed separately. The future scope of the work is to explore other evolutionary algorithms available in the literature is to solve constraint optimization problem framed in this dissertation.

REFERENCES

[Agrawal] Agrawal Rakesh, Peter J. Haas, Jerry Kiernan, "Watermarking relational data: framework, algorithms and analysis", The VLDB Journal, 2003, pp. 157-169.

[Al-Haj] Al-Haj Ali and Ashraf Odeh, "Robust and Blind Watermarking of Relational Database Systems", Journal of Computer Science 4 (12), pp. 1024-1029, 2008.

[Bhattarcharya] Bhattacharya S., A. Cortesi,"A Distortion Free Watermark Framework for Relational Databases". ICSOFT (2) 2009, pp. 229-234.

[Chen'10] Chen H., Y. Zhu, K. Hu, and X. He, "Hierarchical swarm model: a new approach to optimization," Discrete Dynamics in Nature and Society, vol. 2010, Article ID 379649, 30 pages, 2010.

[Chen'11] Chen Hanning , Yunlong Zhu, and Kunyuan Hu," Adaptive Bacterial Foraging Optimization" , Abstract and Applied Analysis,Volume 2011 (2011), Article ID 108269, 27 pages.

[Cui] Cui Haiting, Xinchun Cui, Mailing Meng, "A Public Key Cryptography Based Algorithm for Watermarking Relational Databases", IEEE proceedings of International Conference on Intelligent Information Hiding and Multimedia Signal Processing 2008, pp. 1344-1347.

[Deshpande] Deshpande Arti, Mr. Jayant Gadge," New Watermarking technique for Relational Databases", IEEE proc. on ICETET-09, pp. 664-669.

[Dorigo'96] Dorigo M., V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," IEEE Transactions on Systems, Man, and Cybernetics. Part B, vol. 26, no. 1, pp. 29–41, 1996.

[Dorigo'99] Dorigo M., G. Di Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," Artificial Life, vol. 5, no. 2, pp. 137–172, 1999.

[Das] Das Swagatam, Arijit Biswas, Sambarta Dasgupta, and Ajith Abraham," Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications", Foundations of Computational Intelligence, Vol 3 (2009), Springer, pp: 23–55.

[Eberchart] Eberchart R. C. and J. Kennedy, "A new optimizer using particle swarm theory," in Proceeding of the 6th International Symposium on Micromachine and Human Science, pp. 39–43, Nagoya, Japan, 1995.

[Floudas] Floudas C. A., Deterministic Global Optimization: Theory, Methods and Applications, vol. 37 of Nonconvex Optimization and Its Applications, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[Goldberg] Goldberg D.E., "Genetic algorithms in search, optimization and machine learning", Addison-Wesley, Reading, MA, 1989

[Gupta] Gupta V.K., "Copyright Issues Relating to Database Use" DESlDOC Bulletin of k\tom\ation Technology, Vol. 17, No. 4, July 1997, pp. 1 1-1 6 ,1997, OESIOOC.

[Hanyurwimfura] Hanyurwimfura Damien, Yuling Liu and Zhijie Liu, "Text Format Based Relational Database Watermarking for Non-numeric Data", IEEE ICCDA 2010, pp.v4-312-316.

[Holland] Holland J. H., "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to B, control, and Artificial Intelligence", University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[Hu] Hu Zhongyan, Zaihui Cao, Jianhua Sun," An Image Based Algorithm for Watermarking Relational Databases", IEEE 2009 International Conference on Measuring Technology and Mechatronics Automation, pp. 425-428

[Jiang] Jiang Chuanxian, Xiaowei Chen , Zhi Li "Watermarking Relational Databases for Ownership Protection Based on DWT", Fifth

International Conference on Information Assurance and Security 2009, pp. 305-308.

[Khanduja] Khanduja Vidhi , O.P.Verma,"Identification and Proof of Ownership by Watermarking Relational Databases", 2011 IEEE International Conference on Network communication and Computer (ICNCC-2011), pp. 257-260.

[Mathew] Mathew Tom V.,"Genetic Algorithms", Lecture Notes, IITB, Jan, 2005.

[Odeh] Odeh Ashraf and Ali Al-Haj, "Watermarking Relational Database Systems", 2008 IEEE, pp. 270-274.

[Passino] K. M. Passino, "Biomimmicry of bacterial foraging for distributed optimization and control," IEEE Control Systems Magazine, 2002, Vol. 22, Issue 3, pp. 52–67.

[Mishra] S. Mishra, "A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation," IEEE Transactions on Evolutionary Computation, vol. 9, no. 1, pp. 61–73, 2005

[Schneier] Schneier, B.: Applied Cryptography. John Wiley, New York (1996).

[Shehab] Shehab Mohamed, Elisa Bertino, and Arif Ghafoor, "Watermarking Relational Databases Using Optimization-Based Techniques", IEEE Transactions On Knowledge And Data Engineering, Vol. 20, No. 1, January 2008.

[Sion] Sion R., S., M. Atallah and S. Prabhakar, "Rights protection for relational data", IEEE Transactions on Knowledge and Data Engineering, 2004, pp. 1509-1525.

[Spall] Spall J. C., "Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control", Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley-Interscience, Hoboken, NJ, USA, 2003.

[Sun] Sun Jianhua, Zaihui Cao, Zhongyan Hu," Multiple Watermarking Relational Databases Using Image", IEEE proc. on 2008 International Conference on MultiMedia and Information Technology, pp-373-376.

[Tripathy] Tripathy M., S. Mishra, L. L. Lai, and Q. P. Zhang, "Transmission loss reduction based on FACTS and bacteria foraging algorithm," 9th International Conference on Parallel Problem Solving from Nature (PPSN '06), vol. 4193 of Lecture Notes in Computer Science, pp. 222–231, September 2006.

[wiki] http://en.wikipedia.org/wiki/Digital_watermarking.

[wiki] http://en.wikipedia.org/wiki/Copyright.

[Vaas] Vaas L.,"Putting a Stop to Database Piracy," eWEEK, EnterpriseNews and Revs., Sept. 2003.

[Zhang] Zhang Zhi-Hao , Xiao-Ming jin, Jain-Min wan , "Watermarking Relational Database Using Image" , IEEE, Third International Conference on Machine Learning and Cybernetics, 2004, pp. 1739-1744.