**PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA**
**MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH**
**UNIVERSITY MOHAMED BOUDIAF - M'SILA**

**FACULTY: MATHEMATICS AND INFORMATICS**
**DEPARTEMENT : CMPUTER SCIENCE**
**N° :……..**

**DOMAIN : MATHEMATICS AND COMPUTER SCIENCE**
**FIELD : COMPUTER SCIENCE**
**OPTION : SIGL**

**Dissertation submitted in partial fulfillment of the requirements for The degree of MASTER**

**By**: Bahache Anwar Nour Eddine

**Subject**

# A Metaheuristic Based Approach for Solving the Index Selection Problem in Data Warehouses

**Publicly defended on:** 24/06/2018 **before the jury composed of :**

| | | |
|---|---|---|
| Bahache Mouhamed | University of M'sila | Chair |
| Mr. Ariouat Youcef | University of M'sila | Supervisor |
| Boucetta Mouhammed | University of M'sila | Examiner |

**College year : 2017 /2018**

**PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA**
**MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH**
**UNIVERSITY MOHAMED BOUDIAF - M'SILA**

**FACULTY: MATHEMATICS AND INFORMATICS**
**DEPARTEMENT : CMPUTER SCIENCE**
**N° :……..**

**DOMAIN : MATHEMATICS AND COMPUTER SCIENCE**
**FIELD : COMPUTER SCIENCE**
**OPTION : SIGL**

**Dissertation submitted in partial fulfillment of the requirements for The degree of MASTER**

**By**: Bahache Anwar Nour Eddine

**Subject**

# A Metaheuristic Based Approach for Solving the Index Selection Problem in Data Warehouses

**Publicly defended on:** 24/06/2018 **before the jury composed of :**

| | | |
|---|---|---|
| Bahache Mouhamed | University of M'sila | Chair |
| Mr. Ariouat Youcef | University of M'sila | Supervisor |
| Boucetta Mouhammed | University of M'sila | Examiner |

**College year : 2017 /2018**

# *Acknowledgment*

*First and foremost, heartfelt gratitude and praises go to the Almighty Allah who guided me through and through.*

*This work could not have reached fruition without the unflagging assistance and participation of so many people whom I would never thank enough for the huge contribution that made this work what it is now.*

*I would like to thank profoundly Mr Y.Ariouat for his scientific guidance and corrections, suggestions and advice, pertinent criticism and pragmatism, and particularly for his hard work and patience. I am very grateful to him, for without his help an important part of this work would have been missed.*

*I would also like to thank all the Jury Members, who have agreed to review this work.*

*I thank also Abir (Panda) for motivating and encouraging me on everytime i was down.*

*I would also thank those who helped me to finish this work whom without their assisstance it would not have been done : Imane, Asma.*

*I thank all the teachers who guided us throughout our journey.*

*Thanks to all who helped me*

*Thanks*

# *Dedication*

*This dessertation is dedicated to My parents and for their encouragement, prayers,motivations and beeing there. May god bless them .*

*My preciouse brothers(Sami and Amine) and sisters (Houda and Wafa) who've been there for me in my entire journey, all of my nephews and my nieces.*

*All of my friends and colleagues.*

*And for last the uique Abir (Panda) for her motaviations and encouragements.*

# TABLE OF CONTENTS

# FIGURE LIST

# TABLE LIST

# ABBREVIATION LIST

***DW:*** Data Warehouse

***DB:*** Data Base

***ETL:*** Extract, Transform and Load

***OLTP :*** On-Line Transactional Processing

***OLAP :*** On-Line Analytical Processing

***MOLAP:*** Multidimensional On-Line Analytical Processing

***ROLAP:*** Relational On-Line Analytical Processing

***HOLAP:*** Hybrid On-Line Analytical Processing

***HF:*** Horizontal Fragmentation

***PHF:*** Primary Horizontal Fragmentation

***DHF:*** Derived Primary Horizontal Fragmentation

***ISP:*** Index Selection Problem

***BJI:*** Binary Join Index

***SBJI:*** Single Binary Join Index

***MBJI:*** Multi Binary Join Index

***Q:*** Query Workload

***A:*** Candidtae Attributes

***S: Storage***

***CI:***I ndex Configuration

***BPSO:*** Binary Partical Swarm Optimization

***GA:*** Genetic Algorithm

***AFSA:*** Artificial Fish Swarm Algorithm

***bAFSA:*** Binary Artificial Fish Swarm Algorithm

***IbAFSA:*** improved Binary Artificial Fish Swarm Algorithm

***MIbAFSA:*** Modified improved Binary Artificial Fish Swarm Algorithm

***AFSISbtool:*** Artificial Fish Swarm Index Selection Tool

# GENERAL INTRODUCTION

Data warehouses holds a large amount of data and it plays a crucial role in business intelligence. A data warehouse can be viewed as a database that aims to group and homogenize large datasets from heterogeneous sources for analysis and decision support.

Most of data warehouses nowadays are modeled by a star schema. This model is usually consisted of a central fact table of very large facts (can reach Terra Octets) and several dimension tables which represent the axes of 'analysis. Dimension tables are linked to the fact table via their key attributes being present in the fact table as foreign keys. Data stored in a warehouse modeled according to this schema is queried by decision queries called star join queries. Due to the increasement of data present in data warehouses the cost of its administration rises and its performance decreases which are caused by the join operations between tables while trying to query the data that is stored in it using the star join queries. These queries are complex and require an execution time that can reach several hours or days because of the volumetry of the data and the multitude of operations of star joins between the fact table and the dimension tables. Such a response time is unacceptable by an analyst or decision maker, so it is crucial to use optimization structures and techniques to reduce it.

One of these structuresare Binary Join Indexes (BJI) which are used to pre calculate joins between tables. Two variantsof BJI exists : single bitmap join index (on a single non key attribute SBJI) and multi bitmap join index (on multiple non key attributes MBJI). However, given the large number of dimension tables and attributes that can be indexed, the selection of indexes to create is a known NP-hard problem. As a result, there is no algorithm that offers an optimal solution in a reasonable time. Several research works use heuristics or datamining techniques to reduce the complexity of the problem and propose solutions close to the optimal solution. Generally, these approaches proceed in two steps: selecting an initial configuration in order to reduce the complexity of the problem, followed by a second pruning step that allows the selection of a final index configuration.

In the present work we propose a new heuristic approach based on the improved binary version of the Artificial Fish Swarm Algorithm (noted IbAFSA) using a mathematical cost model. We did some improvements on the IbAFSA by ourselves so it converges to an optimal solution faster. The new approach was named Modified IbAFSA (Noted MIbAFSA).

Several experiments were performed to demonstrate the effectiveness of the proposed approach and the results were compared to a datamining-based approach and a constraint programming-based approach.

# WORKPLAN

The thesis is organized in four chapters:

- **The first chapter :** the first chapter introduces the concept of data warehouses (definition, specificities with regard to databases, architecture and various implementations, etc.) as well as some problems related to the physical design of such a system. Then it discusses the most popular queries optimization techniques in this context and emphasizes the usefulness of different indexing techniques as well as their utility in reducing the response time of queries.
- **The second chapter :** presents the index selection problem in the context of data warehouses (definition, formalization of the problem, complexity, etc.). Then it introduces a state of the art presenting the general approach of index selection in the literature as well as the main works that have addressed this problem. Finally, It contains an analysis of these works to identify their main locks. These will be our motivation to propose a new approach to index selection.
- **The third chapter :** presentation of our approach. We start with the presentation of the general AFSA (Artificial fish swarm algorithm), its basis, variations and the AFSA for global optimization purposes. Then we present our approach in detail followd by some diagrams showing the selection process.
- **The fourth chapter :** presentation of the AFSISbtool (Artificial Fish Swarm Index Selection based tool) that we developed to select the optimum set of SBJI to be indexed, how did we fix the parameters , presentation of the tool's functionalities followed by some experiments. After evaluating and fixing the parameters we've done a comparative study and the results are shown at the end of the chapter.

# CHAPTER 1

# DATA WARHOUSE OPTIMASATION TEQUNIQUES

## 1. Introduction

Computers began to play an important and vital role in the manufacturing process with the Automation era. However, these older processes were independent or stand-alone systems that did not allow the transmission and storage of information onto other computer systems. The users' needs to have access to this data eventually evolved into the creation of distributed database management technology. This software allowed specific information to be pulled from databases located throughout the organization, collected and stored on a computer located in a central location and consolidated and analyzed by the users.

Although the concept of the distributed database management system was good in theory, it still did not resolve the issues with the inability to share information among the various relational databases or incompatible computer systems. The solution to this problem was the onset of data warehousing. Through the concept of client/server technology, data would be copied regardless of computer types and platforms and placed onto a common server.

In general, the benefits of data warehousing are all based on one central premise: warehousing solves the ongoing problem of analyzing separate data and converting it into actionable information you can use. Warehousing also allows you to process large amounts of complex data in an efficient way. When you successfully implement a data warehouse system, it’s possible to access the benefits associated with the practice the very benefits that are making data warehousing a common practice for many businesses today.

Yet, the creation of datawarehouses caused a new problem because of the large amount of the registered data which is the time that the query takes to be done that may take hours to days even months depending on the amount of the stored data, so there was a need for the data warehouse to be optimized.

In the following sections we are going to present a few definitions of a Data Warehouse over time, characteristics, modals and optimization techniques.

-

# 2. Data Warehouse

## 2.1. Definitions

There are many definitions of a Data Warehouse according to many authors :

**Def1 :***A data warehouse is a subject-oriented, integrated, time-variant, nonvolatile collection of data in support of management's decision-making process.***[1]**

**Def2 :***A data warehouse is a system that retrieves and consolidates data periodically from the source systems into a dimensional or normalized data store. It usually keeps years of history and is queried for business intelligence or other analytical activities. It is typically updated in batches, not every time a transaction happens in the source system.***[2]**

**Def3 :***A data warehouse is a relational database that is designed for query and analysis rather than for transaction processing. It usually contains historical data derived from transaction data, but it can include data from other sources. It separates analysis workload from transaction workload and enables an organization to consolidate data from several sources.*

*In addition to a relational database, a data warehouse environment includes an extraction, transportation, transformation, and loading (ETL) solution, an online analytical processing (OLAP) engine, client analysis tools, and other applications that manage the process of gathering data and delivering it to business users.***[3]**

**Def4 :***A Data Warehouse (DW) can be seen as a set of materialized views defined over the source relations.***[4]**

**Def5 :***A data warehouse is a centralized, integrated repository of information that must support complex decision support queries without performance degradation . A data warehouse simply provides a means to manage data outside of the operational systems where the data is originated.***[5]**

## 2.2. Characteristics of a Data Warehouse

The data of a Data Warehouse must respect the following characteristics :

- **Subject-Oriented**: The Data Warehouse is designed to help you analyze data. For example, to learn more about your company's sales data, you can build a warehouse that concentrates on sales, production and marketing... (as shown in figure 1.1). Using this warehouse, you can answer questions like "Who was our best customer for this item last year?" This ability to define a Data Warehouse by subject matter, sales in this case, makes the Data Warehouse subject-oriented.

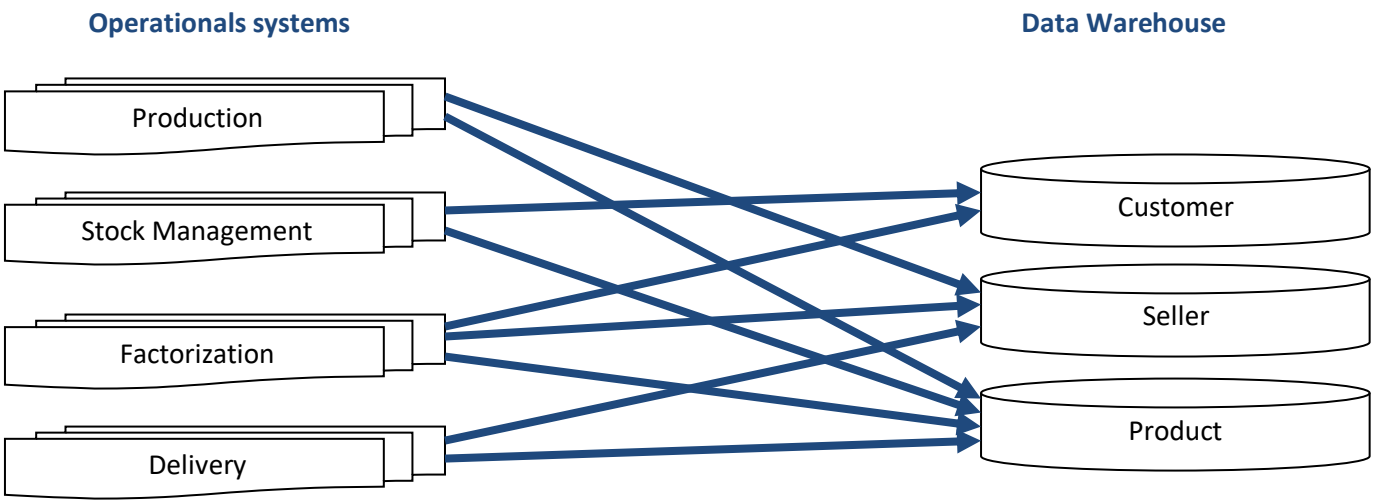


**Figure 1. 1:** Subject-Oriented Data in a Data Warehouse

- **Integrated:** Integration is closely related to subject orientation. Data Warehouses must put data from disparate sources into a consistent format. They must resolve such problems as naming conflicts and inconsistencies among units of measure. When they achieve this, they are said to be integrated.
- **Non-volatile:** Nonvolatile means that, once entered into the warehouse, data should not change. This is logical because the purpose of a warehouse is to enable you to analyze what has occurred.
- **Time-Variant:** In order to discover trends in business, analysts need large amounts of data. This is very much in contrast to online transaction processing (OLTP) systems, where performance requirements demand that historical data be moved to an archive. A Data Warehouse's focus on change over time is what is meant by the term time-variant.

### 2.3. Data Base VS Data Warehouse

The data bases are used in the enterprises to organize important volumes of information contained in their operational systems. These data are managed according to transactional online processes (OLTP **:** "On-Line Transactional Processing"). They are destined to inserting, modifying and deleting the records of a data base that is designed in the third normal form. A data warehouse is designed to support analysis operations that are useful for decision-making. These decision operations use online data analysis processes (OLAP : "On-Line Analytical Processing"). These processes meet the specific needs of information analysis.**[6]**

Table 1.1 summarize the differencesbetween database management systems and the data warehouses **[7]** :

| | Data Base | Data Warehouse |
|---|---|---|
| **Purpose** | support the current operation of the enterprise | Support management decisions |
| **Data** | Detailed | Detailed and summarized (abstract) |
| | Application – oriented | Subject- oriented |
| | Up to date | Time variant ( historical) |
| | Dynamic | Static |
| | Gega-octets | Tera-octets or more |
| **Users/use** | Clerk | decision-maker analyst |
| | Large (numerous) | Small (few) |
| | Simultaneous Access | Non Simultaneous |
| | Update query | Interrogation |
| | predefined query | Query « one use » |
| | Immediate response | Slow response |

**Table 1. 1: D**ifferences between DB and DW

## 2.4. Data Model for datawarehouses

The stored data to be analyzed must be represented in a way that facilitates the decision making**.** A data structure according to many analyzing axes ( ex : time, geographic location,...) is primordial. The traditional modelisation under the form of relations is inadequate to support the efficiency of these analyzes (called multidimensional analyzes).Thus, the data is represented in the form of a hyper-cube (data cube).It is what we call : Multidimensional modeling.

The figure 1.2 illustrate an example of a data cube. It represents sales distribution according to many analyze axes : time, location or region and category.

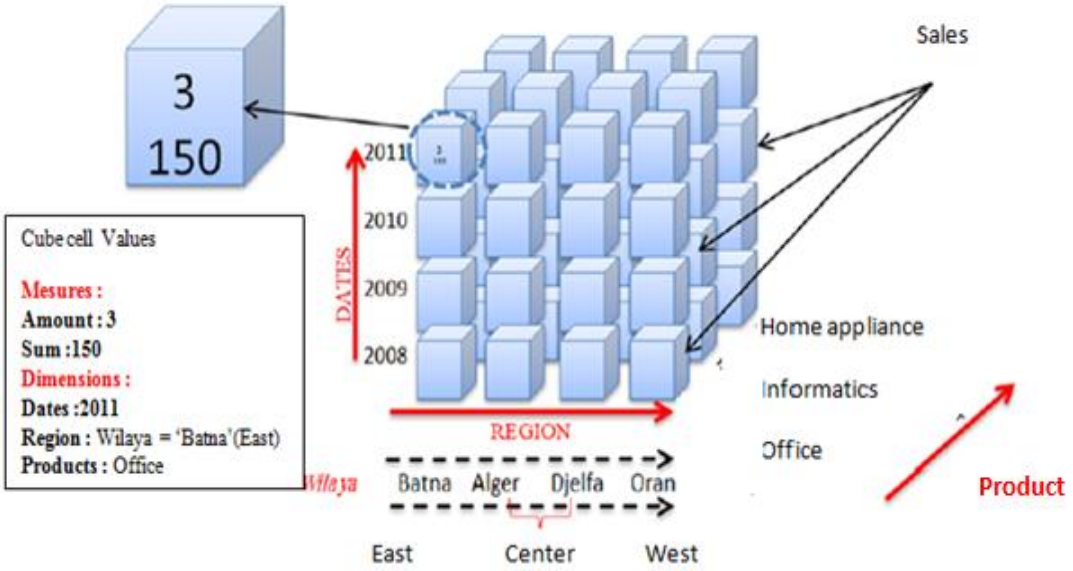


**Figure 1. 2:** Data Cube example

This multidimensional modeling gave birth to the concepts fact and dimension **[8]**

**Fact :** The unit of information in a data warehouse. The fact is a cell in the multidimensional space, limited by the units of analysis. Facts are stored in a table (when used in a relational database) or are a cell in a multidimensional database. Data Modeling for DataWarehouses. Every fact contains the basic information about the fact (revenue, value, satisfaction note, etc.), and relates to the dimensions. In some cases the pure occurring of the fact is information enough for the data warehouse, when all of the necessary information is stored in the dimensions. We talk then about a factless fact. The fact that the measurements are numerical allows to summarize a large number of recordings in a few recordings (we can add,calculate the minimum, the maximum or the average).For example, in the fact "Sales", we can see the measure sum of the product sold by a wilaya.

**Dimension :** The axes of a coordinate system that bind the space defined by that coordinate system. The coordinate system in a data warehouse defines the data cells, which contain facts. An example of a coordinate system is the Cartesian coordinate system with x and y dimensions. In a data warehouse, one of the dimensions is always time. For example, the management of « sales » can be analyzed according to the dimensions : Product, Region, and Dates. For the dimension Dates we can obtain the hierarchy : Year, Semester, Trimester, Month, Week and Day.

## 2.5. Implementation of multidimensionalmodels

The storage mode of a partition affects the query and processing performance, storage requirements, and storage locations of the partition and its parent measure group and cube. The choice of storage mode also affects processing choices.

A partition can use one of three basic storage modes:

- Multidimensional OLAP (MOLAP)
- Relational OLAP (ROLAP)
- Hybrid OLAP (HOLAP)

Microsoft SQL Server Analysis Services supports all three basic storage modes. It also supports proactive caching, which enables you to combine the characteristics of ROLAP and MOLAP storage for both immediacy of data and query performance. (According to how the data cubes are stored, it exist three ways of constructing a multidimensional model based system: The MOLAP approach, the ROLAP approach and a third Hybrid one.)

### 2.5.1. The MOLAP approach

The MOLAP storage mode causes the aggregations of the partition and a copy of its source data to be stored in a multidimensional structure in Analysis Services when the partition is processed. This MOLAP structure is highly optimized to maximize query performance. The storage location can be on the computer where the partition is defined or on another computer running Analysis Services. Because a copy of the source data resides in the multidimensional structure, queries can be resolved without accessing the partition's source data. Query response times can be decreased substantially by using aggregations. The data in the partition's MOLAP structure is only as current as the most recent processing of the partition.

As the source data changes, objects in MOLAP storage must be processed periodically to incorporate those changes and make them available to users. Processing updates the data in the MOLAP structure, either fully or incrementally. The time between one processing and the next creates a latency period during which data in OLAP objects may not match the source data. You can incrementally or fully update objects in MOLAP storage without taking the partition or cube offline. However, there are situations that may require you to take a cube offline to process certain structural changes to OLAP objects. You can minimize the downtime required to update MOLAP storage by updating and processing cubes on a staging server and using database synchronization to copy the processed objects to the production server. You can also use proactive caching to minimize latency and maximize availability while retaining much of the performance advantage of MOLAP storage.**[9]**

### 2.5.2. The ROLAP approach

The ROLAP storage mode causes the aggregations of the partition to be stored in indexed views in the relational database that was specified in the partition's data source. Unlike the MOLAP storage mode, ROLAP does not cause a copy of the source data to be stored in the Analysis Services data folders. Instead, when results cannot be derived from the query cache, the indexed views in the data source is accessed to answer queries. The query response is generally slower with ROLAP storage than with the MOLAP or HOLAP storage modes. Processing time is also typically slower with ROLAP. However, ROLAP enables users to view data in real time and can save storage space when you are working with large datasets that are infrequently queried, such as purely historical data.**[9]**

### 2.5.3. The HOLAP approach

The HOLAP storage mode combines attributes of both MOLAP and ROLAP. Like MOLAP, HOLAP causes the aggregations of the partition to be stored in a multidimensional structure in an SQL Server Analysis Services instance. HOLAP does not cause a copy of the source data to be stored. For queries that access only summary data in the aggregations of a partition, HOLAP is the equivalent of MOLAP. Queries that access source data—for example, if you want to drill down to an atomic cube cell for which there is no aggregation data—must retrieve data from the relational database and will not be as fast as they would be if the source data were stored in the MOLAP structure. With HOLAP storage mode, users will typically experience substantial differences in query times depending upon whether the query can be resolved from cache or aggregations versus from the source data itself.**[9]**

In this thesis, we are interested in data warehouses modeled according to a multidimensional ROLAP model, we describe in the following the most used multidimensional data structure in this model.

**Star schema**

This model works with a fact table, it's the center of the schema. The star schema represents a fact table connected to a set of dimension tables. Each record in the fact table is a fact, (the basic unit). The granularity of the schema makes it possible to determine what will be a fact.This model is recommended because of its low complexity, ease of understanding for the end user, and direct links to the logical structures of the data. .**[10]**

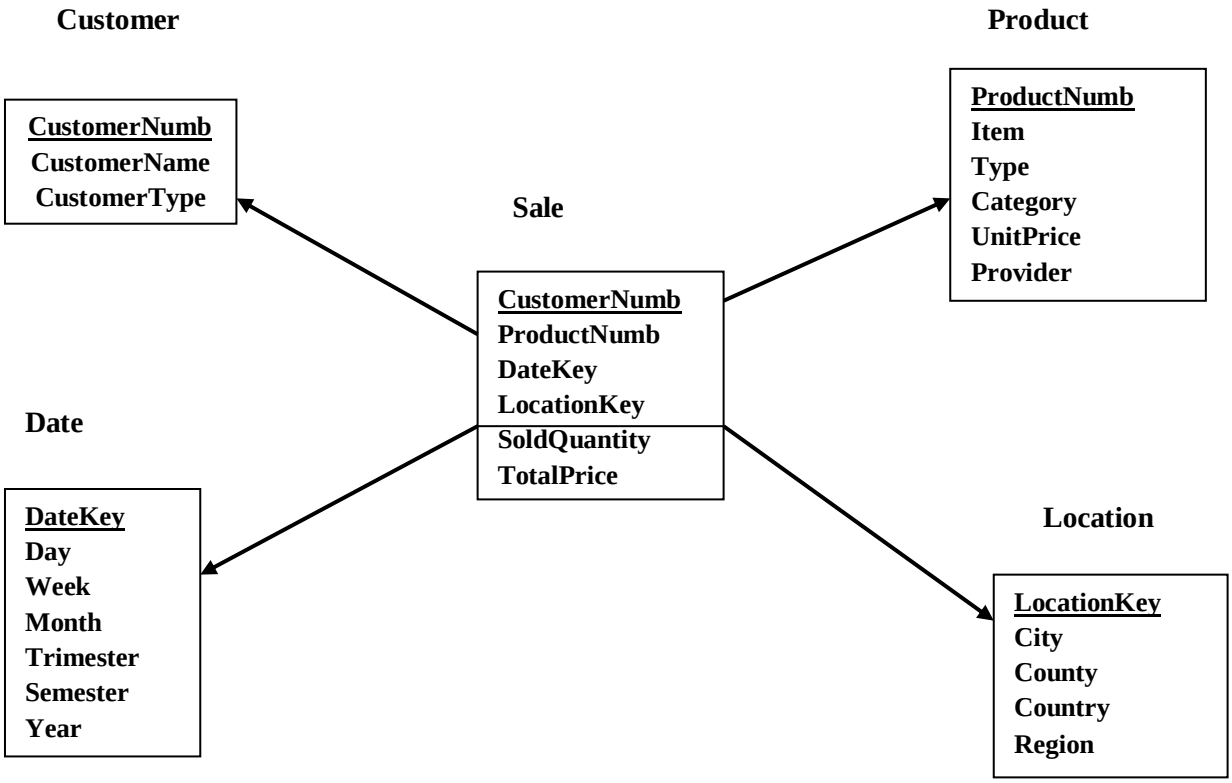


**Figure 1. 3:** Star Scheme **[22]**

In the figure 1.3 we can see that the fact table is Sale and the dimension tables are Client, Date, Product and Location. The last are all connected to the fact table with a key (Primary key).

**Variants of the star schema**

1. **In the snowflakescheme** we break (split) the dimension tables into sub-tables according to the hierarchy of this dimension. This can be seen as a normalization of dimension tables. This scheme avoids redundancies of information which saves disk space and facilitates the supply. But can alter the performance of the warehouse because it requires joins at aggregates on the dimensions.
2. **The Fact constellation schema** consists of merging several star models that use Common dimensions**[6]**Therefore a constellation model includes several facts and common dimensions or not.

## 3. Data Warehouse optimization Techniques

The analyzing process is performed using complex queries with multiple joins between the fact table and adjacent dimension tables and aggregate operations on very large tables. A response time that can last hours or even days is unacceptable given the interactive nature of data warehouses and the need for an acceptable response time to make timely decisions. Therefore, there is an urgent need for efficient and sophisticated optimization techniques in physical design.**[11]** Without a query optimization technique, querying a data warehouse would be complex.

The existing works on decision queries performance optimization classify the optimization structures into two categories :**[12][13][14]**

**1. Non-redundant techniques:** do not duplicate data but allow to reorganize their physical representation **[14] :**horizontal fragmentation and parallel processing.

**2. Redundant techniques :**the use of these techniques produce a duplication of data : indexes, materialized views, and vertical fragmentation.

Figure 1.4 illustrates a classification of the main optimization techniques.

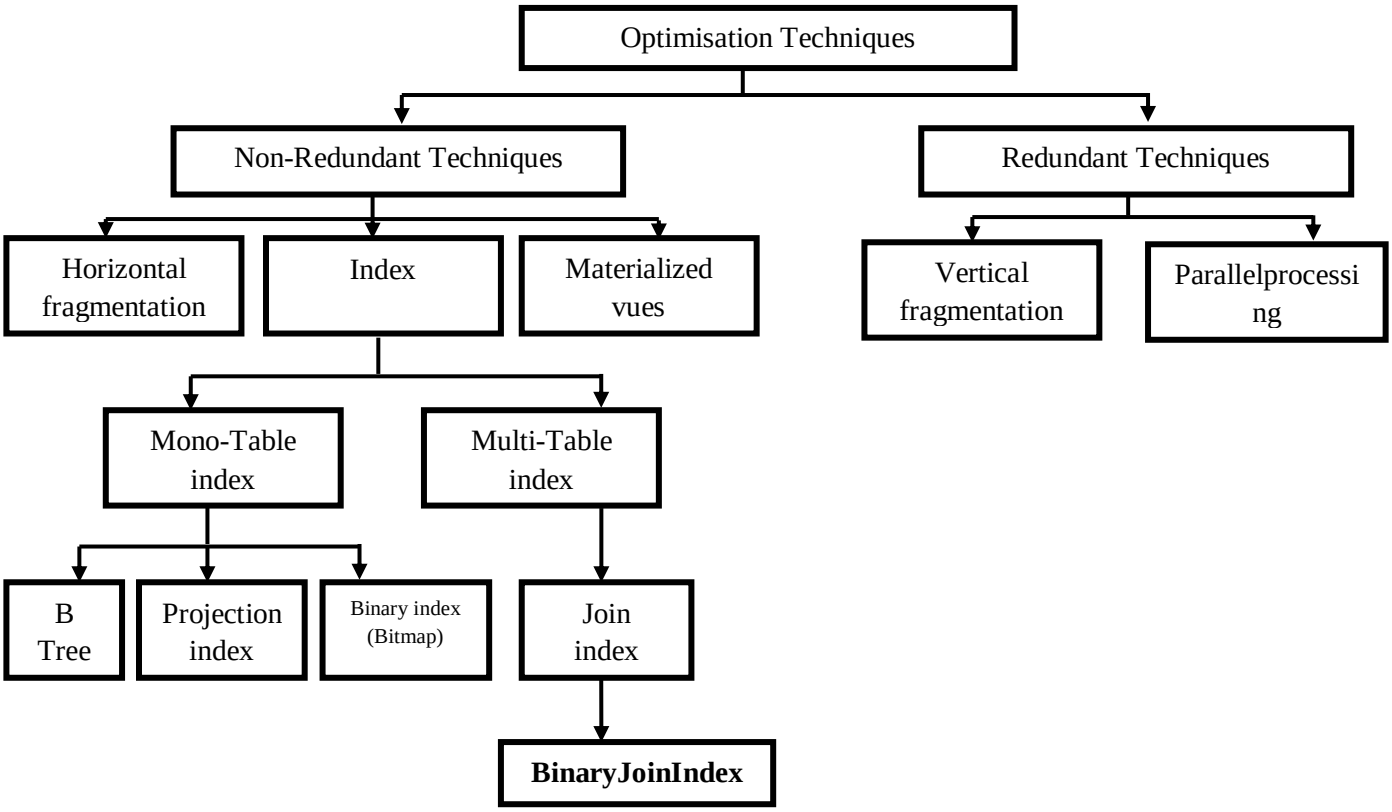


**Figure 1. 4:**Optimization techniques classification

### 3.1. Non-Redundant Optimization Techniques

In the non-redundant optimization techniques there is no duplication of data and in fact they do not require any storage cost or maintenance cost.

#### 3.1.1. Horizontal Fragmentation

In the context of relational data warehouses, horizontal fragmentation HF consists in segmenting tables, indexes and materialized views into several sets of fragments (partitions) of the same structure as the initial table.Each fragment is defined by a set of selection predicates, physically stored and accessed separately **[15].**

The purpose of the data warehouse table fragmentation is to isolate parts of the warehouse with different access probabilities. As a result, the fact of grouping the data corresponding to a set of similar requests makes it possible to limit the search times and to improve the management of the cache memories according to the frequency of access.

**Horizontal fragmentation types :**

Two main types of horizontal fragmentation exist: primary HF, derived HF.

1. **Primary Horizontal Fragmentation PHF :**(also called mono table HF **[16]** In this type of fragmentation a table is partitioned using the conjunction of selection predicates defined on its own attributes.PHF can be used to optimize selections, especially when the partitioning keys match with the selection attributes. In the context of a data warehouse, the PHF is well suited for fragmenting dimension tables.
2. **Derived Horizontal Fragmentation DHF :**(also called table depend HF **[16]**), The derived horizontal fragmentation is performed with selection predicates defined on another table (i.e. following the primary horizontal fragmentation of another table). For example, a fact table can be fragmented based on the fragmentation schemas of the dimension tables.This fragmentation is possible if there is a foreign key that points from the first table to the second (father-son relationship).**[17].[18]**DHF optimizes selections and joins at the same time. Therefore, it is well suited for star join queries.**[17]. [19]. [20].**

#### 3.1.2. Parallel Processing

Parallelism is one of the most relevant solutions to deal with the large volumes of data managed by the warehouse and complex decision requests. Rather than rely on a single, regular processor, parallel systems use fast, low-cost microprocessors to optimize the flow of a set of potentially interdependent operations in a single query.There are three widely used architectures for processing parallelism: (a) shared memory, (b) shared disk, and (c) not

sharing anything. The difference lies essentially in the sharing of memory. For one or the other of the architectures, one can parallelize queries thanks to the CPU but also thanks to the disks.Once the architecture is chosen, a fragmentation (primary horizontal , derived horizontal or vertical) is necessary to be able to share the data on the different processors. Then, a data allocation process puts the generated fragments on the nodes of the parallel machine. To be able to run on such an architecture, the decision requests are rewritten on the data fragments, and during their execution, the workload balancing on the nodes must be checked. **[21].**

## 3.2. Redundant Optimization Techniques

### **3.2.1.** Vertical fragmentation

Vertical fragmentation is a redundant optimization technique that consists of separating the different attributes or columns of a relation and saving them in separate physical spaces. For this purpose, projections are applied to the relation in order to divide it into several sub-relations each containing a subset of the non-key attributes (columns) as well as the primary key of the initial relationship that allows the reconstruction of the initial relation by join operation.

There many reasons for performing vertical fragmentation, for example:

- Vertical fragmentation speeds up the projection operations by loading only the sub-relations whose attributes appear in the query. Knowing that a query hardly ever needs all the attributes of the tables it is addressed to. It is particularly interesting when applied to fact tables.
- A line with fewer attributes takes up less space therefor updates and query execution are more efficient because the system buffer can hold more rows at a time, and so the I / O count decreases.
- Decomposing a relation vertically also allows a number of transactions to be executed simultaneously. For example, in the same company, the human resources department may keep information such as name, address and rank with the customer number as a key. While the payroll service maintains the customer's bank account number and credit information, with the same customer number as a key.

The main disadvantage of vertical fragmentation is that it requires additional joins when a query accesses multiple fragments**[18].**

### **3.2.2.** Materialized Views

A materialized view is a table containing the results of a query. Views improve the execution of queries by pre-calculating the most expensive operations such as joins and aggregations, by storing their results in the database. As a result, some queries only require access to the materialized views and are thus executed more quickly. However, updating the data always involves the materialized views calculated from these data to maintain the consistency and integrity of the data. This induces a system overhead related to the maintenance cost of the materialized views. In addition, the materialization of views requires additional storage space that the administrator allocates to these views.

However, manipulating materialized views presents two important issues: the selection of materialized views and the maintenance of materialized views.

### **3.2.3.** Indexes

An index is a data structure used and maintained by the database management system (DBMS) to enable it to quickly search the information needed for a query without going through all the data. It allows from an index key to find the physical location of the searched tuples.

A database index works like an index in the back of a book. An index of a book associates the information of interest with a page number. To locate information in a book, it is usually much faster to inspect the index first, determine the page number, and then turn to the desired page. If there were no indexes, we will have to inspect each page of the book to find the information.

For a data warehouse, using an index simplifies and speeds up the search, sort, join, or aggregate operations performed by the DBMS. For example, using indexes can optimize queries that require tuples to be ordered according to one or more attributes such as GROUP BY and ORDER BY clauses. If the join attribute between two or more tables is indexed, it is sufficient to calculate the join using the indexes without accessing the tables, which considerably reduces the number of input-output operations. As with materialized views, at the physical design level, the warehouse administrator must select indexes to speed up query execution. Index selection is difficult because their number is exponential in the total number of attributes in the warehouse. This problem is known as the Index Selection Problem (ISP).**[22].** In our thesis we are interested in this problem and it will be treated in detail in the next chapter.

In the next section, we describe some indexing techniques that have been studied / used for academic / industrial purposes in the context of databases or data warehouses and we show the interest of each indexing technique for data warehouses.

# 4. Indexing Techniques

There are many indexing techniques, each technique is appropriate for a particular situation. We will use the example in Figure 1.5 to explain these indexing techniques throughout this section. The figure illustrates the example of a star schema with a central fact table called SALES and two dimension tables called PRODUCTS and CLIENTS.

**Product Table**

| Product_ID | Wieght | Size | Package_Type |
|---|---|---|---|
| P10 | 10 | 10 | A |
| P11 | 50 | 10 | B |
| P12 | 50 | 10 | A |
| P13 | 50 | 10 | C |
| P14 | 30 | 10 | A |
| P15 | 50 | 10 | B |
| P16 | 50 | 10 | D |
| P17 | 5 | 10 | H |
| P17 | 50 | 10 | I |
| P19 | 50 | 10 | E |
| P20 | 40 | 10 | I |
| P21 | 50 | 10 | F |
| P22 | 50 | 10 | G |
| P23 | 40 | 10 | J |
| P24 | 50 | 10 | G |
| P25 | 10 | 10 | F |

**Client Table**

| Client_ID | Gender | City | Region |
|---|---|---|---|
| C101 | F | Algies | Center |
| C102 | F | Algies | Center |
| C103 | M | Djelfa | Center |
| C104 | M | Algies | Center |
| C105 | F | Batna | East |
| C106 | F | Djelfa | Center |
| C107 | M | Algies | Center |
| C108 | F | Annaba | East |
| C109 | M | Algies | Center |
| C110 | F | Oran | West |

**Sales Table**

| Product_ID | Client_ID | Total_Sales |
|---|---|---|
| P10 | C105 | 100 |
| P11 | C102 | 100 |
| P15 | C105 | 500 |
| P10 | C107 | 10 |
| P10 | C106 | 100 |
| P10 | C101 | 900 |
| P11 | C105 | 100 |
| P10 | C109 | 20 |
| P11 | C109 | 100 |
| P10 | C102 | 400 |
| P13 | C105 | 100 |

**Figure 1. 5:** Example of a star schema with a SALES fact table and two dimension tables PRODUCTS and CLIENTS

## 4.1. B-Tree Index

The B-tree index is the default index for most relational DBMSs. The highest level of the index is called: root. The lowest level is called leaf. All other levels in between are called branches. The root and branches contain entries that point to the next level of the index. The leaf nodes composed of the index key and pointers pointing to the physical location (Row-ID) in which the corresponding records are stored. A B-tree index for package_type of the PRODUCT table is shown in Figure 1.6.

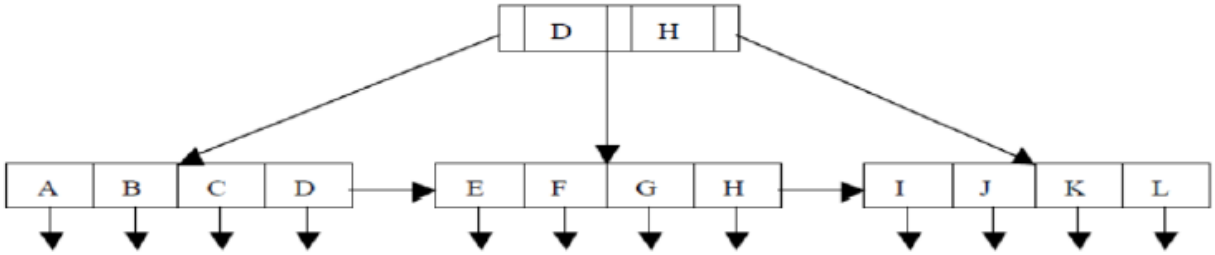


**Figure 1. 6:** B-tree index for package_type of the PRODUCT table **[22]**

The B-tree index is popular in data warehouse applications for the high cardinality column such as names because the use of the index space is independent of the column's cardinality.However, the B-Tree index has features that make it a bad choice for DW queries. First, a B-tree index is of no use for low cardinality data, such as the gender column, because it reduces slightly the I / O count and can use more space than the indexed column. Secondly, each B-tree index is independent and therefore can not work with each other at index level before going to the data source.Finally, the B-tree index retrieves the result data ordered by key values that have unordered Row-IDs and thus, more I / O operations and page faults are generated.

### 4.2. Projection Index

A projection index on an indexed column A in an array T stores all the values of A in the same order they appear in T. Figure 1.7-a shows the projection index on Package_Type of the PRODUCT table.

| Package_Type |
|---|
| A |
| B |
| L |
| C |
| . |
| . |
| G |

**Figure 1. 7:** The projection index on Package_Type of the PRODUCT table **[22]**

Generally(Typically), queries on a data warehouse retrieve a subset of data from the columns of the table; Having a projection index on these columns greatly reduces the cost of polling because a single I / O operation can bring more values into memory.

### 4.3. Bitmap Index

Bitmap indexes are one of the most efficient indexing methods available to speed up multidimensional queries. Queries are evaluated efficiently by logical operations (eg, AND, OR, XOR, NOT operations) which are well supported by the computer hardware. For an attribute with c distinct values, the bitmap index generates c bitmaps with N bits each, where N is the number of records (rows) in the dataset. Each bit in a bitmap is set to "1" if the attribute in the record is of a specific value, otherwise the bit is set to "0". Figure 1.8 shows an example of the Bitmap index on the Package_Type column of the PRODUCT table.

| $B_A$ | $B_B$ | $B_A$ | $B_C$ | $B_D$ | $B_E$ | $B_F$ | $B_G$ | $B_H$ | $B_I$ | $B_J$ | $B_K$ | $B_L$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| . | . | . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . | . | . |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

**Figure 1. 8:** An example of a bitmap index for Package_type from the PRODUCT table **[22]**

To respond to a request (query), the bitmap vectors of the values specified in the condition predicate are read into memory. If there is more than one bitmap vector read, a Boolean operation is performed on them before accessing the data. However, the problem occurs if the bitmap index is built on a high cardinality column, which then requires more space and query processing time.To reduce the size of a binary index, a set of compression techniques has been proposed **[23]**. A variant of the binary index has also been proposed to reduce its size, the encoded binary index **[24]**. In this type of index, each value of the indexed attribute A is encoded using a number of bits. A correspondence table makes it possible to establish the correspondence between the value of A and its encoded representation

### 4.4. Join Index

A join index is a structure that contains pairs of Row-IDs (tuple identifier) of tuples of two relations that we want to make easier to join. We record in a join index the pairs of Row-IDs of the tuples that satisfy a given join criterion. This index can be seen as a pre-calculated join.**[18]**

For OLTP systems that often have joins between two tables, this type of index is desirable. But, these indexes are limited for data warehouses modeled by a star schema because its

queries are characterized by several joins between the fact table and several dimension tables**[19].** To solve this problem, the authors in **[25].** proposed a new index called star join index, adapted to this type of schema. A star join index contains all possible combinations between the identifier of the fact table and the foreign keys of the dimension tables.

The index is implemented using one of two representations: Row-id or bitmap, depending on the cardinality of the indexed column. A Row-id representation is used for data with high cardinality. While a bitmap representation, which is called a binary join index (BJI), is used with low cardinality data.

### 4.5. Binary Join Index

A Binary Join Index is a bitmap index described via a join request(query). It is defined on a base table and stores the row identifiers of the base table with the indexed columns for the joined tables.

For example, the binary join index on the "gender" column can be constructed using the gender column in the CLIENT table and the foreign-client ID-key in the SALES table. Note that the SALES table does not contain the gender column. The binary join index on the gender column is created by assigning a 1 to the bit corresponding to a client ID whose gender is "M" in the SALES table. Otherwise, the bit is set to 0 as shown in Figure 1.9.

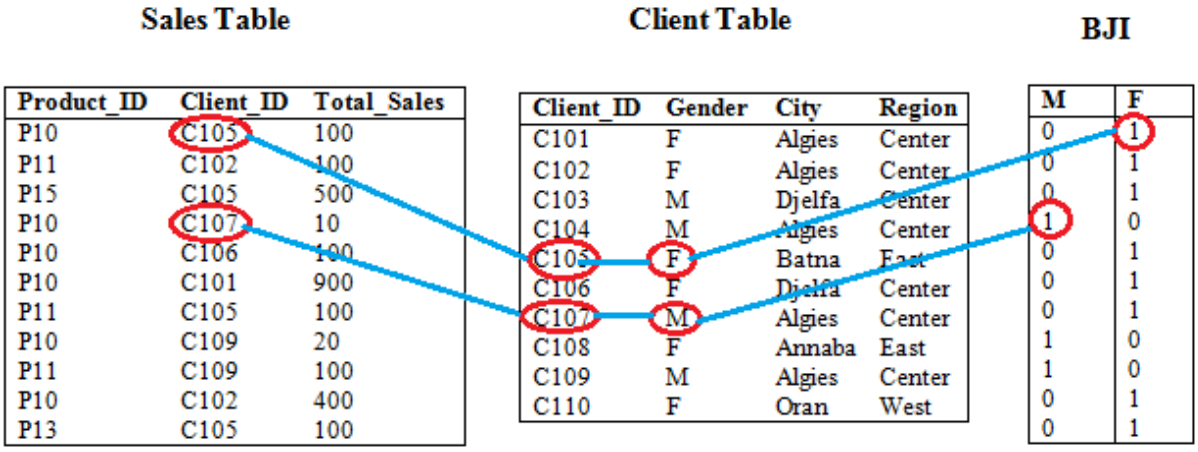


**Figure 1. 9:** Example of a Binary Join Index on the Gender column of the CUSTOMER table

Thus, the join bitmap is constructed by joining foreign keys and can span any field. Bitmap indexes are also an interesting structure for optimizing data searches. Binary join indexes allow to optimize both star joins and selection operations defined on dimension tables.

Example: To illustrate the use and interest of bit join indexes, consider the following Q query that the administrator wants to optimize:

```
SELECT Count(*)
FROM CUSTOMERS C, SALES S
WHERE C.Gender =’M’ AND C.CID=S.CID
```

To do this, the administrator creates a binary join index between the SALES fact table and the CLIENT dimension table based on the gender attribute as follows:

```
CREATE BITMAP INDEX SC_IDX
ON SALES(CUSTOMER.Gender)
FROM SALES S, CUSTOMERS C
WHERE S.CID= C.CID
```

To run the above query, the query optimizer accesses only the bitmap with the male bit = '1' and counts the number of occurrences, without accessing and joining the SALES and CUSTOMERS tables.

## 5. Conclusion

Data warehouses have now become not a fad but a vital tool for the smooth running of the organization. They are the basis of any strategy and decision making of the company. It is necessary to extract the necessary information for the decision-making and also their structure. From this warehouse multidimensional databases are extracted, because they make it possible to look at the organization under different angles or dimensions, the latter consist only of data specific to the decision.

In this chapter, we have presented the main techniques used to optimize query execution time in data warehouses. These are fragmentation, materialized views, and index creation.

Conventional indexing techniques (B-trees, binary indexes, projection indexes, etc.) have shown their performance in traditional databases, but they are not suitable for the context of modeled relational data warehouses according to a star pattern. This is due to queries executed on this type of schema (so-called star join queries), because of expensive join operations, especially when they are executed on a large volume of data, as in the case of warehouses. Binary join indexes make it possible to reduce the complexity of these queries by pre-calculating the joins contained therein. However, choosing an index configuration is a difficult task because of the large number of attributes and tables that can be indexed. This problem is known as the index selection problem and is known NP-Complete.

For this, we will review in the next chapter the problem of selecting indexes in data warehouses. Then we present the main index selection work. To finally analyze these works and introduce our index selection technique.

# CHAPTER 2
# ISP INDEX SELECTION PROBLEM

## 1. Introduction

Faced with the complexity of processing data warehouses or increasingly large databases, the warehouse or databases administrator has the difficult task of choosing a policy for optimizing queries and operations performed on the data. This optimization is done during the physical design phase and must be as optimal as possible. It requires choosing the optimization techniques (index, fragmentation, views, etc.) to implement. In addition, the physical architecture of the data must be able to support the addition of new data, maintenance operations must not alter the execution of processing requests(queries) and the storage space must be optimized. Thus, several works are interested in the selection of optimization techniques in order to choose the most appropriate techniques, which can optimize queries and reduce the cost of access to data.

Among the indexing techniques, the BJI **[26]** binary join indexes are particularly suited to the context of data warehouses because they not only make the logical (And, Or, Not...) and counting operations on bitmaps efficient, but they also make it possible to -calculate the joins at the time of the creation of the indexes and they are thus not calculated during the execution of the requests(queries). In addition, the space required for storing bitmaps is reduced because of their binary representation and the possible compression especially when the indexed attributes are of low and medium cardinality, which is generally the case in the dimensions of a warehouse **[27].**

Selecting a set of indexes that reduce the execution time of warehouse-defined queries is a difficult task because of the large number of attributes and tables that can be indexed. In addition, the indexes can be defined on several attributes coming from several tables. The problem of index selection is known NP-Complete **[28][29][30]** as a result , there are no algorithms that offer an optimal solution in a finite time. Several research works rather propose solutions close to the optimal solution by using greedy algorithms, heuristics and Datamining techniques in order to reduce the complexity of the problem.

In this chapter, we will present a state of the art on the problem of selection of BJI in the context of data warehouses as well as a summary of the main works that have dealt with this problem. We finish with an analysis of these works, which will allow us to identify their main

locks. These will be our motives for implementing a new approach to BJI selection in data warehouses.

## 2. BinaryJoin Indexes

The Binary Join Index (BJI) is used to precompute joins between one or more dimension tables and the fact table in data warehouses modeled by a star schema**[31]** A BJI is a bitmap index described via a join request(Query).It is an indexing structure that covers multiple tables and increases the performance of the joins of these tables.In the context of data warehouses, a BJI allows you to pre-calculate the join between the fact table and one or more dimension tables, which optimizes both the star joins and the selections operations defined on the dimension tables. Most commercial DBMS support BJI.**[22]**

A BJI is defined on one or more non-key attributes of the low and medium cardinality dimension tables, called indexable attributes. An attribute is indexable if it is used in a decision request (query) by a selection predicate. Given the large number ofindexable attributes and given the number of dimension tables and the number of non-key attributes of each table, the selection of a BJI configuration is usually a difficult task (algorithmic complexity) compared to other types of indexes**[13].**

### 2.1. Constructing a BJI

To show how a BJI is constructed, suppose an attribute *A* having *n* distinct values $v_1, v_2, ..., v_n$ belonging to a dimension table *D*. Assume that the fact table *F* is composed of m instances. The construction of the binary join index defined on attribute *A* is as follows:

1. Create *n* vectors each composed of *m* entries;
2. The *i-th* bit of the vector corresponding to a value $v_k$ is set to 1 if the tuple of rank *i* of the fact table is joined with a tuple of the table of dimension *D* such that the value of *A* of this tuple is equal to $v_k$. It is set to 0 in the opposite case.

Note that the size of a BJI is proportional to the cardinality of the indexed attributes (number of distinct values). Because of this, they are often recommended for low cardinality attributes. To reduce the size of a binary index, several compression techniques have been proposed **[32]** .

## 2.2. Automatic Index Selectionproblem

The index selection problem consists in selecting an index configuration minimizing the cost of executing a set of decisional requests (queries, called the request load or query load ) that is supposed to be representative of all the requests addressed to the system. This selection can be made under certain constraints, such as the storage space allocated to the indexes to be selected. **[33]**

The problem is formalized in the following way **[48]**:

Given :

1. A relational data warehouse modeled by a star schema having a set of tables of dimension $D = \{D_1,\dots ,D_d\}$ and a table of facts *F*.
2. A representative set of decisional queries $Q = \{Q_1,\dots ,Q_m\}$ with their frequencies $Freq = \{Freq_1,\dots , Freq_m\}$.
3. Storage space *S* allocated by the administrator to the indexes to be selected.

So, we have to find a *CI* index configuration such as:

1. The cost of running queries using this *COST (Q / CI)* configuration is minimal.
2. 2. The space allocated for the storage of these indexes does not exceed S:

$$\sum_{I\in CI} Size(I) \leq S \quad (1)$$

The cost of executing requests load( query load) *Q* is expressed in the number of input outputs necessary to execute it.

It is calculated by the formula:

$$COST\ (Q)=\sum_{i=1}^{m} COST(Q_i) \times Freq_i \quad (2)$$

## 2.3. Complexity

Let $A= A_1, A_2 ,\dots, A_n$ be a set of candidate indexable attributes for the selection of an BJI configuration. The number of possible BJIs that we must consider to select a single BJI is given by the formula **[13].**

$$BJI_{simple} = \sum_{k=1}^{n} \binom{n}{k} = 2^n - 1 \quad (3)$$

To choose more than one index, the total number of possible configurations of **[13]**

$$BJI_{multi} = \sum_{k=1}^{2^{n-1}} \binom{2^n - 1}{k} = 2^{n-1} - 1 \quad (4)$$

Thus, if the number of indexable attributes is equal to *5 (n = 5)*, then the number of possible BJIs is equal to $2^{31}$ - 1 = 2.147.483.647 .

# 3. State of the art

In the following we will discuss the state of the art in index selection work. We begin by presenting the general approach that most of the work follows to address this problem, and we briefly mention some of the work done in the context of databases. Then with more details, we present some works closely related to our context.

## 3.1. General approach to index selection in DWs

Generally, the algorithms proposed for index selection comprise two steps: (1) Selection of the set of candidate indexes, (2) Construction of an index configuration. **[22].**

### 3.1.1. Candidate index set selection

In this first step, a set of candidate indexes is built from a load of queries representing queries to the DBMS over a period of time. This charge is extracted either directly from the DBMS transaction log or through an external application **[22]**

The selection of candidate attributes for indexing can be done in two ways, *manually or automatically*:

1. Manually by the administrator according to his expertise **[34]** , the choice is then subjective because it depends on the degree of expertise of the administrator, but it becomes quite difficult to achieve when the number of requests in the charge of queries is very big.
2. The identification of the candidate attributes can be done automatically by using a query analyzer, in order to identify the interesting attributes to index, which are mainly present in the clauses WHERE, GROUP BY, ORDER BY, HAVING and UPDATE of the SQL queries **[27][35][36].**

Most index selection jobs use automatic selection of candidate attributes, the goal being to facilitate the manual tasks of the system administrator. **[33][27][35][36][37].**

**Reducing the number of candidate indexes**

The main difficulty in index selection is the large number of dimension attributes that can participate in the selection process (several tens or hundreds of attributes). It is therefore important to reduce the number of candidate indexes to reduce the search space of the algorithm that selects the final index configuration. To reduce the number of candidate indexes, several methods are used **[13]**:

- Estimation of the contribution of each index in the optimization of the load and remove indexes that do not bring a great benefit **[37]**

- Removing indexes that cause the total space reserved for storing indexes to be exceeded. **[38]**
- Removing indexes that cannot be created or whose creation clause has an anomaly **[39].**
- Using Datamining Techniques: The main idea of these approaches is to compute the set of frequent groups of indexable attributes rather than all possible combinations. The groups generated in this step are then used in the second pruning step to select the final index configuration. **[11][40][39][41][42].**

3.1.2. Building a final index configuration

The candidate index configuration obtained may not respect the storage constraint. Also, the number of candidate indexes can be significant and some DBMS allow only a predefined index number per table **[43]**.Thus we must reduce the number of indexes to generate.

The methods used to build a final index configuration from candidate indexes can be classified into:

1. Simple algorithmic methods (greedy) ascending or descending.
2. Heuristicmethods.

**A- Greedymethods**

These methods can be classified into two categories: ascendant and descendant .

1. Ascending greedy methods start from an empty set of candidate indexes and progressively add indexes that minimize the cost of the load. This process stops when the cost stops decreasing or the storage constraint is violated.
2. On the contrary, the gluttonous descending methods consider all candidate indexes as a starting point. Then, at each iteration, the indexes are pruned until the cost of the load no longer decreases. It should be noted that most of the selection work of a final index configuration uses ascending construction**[11][35][36][44][39]**

**B- Heuristicmethods**

The principle of these approaches is to find the best subset of candidate indexes that provides the most benefit for the query load and respects the storage constraint. The problem is seen as an optimization problem where one seeks to minimize an objective function computed by a cost model that estimates the cost of executing the query load if the current set of indexes is implemented. **[22]**

1. *Methods assimilating the problem of selection to the knapsack problem optimization :* The index selection problem has been equated in some works **[35][37][45][38]**with the knapsack problem . The indexes are assimilated to objects

that may or may not be put in the bag, the size of the indexes corresponds to the weight of the objects. The improvement in the estimated execution time that an index contributes to all the requests that exploit corresponds to the benefit of the corresponding object. The disk space allocated to the storage of the final index configuration corresponds to the size of the backpack.

2. *Methods derived from genetic algorithms :* The principle of these methods is to imitate natural selection to produce a population (index configuration) with better adaptability (convergence towards the optimum). **[46]**

#### 3.1.3. Costmodels

To evaluate the cost of an index and eliminate the least advantageous indexes (having a higher cost), two methods are used to estimate the cost of an index configuration.

1. The first **[11][17][33][40]**is to implement a mathematical cost model to estimate the cost of using an index. The advantages of this method are its speed and it is completely independent of the DBMS, but the simplicity of these models that do not accurately reflect the reality of query execution is its great disadvantage.
2. The second **[35][44]**is to use the DBMS Query Optimizer to estimate this cost based on actual estimates of table sizes, loaded data, tuples, and so on. It is therefore more reliable than thefirst, but its main disadvantage is that it makes the selection technique dependent on a given DBMS. In addition, it is greedy in computing time due to frequent calls from the optimizer.**[27]**

### 3.2. Index selection works in databases

The problem of index selection was initially studied in the context of databases. The first works that dealt with this problem date back to the 1970s and it was demonstrated NP-Complete in 1978 **[30]**. As a result, this work has proposed solutions to find a set of indexes close to optimal without evaluating the cost of all these sets.

#### 3.2.1. Chaudhuri et al. 1997

As part of the AutoAdmin project launched by Microsoft for the self-administration of databases. The authors in **[35]** have proposed a single index and multi-attribute Index Selection Tool (IST) that takes a query load and a storage constraint as input and it proceeds in four steps:

1. As a first step, a single-attribute index configuration is created from the load.
2. Then, a greedy algorithm will use the estimates of the cost obtained by the query optimizer to choose the best indexes of this configuration.

3. Indexes on two attributes are constructed from single-attribute indexes using the same approach as step 2. The process is reiterated to create indexes of increasingly larger size.
4. The last step built is to enumerate the configurations and choose the k best indexes among the n chosen in the previous step using a greedy algorithm and using the optimizer.

The authors extended the query optimizer of the Microsoft SQL server DBMS with a module called "what-if" to simulate the presence of indexes that do not exist and also extended the cost optimizer module, such as so that, given a Q query and a C configuration, the cost of Q when the physical model is C, can be calculated.

The explosion of the number of indexes generated in step 3 and the frequent calls of the optimizer are the main drawbacks of this approach **[22]**

3.2.2. Valentin et al. 2000

The authors in **[35]** propose an extension of the IBM DB2 Advisor Query Optimizer. The proposed approach takes as input a query load and proceeds in three steps:

1. The first step is to generate so-called virtual indexes which constitute a set of candidate indexes for each query.
2. The optimizer then generates the optimal execution plan for each query and recommends an index configuration.
3. 3. In the last step, the problem is modeled as a knapsack problem and solved by a heuristic that allows indexes to be selected in descending order of the gain / storage ratio.

3.2.3. Feldman et al. 2003

The authors in **[38]**propose an index selection tool named DINNER that first uses a knowledge base to build its cost model. Then, it proceeds to the representation of the requests of a load in the form of a graph in which each path represents a plan of execution of a request by using or not indexes. A final step is to use heuristics to eliminate and unify unnecessary nodes and prune bad solutions.

3.2.4. Kratica et al. 2003

The authors in **[46]** propose a genetic algorithm in which the query processing time is considered as objective function. An initial population of individuals is constructed from the candidate indexes (each index is assimilated to an individual). To choose the set of individuals that will survive the next generation, a configuration consisting of the best individuals (indeindex) is constructed by crossbreeding, mutation and genetic selection.

### 3.3. Index selection works in DW

In this section, we present the main work on the problem of index selection in the context of data warehouses, classified into two main categories and at last we will discuss the linear approach that was proposed in 2015 doesn't belong to any of the previous categories**[22]**:

a) Heuristic-based approaches.
b) Datamining-based approaches.

**A. Heuristic-basedapproaches**

3.3.1. Golfareli et al. 2002

In their work, Golfareli and al **[36]**propose a heuristic approach for the selection of indexes of type list of values and the bitmap indexes in relational data warehouses implemented in star schema.

In order to determine a set of indexes that minimize the cost of running the query load within the constraint of storage space, the authors define an optimizer model that creates an execution plan for each query and a cost model to compare different solutions. A set of potentially useful candidate indexes is determined and then a greedy algorithm gradually selects from them the most beneficial indexes taking into account the constraint of the storage space.

The algorithm proposed by Golfarelli et al. proceeds in three distinct steps :

1. The first is to initialize:

- *The set of candidate indexes C:* for each indexable attribute, the most beneficial index (according to its calculated cost) is added to C.
- *Set of optimal indexes O:* For each dimension table, a list of values index built on its primary key is inserted in O.
- *The storage space S* needed to store the indexes to select.

2. The second step is to pick from C in a gluttonous way the indexes bringing the best benefit and add them to O. If after an addition, it happens that all the attributes composing the primary key of a table of facts are indexed , then these indexes are transformed into a single multi-attribute index built on the primary key of the fact table.
3. In the third step, primary indexes for the remaining fact tables are chosen.

3.3.2. Boukhalfa et al. 2010

In **[47]**, the authors propose an algorithmic approach for the selection of single-attribute and multi-attribute BJIs based on an attribute elimination heuristic based on criteria such as frequency, cardinality of attributes, etc.

Two algorithms have been proposed for the selection of BJIs, the first for the selection of mono-attributes BJI and the second for multi-attributes BJI. A cost model based on the one proposed in **[11]**for estimating the size of BJIs and the cost of executing queries in the presence of a BJI configuration is used to evaluate the quality of the resulting index configurations. . Figure 2.1 represents the general architecture of the proposed approach.

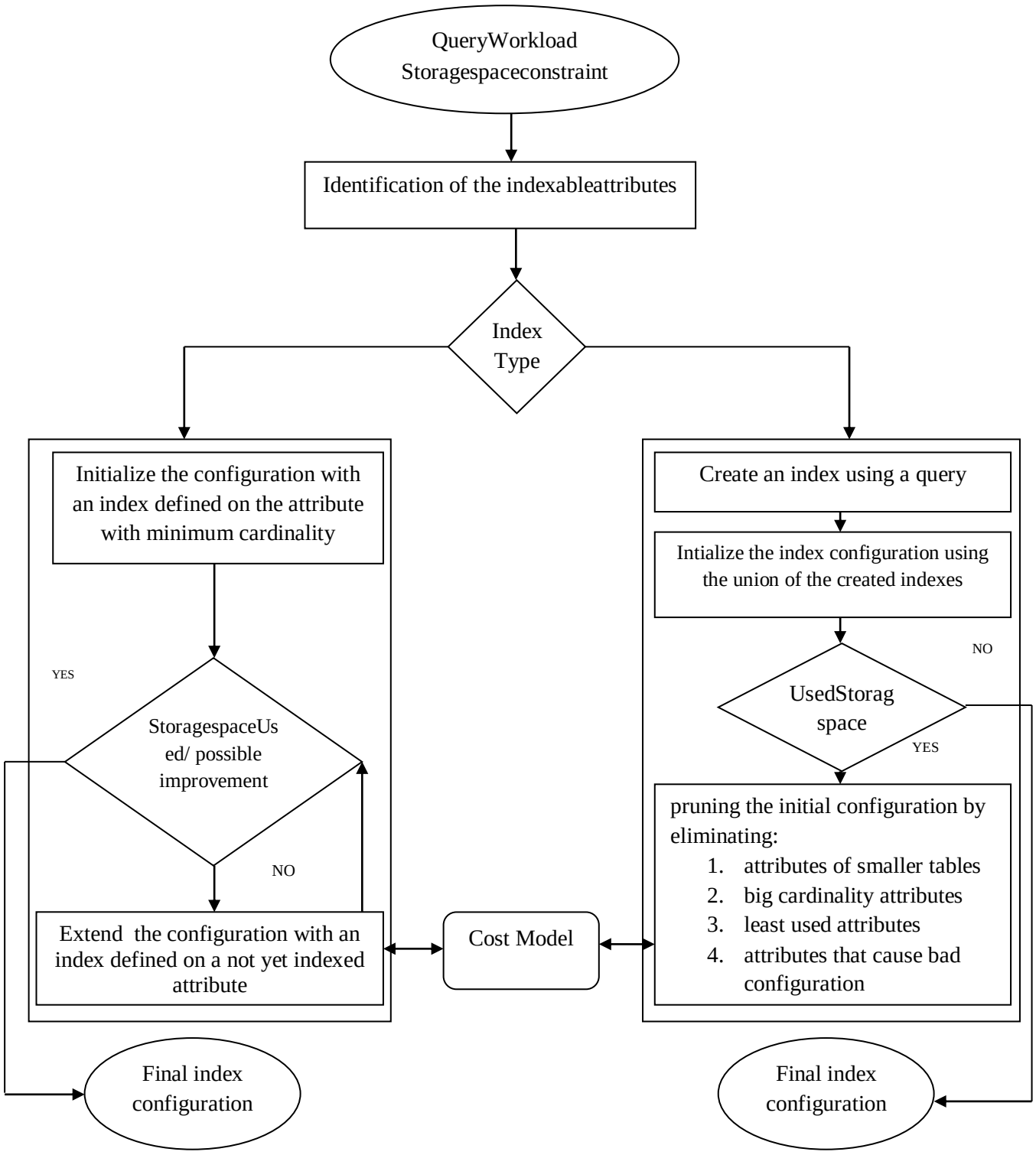


**Figure 2. 1:** The general architecture of Boukhalfa&al's approach **[22]**

**Selecting a mono-attribute index configuration Algorithm :**

Selecting a mono-attribute index configuration takes place in three steps:

1. Identification of the low and medium cardinality attributes contained in the WHERE clauses of the query load.
2. An initial configuration is built with a mono-attribute index defined on the attribute having a minimum cardinality.
3. The initial configuration is extended by the addition of new indexes defined on other attributes not yet indexed until no improvement is possible and / or the storage space is consumed.

**Selecting a multi-attribute index configuration Algorithm :**

Selecting a multi-attribute index configuration requires four steps:

1. *Identifying indexable attributes:* this is the same step as selecting a single-attribute index configuration
2. *The construction of a configuration by request:* For each query, an index allowing to pre-calculate all the joins of the query is selected. This is the ideal scenario to significantly reduce the cost of running the query but may violate the storage constraint.
3. *The construction of an initial configuration*: This initial configuration is built by performing the union of the indexes obtained from the previous step and by deleting the indexes shared by several queries.
4. The construction of a final configuration: if the initial configuration respects the storage constraint, it is retained, otherwise some attributes must be eliminated. To choose the attributes to eliminate the authors considered the following four strategies:
   a- *Elimination of strong cardinality attributes:* In this strategy, the strong cardinality attribute is removed from all indexes constituting the initial configuration until the storage constraint is respected.
   b- *Eliminate Attributes from Smaller Tables:* The joins between the fact table and the largest dimension tables are the most expensive and therefore have higher priority. For this, the attributes belonging to the smaller tables are eliminated starting with those having a strong cardinality.

c- *Eliminate Less Used Attributes:* This strategy assumes that it is a good idea to create indexes on the most used attributes to satisfy more queries. As a result, the least used attributes are eliminated first.

d- *Eliminate attributes with less cost reduction:* This strategy uses a cost model to eliminate attributes that cause bad configurations. For that, for each candidate attribute, it is eliminated, one calculates the cost of the load after elimination to keep in the end those generating a configuration bringing the best reduction of cost.

3.3.3. Toumi et al 2013

In this work the authors propose a new approach based on Binary Particle Swarm Optimization (BPSO) for solving the BJISP in data warehouses. This approach selects the optimal setof bitmap join indexes based on a mathematical cost model.

the general schema of the bitmap join indexes selection process is based on BPSO and illustrated in Fig 2.2. First, the queries workload Q is parsed and a set of candidate attributes A is generated. The BPSO algorithm then takes the set A along with the storage constraint S and the cost models as input and undergoes the selection process. The process continues until a feasible solution is found, or the generation limit is exceeded (if the storage constraint is violated, then the solution is marked as infeasible). A feasible solution, if found, is the final configuration of SBJI indexes. **[48]**

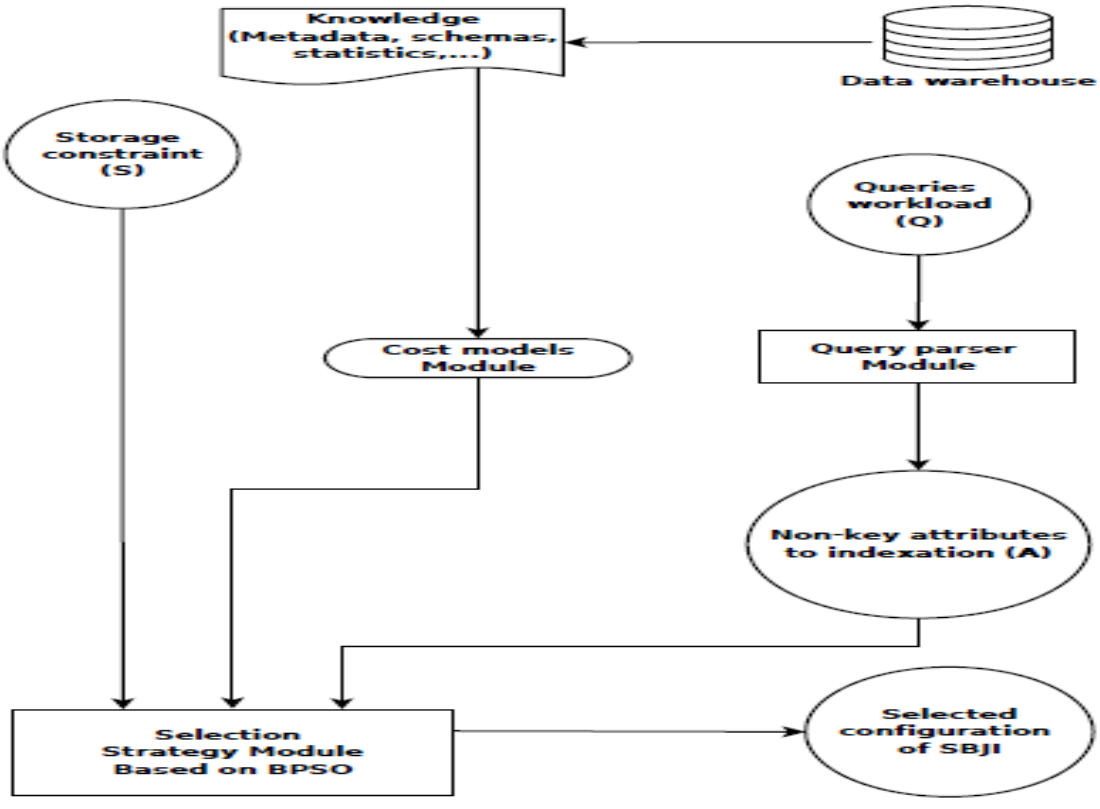


**Figure 2. 2:** The BPSO based approach proposed by LyazidToumi for SBJISP **[48]**

3.3.4. LyazidToumi et al 2015

In their work, the authors proposed a mixed-integer linear programming *(MILP)*formulation of the SBJISP (Single Binary Join Index Selection Problem) to obtain an optimal solution. The approach is different from the stochastic and statistical approaches that exist to solve the problem.An internal bitmap is utilized for accurately incorporating the cost of joins involved into the model. The *MILP* can be solved using a commercial *MILP* solver (such as CPLEX 10.1). This approach proved it's effectiveness by outperforming the *BPSO* ,*GAI* and *DM* approaches

**B. Datamining-basedapproaches**

3.3.5. Aouiche2005

In **[33]**the author assumes that an index is considered interesting, if it is built on an attribute or a group of attributes most frequently used in the set of queries workload. To highlight this correlation, the author uses a Datamining technique to generate the most frequent set of attributes (single-attribute index) or attribute group (multi-attribute index). therefore more relevant instead of calculating all possible index combinations. The set of candidate indexes generated is then used by a greedy algorithm for the selection of final indexes through a cost model.

To further reduce the number of candidate indexes that may be huge ($2^k$, where k is the number of attributes), the author has resorted to closed frequent patterns that represent the maximum set of items common to a set of patterns and therefore they are much less numerous. The algorithm used to extract closed patterns is Close **[49]**

The automatic selection procedure of Binary Join Index proposed by the author and whose main steps are represented in figure 2.2, takes place in five steps:

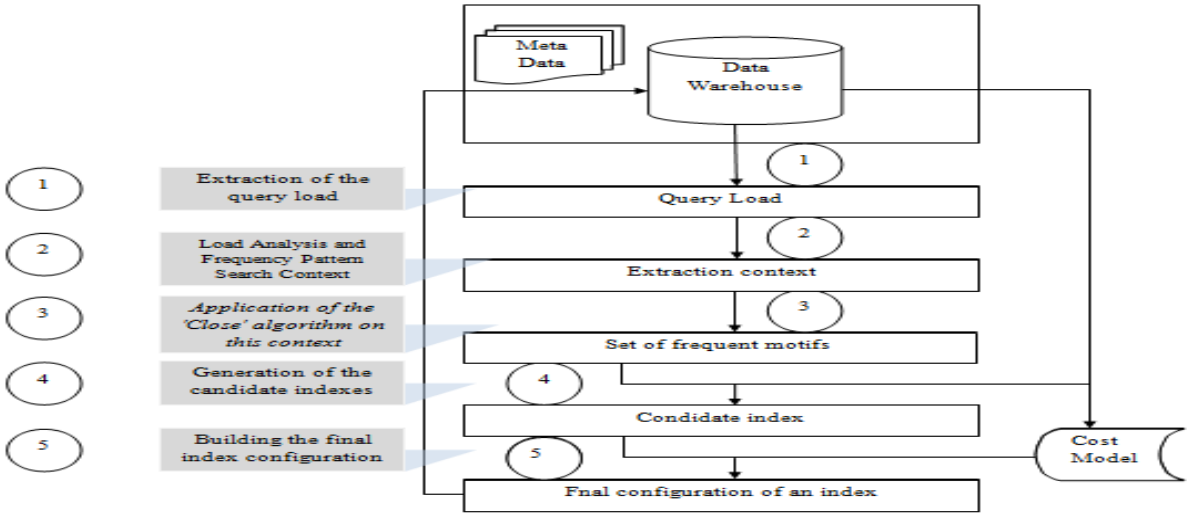


**Figure 2. 3:** Main steps of Aouiche's approach **[22]**

1. *Extraction of the query load:* During this step, a query load represented by a set of decision requests sent to the DBMS is extracted from the transaction log.
2. *Load Analysis and Frequency Pattern Search Context Building:* In this step, a parser processes the query load to retrieve a set of candidate attributes from those found in the "WHERE" clauses of the query requests. . These attributes are used to build a query-attribute matrix where the rows represent the load requests and the columns represent the candidate attributes.
3. *Application of the 'Close' algorithm on this context:* The result matrix of the previous step is exploited by the Close algorithm and gives rise to a set of closed frequent motifs. Each extracted pattern is composed of a set of attributes of the data warehouse whose support is greater than or equal to a threshold chosen by the author.
4. *Generation of the candidate indexes:* During this step and using the data warehouse schema (foreign keys of the fact table, primary keys of the dimension tables, etc.) each pattern extracted in the previous step will be analyzed. to check whether it can generate a candidate or non-candidate binary join index and a set of candidate indexes is created.
5. *Building the final index configuration:*An ascending pruning approach based on greedy search in all candidate indexes is chosen by the author for the construction of the final index configuration.

A mathematical cost model that calculates the size of the index as well as the cost of running the query load (cost of accessing data through this index) is used to define three objective functions: 1) Profit, 2 ) Ration profit / space and 3) Hybrid.

- *The objective function Profit:* favors the indexes that bring the most profit when executing queries. It isusedwhenstoragespaceis not limited.
- *The objective function Profit Ratio / space*: favors indexes bringing the most profit while occupying a minimum of storage space. It isusefulwhenstoragespaceislow.
- *The Hybrid objective function:* combines the first two, to first select the indexes that bring the most profit and then keep only those that occupy the least amount of storage space when it becomes weak. It isinterestingwhenthisspaceismoderately large

This approach is among the few works that have addressed the problem of selecting IJBs in the context of data warehouses modeled by a star schema **[13]**Its modular architecture allows it to be easily adaptable to other indexing techniques (just adapt the index generation module and apply the cost models corresponding to these indexes.). It draws strength from the capabilities of data mining techniques to extract useful knowledge of data and has the

advantage of generating mono and multi-attribute indexes in a single pass. It therefore avoids executing several iterations to generate multi-attribute indexes as in **[37]**. However, relying on attribute frequency as the only parameter to choose the most advantageous indexes has been shown to be insufficient in **[11]**

3.3.6. Bellatreche et al. 2007

The authors of **[11]**have based on the work that has been done on vertical fragmentation **[50]**where it has been shown that the single frequency parameter is not sufficient to have a good fragmentation pattern. They found that in Aouiche's approach we can eliminate indexes built on attributes that are not frequently used but that belong to large dimension tables, or that the cost of join operations strongly depends on the size of the joined tables.

These findings allowed the authors to judge that the frequency of using an index by a maximum of queries as the only parameter to generate frequent closed patterns would not be effective for selecting a set of candidate indexes. The authors showed the limit of Aouiche's approach through an example (see **[11]**) and proposed to include other parameters in the generation of frequent patterns such as the size of dimension tables, the length of tuples, the size of the system page, etc.

The proposed approach proceeds in two main steps: (1) generation of the candidate indexes and (2) selection of the final index configuration.

**Generation of candidate indexes :**

As with the Aouiche approach, Bellatrche et al's approach begins with the construction of the search context for frequent patterns using the candidate attributes. But to generate the closed frequent motifs, the authors use two adaptations of two closed frequent motif search algorithms: Close and Charm called *DynaClose* and *DynaCharm* which instead of using the frequency parameter as the only pruning parameter, they use a finesse function called Fitness sets on the non-key attributes of the patterns as follows:

$$Fintness(X) = \frac{1}{n} \times (\sum_{i=1}^{n} \alpha_i \times sup_i) \qquad (5)$$

Where: $n$ represents the number of non-key attributes $A_i$in the $X$ pattern. $sup_i$represents the support of $A_i$ and $\alpha_i$ is a parameter penalizing the attributes belonging to smaller dimension tables and defined by the following equation:

$$\alpha_i = \frac{|D_i|}{|F|} \quad (6)$$

Where: $|F|$,$|D_i|$ represent respectively the number of system pages necessary for the storage of the table of Facts $F$ and the dimension table $D_i$ which contains$A_i$. For pruning the index

search space, a pattern *X* is retained when its support is greater than or equal to a minfit threshold calculated using a given minimum support minsup:

$$minfit = \frac{minsup}{|F|} \times \sum_{j=1}^{d} \frac{|D_j|}{d} \quad (7)$$

Where *d* is the number of dimension tables. The frequent closed patterns generated by the *DynaClose* and *DynaCharm* algorithms are then purified to avoid non-coherent indexes.

**Selection of the final index configuration:**

Once the patterns are recovered and purified, a greedy algorithm based on a mathematical cost model adapted from that of Aouiche**[33]**is used for the creation of the final index configuration under constraint of the available storage space. The algorithm begins with a configuration with an index set to a lower cardinality attribute, recalculates the cost, and iteratively improves the initial configuration by considering other attributes until no further reduction in total processing cost requests are not possible or the storage constraint is violated.

3.3.7. Ziani et al 2010

The authors **[41]**propose to exploit the maximal frequent patterns that are less numerous than the frequently closed patterns in order to reduce the search space of the candidate indexes.

Therefor , the extraction context built from query load is often dense because of the correlation of the queries due to the correlation of interrogated data **[11]**. In this type of context, the generation of all frequently closed patterns is not trivial and suffers from the same problems as the frequent patterns. A search algorithm based on the search for closed patterns may have a number of closed patterns exponentially larger than the set of maximal generated patterns.

By definition, a Maximum Frequency Pattern is a pattern that all supersets are infrequent. It is therefore clear that the set of maximal frequent motifs is included in all the frequent motifs. This particularity will make it possible to generate fewer and more relevant indexes **[41]**Another peculiarity of the maximal frequent motifs is that the set of frequent motifs can be generated directly from the set of maximal frequent motifs. It thus preserves all the frequent attributes (candidates for indexing).

Several algorithms have been proposed for the search of maximal frequent motives. In their work, the authors chose to use their own Java implementation of the FP-Max algorithm **[51]** . The latter is presented in **[52]** as being the best performer.

### 3.3.8. YoucefAriouat et al 2013

In this approach the authors, **[22]**addressed the bitmap join indexes selection problem using a constraint-based itemset mining Approach (constraineditemset extraction).Unlike previous approaches, the selection is performed in one step by introducing constraints in the selection process.They constrained the input data earlier in the selection process thereby reducing the output size to a set of indexes that are of interest for the administrator. For example, to select a set of indexes, the administrator may put limits on the number of attributes or the cardinality of the attributes to be included in the indexes configuration he is seeking.Its effectiveness has been shown thatthe index configuration generated in one step allowed a significant improvement of the execution time of the workload

## 3.4. Summary

The indexation is an important method in the databases/data warehouses physical design. The indexation is largely used to optimize OLAP queries in the data warehouse environment. We have presented previously the well-known works to solve indexes selection problem on both databases and data warehouse. The proposed approaches start by common step that is analyzing the query workload using automatically using a parser or manually by the DBA. To select the optimal/sub-optimal configuration a greedy algorithm or a genetic algorithm based on a mathematical cost model or on DBMS optimizer are used.

To reduce the size of the problem, Aouiche et al proposed a data mining technique(mining frequent patterns) to prune the research space, this approach is improved by Bellatreche et al to take in consideration the cost features. The integer linear programming (ILP) has been used to solve the ISP in classical databases and found to be effective for obtaining higher quality solutions. Also the MILP has prooved the same and even that it is more effective in DW and DB.

The Table 2.1 summarize the well-known approaches to solve the indexes selection problem and the BJI selection problem in both databases and data warehouses.

| Standard/Cretiria ——— Works | Context | Selected index type | Candidate index selection | Construction of the final configuration | Cost model |
|---|---|---|---|---|---|
| Chaudhuriet al. 1997 | DB | Generic on a table | Automatic | Greedy (Ascndant) | Optimizer+ What-if model |
| Valentin et al. 2000 | DB | Generic on a table | Automatic | Heuristic (Knapsack) | Optimizer |
| Feldman et al. 2003 | DB | Generic on a table | Automatic | | Mathematic |
| Kratica et al. 2003 | DB | Generic on a table | | Heuristic (Genetic) | Mathematic |
| Golfareli et al. 2002 | DW | list of values and bitmap | Automatic | Greedy (Ascendant) | Optimizer |
| Boukhalfa et al. 2010 | DW | BJI | Automatic (Index per query) | Heuristic (Attributeelimination) | Mathematic |
| Aouiche et al 2005 | DW | BJI | Automaticwith datamining | Greedy(Ascendant) | Mathematic |
| Bellatreche et al. 2007 | DW | BJI | Automaticwith datamining | Greedy (Ascendant) | Mathematic |
| Ziani et al. 2010 | DW | BJI | Automatic with datamining | Datamining | Mathematic |
| Toumi et al 2013 | DW | BJI | Automatic | Greedy | Mathematic |
| Ariouat et al 2013 | DW | MBJI | Automatic/ Manual | Datamining with constrains | Mathematic |
| Toumi et al 2015 | DW | SBJI | Automatic | MILP | Mathematic |

**Table 2. 1 :** Summary of the well-known approaches for the ISP resolution

## 4. Conclusion

In this chapter, we have presented the indexation techniques used in both classical databases and data warehouses in detail (for each indexation type, we have presented general information, the objectives and usage). We have also presented the background on how to select indexes in classical databases (ISP). Finally, we have presented the ISP in data warehouses in detail and reviewed the existing approaches to solve the problem. In the next chapter we will present our proposed approach based on Artificial Fish Swarm Algorithm AFSA for resolving the ISP.

# CHAPTER 3

# MibAFSA : Modified improved binary Artificial Fish Swarm Algorithm for ISP resolving

## 1. Introduction

In this chapter, We propose a new approach based on Binary Artificial Fish School Algorithm (bAFSA) using a mathematical cost models for solving the BJISP. This approach is named MIbAFSA (Modified Improved Binary Artificial Fish School Algorithm) that was originally proposed in **[54]**named IbAFSA(improved binary artificial fish school algorithm) to solve the multi-dimensional 0-1 knapsack problem and we adapted it to suit our ISP.

In the following we will present the AFSA, its variations,AFSA for global optimisation and the proposedMIbAFSA for solving the ISP and it's structure with some pseudocodes.

## 2. Artificial Fish School (swarm) Algorithm/ Fish School Search

Artificial Fish School Algorithm (AFSA) is a bio-inspired algorithm used for global optimization problem. Proposed by Bastos Filho and Lima Neto in 2007 is, in its basic version,**[55]**an unimodal optimization algorithm inspired on the collective behavior of fish schools. The mechanisms of feeding and coordinated movement were used as inspiration to create the search operators. The core idea is to make the fishes "swim" toward the positive gradient in order to "eat" and "gain weight". Collectively, the heavier fishes are more influent in the search process as a whole, what makes the barycenter of the fish school moves toward better places in the search space over the iterations.**[56]**

The AFSA uses the following principles**:[57]**

1. Simple computations in all individuals (i.e. fish)
2. Various means of storing information (i.e. weights of fish and school barycenter)
3. Local computations (i.e. swimming is composed of distinct components)
4. Low communications between neighboring individuals (i.e. fish are to think local but also be socially aware)
5. Minimum centralized control (mainly for self-controlling of the school radius)
6. Some distinct diversity mechanisms (this to avoid undesirable flocking behavior)
7. Scalability (in terms of complexity of the optimization/search tasks)
8. Autonomy (i.e. ability to self-control functioning

### 2.1. Components of Artificial Fish School (swarm) Algorithm

AFSA is a population-based search algorithm inspired in the behavior of swimming fishes that expand and contract while looking for food. Each fish *n* dimensional location represents a possible solution for the optimization problem. The algorithm makes use of weights for all the fishes which represents cumulative account on how successful the search for each fish in the school has been. AFSA is composed of the feeding and movement operators, the latter being divided into three sub-components, which are: **[58]**

### 2.2. Individual component of the movement

Every fish in the school performs a local search looking for promising regions in the search space. It is done as represented below in equation (8) :

$$x_i(t+1) = x_i(t) + rand(-1,1)\ step_{ind} \qquad (8)$$

where $x_i(t)$ and $x_i(t+1)$ represent the position of the fish *i* before and after the individual movement operator, respectively. $rand(-1,1)$ is a uniformly distributed random number varying from -1 up to 1 and $step_{ind}$ is a parameter that defines the maximum displacement for this movement. The new position $x_i(t+1)$ is only accepted if the fitness of the fish improves with the position change. If it is not the case, the fish remains in the same position and $x_i(t+1) = x_i(t)$ .

**2.2.1.** Collective-instinctive component of the movement

An average of the individual movements is calculated based on the following equation (9) :

$$I = \frac{\sum_{i=1}^{N} \Delta x_i \, \Delta f_i}{\sum_{i=1}^{N} \Delta f_i} \qquad (9)$$

The vector *I* represents the weighted average of the displacements of each fish. It means that the fishes that experienced a higher improvement will attract fishes into its position. After the vector *I* computation, every fish will be encouraged to move according to equation (10):

$$x_i(t+1) = x_i(t) + I \qquad (10)$$

**2.2.2.** Collective-volitive component of the movement

This operator is used in order to regulate the exploration/exploitation ability of the school during the search process. First of all, the barycenter *B* of the school is calculated based on the position and the weight $W_i$ of each fish as shown in (11):

$$B(t) = \frac{\sum_{i=1}^{N} x_i(t) W_i(t)}{\sum_{i=1}^{N} W_i(t)} \qquad (11)$$

and then, if the total school weight $\sum_{i=1}^{N} W_i$ has increased from the last to the current iteration, the fishes are attracted to the barycenter according to equation A . If the total school weight has not improved, the fishes are spread away from the barycenter according to equation *B*:

Eq. A:

$$x_i(t+1) = x_i(t) - step_{vol}\, rand(0,1) \frac{x_i(t) - B(t)}{distance(x_i(t), B(t))} \quad (12)$$

Eq.B :

$$x_i(t+1) = x_i(t) + step_{vol}\, rand(0,1) \frac{x_i(t) - B(t)}{distance(x_i(t), B(t))} \quad (13)$$

Where $step_{vol}$ defines the size of the maximum displacement performed with the use of this operator. $distance(x_i(t), B(t))$ is the Euclidean distance between the fish position and the school barycenter.$rand(0,1)$ is a uniformly distributed random number varying from 0 up to 1.

Besides the movement operators, it was also defined a feeding operator used in order to update the weights of every fish according to (14):

$$W_i(t+1) = W_i(t) + \frac{\Delta f_i}{\max(|\Delta f_i|)} \quad (14)$$

Where $W_i(t)$ is the weight parameter for fish $i$ ,$\Delta f_i$ is the fitness variation between the last and the new position, and $|\Delta f_i|$ represents the maximum absolute value of the fitness variation among all the fishes in the school. $W$ is only allowed to vary from 1 up to $W_{scale}/2$ , which is a user defined attribute. The weights of all fishes are initialized with the value $W_{scale}/2$.

**2.2.3.** The pseudo-code for AFSA

1 : Initialize user parameters
2 : Initialize fishes positions randomly
3 : while Stopping condition is not met do
4 : Calculate fitness for each fish
5 : Run individual operator movement
6 : Calculate fitness for each fish
7 : Run feeding operator
8 : Run collective-instinctive movement operator
9 : Run collective-volitive movement operator
10 : end while

The parameters $step_{ind}$ and $step_{vol}$ decay linearly according to :

$$step_{ind}(t+1) = step_{ind} - \frac{step_{ind}(initial)}{It_{max}} \quad (15)$$

Andsimilarly :

$$step_{vol}(t+1) = step_{vol} - \frac{step_{vol}(initial)}{It_{max}} \quad (16)$$

Where $step_{ind}(intial)$ and $step_{vol}(intial)$ are user defined initial values for $step_{ind}$ and $step_{vol}$ , respectively. $It_{max}$ is the maximum number of iterations allowed in the search process.

# 3. Variations of the AFSA

## 3.1. dAFSA(Density based AFSA)

This version excels for multimodal hyper-dimensional functions. It includes modifications in the previous operators: Feeding and Swimming, as well as new: Memory and Partition operators. The latter two were introduced to account for the partition of the main school into subgroups. Some changes were also included in the stop conditions that now also have to consider subswarms.

## 3.2. wAFSA(Weight based AFSA)

wAFSA is a weight based niching version of AFSA intended to produce multiple solutions. The niching strategy is based on a new operator called link formator. This operator is used to define leaders for the fishes in order to form sub-schools. **[59]**

## 3.3. AFSA-SAR(Stagnation Avoidance Routine AFSA)

In the original version of the algorithm, the individual movement component is only allowed to move a fish if it improves the fitness. However, in a very smooth search space, there would be many moving trials with no success and the algorithm could fail to converge. To solve these issues, was introduced a parameter *X* for which *0* <= *X* <= 1 in the individual component of the movement. *X* decays exponentially along with the iterations and measures a probability for a worsening allowance for each fish. It means that, every time a fish tries to move to a position that does not improve its fitness, a random number is chosen and if it is smaller than *X* the movement is allowed. **[58]**

## 3.4. bAFSA(Binary AFSA)

The bFSS intended to cope with premature convergence. Proposing the use of a binary encoding scheme for the internal mechanisms of the fish school search. It combined the FSS with fuzzy modeling in a wrapper approach for Feature Selection.**[60]**

## 3.5. MOAFSA(Multi-Objective AFSA)

In the MOAFSA the operators are adapted to solve multi-objective problems. The algorithm deploys an External Archive to store the best non-dominated solutions found during the search process. This approach has been extensively used for different bio-inspired multi-objectiveoptimizers**.[61] [62]**

Furthermore, the solutions within the External Archive are used to guide the fish movements in the proposal version.**[63]**

### 3.6. IBAFSA

IBAFSA stands for improved binary AFSA this algorithm was originally proposed in **[64]** to solve the multi-dimensional knapsack problem and we inspired our solution and our own version of the BAFSA from this algorithm.

## 4. Artificial Fish School Algorithm for global optimization problems

In thissection,we give a brief description of AFSA proposed in **[65]**for box constrained global optimization problems of type minimize$x \in \Omega\ f(x)$ . Here $f: R^n \to R$ is a non linear function that is to be minimized and is $\Omega = \{x \in R^n : l_j \leq x_j \leq u_j\ , j = 1,2, \dots, n\}$ the search space. $l_j$and $u_j$are the lower and upper bounds of $x_j$, respectively, and $n$ is the number of variables of the optimization problem.

AFSA works with a population of N points $x^i, i = 1,2, \dots, N$ identify promising regions looking for a global solution**[66]**.$x^i$ is a floating point encoding that covers the entire search space $\Omega$. The crucial issue of AFSA is the 'visual scope' of each point $x^i$. This represents a closed neighborhood of $x^i$ with a radius equal to a positive quantity $v$ defined by (17):

$$v = \sigma_{j\epsilon\{1,2,\dots,n\}}(u_j - l_j) \quad (17)$$

Where $\delta\epsilon\ (0,1)$ is a positive visual parameter. This parameter may be reduced along the iterative process. Let $\boldsymbol{I}^i$ be the set of indices of the points inside the 'visual scope' of point $x^i$, where $i \notin \boldsymbol{I}^i$ and $\boldsymbol{I}^i \subset \{1,2, \dots, N\}$, and let $np^i$ be thenumber of points in its 'visual scope'. Depending on the relative positions of the points in the population, three possible situations may occur:

a- When $np^i = 0$, the 'visual scope' is empty, andthe point $x^i$, with no other points in its neighborhood, moves randomly looking for a better region;

b- Whenthe 'visual scope' is not crowded, the point $x^i$ is able either to chase moving towards the best point inside the 'visual scope', or, if this best point does not improve the objective function value corresponding to $x^i$, to swarm moving towards the central pointof the 'visual scope';

c- Whenthe 'visual scope' is crowded, the point $x^i$ has some difficulty in following any particular point, and searches for a better region by choosing randomly another point (from the 'visual scope') and moving towards it.

The condition that decides when the 'visual scope' of xi is not crowded is :

$$C_f \equiv \frac{np^i}{N} \leq \theta \qquad (18)$$

Where $C_f$is the crowding factor and $\theta \in (0,1)$ is the crowd parameter. In this situation, the point $x^i$ has the ability to swarm or to chase.The swarming behavior is characterized by a movement towards the central point inside the 'visual scope' of $x^i$ defined by :

$$\overline{x} = \frac{\sum_{l \epsilon I^i} x^i}{np^i} \qquad (19)$$

## 5. Our approach: MIbAFSA for ISP

In this section we will present the proposed MIbAFSA to solve the ISP. The outline of the algorithm is described in the following.

### 5.1. Initialization (coding) of the population

The first step to design the MIbAFSA for solving the ISP is to devise a suitable representation scheme of a point/solution (set of indexes) from the population. Since we consider the ISP, N solutions, $x^i, i = 1,2, \dots, N$ are randomly initialized taking in consideration the *S* (storage) constraint, each represented by a binary 0/1string of length *n*(here *n* represents the number of candidate attributes to be indexed and each bit in the binary string represents a candidate attribute and each of them has a specific storage cost and gain ).. We remark that the maximum population size *N* of binary 0/1 strings of length *n* is $2^n$ In our MIbAFSA we decided to modify the original IbAFSAalgorithm by a new approach were we generate only feasible solutions.The steps we took are defined as follow :

a- Since our solution representation is binary we know that the possible solution sets are going to be a binary strings of length *n* where*n* represents the number of candidate attributes there for the number of possible solution is going to be $2^n$.

b- Our algorithm takes*N* as an initial number of solution (population) and we start generating *N* random decimal numbers betwen 0 and $2^n$.

c- Every time we generate a new random number we convert it to a binary string then call the cost model to check if it doesn't violate the *S* constraint

d- After checking the S constraint now we check if the solution isalready contained in the Solutions set (population)or not. if it is, we generate a new one, or else it will be added to the population.

The previous steps were taken to improve the quality of the generated solutions so therewont be any repetition to skip the local optimum problem which represents the major

drawback of the AFSA nor constraint violation. Also, all generated solutions are feasible not like the original IbAFSA algorithm.

### 5.2. Generating trial points in MIbAFSA

In MIbAFSA the Hamming distance, $H_d$, is used to identify the points inside the 'visual scope' of point $x^i$. The Hamming distance between two bit sequences of equal length is the number of positions at which the corresponding bits are different. Aftercalculating the Hamming distance between all pair of points from the population, the $np^i$ points inside the 'visual scope' of $x^i$ are identified as the points $x^j$ that satisfy the condition $H_d(x^i, x^j) \leq v$, for $j \in \{1,2, \dots, N\}, j \neq i$, where :

$$v = \delta \times n \quad (20)$$

$\delta \in (0,1)$and $n$ represents the maximum Hamming distance between two binary points. After computing $np^i$, the crowding factor $C_f$ of $x^i$ is calculated using (18).

Depending on the value of $C_f$, the 'visual scope' can be empty, not crowded or crowded. In IbAFSA, the behavior that generate the trial points are outlined as follows.

#### 5.2.1. Chasing behavior

If the 'visual scope' of $x^i$ is not crowded and the point that has the best objective function value (the solution's augmentation in excuting the query workload) inside the 'visual scope', denoted by $x^{best}(best \in \boldsymbol{I}^i)$, satisfies $z(x^{best}) > z(x^i)$, the chasing behavior is to be implemented. In chasing, each bit of the trial point, $y^i$, is generated by copying the corresponding bit from $x^i$ or from $x^{best}$ with equal probability. This operation is similar to the uniform crossover present in genetic/evolutionary algorithms.

#### 5.2.2. Swarming behavior

When the 'visual scope' is not crowded and $z(x^{best}) \leq z(x^i)$, (chasing is not possible), then if $z(\bar{x}) > z(x^i)$, where $\bar{x}$ is the central point inside the 'visual scope' of the point $x^i$, the swarming behavior is to be implemented. The central $\bar{x}$ is the point closest to all the other points in the 'visual scope', in the sense that the average Hamming distance to all other points in the 'visual scope' is minimal. Since in IbAFSA, the points are represented by binary 0/1 strings, each bit of $\bar{x}$ takes the majority of the corresponding bits of the other points in the 'visual scope' and is randomly defined in case of tie. The pseudocode to compute the central point is shown in Algorithm 1.

**Algorithm1.** Central point

***Require :*** *Set* $I^i$ *and* $np^i$ *points inside the 'visual scope' of* $x^i$
*1 :* ***for*** *j = 1 to n* ***do***
*2 :* *Compute* $\overline{x_j} = \frac{\sum_{l \epsilon I^i} x_j^i}{np^i}$
*3 :****if*** $\overline{x_j} = 0.5$ ***then***
*4 :* *Set* $\overline{x_j}$ *:= Random Integer {0,1}*
*5 :****else***
*6 :* *Set* $\overline{x_j}$ *:= Round* $(\overline{x_j})$
*7 :****end if***
*8 :****end for***
*9 :* ***return*** *Central point* $\overline{x}$

In swarming, each bit of the trial $y^i$ is created by copying the corresponding bit from $x^i$ or from $\bar{x}$ with equal probability.

**5.2.3.** Searching behavior

The searching behavior is tried in the following situations:

a- When the 'visual scope' is not crowded and neither $x^{best}$ nor $\bar{x}$ improves in objective function value;

b- Whether the 'visual scope' is crowded.

Here, a point $x^{rand}(rand \in \boldsymbol{I}^i)$ inside the 'visual scope' of $x^i$ is randomly selected and the searching behavior is to be implemented if $z(x^{rand}) > z(x^i)$ . Otherwise, a random behavior is implemented. In searching, each bit of $y^i$ is created by copying the corresponding bit from $x^i$ or from $x^{rand}$ with equal probability.

**5.2.4.** Random behavior

When the 'visual scope' of $x^i$ is empty or the other behavior were not performed, the point $x^i$ performs the random behavior. In this case, the trial point $y^i$ is created by randomly setting a binary string of 0/1 bits of length *n*.

**5.3. Constraints handling**

The widely used approach to deal with constraints is based on penalty functions where a penalty term is added to the objective function in order to penalize the constraint violation. The penalty function method can be applied to any type of constraints, but the performance of penalty-type method is not always satisfactory due to the choice of an appropriate penalty parameter. Although several ideas have been proposed about designing efficient penalty functions and tuning penalty parameters, other alternative constraint handling techniques have emerged in the last decades**[67][68].**

In our version of the BAFSA we do not need a decoding function because all of the generated solutions are feasible therefore a simple greedy-like heuristic called **add_item** (aiming to add indexes) (Algorithm 2) is implemented to each feasible solution aiming to improve that point without violating any constraint. Our own function to add indexes is based on randomality since the indexes has no fixed gain thus we decided that it would be better to pick randomly an index and try to add it to the solution in order to improve it and if the new improved solution is feasible we maintain the 1 bit and try to add a new one or else the bit is set to 0 then randomly select a new one then try the same, in both cases all of the indexes must be tried.

**Algorithm 2. Add_item used in MIbAFSA.**

***Require :*** *feasible point $x^i$, rand function,remove function, set J={1 ,2,...,n}, S, Size function*
*1 :* $y^i = x^i$
*2 :* ***for*** *j=1 to n*
*3 :rnd= random(J)*
*5 : remove(rnd,J)*
*6 :* $y^i_{rnd}$ *:=1*
*7 :* ***if*** *z* $(x^i)$*<z* $(y^i)$ *and Size* $(y^i)$*<= S*
*8 :* $x^i = y^i$
*9 :* ***end if***
*10:* ***end for***
*11:* ***return*** $x^i$

### 5.4. Selection of a new population

Among the trial points $y^{i,t}$ and the current points $x^{i,t}$ , $i = 1,2,\dots,N$ atiteration *t*, in order to decide whether or not they should become members of the population in the next iteration, *t+1* , the trial point is compared to the current point using the following greedy criterion:

$$x^{i,t+1} = \begin{cases} y^{i,t} \; if \; z\left(y^{i,t}\right) \geq z\left(x^{i,t}\right) \\ x^{i,t} \qquad otherwise \end{cases}, \quad i = 1,2,\dots,N. \qquad (21)$$

### 5.5. Leaping behavior

When the best gain function value in the population does not change for a certain number of iterations, the algorithm may have stagnated. The other points of the population will eventually converge to that gain function value. To be able to escape from this region and to try to converge to the optimal solution, the algorithm performs the leaping behavior, at every $L$ iterations. In the leaping, a point $x^{rand}(rand \in \{1,2,\dots,N\})$ is randomly selected from the current population and some randomly selected bits of the point are changed from 0 to 1 or

vice versa with probability $p_m$. The value $p_m = 0.01$ is widely used in binary represented methods. The described operation is similar to a mutation with probability $p_m$ of genetic/evolutionary algorithms.Afterwards, the add_item heuristic is implemented, and the new point replaces the point $x^{rand}$.

### 5.6. Termination conditions

Let $T_{max}$ be the maximum number of iterations. Let $z_{max}$ be the maximum objective function value attained at iteration $t$ and $z_{opt}$ be the known optimal value available in the literature. The proposed MIbAFSA terminates if one of the conditions:

a- $t < T_{max}$ The maximum number of iteration is reached.

b- $or\ |z_{max} - z_{opt}| \leq \varepsilon$ This condition is used in the case of comparing between an existed optimal solution found by another approach and ours. If we are not comparing $z_{opt}$ is always set to 100 and it continues execution until $T_{max}$ is exceeded.

### 5.7. Reinitialization of the population

Past experiments with bAFSA have shown that, from a certain iteration on, all the individual points in a population converge to a non-optimal point, even after the leaping behavior has been performed. To diversify the search, we propose to reinitialize the population randomly, every *R*iterations, keeping the best solution found so far. In practical terms, this technique has greatly improved the quality of the solutions and increased the consistency of the proposed version of bAFSA.

### 5.8. The algorithm

The pseudocode of the proposed algorithm for solving the ISP is show in Algorithm 3.

**Algorithm 3. MIbAFSA.**

*$T_{max}$ and $z_{opt}$ and other values of parameters*
*1 : set t :=1. Initialize population $x^{i,1}, i = 1,2, \dots N$*
*2 : Perform and add_item, evaluate the population and identify $x^{max}$ and $z_{max}$*
*3 :**while** 'termination conditions are not met' **do***
*4 :**if** MOD(t,R) = 0 **then***
*5 : Reinitialize population $x^{i,t}, i = 1,2, \dots N\text{-}1$*
*6 : Perform add_item, evaluate the population and identify $x^{max}$ and $z_{max}$*
*7 :**end if***
*8 :**for all $x^{i,t}$ do***
*9 : Compute 'visual scope' and 'crowdingfactor'*
*10 :**if** 'visual scope' is empty **then***
*11 : Perform random behavior to create trial point $y^{i,t}$*
*12 :**else if** 'visual scope' is notcrowded**then***
*13 :**if** $z(x^{best}) > z(x^{i,t})$**then***
*14 : Perform chasing behavior to create trial point $y^{i,t}$*
*15 :**else if** $z(\bar{x}) > z(x^{i,t})$**then***
*16 :Perform swarming behavior to create trial point $y^{i,t}$*
*17 :**else if** $z(x^{rand}) > z(x^{i,t})$**then***
*18 : Perform searching behavior to create trial point $y^{i,t}$*
*19 :**else***
*20 : Perform random behavior to create trial point $y^{i,t}$*
*21 :**end if***
*22 :**else if** 'visual scope' is crowded **then***
*23 :**if** $z(x^{rand}) > z(x^{i,t})$**then***
*24 :Perform searching behavior to create trial point $y^{i,t}$*
*25 :**else***
*26 : Perform random behavior to create trial point $y^{i,t}$*
*27 :**end if***
*28 :**end if***
*29 :**end for***
*30 : Perform add_item to get $y^{i,t}, i = 1,2, \dots N\text{-}1$ and evaluate them*
*31 : Select new population $x^{i,t+1}, i = 1,2, \dots N\text{-}1$*
*32 :**if** MOD(t,L) = 0 **then***
*33 : Perform leaping behavior, add_item and evaluate*
*34 :**end if***
*35 : Identify $x^{max}$ and $z_{max}$*
*36 : Set t :=t+1*
*37 :**end while***
*38 : **return $x^{max}$ and $z_{max}$***

In the following section we present the steps of the algorithm in the data warehouse environment and the selection process with simple diagrams :

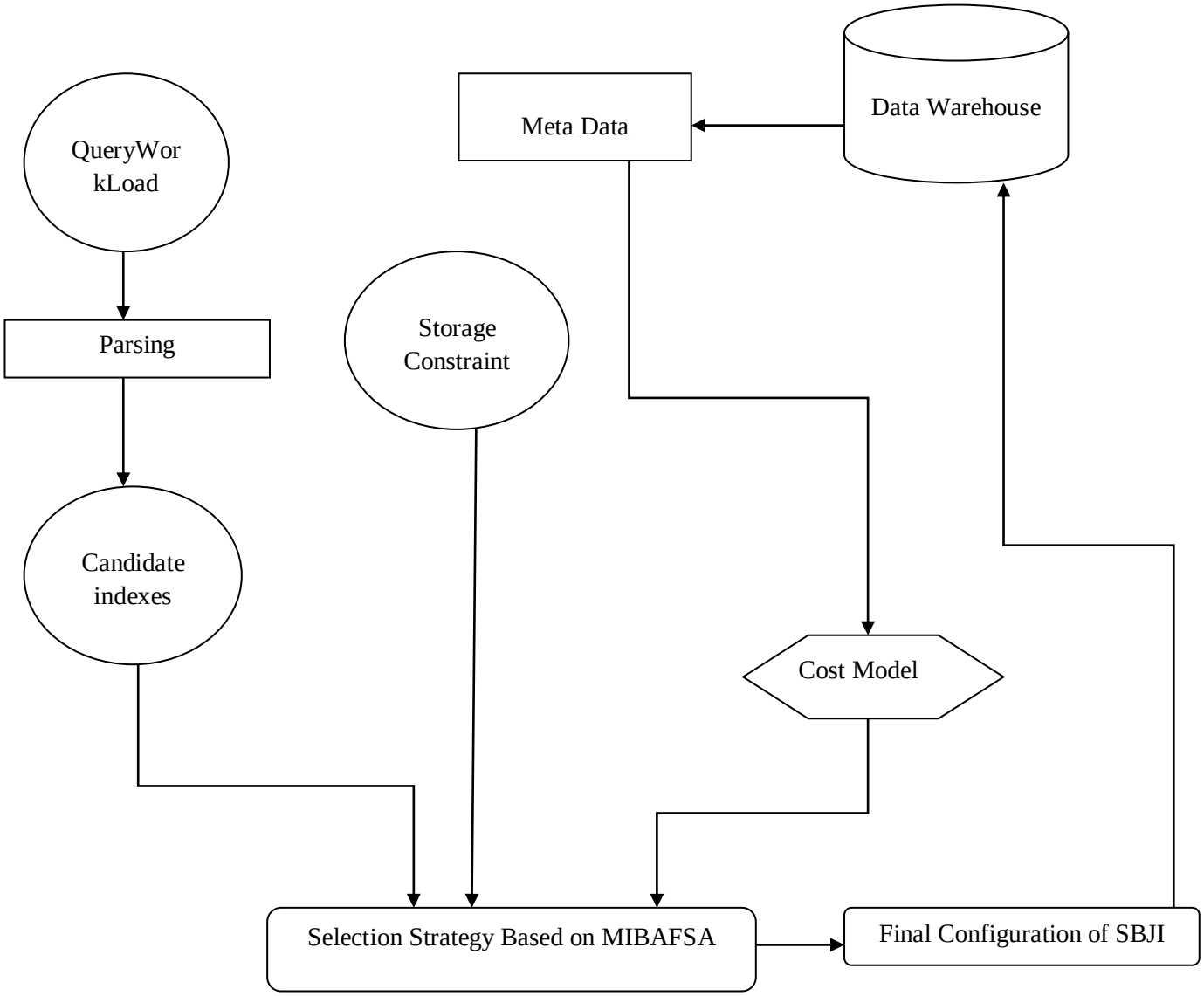


**Figure 3. 1:**The Globalarchitecture of MiBAFSA for solving ISP in data warehouses

The index selection stageof our approach is presented in the next diagram (Figure 3.2)

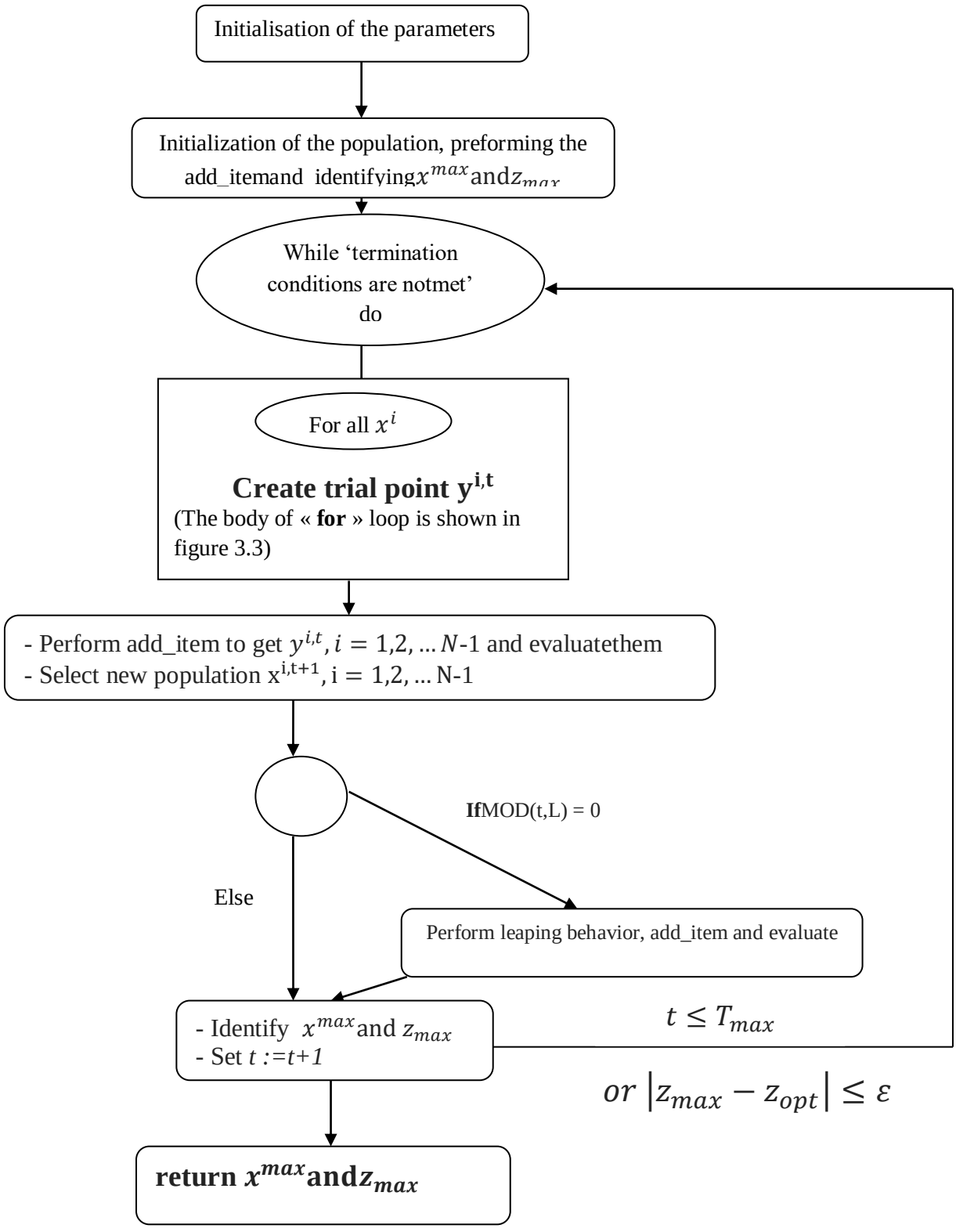


**Figure 3. 2:** the index selection process of MiBAFSA

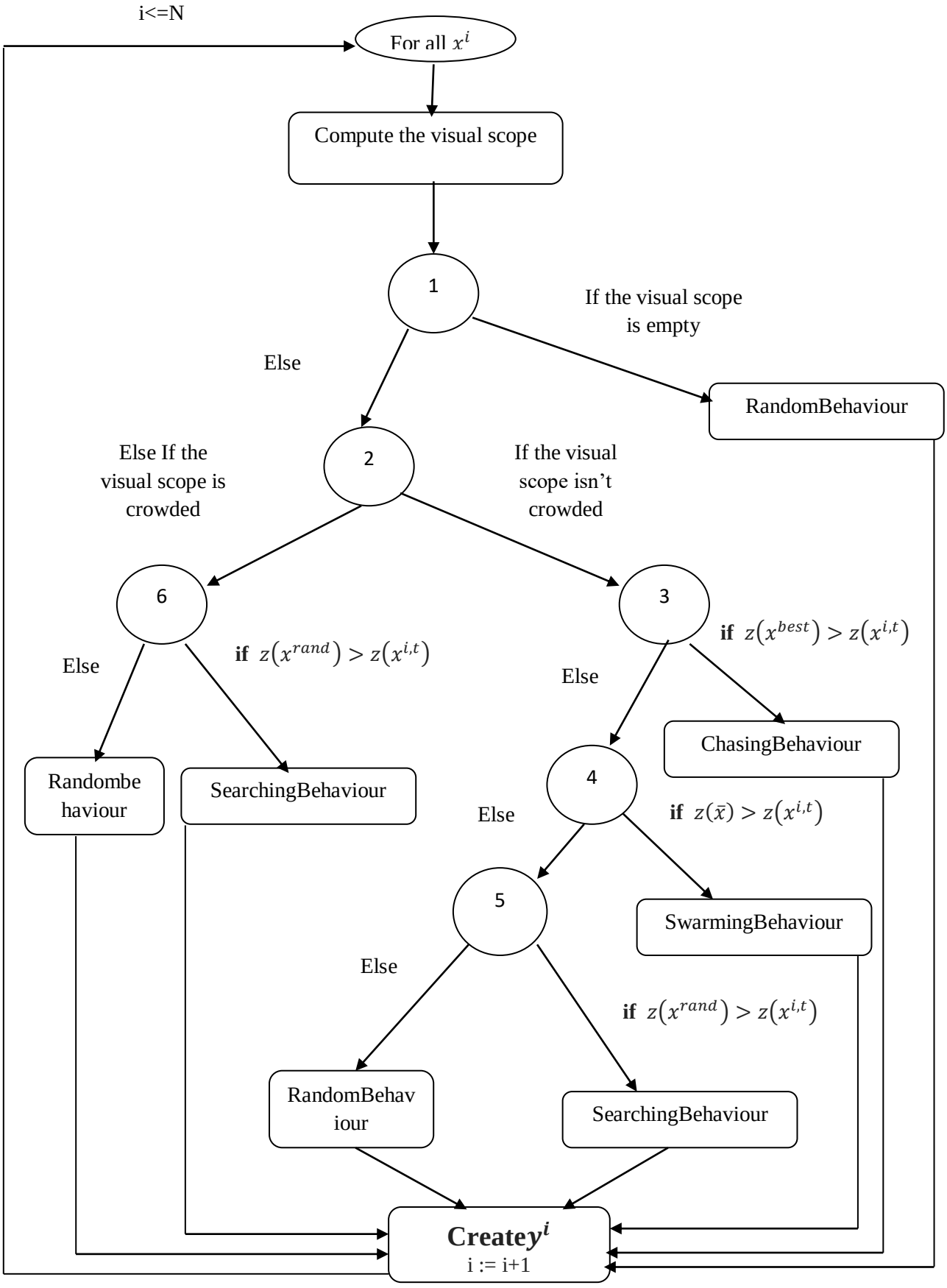


**Figure 3. 3:** the « For » loop body for creating trial points

In the following we present the cost model that was presented in the algorithm as the objective function *z* that is to be maximize. this cost model is used to estimate the gain and the size of the index configuration.

## 6. Cost model

To estimate both the storage and workload costs, we use the cost models that are initially proposed in **[33]** and are widely used in the most closely related works.

Below, we briefly recall the used cost models. The parameters of this cost model are summarized in Table 3.1.

| Symbole | Description |
|---|---|
| $\|X\|$ | Number of tuples of a table X or the cardinality of the attribute X |
| $S_p$ | Disk page size in bytes |
| $p_X$ | Number of pages need to store the table X |
| $S_{Pointer}$ | Page pointer size in octets |
| $m$ | B-tree order, $m = 1 + \frac{S_p}{w(A)+S_{Pointer}}$ |
| $d$ | Number of bitmaps used to evaluate a given query |
| $N_r$ | Number of read tuples for a givenquery, $N_r = d\frac{\|F\|}{\|A\|}$ |
| $w(X)$ | Size of n-uplets of a tabke X or the cardinality of the attribute X |
| $Size_{RowId}$ | RowID size = 10 Bytes |

**Table 3. 1:** Symbols used in the cost model

### 6.1. The size of a BJI

The size « S » of a BJI on an attribute « A » is estimated by :

$$SIZE(BJI_A) = \frac{|A|\times|F|}{8} + Size_{RowId} \times |F| Bytes \qquad (22)$$

### 6.2. The cost of accessing data with a BJI

the number of I/Os performed when executing a query is used as a cost metric for an index configuration. The cost of processing a query qi using an index BJIA created on an attribute A is given by:

$$COST(Qi/\ BJI_A) = Log_m|A| - 1 + \frac{|A|}{m-1} + d\frac{|F|}{8S_p} + (1 - e^{-\frac{N_r}{p_F}}) \quad (23)$$

### 6.3. The cost of accessing data without BJI

All joins are implemented by the hash-join method. The number of I/O operations needed to join two tables T1 and T2 using hash-join method is estimated by :

$$COST_{hash} = 3(p_{T1} + p_{T2}) \quad (24)$$

## 7. Conclusion

In this chapter we presented the algorithm that was the base of our approach, it's variations and how to use it for global optimization.

Also, we have presented the proposed algorithm for solving the ISP in details and in the next chapter we will see the experimental results in addition to a comparing study.

# CHAPTER 4
# IMPLEMENTATION AND EXPERIMENTAL RESULTS

## 1. Introduction

In order to validate our approach, we implemented it so that we could evaluate it and apply tests on real data. For this, we have developed a tool called AFSISbtool (Artificial Fish Swarm Index Selection based tool) which provides the administrator with a simple and intuitive interface allowing him to specify the parameters and apply them to a workload of decisional requests to get out a configuration of indexes to optimize the load. In this chapter, we will present in the first part the tool for selecting binary join indexes realized and called AFSIStool . Then, in the second part, we carry out a series of tests on a real warehouse resulting from the benchmark APB1 **[69]** in order to examine the behaviors of our approach according to different parameters values. To do this, we start by introducing our experiment environment, the query load we worked on and end up evaluating our results VS a datamining constraint based approach results.

## 2. AFSISbtool : Artificial Fish Swarm Index Selection based tool

AFSISbtool is the implementation of our index selection approach. It's a tool that helps the administrator of a data base/data warehouse or data scientists to chose an optimum SBJI set to optimize the response time of decisional queries addressed to the datawarehouse/database.This tool also gives a theorical storage space that is needed to create the optimum index configuration and a percentage of the gain in the execution time of the decisional queries.

In the following sections we present through screen-shots the functionality of the AFSISbtool.

### 2.1. Development language and environment

The tool is developed under the Netbeans Integrated Development Environment. Besides its portability, java is chosen for its automatic management of memory. This feature is crucial because patterns are implemented through lists and sets. The Java language also offers a multitude of functions to easily handle this type of data structures.

## 2.2. Features overview

The figure 4.1 represents the main interface of the AFSISbtool that contains the algorithm's parameters and the constraints. The interface also has a zone to show the optimum index set and it's gain.

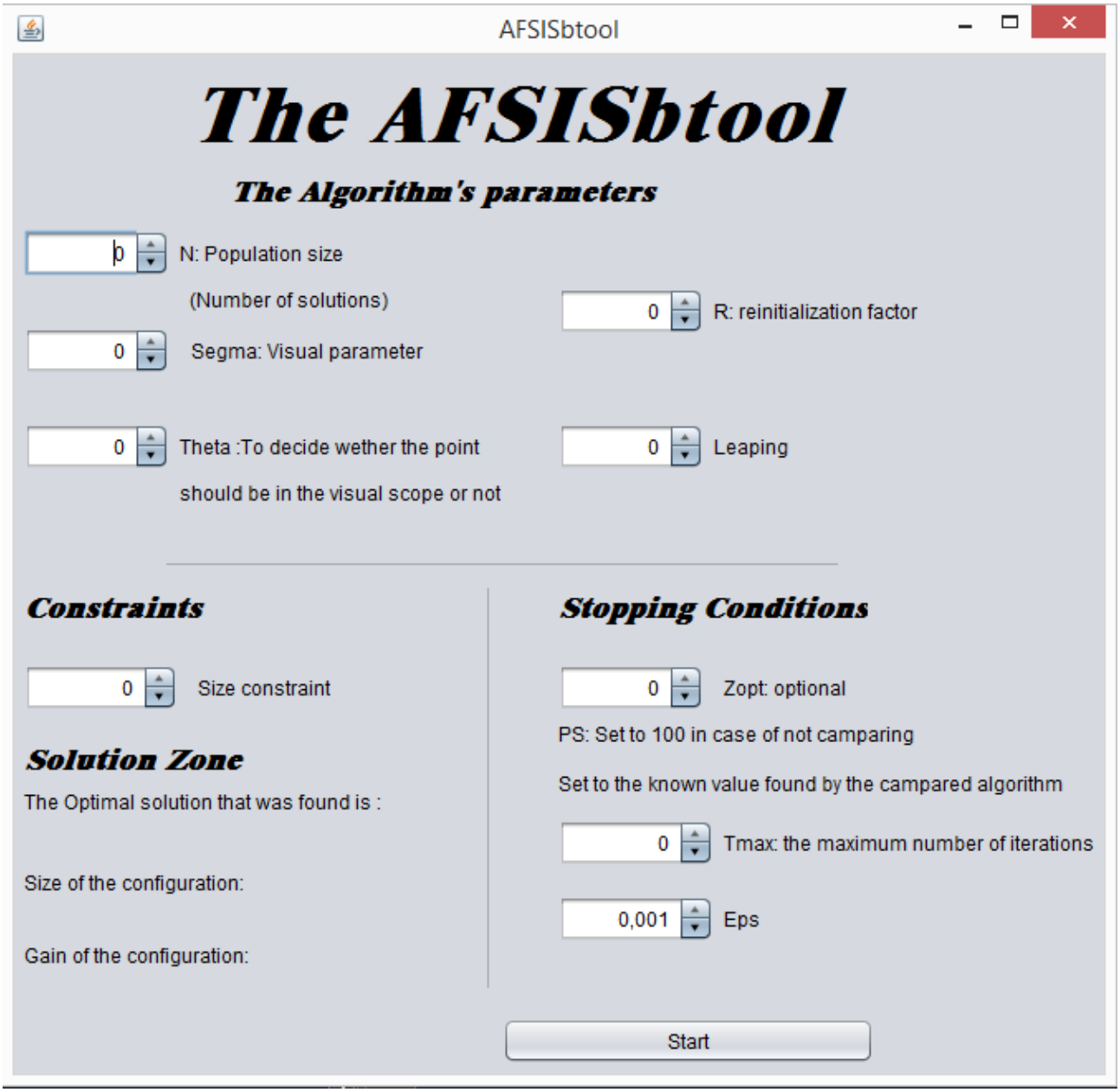


**Figure 4. 1:** The interface of the AFSISbtool

The main interface could be divided into 4 major parts that are: the algorithm's parameter, stopping conditions, constraints and the solution's zone that contains the final index configuration.

**2.2.1.** Algorithm's parameters part

In this part we can initialize the basic parameters of our approach (MIbAFSA). The following figure shows that.

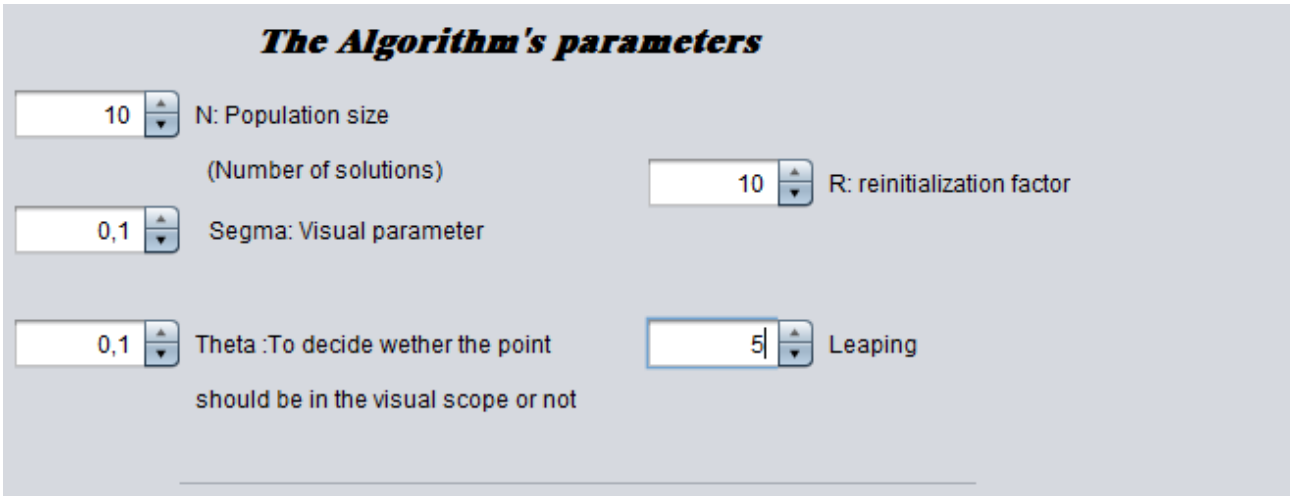


**Figure 4. 2:** The algorithm's parameters

**2.2.2.** The stopping conditions part

This part represents the conditions that specifies wether the test is a comparaison by setting the *Zopt*to a known value or an optimum solution search by setting Zopt to 100. In addition it contains the max number of iterations.

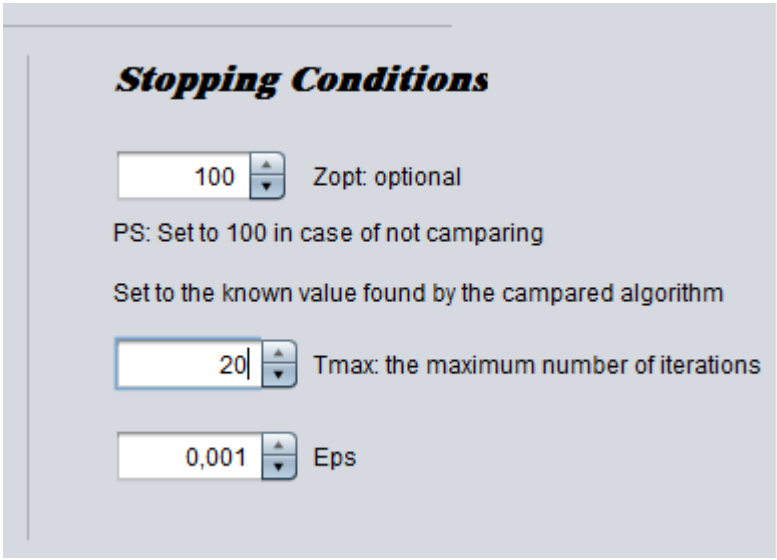


**Figure 4. 3:** Stopping conditions part

**2.2.3.** The size constraint and solution zone part

In this part we can specify the size constraint and see the information of the solution that was found such as : index configuration (index set), size of the configuration and the gain of the configuration.

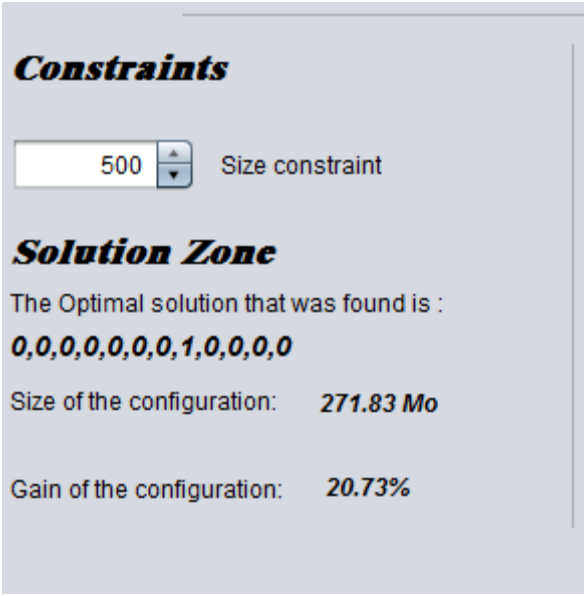


**Figure 4. 4:** Size constraint and solution zone part

In addition to this zone that shows the best solution ( index configuration ) that was found with its profit and size we can also track the best solution that was found on every iteration with it's profit and size and also the final best solution on the console of netbeans. the Following figure shows that.

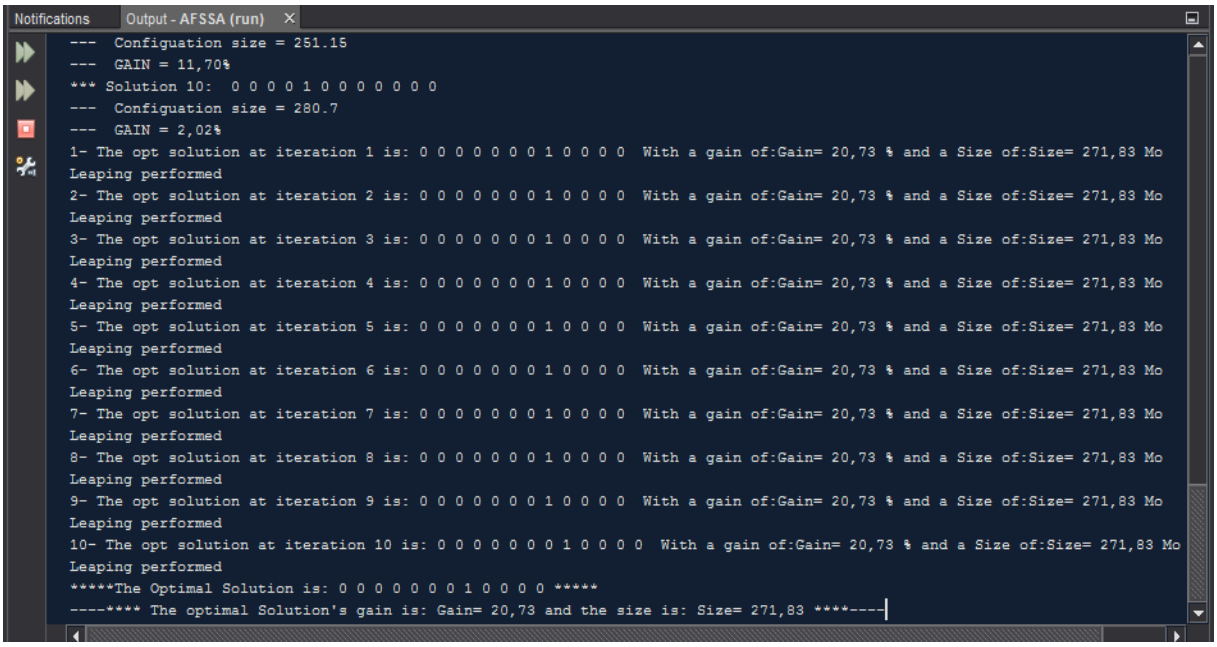


**Figure 4. 5:** Tracking the solution on console

## 3. Experimentations

In order to validate our approach, we conducted experiments that allowed us to evaluate it and compare it with other binary join index selection techniques. In this section we present our experimental environment and we discuss the results obtained.

In this section we will present the benchmark (the data warehouse that was used) and discuss how did we fix the parameters of the algorithm in addition to a comparative study between our approach and a constrained method.

### 3.1.Experimental environment

The experiments were done on a PC having 1.9GHz Intel(R) Core(TM) i3-4030U CPU and 4Gb RAM with a 500 Gb HDD that operates a data warehouse generated by a Benchmark DataBenchmark APB-1 Release II and a query workload with their frequencies.

We consider that :

- The size of a System Page is 65536
- The line identifier (RowId) is coded on 10 bytes

#### 3.1.1. Test Benchmark

Benchmarks allows us to measure the performance of a system to compare it to others. In the context of data warehouses, the best known is APB-1. Published in 1998 by an association of four solution vendors of OLAP, APB-1 simulates a real OLAP sales situation. This benchmark is used by most index selection works which allows us to compare our approach to them. The logical schema of the data warehouse resulting from this benchmark is a star schema consisting of a Actvars fact table and four dimension tables :ProdLevel, TimeLevel, CustLevel and ChanLevel.

The star schema of this warehouse is given in Figure 4.6.

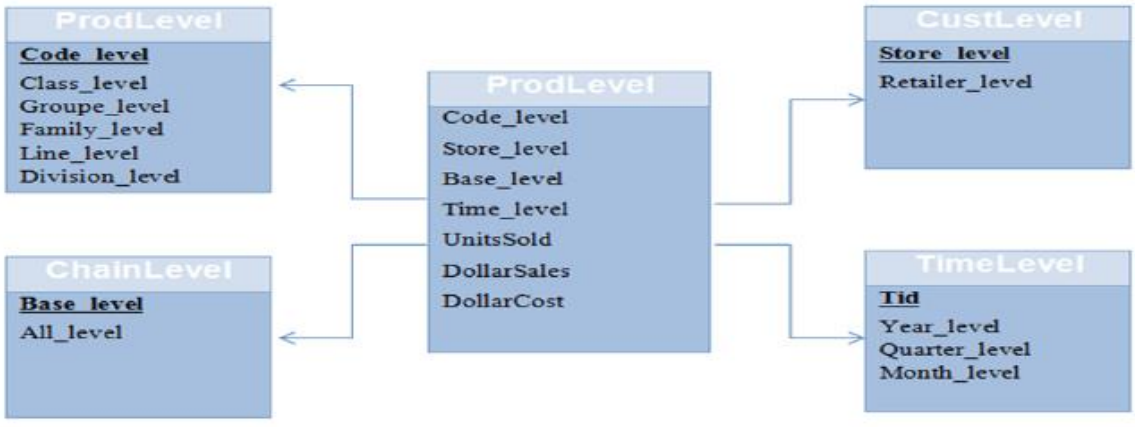


**Figure 4. 6:**data warehouse diagram of the Benchmark APB-1 release II

The characteristics of the tables that form the warehouse are given in the Table 4.1.

| Table | Number of records | Record size |
|---|---|---|
| Actvars | 24 786 000 | 74 |
| ProdLevel | 9 | 24 |
| TimeLevel | 900 | 24 |
| CustLevel | 9000 | 72 |
| ChanLevel | 24 | 36 |

**Table 4. 1:** Characteristics of the tables of the warehouse used

### 3.1.2. Query workload

We considered a load of 60 star join queries defined on the above warehouse. These queries are chosen so that they:

- must be composed of a single block and not nested queries.
- must cover all the attributes of the warehouse.
- belong to different categories:
- queries of type count (*) with and without aggregations,
- queries using aggregation functions such as Sum, Min, Max,
- queries with dimension attributes in the SELECT clause
- , ... etc ..

Figure 4.7 show an extract of three queries from the 60 query that represents our initial query workload.

```
--Q54:
SelectMin(Unitssold)
fromActvars A,CHANLevelH,CUSTLevelC,ProdLevel P,TimeLevel T
whereA.Customer_Level=C.Store_Level
      andA.Product_Level=P.Code_Level
      andA.Channel_Level=H.Base_Level
      andA.Time_Level=T.TID
      andT.Year_Level='1995'
      andT.Quarter_Level='Q1'
      andT.Month_Levelin('2','3','4')
      andP.Class_Level='OGTBD3Z92GQ4'
      andC.Retailer_Levelin('F7C6MZYU0MHV','X2DKZGVZ6CMO')
      andC.Gender_Level='F'
      andC.City_Level='Msila'
      andH.ALL_Level='ABCDEFGHIJKL';
```

**Figure 4. 7:** Query workload extract

The query load contains 12 selection attributes, these attributes represent the list of candidate attributes for indexing.

The cardinality of these attributes are given in Table 4.2.

| Attribut | Cardinalit |
|---|---|
| ClassLevel | 605 |
| GroupLevel | 300 |
| FamilyLevel | 75 |
| LineLevel | 15 |
| DivisionLevel | 4 |
| YearLevel | 2 |
| MonthLevel | 12 |
| QuarterLevel | 4 |
| RetailerLevel | 99 |
| CityLevel | 4 |
| GenderLevel | 2 |
| AllLevel | 5 |

**Table 4. 2:** Cardinality of the condidate attributes

Each query is characterized by its frequency of access, the frequency of each query is given in Figure 4.8. We think that this frequency is an important factor because if a query is frequent, its attributes should be indexed more than the attributes of an infrequent query. Therefore, when constructing the extraction context we must take into account this frequency, which is different from the occurrence frequency of the attributes in the 60 queries considered. For this, we repeated each query as many times as its frequency, which gives the final context with 671 entries. Thus, less frequent queries are penalized and more frequent queries are given more interest.

| Query | Frequency | Query | Frequency | Query | Frequency | Query | Frequency |
|---|---|---|---|---|---|---|---|
| Q1 | 3 | Q16 | 7 | Q31 | 12 | Q46 | 5 |
| Q2 | 7 | Q17 | 10 | Q32 | 4 | Q47 | 21 |
| Q3 | 10 | Q18 | 10 | Q33 | 12 | Q48 | 12 |
| Q4 | 10 | Q19 | 20 | Q34 | 4 | Q49 | 22 |
| Q5 | 12 | Q20 | 4 | Q35 | 21 | Q50 | 10 |
| Q6 | 3 | Q21 | 12 | Q36 | 3 | Q51 | 15 |
| Q7 | 3 | Q22 | 4 | Q37 | 15 | Q52 | 30 |
| Q8 | 16 | Q23 | 21 | Q38 | 10 | Q53 | 5 |
| Q9 | 4 | Q24 | 21 | Q39 | 3 | Q54 | 20 |
| Q10 | 12 | Q25 | 21 | Q40 | 12 | Q55 | 18 |
| Q11 | 4 | Q26 | 3 | Q41 | 12 | Q56 | 3 |
| Q12 | 21 | Q27 | 7 | Q42 | 10 | Q57 | 4 |
| Q13 | 21 | Q28 | 10 | Q43 | 18 | Q58 | 10 |
| Q14 | 21 | Q29 | 10 | Q44 | 11 | Q59 | 2 |
| Q15 | 3 | Q30 | 20 | Q45 | 12 | Q60 | 5 |

**Figure 4. 8:** Frequency of access to the query of the load

### 3.2.Parameters fixing

In this part we present a detailed study on how we fixed our parameters after several tests. to fix every single parameter we decided to start one by one until we reach the end. The following graphs represents the test that were done on every parameter and how we decided to pick it's value.

**a. Fixingthe population's size"N"**

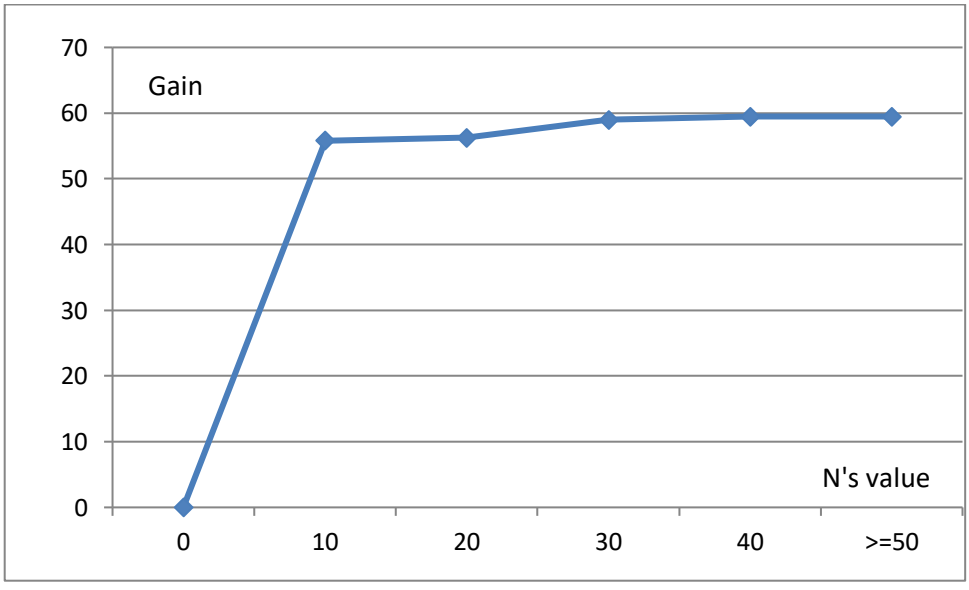


**Figure 4. 9:**Effect of the population's size parameter N on the quality of the solution

The population's size N represents the number of initial solutions.After experimentation, we found that it is the key factor that decides whether the solution will be found on the early

iterations or not specially that we changed the initialization and Add_Item function so the algorithm converges to an optimal solution on it's early stages as we can see after the first step from 0 to 10 the algorithm already made a huge solution jump in the other hand after few steps of 10 as well the solution didn't change that much . From the graph we see that that best size is 30 but after few experiments the size 30 took more iterations to stagnate to the optimum solution but when we the size of >=40 the algorithm converges to the optimum solution on its early iterations so we found that it should be better to pick 2 values , the first value is to show that the solution is getting improved over the iterations which is the value of 20, the second value is 50 so we can see that the algorithm can find it's optimum solution on the early iterations as well.

**b.Fixing the number of iterations"***Tmax"*

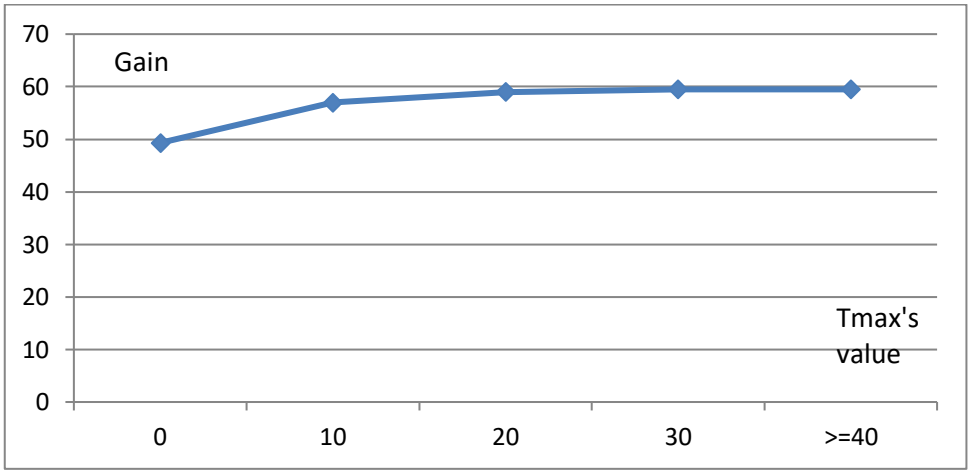


**Figure 4. 10: E**ffect of the parameter number of iterations Tmax

Tmax is the maximal number of iterations. The experiments showed that the best value of Tmax should be around 30 and 40 but we decided that it would be better to fix it at 50 just to ensure that the found solution is going to be optimal.

**c. Fixingthe crowding factor "$\theta$"**

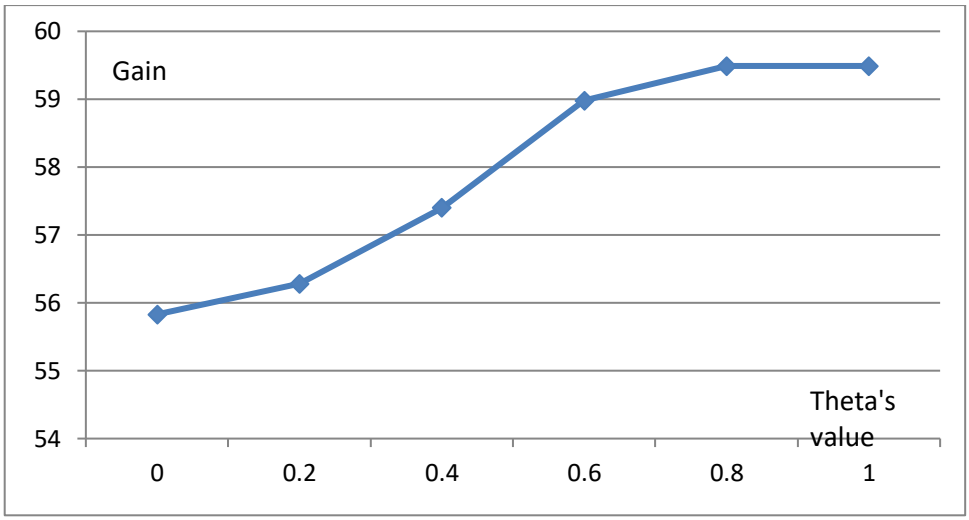


**Figure 4. 11:** Effect of the crowding factor parameter "Theta θ"

theta represents the crowding factorand as we can see in the previous graph the value of theta influences the solution's valuegreatly and after a few experiments we decided that it should be fixed at 0.8.

**d.Fixing the visual parameter "$\sigma$"**

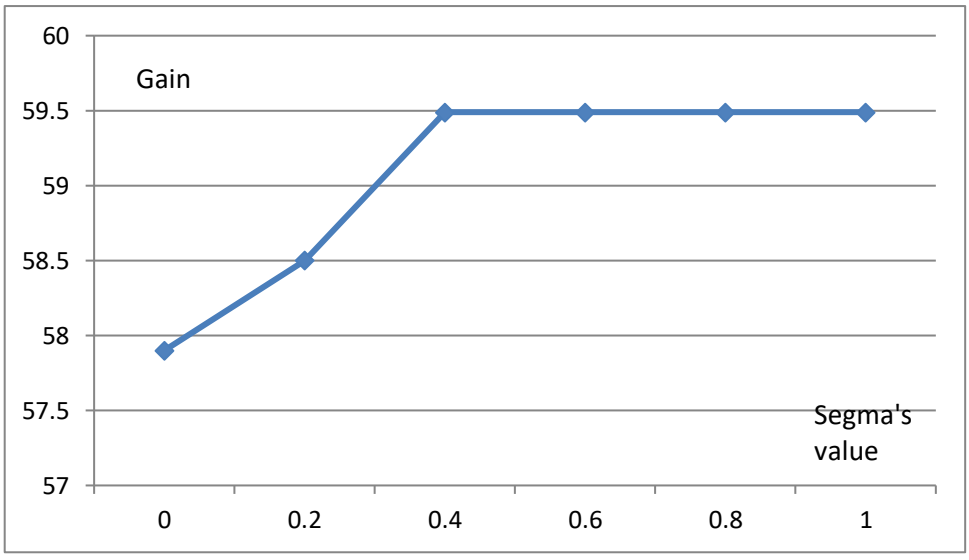


**Figure 4. 12:** Effect of the visual parameter "Segma σ"

After few tests we saw that the visual parameter Segma's value influences the solution's value but the influence is not that huge because of the new initialization and Add_Item functions in addition to the max number of iterations yet its influence should not be ignored but taken in consideration since it improves the solution's quality. After some point increasing in segma's value doesn't change a thing so we decided that it should be fixed at 0.4.

**e. Fixingthe reinitialization parameter "*R*"**

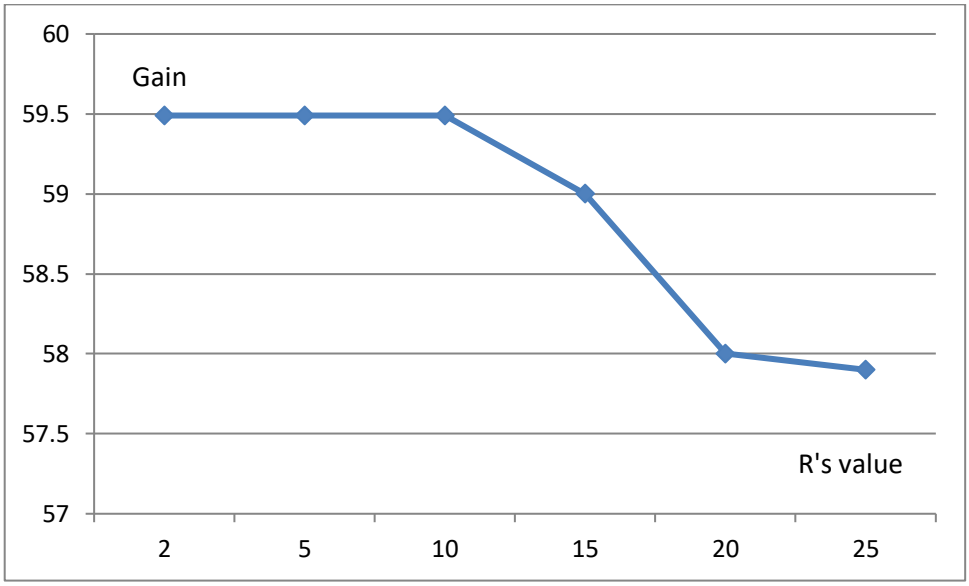


**Figure 4. 13: E**ffect of the reinitialization parameter R

The factor R represents the reinitialization factor. Every R iteration the population gets reinitialized to avoid the local optimum problem. And after few experiments we found that the reinitialization factor does not change much between the value 2 to10 so we decided to take the value 10 instead of 2 or 5 because the reinitialization does not change much in the previoustwo , instead picking 10 means a lower calculation time therefore the execution time is reduced.

**f. Fixing the leaping parameter "*L*"**

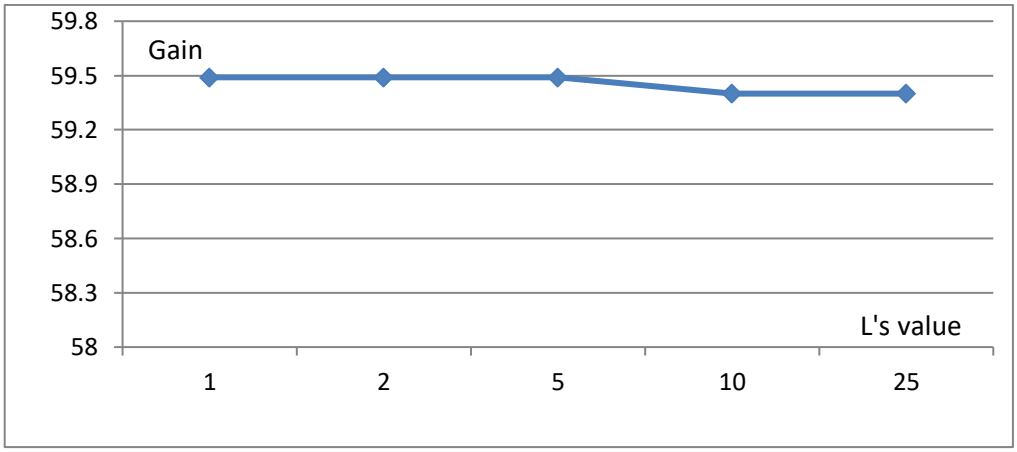


**Figure 4. 14: E**ffect of the parameter L(Leaping)

In the experiments that we've done and after fixing the previousparameters on the early stages of the tests we realized and noticed that the efficiency of the leaping parameter has

decreased but not completely gone so we found that it can be eliminated or fixed to a high value were it can be useful and not time consuming ( the value should be > 2 preferably 5).
For the other parameters such as $\varepsilon$ and $Z_{opt}$ they dont affect the quality of the solution by any mean and they are only considered in the algorithm in the case of comparing between the solution that may be found by our appraoch and another approach.
As we already stated in the past chapter $Z_{opt}$ should be always initialized to 100 in the case of not comparing and in this case it does not really matter if we initialize $\varepsilon$ or not because it wont affect the break constraint.

At the end of these tests we concluded that the effective parameters should be fixed at the following values :

| **The parameter** | **The value** |
|---|---|
| Population size *N* | 20or 50 depending on the size constraint |
| Iterations: Tmax | 30 |
| Crowding factor theta $\theta$ | 0.8 |
| Visual parameter $\sigma$ | 0.4 |
| Reinitialization parameter *R* | 10 |
| Leaping parameter *L* | 5 |

**Table 4. 3:** The parameters values

In the next section we present the results that were obtained by our approach and the constraint method approach.

### 3.3. Comparative study

We have compared our approach to a new one based on “constraint programming for solving the ISP problem”. Being an exact method, this approach has proven its effectiveness comparing to other approaches using approximation methods.
The following table contains the test results of the comparing study. in this study we decided to start from 500 Mo storage size and move with 500 steps until we reach the solution that contains the entire indexes.

| storage (MO) | Index configuration found by constraint based approach | Size (MO) | Gain % | Index configuration found by MIbAFSA approach | Size (MO) | Gain % |
|---|---|---|---|---|---|---|
| **500** | 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 | 271 | **20,88** | 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 | 271 | **20,88** |
| **1000** | 0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 1 , 0 | 771 | **34,54** | 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 1 , 0 | 980 | **34,71** |
| **1500** | 0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 1 , 0 | 1300 | **42,42** | 0 , 1 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 1 , 1 | 1477 | **45,22** |
| **2000** | 0 , 1 , 0 , 0 , 0 , 1 , 0 , 1 , 1 , 0 , 1 , 1 | 1796 | **51,16** | 0 , 1 , 0 , 1 , 0 , 1 , 1 , 1 , 0 , 0 , 1 , 1 | 1967 | **51,35** |
| **2500** | 0 , 1 , 0 , 1 , 0 , 1 , 1 , 1 , 1 , 0 , 1 , 1 | 2496 | **57,41** | 0 , 1 , 0 , 1 , 0 , 1 , 0 , 1 , 1 , 1 , 1 , 1 | 2496 | **58,04** |
| **3000** | 0 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 0 , 1 , 1 | 2777 | **58,65** | 0 , 1 , 0 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 | 2739 | **59,5** |
| **3500** | 0 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 | 3019 | **60,74** | 0 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 | 3019 | **60,74** |
| **4000** | 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 0 , 1 , 1 | 3900 | **60,76** | 0 , 1 , 1 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 | 3861 | **61,6** |
| **4500** | 1 , 1 , 0 , 1 , 0 , 1 , 0 , 1 , 1 , 0 , 1 , 1 | 4278 | **64,52** | 1 , 1 , 0 , 0 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 | 4305 | **64,52** |
| **5000** | 1 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 0 , 1 , 1 | 4801 | **67,45** | 1 , 1 , 0 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 | 4763 | **68,42** |
| **5500** | 1 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 | 5043 | **69,67** | 1 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 | 5043 | **69,67** |
| **6000** | 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 0 , 1 , 1 | 5924 | **69,55** | 1 , 1 , 1 , 1 , 0, 1 , 1 , 1 , 1 , 1 , 1 , 1 | 5885 | **70,51** |
| **6500** | 1 , 1 , 1 , 1 , 1, 1 , 1 , 1 , 1 , 1 , 1 , 1 | 6166 | **71,76** | 1 , 1 , 1 , 1 , 1, 1 , 1 , 1 , 1 , 1 , 1 , 1 | 6166 | **71,76** |

**Table 4. 4:** The comparing study results

The results that were attained in the tests and then captured in this table are transformed into a graph that is showed in the figure 4.15 and figure 4.16.

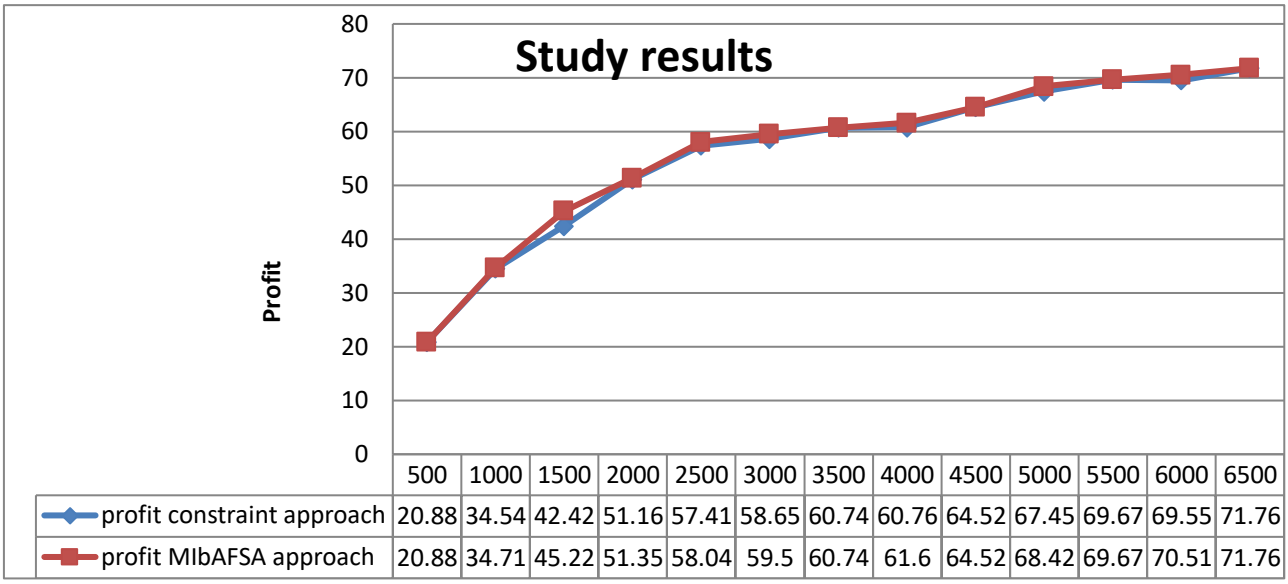


**Figure 4. 15:** Comparative study results

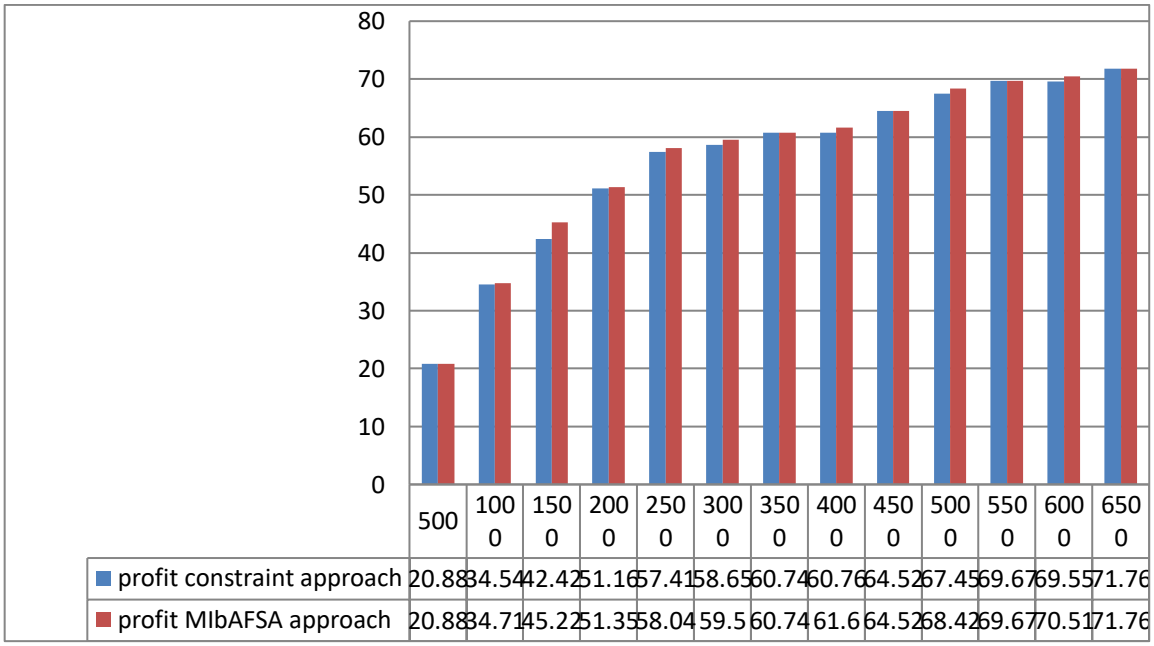


**Figure 4. 16:** Comparative study results

As we can see in the past graph our approach proved that its results are superior or equal to the constraint based method but never less. It showed that it is better than the other approach in 70% of the cases. Our approach proved its efficiency and superiority against the constraint method .

# GENERAL CONCLUSION AND PERSPECTIVES

The work developed in this dissertationis part of the optimization of the physical design of relational data warehouses modeled in a star schema. Given the gigantic size of these warehouses and the complexity of queries called star-join queries which interrogates them, many optimization techniques have been proposed in the literature, the main ones being materialized views, index creation and fragmentation. . Among indexing techniques, binary join indexes have shown their effectiveness in optimizing these queries. On the other hand, the selection of the indexes to create is a NP-complete problem.

In this context, we have proposed a binary join index selection approach based on the AFSA (Artificial Fish Swarm Algorithm) .The contribution of our work is the ability to target the index selection process automatically using our own version of the bAFSA (binary AFSA).Our contribution is to select the optimal index set without violating the storage constraint and ensuring that this set has the best gain ( query workload time response improvement).

We’ve implemented our approach under the form of an index selection tool that we named AFSISbtool (Artificial Fish Swarm Index Selection based tool) allowing the administrator to easily express the storage constraint that we mainly focus on not violating it ensuring that the selected index set has the best gain.

Finally, we experimented our approach through this tool on a test data warehouse. In these experiments we ‘ve fixed the parameters of the algorithm to guarantee the best solution and the best execution time without violating the storage constraint. Also we’ve done a comparative study between our approach and a datamining constraint based approach. And we’ve concluded that our approach with the best parameters proved its superiority to the datamining constraint-based approach.

# PERSPECTIVES

Different perspectives are envisaged:

- In the first place, note that in our approach we did not consider the constraint of index order. So we aim to consider this constraint in the future so we can improve the quality of the solution given by our approach even more than it is right now.
- The benchmark that we used in our tests contains only 12 candidate attributes for indexing so we aim to work with a larger benchmark with hundreds of condidateattributes or even thousands.
- In this dissertation we've only worked on the SBJI(Single binary join index) so mainly in the future we will focus on working on MBJI( multi binary join index) because it is widely used and known in the data warehouses.
- Considering the cardinality of each attribute to select the optimal set of indexes.
- For the AFSISbtool we have developed, the following improvements are possible:
  - Giving the ability the administrator to generate the index creation queries after selecting the optimal index set.
  - The ability to work with MBJI and generating the creation queries .
  - The ability to work with every possible index order because it greatly affect the quality of the solution.
- In the experimental part, a validation of our approach under a DBMS such as Oracle is to be considered.

# BIBLIOGRAPHICAL REFERENCES


**[1]** Inmon, W.H., Hackarton, R.D.: Using the Data Warehouse. John Wiley& Sons, 1994, p3.

**[2]** Vincent Rainardi ,Building a Data Warehouse: With Examples in SQL Server , 2008 , p4.

**[3]** Oracle's Data Warehousing Guide: «https://docs.oracle.com/cd/B10500_01/server.920/a96520/concept.htm#50413 »,consulted in : 15/01/2018

**[4]** Ashadevi. B / International Journal of Engineering and Technology Vol.3 (6), 2011-2012, p447-457

**[5]** Steve S. Daino : DATA WAREHOUSING IN COMPUTER INTEGRATED MANUFACTURING (CIM), 2004,p6.

**[6]** O. Teste, «Modélisation et manipulation d'entrepôts de données complexes et historisées». Thèse de Doctorat en Informatique, Université Paul Sabatier. 2000.

**[7]** A. Bouzeghoub, « Modélisation des Entrepôts de donnéesXML : Application au domaine de la sécurité social », mémoire de magister, Institut National de formation en Informatique (INI), 2008

**[8]** R. Kimball. « Entrepôts de données, Guide pratique du concepteur de datawarehouse». John Wiley and Sons, Inc., 1996.

**[9]** Partitions - Partition Storage Modes and Processing : « https://docs.microsoft.com/en-us/sql/analysis-services/multidimensional-models-olap-logical-cube-objects/partitions-partition-storage-modes-and-processing », consulted in : 20/02/2018

**[10]** Vassiliadis, P., Simistsis, A., Skiadopoulos, S.: Conceptual modeling for ETL processes. National Technical University of Athens, Athens, Greece , 2002 .

**[11]** L. Bellatreche, R. Missaoui, H. Necir et H. Drias, "A Data Mining Approach for Selecting Bitmap Join Indices", Journal of Computing Science and Engineering, vol.2(1), January, 2008, pp. 206-223.

**[12]** E. Ziyati, « Optimisation de requêtes OLAP enEntrepôts de DonnéesApprochebasée sur la fragmentation génétique», », thèse de doctorat, UniversitéMohammed V –AGDAL, Maroc; 2010

**[13]** K. Boukhalfa, « De la conception physique aux outilsd'administration et de tuning des entrepôts de données », Thèse de doctorat, l'écolenationalesupérieure de mécanique et d'aérotechnique, 2009

**[14]** R. Bouchakri, L. Bellatreche et K. Boukhalfa, « Administration et Tuning des Entrepôts de Données :Optimisation par Index de Jointure Binaires et Fragmentation Horizontale ». Doctoriales STIC'09,Msila, Décembre 2009

**[15]** A. Sanjay, V. R. Narasayya, et B. Yang. «Integrating vertical and horizontal partitioning into automated physical database design. », Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 359–370 , June 2004.

**[16]** R. Bouchakri , L. Bellatreche, « On Simplifying Integrated Physical Database Design », 15th East European Conference on Advances in Databases and Information Systems (ADBIS'2011), LNCS, Springer, September, 2011, pp. 333-346

**[17]** R. Bouchakri, « Une approchedirigée par la classification des attributs pour fragmenter et indexer des entrepôts de données », mémoire de magister, écolenationalesupérieured'informatique, 2009)

**[18]** L. Bellatreche, « Techniques d'optimisation des requêtesdans les data warehouses », Sixth International Symposium on Programming and Systems, 2003, pp.81-98.

**[19]** M. Barr, « Approchedirigée par les fourmis pour la fragmentation horizontale des entrepôts de donnéesrelationnels», mémoire de magister, écolenationalesupérieured'informatique, 2009

**[20]** L. Bellatreche, « Dimension Table Selection Strategies to Referential Partition a Fact Table of Relational Data Warehouses », Recent Trends in Information Reuse and Integration Book, Springer, 2012, pp. 19-42.

**[21]** T. Stöhr, H. Märtens, and E. Rahm. « Multidimensional database allocation for parallel data warehouses ». Proceedings of the International Conference on Very Large Databases, pages 273–284, 2000.

**[22]** ARIOUAT Youcef, Une approchebasée sur l'extraction de motifs sous contraintes pour la sélection des Index de Jointure Binaires,MAGISTER, UNIVERSITE AMAR TELIDJI – LAGHOUAT-, 2013.

**[23]** T. Johnson. « Performance measurements of compressed bitmap indices ». Proceedings of the International Conference on Very Large Databases, 1999

**[24]** C. Chee-Yong. « Indexing techniques in decision support systems. » Phd. thesis,University of Wisconsin - Madison, 1999.

**[25]** Red Brick Systems. « Star schema processing for complex queries ». White Paper, July 1997

**[26]** P.E. O'Neil et G. Graefe : « Multi-table joins through bitmapped join indices ». SIGMOD Record, 24(3) :8–11, 1995.

**[27]** J. Darmont, « Optimisation et évaluation de performance pour l'aide à la conception et à l'administration des entrepôts de données complexes » , Habilitation à Diriger des Recherches, Université Lumière Lyon 2 , 2006.

**[28]** H. Gupta, V. Harinarayan, A. Rajaraman, et J. D. Ullman. «Index Selection for OLAP ». In ICDE, pages 208–219, 1997.

**[29]** W. J. Labio, D. Quass, and B. Adelberg. «Physical Database Design for DataWarehouses». In ICDE, pages 277–288,1997.

**[30]** D. Comer , « The difficulty of optimum index selection », ACM Transactions on Database Systems (TODS), 3(4) :440-445. 1978.

**[31]** O'Neil et Graefe, 1995;O'Neil et Quass, 1997.

**[32]** Lemire et al., 2010; Wu et al., 2010.

**[33]** K. Aouiche. «Techniques de fouille de données pour l'optimisationautomatique des performances des entrepôts de données». Ph.d. thesis, Université Lumière Lyon 2 ,December 2005.

**[34]** M.R. Frank, E. Omiecinski, et S.B. Navathe, « Adaptive and automated index selection in RDBMS », 3rd International Conference on Extending Database Technology (EDBT 92), Vienna, Austria ; LNCS, Vol. 580, Springer, Heidelberg, 1992, pp. 277-292.

**[35]** G. Valentin, M. Zuliani, D. Zilio, G. Lohman et A. Skelley : " DB2 advisor : An optimizer smart enough to recommend its own indexes". 16th International Conference on Data Engineering (ICDE 00), San Diego, USA, pages 101–110, 2000.

**[36]** M. Golfarelli, E. Rizzi, et S. Saltarelli. «Index selection for data warehousing». Proceedings 4th International Workshop on Design and Management of Data Warehouses (DMDW'2002), Toronto, Canada, pp 33–42, 2002.

**[37]** S. Chaudhuri, V. Narasayya, « An efficient cost-driven index selection tool for Microsoft sql server » In : Proceedings of the International Conference on Very Large Databases, August 1997, pp. 146–155 (1997).

**[38]** Y.A. Feldman , J. Reouven , « A knowledge_ based approach for index selection in relational databases », Expert System with Applications, 25(1) :15_37. 2003.

**[39]** K. Aouiche, O. Boussaid, et F. Bentayeb, « Automatic Selection of Bitmap Join Indexes in Data Warehouses ». 7th International Conference on Data Warehousing and Knowledge Discovery (DAWAK05), p. 64-73. 2005.

**[40]** S. Taktak, J. Feki : "A maintenance Approach of a BJI Index Configuration" In : ICSEA 2011, The Sixth International Conference on Software Engineering Advances, October 2011, Pages : 221-226

**[41]** B. Ziani, Y. Ouinten, «Versl'auto-sélection des index dans les entrepôts de données: uneapprochebasée sur la recherche des motifs fréquentsmaximaux», 10e ColloqueAfricain sur la Recherche en Informatique et enMathématiquesAppliquées, Yamoussoukro, Côte d'Ivoire, pages 301–308, 2010

**[42]** .L.Bellatreche and K. Boukhalfa, « Yet Another Algorithms for Selecting Bitmap Join Indexes », Proceeding of 12th International Conference on Data Warehousingand Knowledge Discovery (DAWAK'08), Springer-Verlag, September 2010.

**[43]** K. Aouiche, J. Darmont, O. Boussaid, "Sélectionautomatiqued'index dans les entrepôts de données", 1er atelier Fouille de Données Complexes dans un processusd'extraction des connaissances, EGC 04, Clermont-Ferrand, Janvier 2004, 91-102.

**[44]** S. Chaudhuri, M. Datar, et V. Narasayya, « Index Selection for Databases : A Hardness Study and a Principled Heuristic Solution ». IEEE Transactions on Knowledge and Data Engineering, VOL. 16, NO. 11. 2004.

**[45]** T.I. Gundem ., « Near optimal multiple choice index selection for relational databases », Computers & Mathematics with Applications, 37(2) :111_120. 1999.

**[46]** J. Kratica, I. Ljubic, et D. Tosic, « A Genetic Algorithm for the Index Selection Problem». In Applications of EvolutionaryComputing, EvoWorkshop. 2003.

**[47]** K. Boukhalfa, L. Bellatreche et B. Ziani «Index de Jointure Binaires :Stratégies de Sélection& Étude de Performances», 6ème Journées Francophones sur les Entrepots de Données et AnalyseenLigne (EDA10). - Jerba-Tunisie : 2010. - pp. 175–190.

**[48]** M. :LyazidTOUMI ,Thème : Optimisation des performances par administration et tuning d'entrepôt de données, 2015, pp 62,63

**[49]** N. Pasquier, Y. Bastide, R. TaouiletL.Lakhal, « Pruning closed itemset lattices for association rules ». Actes des 14emes journées Bases de DonnéesAvancées BDA'98 , 1998, pp. 177-196

**[50]** C.-H Fung, K. Karlapalem, Q. Li, : « Cost-driven vertical class partitioning for methods in object oriented databases." VLDB Journal 12 (3), 187–210 (2003)

**[51]** B. Ziani, Y. Ouinten, « Mining Maximal Frequent Itemsets : a java implementation of FPMAX algorithm », 6th International Conference on Innovations in Information Technology (IIT09), 2009.

**[52]** J. Roberto Jr. Bayardo Bart Goethals and Mohammed JaveedZaki, FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004

**[53]** LyazidTOUMIa, Abdelouahab MOUSSAOUI, Ahmet UGURb "A linear programming approach for bitmap join indexes selection in data warehouses" , 6th International Conference on Ambient Systems, Networks and Technologies, ANT 2015.

**[54]** Md. AbulKalamAzad ,AnaMariaA.C.Rocha , EditeM.G.P.Fernandes "Improved binary artificial fish swarm algorithm for the 0–1 multidimensional knapsack problems" Algoritmi R&D Centre, School of Engineering, University of Minho, 4710-057 Braga, Portugal, 12-03-2013.

**[55]** C. J. A. B Filho., F. B. de Lima Neto, A. J. C. C..Lins, A. I. S. Nascimento., and M. P. Lima, "A novel search algorithm based on fish school behavior," Systems, Man and Cybernetics, SMC 2008. IEEE International Conference on, 2008, pp. 2646-2651.

**[56]** de Lima Neto, Fernando Buarque, and Marcelo Gomes Pereira de Lacerda."Multimodal Fish School Search Algorithms Based on Local Information for School Splitting." 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence. IEEE, 2013)

**[57]** Computational Intelligence Research Group(CIRG) :http://www.fbln.pro.br/fss/Consulted in : 12-05-2018

**[58]** J. B. Monteiro, I. M. C. Albuquerque, F. B. L. Neto, and F. V. S. Ferreira, "Optimizing multi-plateau functions with FSS-SAR (Stagnation Avoidance Routine)," Submitted to IEEE Symposium Series on Computational Intelligence, 2016.

**[59]** F. Buarque De Lima Neto and M. Gomes Pereira de Lacerda, "Weight based fish school search," in Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on. IEEE, 2014, pp. 270–277.

**[60]** Sargo, João AG, et al. "Binary Fish School Search applied to feature selection: Application to ICU readmissions." 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). IEEE, 2014.

**[61]** Deb, K., Thiele, L., Laumanns, M., &Zitzler, E.(2002) Scalable Multi-Objective Optimization Test Problems, In: IEEE Congress on Evolutionary Computation (pp. 825–830).

**[62]** Nebro, A. J., Durillo, J. J., Garça-Nieto, J., CoelloCoello, C. A., Luna, F., & Alba, E. (2009) SMPSO: A new PSO-based metaheuristic for multi-objective optimization, In: IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (pp. 66–73). doi:10.1109/MCDM.2009.4938830

**[63]** Bastos-Filho, Carmelo JA, and Augusto CS Guimarães. "Multi-Objective Fish School Search." International Journal of Swarm Intelligence Research (IJSIR) 6.1 (2015): 23-40.

**[64]** C.C. Petersen, Computational experience with variants of the balas algorithm applied to thes election of R&D projects, Management Science13 (9) (1967) 736–750.

**[65]** A.M.A.C.Rocha,E.M.G.P.Fernandes,T.F.M.C.Martins,Novel fish swarm heuristics for bound constrained global optimization problems,in: B.Murgante, etal.,(Eds.),Computational Science and Its Applications,ICCSA 2011, PartIII, Lecture Notesin Computer Sciences,vol.6784., Springer-Verlag, Heidelberg, 2011,pp.185–199.

**[66]** C.R. Wang, C.-L.Zhou, J.-W.Ma, An improved artificial fish swarm algorithm and its application in feed forward neural networks, in: Proceeding sof the Fourth International Conference on Machine Learning and Cybernetics, 2005, pp. 2890–2894

**[67]** F.Djannaty, S.Doostdar, A hybrid genetic algorithm for the multidimensional knapsack problem, International Journal of Contemporary Mathematical Sciences 3 (9) (2008) 443-456.

**[68]** D. Zou, L.Gao, S.Li, Z.Wu, Solving0–1 knapsack problem by a novel global harmony search algorithm, Applied Soft Computing 11(2011)1556–1564.

**[69]** O. COUNCIL, " Apb-1 OLAP benchmark, release II" :« Http://www.olapcouncil.org/research/resrchly.htm», Consulted in : 25/4/2018

## ملخص

الاستعلامات التحليلية التي تم تحديدها في مستودع بيانات مبني على مخطط النجمة معقدة للغاية وتستغرق وقتًا طويلاً بسبب عمليات الربط بين جداول الأبعاد والحقيقة. وقد ظهرت عدة تقنيات لتقليل التكلفة وزمن الاستجابة في العقود الماضية ومن أهمها الفهارس الالكترونية.تعتبر فهارس الربط الثنائية BJI ( Binary Join Indexes) إحدى الفهارس المعروفة ، لكن يعتبر اختيارها كمشكلة معقدة. هذه المشكلة أمر بالغ الأهمية في تصميم تخزين البيانات المادية. لحل هذه المشكلة ، يوجد منهجان قائمان على أساس إحصائي ونهج قائم على الاستكشافات. في هذه المذكرة نقترح نهجًا جديدًا يستند إلى metaheuristic. يعتمد هذا النهج على النسخة المحسنة من خوارزمية سرب الأسماك الاصطناعية AFSA لحل مشكلة اختيار فهارس الربط الثنائية. ويهدف هذا النهج إلى اختيار المجموعة المثلى من BJI التي تم قياسكفاءتها على أساس نموذج تكلفة رياضي . تم اختبار هذه الطريقة ضد طريقة تستند إلى القيد datamining وأثبتت فعاليتها بل وحتى تفوقها.



**الكلمات المفتاحية:**

الاستعلامات التحليلية، مستودعالبيانات،فهارسالضمالثنائية، خوارزمية سرب الأسماك الاصطناعية،



## ABSTRACT

Analytical queries defined on a star schema modeled data warehouse are very complex and time consuming due to the join operations between the fact and dimension tables. Several techniques to reduce the cost and response time has been emerged in the past decades such as indexes. Binary Join Indexes (BJI) are one of the well-known indexes and its selection is considered as a problem itself (noted Index Selection Problem: ISP). this problem is crucial in data warehousing physical design. To solve this problem two approaches exists statistics-based approach and metaheuristic-based approach. In this dissertation we propose a new metaheuristic-basedapproach. This approach is based on the improved version of the artificial fish swarm algorithm for solving the binary join index selection problem. This approach aims to select the optimal set of BJI based on a mathematical cost model. This method was tested against a datamining constraint-based method and proved its effectiveness and even its superiority to the datamining constraint method.



**Keywords:**

Analytical queries, Data warehouse, Binary join index, Artificial fish swarm algorithm,

# RESUME

Les requêtesanalytiquesdéfinies sur un entrepôt de donnéesmodéliséenétoilesonttrès complexes et prennent beaucoup de temps à cause des jointures couteuses entre les tables de faits et de dimensions. Plusieurs techniques pour réduire le coût et le temps de réponse existent dans la littérature, parmi eux les index. Les indexes de jointure binaire (BJI) sontl'un des indixes bien qui ont montrés leurs efficacités face à ce type de requêtes. Cependant leursélectionestconsidéréecomme un problème NP-Complet (nommé problème de sélectiond'index, noté:PSI). Ce problèmeest crucial dans la conception physique de l'entrepot de données. Pour résoudreceproblème, deux approches existes :(1) approches basées sur les statistiques et (2) approches basées sur la métaheuristique. Dans ce mémoire, nous proposonsune nouvelle approchemétaheuristique. Cetteapprocheestbasée sur la version améliorée de l'algorithme de l'essaim de poissons artificiels pour résoudre le PSI de jointure binaire. Cetteméthode a ététestée par rapport à uneméthodebasée sur des contraintes de datamining et a prouvé son efficacité et mêmesasupériorité par rapport à la méthode de contrainte de datamining.

**Mots-clés:**

Les requêtesanalytiques, Entrepôt de données, Les indexes de jointure binaire, l'algorithme de l'essaim de poissonartificiel,