DISTRIBUTED QUERY PROCESSING USING ARTIFICIAL IMMUNE SYSTEM

A dissertation submitted to the Jawaharlal Nehru University
in partial fulfillment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE AND TECHNOLOGY

BY

RUBY RANI

School Of Computer and Systems Sciences

Jawaharlal Nehru University

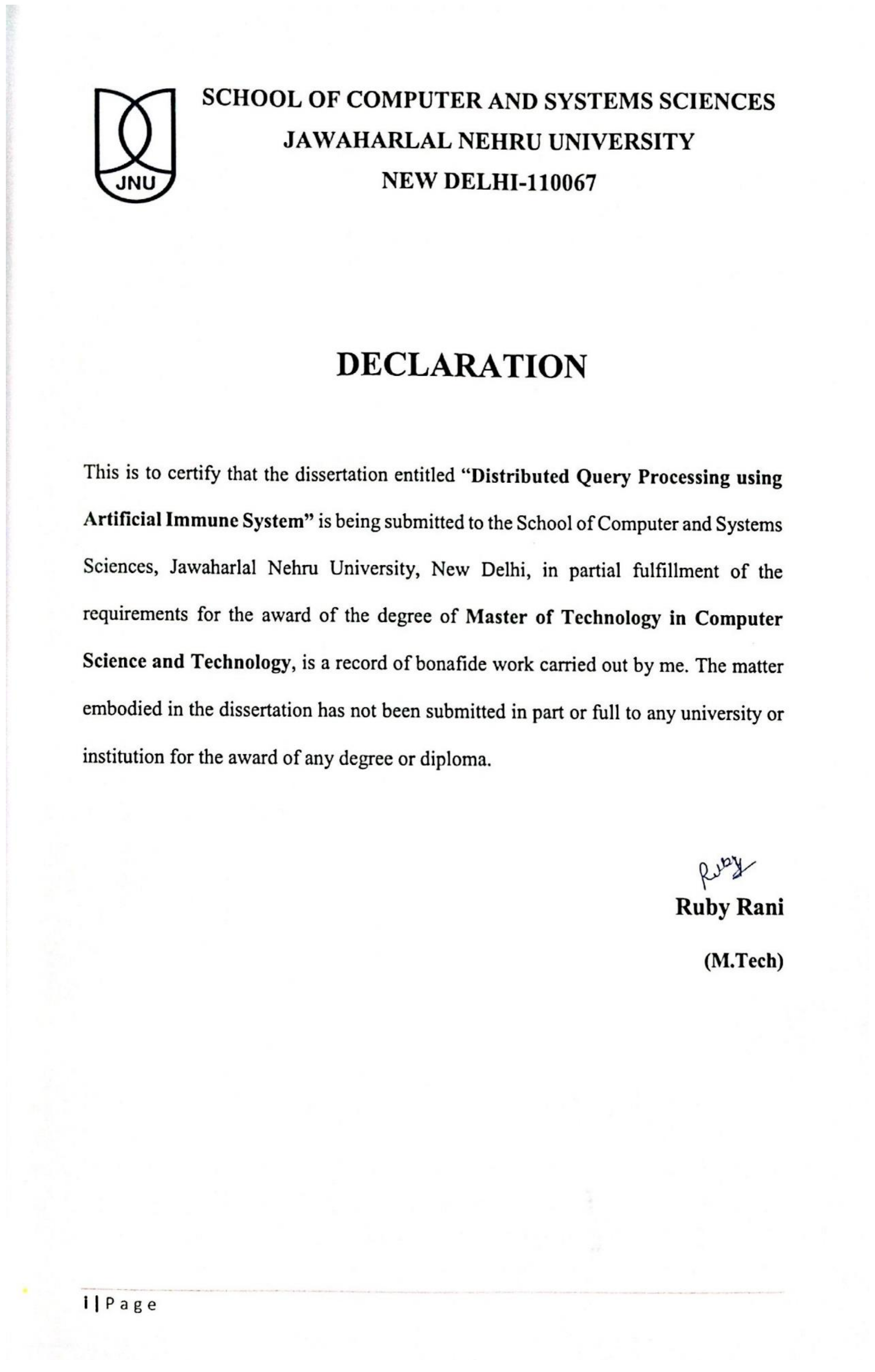
New Delhi-110067

JULY, 2015

Dedicated

To

My Family

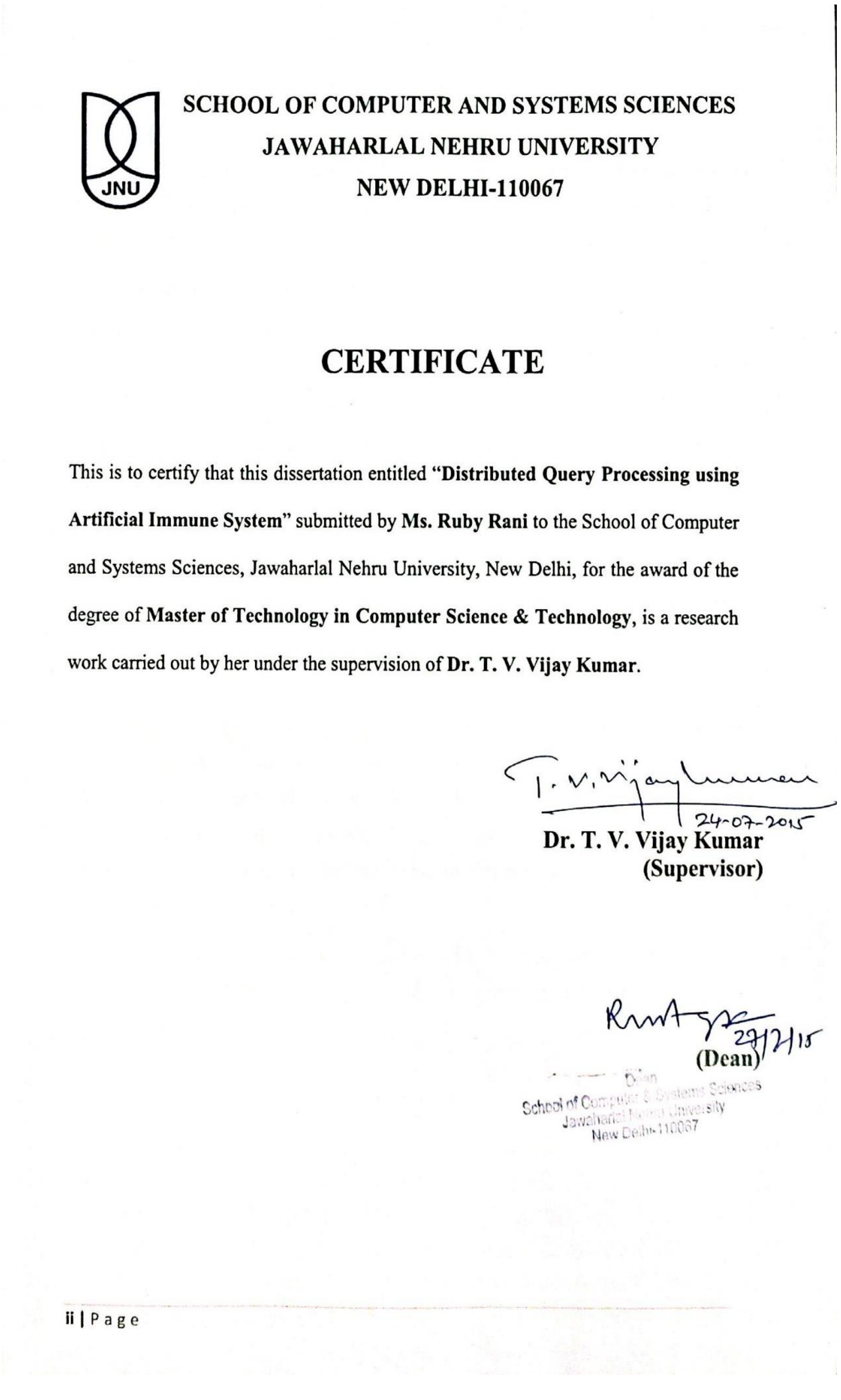


SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI-110067

DECLARATION

This is to certify that the dissertation entitled **"Distributed Query Processing using Artificial Immune System"** is being submitted to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Computer Science and Technology**, is a record of bonafide work carried out by me. The matter embodied in the dissertation has not been submitted in part or full to any university or institution for the award of any degree or diploma.

Ruby
Ruby Rani
(M.Tech)



SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI-110067

CERTIFICATE

This is to certify that this dissertation entitled **"Distributed Query Processing using Artificial Immune System"** submitted by **Ms. Ruby Rani** to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, for the award of the degree of **Master of Technology in Computer Science & Technology**, is a research work carried out by her under the supervision of **Dr. T. V. Vijay Kumar**.

[illegible]
24-07-2015
Dr. T. V. Vijay Kumar
(Supervisor)

[illegible] 27/7/15
(Dean)

Dean
School of Computer & Systems Sciences
Jawaharlal Nehru University
New Delhi-110067

ACKNOWLEDGEMENT

Without the support, patience and guidance of the following people, this study would not have been completed. I owe my gratitude to all those people who have made this dissertation possible

My deepest gratitude is to my supervisor, **Dr. T.V Vijay Kumar** for the great guidance in my dissertation. His patience and support helped me overcome many crisis situations and finish this dissertation. His wisdom, knowledge and commitment to the highest standards inspired and motivated me.

I would also like to thank Dean **Prof. C.P Katti**, **SC&SS, JNU Delhi,** for their support, encouragement and providing a research oriented environment. I would like to express my sincere thanks to **SC & SS staff** for providing all the needed resources and their helpful nature.

I thank my lab mates **Mr. Dilip Kumar**, **Mr. Biri Arun**, **Mr. Jay Prakash Soni**, **Mrs. Neha Singh** and sincere thanks to **Ms. Monika Yadav** for the stimulating discussions and their moral support.

To my friends, thank you for listening, offering me advice, and supporting me through this entire process. Special thanks to **Mr. Ashish Kumar**, **Ms. Hema**, **Mr. Mahender Kumar**, **Mr. Mayank**, **Ms. Nirmal** and **Mr. Sudhakar** for the sleepless nights we were working together before deadlines. The editing advice, general help and friendship were all greatly appreciated and **Robin Canteen**, where most of my research-oriented discussions have taken place.

Last but not the least, I would like to thank my family: my parents, for giving birth to me at the first place, their priceless love and care, their spiritual support throughout my life, their prayers.

Ruby Rani

ABSTRACT

The objective of distributed query processing is to generate efficient and optimized distributed query plans. Optimization of distributed query plan is based on local processing cost, and site to site cost. Among these, Site to site cost is the most major cost dictating the query performance of a distributed database system. Thus, the aim is to generate query plans that have a minimum site to site communication. One such GA based distributed query plan generation (DQPG) algorithm that generates close query plans for a distributed query has been in [VSV10] [VSV11]. This DQPG problem has been addressed using a bioinspired technique artificial immune system (AIS) in this dissertation. In this regard, two AIS based DQPG models $DQPG_{AIS}$-I and $DQPG_{AIS}$-II are proposed that generates Top-K query Plans for a given distributed query. In $DQPG_{AIS}$-I, the mutation rate is computed by the product of fitness value and the number of sites used in the query plan. On the other hand, generated clones of antibody query plans in $DQPG_{AIS}$-II are mutated with a constant rate using Roulette Wheel selection. $DQPG_{AIS}$-I and $DQPG_{AIS}$-II use different measures to compute the number of clones. Further, experimental based comparison of the two proposed models $DQPG_{AIS}$-I and $DQPG_{AIS}$-II with $DQPG_{GA}$ showed that both the proposed models are able to generate query plans having comparatively lower average QPC. The query plans, so generated would lead to efficient processing of distributed queries.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACID	Atomicity, Consistency, Integrity, Durability
AIS	Artificial Immune System
DBMS	Database Management System
DDBMS	Distributed Database Management System
DDBS	Distributed Database System
DQP	Distributed Query Processor
DQPG	Distributed Query Plan Generation
GA	Genetic Algorithm
LAN	Local Area Network
MAN	Metropolitan Area Network
QPC	Query Processing Cost
QP	Query Plan
QPN	Query Plan Number
RDBMS	Relational Database Management System
WAN	Wide Area Network

LIST OF SYMBOLS

R	Number of Relations used in Relation-Site Matrix
S	Number of sites in Relation-Site Matrix
S_i	Number of times i^{th} site used in a query plan
A	Attribute in a Distributed Query
P	Population Size
A_b	Antibody
N	Top Query Plans in DQPGAIS-I & $DQPG_{AIS}$-II
n1	Top antibodies in CLONALG
N	Number of Relations used in a Query Plan
C_n	Total Number of Clones
RS	Relation-Site Matrix
G_P	Maximum Number of Generations
B	Parameter used to compute number of clones
P_{QP}	Roulette Wheel Fitness function
P_m	Mutation Rate
α	A Constant used to calculate β
A_n	Antigens Population

CHAPTER 1

INTRODUCTION

Data is the basic necessity of computer science field to do various operations such as computational operations, mathematical operations for some purpose. So, database is used as a repository to store coherent, interrelated and similar kind of data to fulfill different aspects **[D95][EN10]**. **Database Management System** is a computerized system, which is used to manage database and to process user queries **[D95].** DBMS provides an environment that is most convenient and efficient to retrieve information and process it. DBMS provides an abstract view of the database and does not provide the physical structure of the database, as it is too complex to understand for users **[D95][EN10]**.

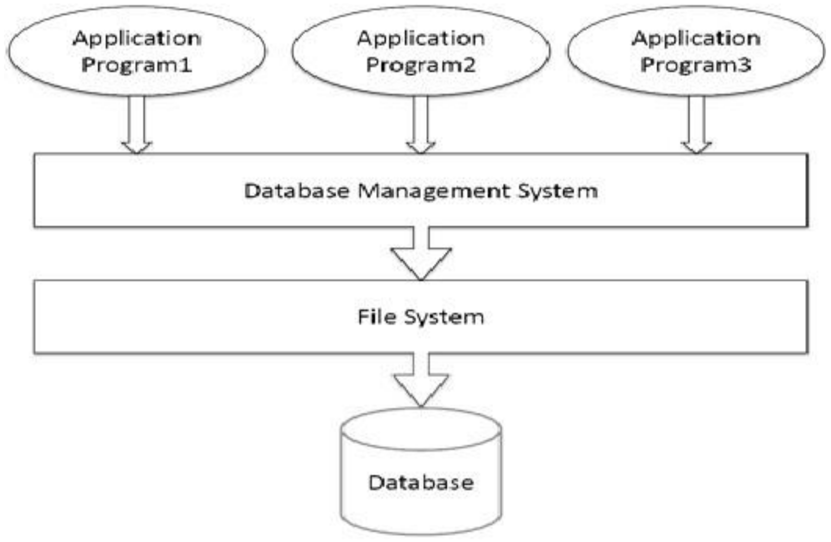


Figure 1.1 Traditional File System [EN10].

Both Traditional file system and Database Management System have been shown in Figure 1.1 and Figure 1.2 respectively. In Traditional File System, data was accessed in a sequential manner, and different copies of data were maintained. But, with time, data grew exponentially. So, database properties such as data integration, data security, database atomicity, consistency, and durability, etc. came into existence and these were difficult to maintain. Thus, the traditional file system was not efficient to serve users' request in an efficient way. To fulfill these requirements, a DBMS was required to keep the data at a place from where data would be accessible according to the requirements of users.

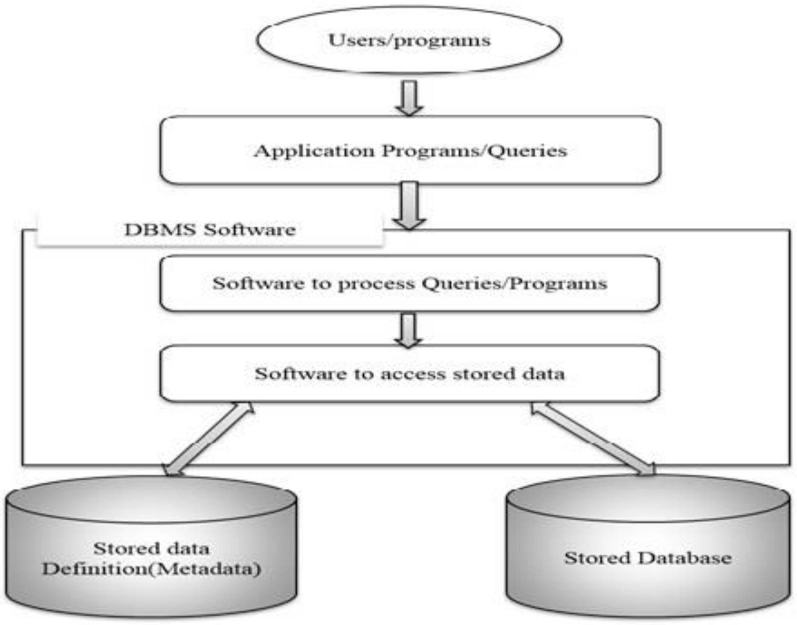

Users/programs
Application Programs/Queries
DBMS Software
Software to process Queries/Programs
Software to access stored data
Stored data Definition(Metadata)
Stored Database

Figure 1.2 Database Management System [EN10].

In RDBMS, data were stored in the form of relations and constraints may be applied as "keys". In the 1980s, a query language known as "Structure Query Language" was developed for RDBMS. The query language was composed of data definition, data manipulation and data control language **[E74] [OV91]**. Later, in the 1980s and 1990s a revolution came in the field of DBMS and various types of database management systems were developed. Some of these are MySQL, Oracle, MS-Access, DB2, SQL Server, RDBMS, dBase etc **[OV91][OV11].** These were centralized database management systems having central control. In central database, a single site was used to serve various users requests in different forms depending on types of requests. The central mainframe, as shown in Figure 1.3, was treated as a database server and clients used it from different terminals of LAN, WAN, MAN **[OV91]**.

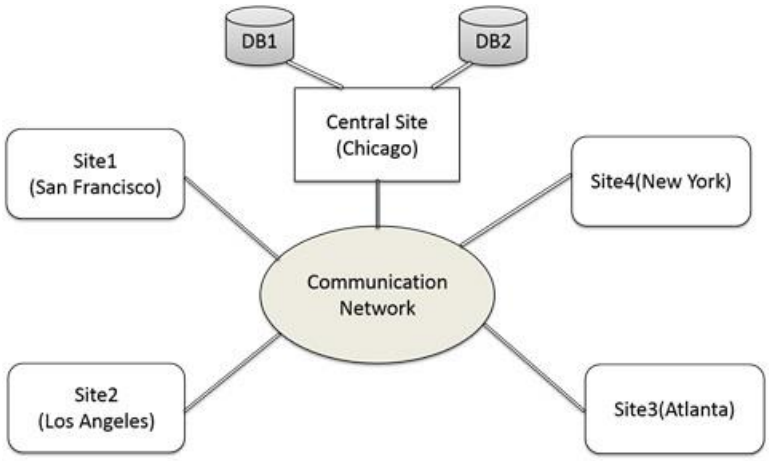


Figure 1.3 Centralized Database Systems [EN10].

These kinds of systems were very helpful in managing the data at a single place as well as security and integrity of data could be easily maintained. The most critical problem was the failure of the centralized server resulting in the loss of that loss of the whole data stored on the central server **[OV91]**. There was no alternative to retrieve the lost data, so this caused the entire system to fail. This made the entire system less reliable. Even communication cost for a remote terminal was very high and traffic over the network was also increased. So, solution to these major problems were provided by a distributed database management system, which is discussed in the next section **[SG98][SL90]**.

1.1. Distributed Database Systems

Distributed database, which is shown in Figure 1.4, is a logically interrelated database dispersed physically across the world on different nodes through a computer network. Distributed database is administered centrally but gives local flexibility and customization. In distributed database, there should be a central body that manages the communication among database instances at different sites **[OV91][SL90]**.

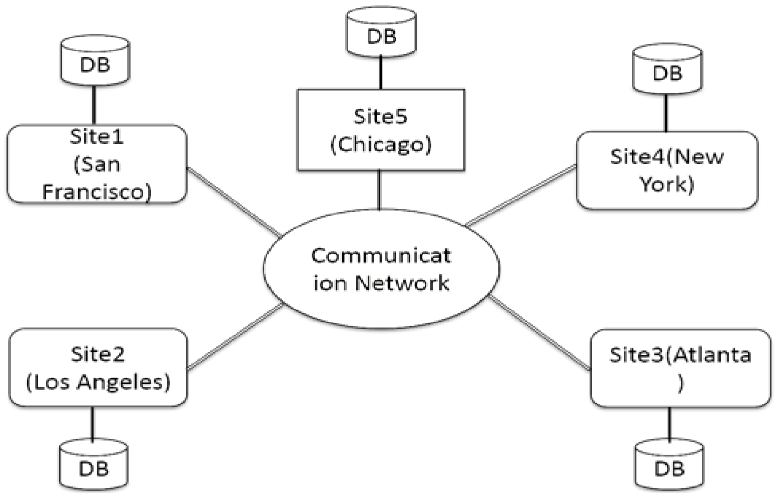


Figure 1.4 Distributed Database Systems [EN10].

1.1.1 Objectives of DDBMS

There are various objectives of DDBMS, whose main purpose is the ease of data access to users at different nodes worldwide some of which are explained below.

Location transparency: The user working on a data need not to know the location of the data. The requested query should be answered automatically if requested data is present at various sites without the intervention of the user. In ideal case, a single query may fulfill the request by joining data from different relations present on various sites as if the whole data are present on a single site**[OV91][OZ97][CP84]**.

Local Autonomy: A local site should administer itself, if central node fails. It should provide the whole access of data to all local users and should manage data security at local level. If local site fails, then recovery of data must be there **[OZ97][CP84]**.

Replication Transparency: In distributed database system, replicated data is present across various sites. Regardless of replicated data, design goal of DBMS says that the developer should treat the whole data at a single node. It is also known as fragmentation transparency. To guarantee it, data integrity is maintained, concurrency transparency and failure transparency must be preserved **[OV11][OZ97][CP84]**.

Failure Transparency: Each node in DDBMS can be affected by same failure, as in centralized systems. An additional failure, known as communication links failure, occurs in DDBMS. A system can be robust only in a situation, if it can detect the failure and reconfigure the whole system so that the computation should be in process and can be recovered with the recovery of a link. In failure transparency, either all the operations of the transaction must complete or none is committed. The integrity of data is maintained by commit protocol **[OV11][CP84]**.

Concurrency Transparency: DDBMS should be designed in such a way that although a distributed system runs many transactions in parallel, but it should appear that only one transaction is in the process, and the result must come out to be the same, as it comes when transactions run in a serial order. It means that there must not be any interference with the other transactions running simultaneously **[OV11] [OZ97]**.

Query Optimization: In distributed database systems, query response is collection of the results from different sites based on the query complexity, availability of data at different sites and how the data is distributed across the network. This whole process is unknown to the user because of the location transparent property **[CHF+10]**. Processing of whole relation is more difficult and time consuming, if only a small part of it is required. However, if the query is processed based on replication and fragmentation; then it is easier and more efficient **[K00][MHH09]**.

Distributed Transaction Management**:** Distributed database transactions should be managed correctly. For this purpose, there is a manager known as transaction manager that manages the transaction logs before and after database changes. The transaction manager also manages the concurrency control and integrity of data during concurrent execution of the transactions **[D95][EN10][OV91][OV11][OZ97]**.

Platform Independence: DDBMS works on different platforms such as on different hardware platforms, on different operating systems and also supports variety of communication networks **[OV11]**.

1.2. DISTRIBUTED QUERY PROCESSING

In distributed database system, query processing may require responses from various sites, which is different from query processing in the centralized system. The response from different sites has to be assembled at a single site. The major decision to be taken in distributed query processing is to formulate a query and intelligence of DDBMS to produce an effective plan for query processing. This computation is done by a DBMS module, known as query processor **[YC84][MHH09]**. Distributed query processing with different phases is shown in Figure 1.5. Distributed Query Optimization involves the processing and retrieval of data from participating sites dispersed physically **[K00][XY10]**. Objectives of Distributed Query Optimization are described in section 1.4.

1.2.1. Phases in DQP

In DDBMS, distributed query processing takes place in four phases such as query decomposition, data localization, global optimization and distributed execution. The task for the first three phases is carried out by central control site with the help of schema information of the global directory. In coordination with the central node, a particular node performs the work of the fourth phase **[SBM98][K00][WY91][YC84]**. These phases are discussed briefly as follows.

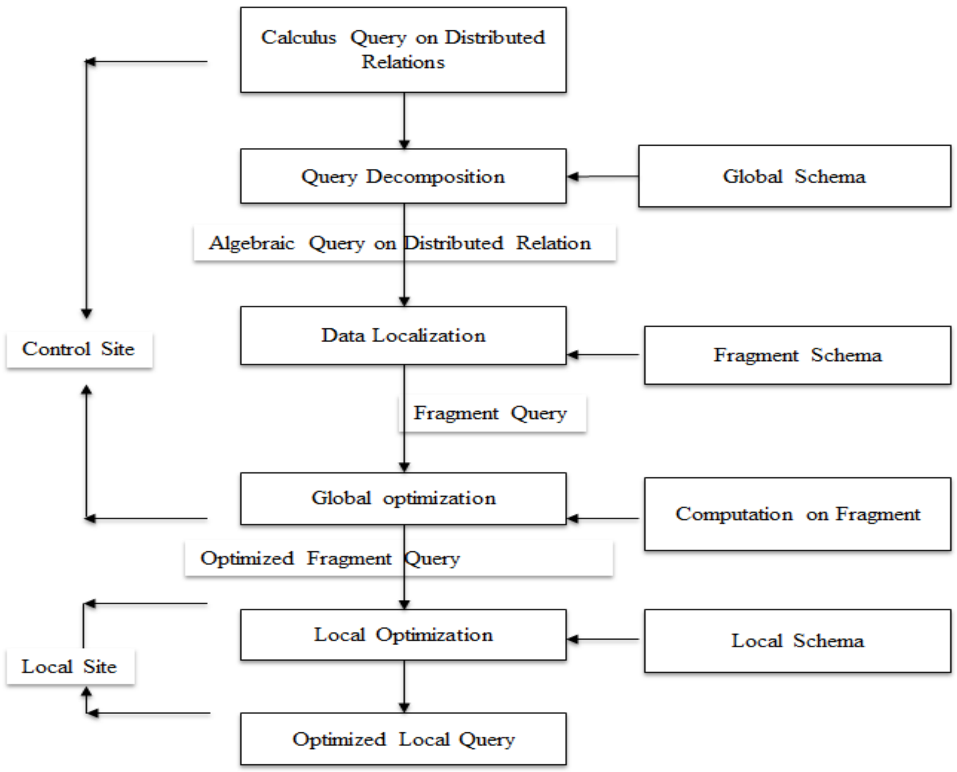


Figure 1.5 Distributed Query Optimization Processing [MHH09].

Query Decomposition: This phase translates the input calculus query into an algebraic query with the help of information on global relations stored in global conceptual schema **[WY91][YC84]**. Query decomposition takes place in four steps. The first step is "**Normalization**", in which query is translated into a normal form, which is more appropriate for subsequent manipulations **[K00]**. All logical operators are arranged in an order. After normalization, query is *"analyzed"* and incorrect queries are detected and rejected by the semantic analyzer followed by parsing the correct queries to the next level in simple form **[K00]**. Redundant queries can be generated when transformations are applied to the query and removed in simplification step followed by conversion in algebraic Query from. There can be various algebraic forms of a query, in which some are better than others. So the best way to get a better algebraic form is to start with an initial query and transform it suitably, to get a better algebraic query by applying suitable rules of transformation **[SBM98][ZHW05]**.

Data Localization: Data is stored on different sites using the fragment schema that contains the information of data distributed over a network. In this layer, the fragments used in a query are selected, and accordingly query transformation takes place which consists of fragments. There are two steps to produce a query comprising of fragments. The first step is to transform resultant of the first phase i.e. an algebraic query in relations, into small query fragments. In the second step, fragmented query is simplified and restructured to obtain an improved query **[EN10][OV11] [SBM98]**.

Global Query Optimization: Global query optimization phase uses communication operators and fragment characteristics to optimize the query. Many equivalent queries can be obtained by permutation of relational operators in a query fragment. Communication cost is a major factor of the cost function. Query optimization minimizes the cost function by using a suitable ordering of relational operators in a fragment query. A pre-computation of query fragment is required to find suitable ordering of relational operators. Ordering computation of relational operators is done by fragment statistics and results cardinalities is estimated by relational operators. So, the fragment allocation and fragment statistics are useful in optimal decision making **[ZHW05]**. The sequence ordering of join operations is also an important aspect of query optimization. The permutation of join operation orders in the user query may lead to fast improvement. Semi-join operator optimizes distributed join operator sequence. In normal join operation in DDBMS, entire Table has to be transferred to a remote computer and Join operation is performed. Join operation involves the transfer of many unrelated rows, which are eliminated by the Join operation. However, communication cost is based on for the amount of data transferred. Semi join significantly prevents unnecessary data transmission during execution of a distributed query, which would result in reduced communication cost **[CY93]**. An optimized algebraic fragment query with embedded communication operators in fragments is produced and is saved as a distributed query execution plan**[SBM98] [XY10][ZHW05]**.

Distributed Query Execution: Execution operation is performed by all the sites involved in distributed fragment query. Sub-queries are executed at local sites by using local schema of the local site. These queries execute in parallel at participating sites and produce optimized result **[SBM98] [ZHW05]**.

1.3. Distributed Query Processor

The main purpose of distributed query processor is to convert a query into small queries on fragmented relations in local data sources. Distributed query processors convert high-level language query plans into low-level language query plans. The input language is relational calculus, and the returned output query is in relational algebra form. Query processor should do a correct mapping from an input language to the output language. Query processing efficiency can vary from system to system **[OV11]**.

1.3.1. Type of Optimization

The most important aspect of query optimization is to choose the best query plan from a large search space of query execution plans. Query execution plan selected by exhaustive search has a large optimization cost **[MHH09][XY10][ZHW05]**. To minimize the optimization cost, some strategies using iterative improvement, and simulated annealing has been proposed. These strategies try to find not the best, but, surely near optimal query plans. An important way in DDBMS is to use semi-joins to reduce the volume of data transfers between intermediate sites **[OV11][CY93]**.

Static query optimization, *dynamic query optimization*, and *hybrid* are the three types of query optimization **[ZHW05]**. In static query plan, optimization is done at compilation time. It reduces the overall query execution time. The other approach is dynamic query plan optimization in which optimization is done in an execution phase, and it minimizes the probability of bad query selection because the size of actual relation is known to the user. However, query execution cost is high in this approach. So, it should be used only for ad-hoc queries, not for repeated queries **[XY10]**. The third approach is a hybrid, a combination of static and dynamic query optimization. Unless there is a large difference between the actual size and estimated size of relations, uses static query optimization is used **[OV11]**. The features of the distributed query processors are explained below.

Statistics: It is used by dynamic query optimizer to arrange the operators according to their execution order. However, static query optimizer uses it to estimate the size of intermediate relations. Database statistics can take different detailed phases and is maintained by frequently updating it **[OV11]**.

Decision Sites: In most DDBS, centralized decision model is used in which one site takes all decisions as the choice of execution plan. Decision making could also be dispersed over several sites to search the best query plan. In DDBMS model, single node has to keep local information only **[OV11]**.

Exploitation of Network Topology: In wide area network, the cost function has to be restricted by data communication cost, a ruling factor **[ZHW05]**. On the basis of this assumption, this problem can be divided into two problems: First, a selection of global query plan based on the inter-site communication cost and second, a selection of local query plans based on the local query processing algorithms. In local area network, only I/O cost is involved, so, it can be used for parallel execution. In the client-server model, client systems are operated by data shipping to perform database operators. So, the optimization problem in such case is to decide which part of the problem has to be solved at client side and which part at the server side **[OV11]**.

Exploitation of Replicated Fragments: A distributed relation is divided into different relation fragments and is replicated on local sites using localization process. This replication is done to increase the reliability and better reading performance. It is considered an independent optimization process by some algorithms and is implemented in order to minimize the communication cost at run time **[OV11][ESW78].**

1.3.2. Use of Semi-joins

Semi-joins help in reducing the data to be communicated between the first and the second site. There might be a situation when the join key attribute from the first site matches completely with the second site **[BC81]**. In this case, the whole file will have to be sent for the join and this will result in increasing data communication overhead instead of decreasing it **[CY92]**. There are some joins, which are not helpful in minimizing the communication overhead but help in to minimize the overall execution cost of the query. These types of semi-joins are known as Gainful. Thus, these will reduce the distributed join query processing communication cost **[RM71][CY92].**

1.4. THE DQPG

The distributed query plan generation problem discussed in **[VSV11]** is the main focus of this dissertation. Based on the problem discussed in **[VSV11]**, query plans are generated based on the property of closeness. The property of closeness of a query plan is the number of sites participating in a query plan and the concentration (number) of relations used by the query, in the participating sites. Lesser the number of sites participating in the query plan and greater the number (concentration) of relation in these sites, closer would be the query plan. Inter-site communication in closer query plan would be less. So, the efficiency of distributed query processing would become more efficient.

For example, consider relations R1, R2, R3, R4, R5 and R6 accessed by a user query. Let us consider there are six sites in DDBS, namely S1, S2, S3, S4, S5 and S6. Relations and their respective host sites are shown in the Table 1.1. Let us consider the following relational query:

```
Select A1, A2, A3
From R1, R2, R3, R4, R5, R6
Where R1.A1=R2.A1 and R3.A2=R4.A2 and R5.A3=R6.A3.
```

Relation\ Site	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**
R1	0	1	1	1	1	1
R2	1	1	0	1	1	0
R3	1	0	1	0	1	0
R4	0	0	0	1	1	0
R5	0	1	1	1	1	0
R6	0	0	0	0	1	1

Table 1.1 Relations along with their Sites.

In Table 1.1, the presence of '1' depicts the presence of relation in the corresponding site and '0' indicates otherwise. There are S sites and R relations. Few of the Query Plans are shown below in Table 1.2.

Query Plan	Relations					
	R1	R2	R3	R4	R5	R6
1	2	1	1	4	3	6
2	3	2	3	4	4	5
3	4	4	5	4	4	5
4	5	5	5	5	5	5
5	2	1	3	4	5	6
6	2	2	5	5	2	5
7	6	2	3	4	5	6
8	3	4	1	5	2	6
9	4	5	3	5	3	5
10	6	1	5	4	5	6

Table 1.2 Valid Query Plans.

These numbers of relation are commonly present in the DDBS. Exhaustive search is not a good approach to pick a query plan with minimum communication cost, but the aim is to produce close query plans with minimum cost. This cost of query plans can be computed in terms of query proximity cost defined in **[VSV11].** The computation method for QPC is given below.

$$QPC = \sum_{i=1}^{S} \frac{S_i}{N}\left(1 - \frac{S_i}{N}\right)$$

Where, S: number of sites participated in the query

N: number of relations participated in the query

S_i: number of times site *i* is used in the query

The value of QPC lies between 0 and 1. Lesser the value of QPC lesser would be the cost of communication i.e. closer the query plan. QPC value of some of the query plans is given in Table 1.3.

QPN	Query Plan	QPC
X1	[5,6,4,4,1,2]	0.7778
X2	[5,1,2,6,4,4]	0.7778
X3	[3,1,1,6,3,2]	0.7222
X4	[5,4,4,6,4,4]	0.5000
X5	[3,2,2,2,3,1]	0.6111
X6	[1,2,4,5,2,4]	0.7222
X7	[6,6,2,2,3,4]	0.7222
X8	[3,3,1,4,3,4]	0.6111
X9	[1,3,6,5,3,1]	0.7222
X10	[3,4,4,2,2,4]	0.6111

Table 1.3 Query Plans with QPC.

1.5. THE AIM OF DISSERTATION

The aim of this dissertation is to solve the problem of distributed query processing using Artificial Immune System given by L.N. De Castro **[LJ02]**. This algorithm is used to generate close query plans. This algorithm is inspired by the natural human immune system**[FPP86]**. In AIS, antibodies represent the query plans, and fitness of these antibodies is computed using QPC. This algorithm is used to solve the DQPG problem. Further, the AIS based DQPG algorithm is compared with GA-based DQPG algorithm **[VSV10][VSV11].**

1.6. OUTLINE OF THE DISSERTATION

This dissertation is organized in the following manner: The Distributed query plan generation (DQPG) problem is described in Chapter 1. In chapter 2, Artificial Immune System algorithm is discussed. Chapter 3 discusses the DQPG using AIS. The conclusion is given in Chapter 4.

CHAPTER 2

ARTIFICIAL IMMUNE SYSTEM

In this chapter, nature-inspired techniques and its classifications are described briefly. Among all these classifications, bio-inspired techniques are emphasized. Artificial Immune System is explained in detail under bio-inspired techniques in the later section of this chapter.

2.1. NATURE INSPIRED TECHNIQUES

Nature always has been a great source of inspiration for human beings. Nature provided techniques tell a human that how to behave against various complex and dynamic problems in real life. These techniques are known as Nature Inspired Techniques **[FM08]**. Nature inspired techniques are based on the principle of self-organization, collective behavior and complex systems **[M94]**. These techniques take an idea from nature and use that idea to develop new techniques, algorithms, and some more computational applications **[H75]**. Nature Inspired techniques provide a novel and better and optimal solution to NP-hard problems in an efficient manner **[M94]**. These techniques are used in various fields such as engineering, physics, and economy management. Nature inspired techniques have various branches such as evolutionary algorithms, swarm intelligence techniques, neural networks, robotics, etc. The complexity of problems is increasing with the increase in the size of computational systems. So, it is hard to predict and control these systems **[H75].**

There are many examples of nature-inspired techniques such as natural ant shows collective behavior in ant colony optimization, Artificial bee colony has good exploration and exploitation ability, Bacteria Foraging is helpful in searching and designing routing algorithms, fish schooling is very advantageous in foraging, birds flocking is used for visualizing tasks and also for optimization tasks, etc. **[H75][HHC+97][MDP+13][M94].**

2.1.1. Nature inspired techniques categories

Nature inspired techniques have many applications in various fields. So, based on these applications, Nature inspired techniques can be categorized as shown in Figure 2.1.

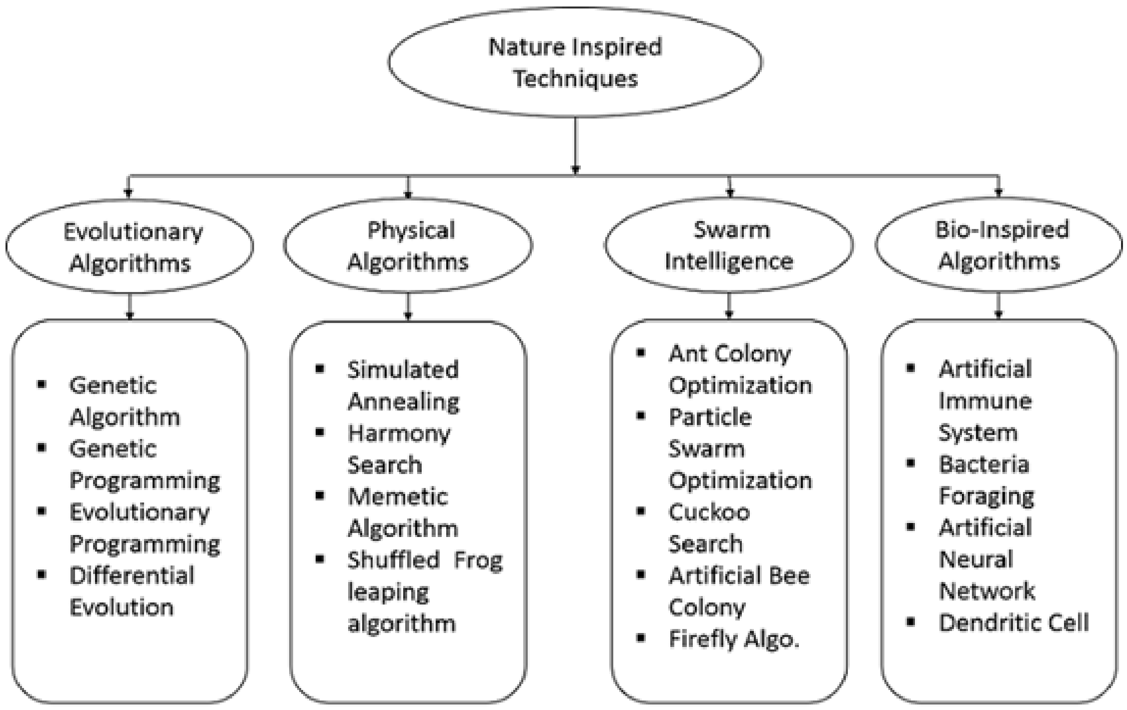


Figure 2.1 Classification of Nature-inspired Techniques [L06].

2.1.1.1. Evolutionary Algorithms

Evolutionary algorithms are inspired by the Darwin's theory of survival of the fittest. These population-based algorithms are the subset of Evolutionary computation **[H75][G85]**.

Some evolutionary algorithms are Genetic algorithms, Genetic Programming, Evolutionary Programming, etc. Among these Genetic algorithm has been discussed below **[FM08].**

Genetic Algorithm: Genetic Algorithm was invented and developed by John Holland and his colleagues in the 1960s **[M90]**. The main goal of the genetic algorithm was to understand the adaptation of natural phenomenon in computer science fields **[M90]**. Genetic Algorithm is used to find the solution for complex problems or the problems having no exact solution **[ML94]**. So, the genetic algorithm provides not best but acceptable solution based on some requirements and restricted to some conditions. According to Holland, Genetic Algorithm is a method that moves from one generation of 'chromosomes population' to next generation based on 'natural selection' together with the genetic operators such as selection, mutation, crossover and inversion **[M90]**.The selection operator selects chromosomes; those can reproduce. Fitter chromosomes are likely to produce fitter offsprings. Crossover operator exchanges two subparts of chromosomes similar to the biological recombination operator between two organisms. Mutation operator changes the allele values at some locations. The inversion operator reverses the previous order of chromosomes arrangement. Mutated chromosomes are added into the initial population. Then, the best chromosomes are selected from the whole population based on their 'Fitness' **[M90] [ML94][G98]**.

2.1.1.2. Physical Algorithms

Physical Algorithms are nature inspired and belong to meta-heuristic and computational intelligence fields. These algorithms are based on physical phenomena such as music, culture interplay, evolution and complex dynamic systems. These are stochastic optimization algorithms based on local and global search techniques **[FJY+13]**. Some of these algorithms are Harmony Search Algorithm, Simulated Annealing, and Memetic Algorithm **[VBS+11]**.

2.1.1.3. Swarm Intelligence:

A swarm is a homogenous collection of agents such as birds, animals; insects, fishes, etc. are interacting with each other in their environment in decentralized manner doing some intelligent task **[M94]**. Swarm Intelligence is the field of artificial intelligence that studies the collective behavior and emergent behavior of self-organized, complex and decentralized systems with social behavior **[EMG99]**. Some of the popular swarm based techniques are Ant Colony Optimization, Particle Swarm Optimization, Artificial Bee Colony, Firefly Algorithm and Cuckoo Search Algorithm, etc. **[EMG99] [MDP+13][M94].**

2.1.1.4. Bio-inspired Algorithms

Bio-inspired Algorithms are based on bio-inspired computing **[VBS+11]**. These algorithms are related to the fields of connectionism, social behavior, and emergence **[H86]**. It is very much related to artificial intelligence, field of mathematics, self-organized, decentralized behavior, nature adaptation and distributed are the intelligence derived by these algorithms from the bio-inspired phenomenon. These algorithms are used to solve partition clustering problem **[FM08]**. Some of the popular bio-inspired algorithms are described as given below **[MTM11][VBS+11].**

Artificial Neural Network: This is a bio-inspired technique and a subfield of Artificial Intelligence **[H86]**. It follows the principle of attention, perception and memory emerging in memory. The main objective of this technique is to acquire knowledge from the environment. This algorithm has the property of self-learning from experience, it gets from the environment **[H75][H86]**. Experience reflects in neurons, and our memory learns from experience **[FM08] [H75].**

Bacteria Foraging Optimization: This approach was introduced by Passino in 2002. This algorithm imitates the foraging strategy of bacteria for finding food. It is based on two types of movements. One is run, and another is a tumble. This algorithm provides good results in dynamic and multimodal environments. It is based on four steps i.e. Chemotaxis, Swarming, reproduction, elimination, and dispersal. Chemotaxis is a movement of bacteria. Swarming represents a cell-to-cell signaling scheme of bacteria and then after some iteration reproduction takes place. In elimination, first half of the bacteria is retained and the second half dies. This technique has been successfully implemented on sensors in wireless networks to enhance coverage and connectivity. This is also used in Clustering **[SAS+09].**

Artificial Immune System: Artificial Immune System is a paradigm used to perform pattern recognition. This technique was proposed by L.N. de Castro and J. Timmis in 2002 **[LJ02].** It is a bio-inspired technique. It is used to solve the most complex computational problems such as **[LJ02][HC96].** Artificial Immune System algorithms use antibodies that fully specify the solution to an optimization, learning, pattern recognition problem and NP

Class problem. Artificial Immune System has been used to solve the DQPG problem discussed in Chapter 1. This technique has been explained in detail in the following section.

2.2. IMMUNE SYSTEM

In many areas of engineering, biological systems are used as a tremendous source of inspiration. In this section, biological immune system has been discussed and based on biological immune system artificial immune system is discussed later in this section.

2.2.1. Biological Immune system

The natural immune system is one of the most complex and intricate biological systems. It is a distributed system, having no central controller **[YZH+08]**. The immune system supports diversification i.e. it does not focus on global optima, instead, it produces antibodies that can deal with different antigens. Natural Immune system is made up of cells, molecules and organs, which have the capability to differentiate self-cells and non-self-cells, but Lymphocytes are a particular interest of Artificial Immune Systems. **[A89][YZH+08].** Lymphocytes are also known as White Blood Cells. These are the antigen detectors of the immune system **[YZH+08]**. Antigens are harmful elements to the body. There are two types of lymphocytes such as T-cells and B-cells. These two types of cells have an effective role in the immune system. B cells can recognize antigens in a free solution like in the blood stream while T cells can recognize antigens if they are present in other accessory cells. Unlike antigens, antibodies are useful to the body **[HC96] [LJ02].**

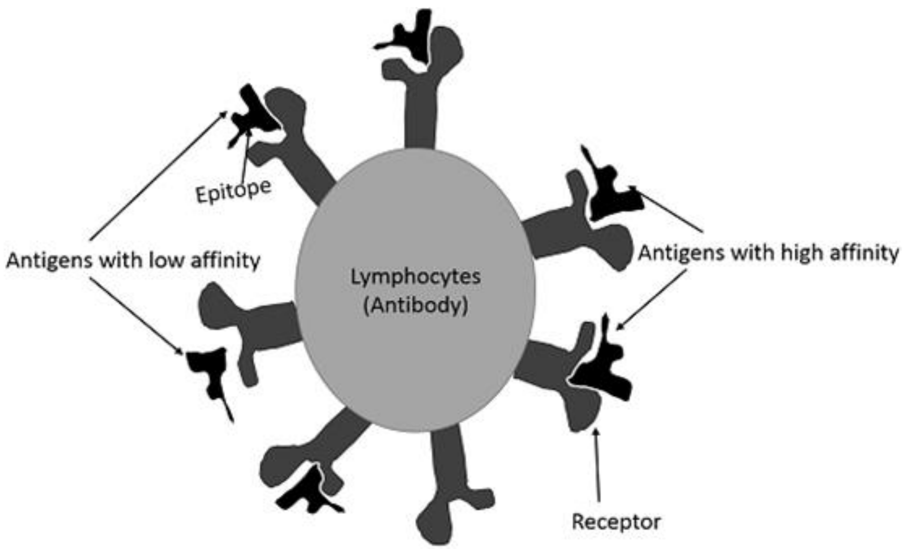


Figure 2.2 Binding between Antigen and Antibody [HC96]

One of the most important features of the immune system is the generation of millions of antibodies from hundreds of antibodies. These antibodies are B-Cells in real and collectively form an immune network **[YZH+08]**. These cells are generated in bone marrow and ensure that once they are generated, they remain in the immune system only until they are not required. When a B-cell encounters with an antigen, a response in the immune system is elicited. As a result, antibody binds with the antigen, as shown in Figure 2.2 and the antigen is neutralized. If B-cell binds with an antigen with sufficient affinity, the B-cell generates mutated clones and is added to the immune network. Diversity in the immune system is maintained. This is because 5% least stimulated antigens die every day and are replaced by an equal number of new antibodies produced by bone marrow, only, if they have affinity to the cells already in it else they die **[HC96][GR94].**

2.2.2. Response in the immune system

There are two types of consideration on creation of an immune system memory **[GR94]**. The most widely accepted theory is that a 'virgin' B cell is stimulated by antigen and produce memory cells and effector cells. The other less accepted theory states that the immune network is dynamic in nature. This means that if something has been learned and is not being used for a long time; it can be forgotten. Here the immune network theory **[HC96][GR94]** has been chosen**.** The *secondary response* states that when the same antigen attacks again, it can be recognized more rapidly and thus results in more number of antibodies production. The secondary response can be elicited from an antigen that need not to be exactly same but, should be similar to the original one that originates in the memory. Thus, the immune system has a content addressable memory **[HC96][GR94].** Artificial Immune System (AIS) mimics the genetic mechanism used to produce antibodies, for antigen/antibody binding and its immune network theory used for its self-organization. This is explained in detail in the next section **[HC96] [LJ02].**

2.3. BIOLOGICAL INSPIRED AIS

AIS is inspired by Natural Immune System and implements learning technique. It is a remarkable property of AIS, which is used to learn foreign agents. AIS can be defined as "An abstract or metaphorical computational system developed using ideas, theories, and components, extracted from the immune system" **[LJ02].** The aim of AIS is to solve complex engineering problems. For example pattern recognition, elimination, and optimization **[HC96][CZ99].**

2.3.1. The Bone Marrow Object

The Bone Marrow Objects perform functions of bone marrow in the body. It decides where an antigen is to be inserted with in the immune network. Along with this, it decides which B cells have to die and which B cells have to proliferate in the immune network **[HC96].**

2.3.2. B cell objects

The B cell objects possess pattern matching mechanism that mimics the genetic mechanisms in which new antibodies are formed in human immune system. These new antibodies attempt to mirror genetic mechanism of gene selection, mutation, proliferation and combine newly produced B cells to the initial population of B objects **[HC96].**

2.3.3. Antibodies

Antibodies bind to the infectious agents known as antigens and destroy these elements in the body. In Artificial Immune System, the antibody contains a *receptor* on its surface. *Receptor* represents a pattern with which the antibody binds with the antigen. When an antigen is encountered with the antibody, a response is elicited. The elicited response is used to calculate the match score i.e. the amount of affinity with which the antibody is attached to the antigen. If match score is equal or above a threshold value, then the antibody attaches to the antigen based on binding strength **[HC96][LJ02].**

2.3.4. Antigen

Antigens are the foreign agents to the body. These are harmful to the body. Antigens induce a response to the body by binding with the antibodies present in the body. These are also known as non-self-agents to the body. The *Paratopes* present on the surface of antigen also known as *epitopes*. *Epitopes* are recognized by receptors present on the surface of the B cells, known as antibodies **[HC96][LJ02].**

2.3.5. Antibody/Antigen Discrimination

Each antibody and antigen possess receptors and epitopes on their surface respectively. Antibodies identify the antigen by complimentary pattern matching operation between the receptor and epitopes **[HC96]**. The binding affinity depends on the closeness of matching of the both. More the closeness between the antigen and antibody, stronger is the binding between molecules and better is the identification. The stimulation level of B cells should be equal to or greater than a threshold, as shown in Figure 2.3, only then the binding between antigen and antibody can take place and B cell replicates and creates new B cells and vice versa **[LJ02][N93][SAL+94][SC92].** Antibodies can also recognize other antibodies present in the body. In this case, receptor present on the antibodies acts as both receptors on one antibody and epitopes on the other antigen. Self-antigens are recognized by T-cells. An organ 'thymus' present in the immune system handles maturation of T cells. During this maturation, the T cells recognize self-antigens, which are excluded from the population of T cells.

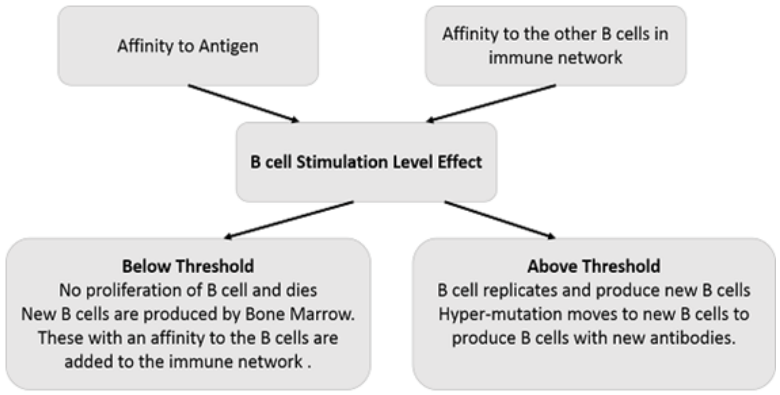

Affinity to Antigen
Affinity to the other B cells in immune network
B cell Stimulation Level Effect
Below Threshold
No proliferation of B cell and dies
New B cells are produced by Bone Marrow.
These with an affinity to the B cells are added to the immune network .
Above Threshold
B cell replicates and produce new B cells
Hyper-mutation moves to new B cells to produce B cells with new antibodies.

Figure 2.3 B cell Stimulation Level Effect [HC96].

This process is known as *negative selection* **[NG94][HC96][LJ02]**. If a B cell recognizes antigens with sufficient affinity, it proliferates and differentiates into the memory. This process is known as *clonal selection* **[LJ02][CZ00]**. *Network theory* process is opposite to the clonal selection theory, in which self-antigen is identified and might result in suppression **[LJ02]**. The clonal selection is described in detail in this chapter.

2.4. CLONAL SELECTION

Clonal selection theory explains about immune system response mounted when an antigenic pattern is identified by an antibody, a subpart of the B cell object. When a B cell receptor recognizes an antigen, with an affinity greater than the threshold, then it is used to proliferate **[LJ02]**. Antibodies are soluble form of B cell objects, which are released from B cell surfaces to encounter against the foreign invaders. Antibodies are binded with antigens followed by elimination of antigens by immune cells. Proliferation in clonal selection is asexual. It is a mitotic process in which cells divide themselves without any crossover operation. In reproduction phase, B cells clones undergo a process of hypermutation **[LJ02]**. In this process, B cells with selective pressure produce new B cells having great affinity with the antigens. The B cells with high affinities are kept as memory cells with long life spans **[LJ02][FPP86][GR94]**. Clonal Selection process, which is shown in Figure 2.4. There are some features of clonal selection; those are relevant to computational point of view.

- An antigen selects many B cells for proliferation. Proliferation rate of each B cell is directly proportional to its affinity with the selected antigen. The greater the affinity of antibody with the antigen, the greater the proliferation rate and vice-versa **[LJ02]**.

- Whereas, mutation rate is inversely proportional to the affinity of antibody and antigen such that greater the affinity lesser the mutation and vice-versa **[LJ02].**

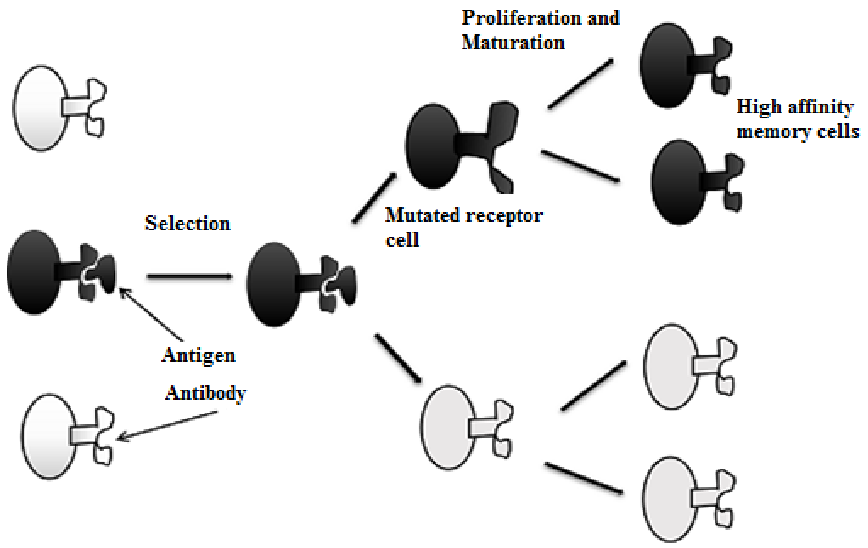


Figure 2.4 Clonal Selection, Proliferation, Affinity Maturation and memory cells maintenance [LJ02].

2.4.1. CLONALG

Clonal selection process is described by an algorithm named as CLONALG. Initially, CLONALG was proposed for pattern recognition. Later, it was used in multi-modal optimization tasks **[BV90]**. It is given below in the Figure 2.5 **[LJ02].**

CLONALG has been explained below with the following considerations.

Population (P): $(Ab_1, Ab_2 \ldots .Ab_i)$ is the population of antibodies.

Individuals: Each antibody Ab_i represents individuals in the population.

Affinity/Fitness: Binding strength between antibody and antigen is known as affinity/fitness of each antibody.

```
Given a set of patterns (Antigens) $A_n$ = ($A_n$1, $A_n$2,… $A_n$i) to be recognized.
Begin
    1.  Randomly initialize the population of individuals P = ($Ab_1$, $Ab_2$ ….$Ab_i$);
    2.  While (stopping criteria)
    3.  For each pattern of $A_n$
    4.  Present it to the population P and determine its affinity (match) with each
        individual of the population P
    5.  Select n1 best individuals of the highest affinity from population P.
    6.  Generate copies of these individuals proportionally to affinity of these individuals to the
        antigen.
    7.  Mutate all these copies with a rate proportional (inversely proportional) to their affinity
        with the input pattern.
    8.  Add these mutated individuals to the population P.
    9.  Re-select n2 of these maturated (optimized) individuals to keep as memories of the system.
    10. End for
    11. End while
    12. Repeat steps 3 to 12 until some stopping criteria is met. For example: minimum pattern
        recognition or classification error or maximum number of generations.
End
```

Figure 2. 5 CLONALG Algorithm [LJ02].

No. of Generations (G_P): Number of times antibodies are calculated based on their fitness to find the optimum solution.

Objective Function*:* It is used to find the optimum solution to optimize individuals to keep as memories.

Given*:* Initially population of antigens is provided as input to be recognized.

Step1*:* Population of antibodies **P** is initialized randomly. In which, Ab_i represents an antibody.

Step2*:* An antigen is selected and encountered with each antibody in the population and affinity between antigen and antibody is computed.

Step3*:* Based on their affinity ***n1*** antibodies are selected for proliferation. The rate of proliferation of each B cell is directly proportional to its affinity with the antigen.

Step4*:* After generation of progenies of each selected antibody, these progenies (clones) are used for mutation, and mutation is inversely proportional to the affinity of the antigen and antibody of B cell Objects.

Step5*:* All these mutated antibodies are added to the immune network i.e. to initial population of B cell objects.

Step6*:* One generation ends here.

Step7: Choose top antibodies equals to the population ***P*** to make memory cells for future.

Step8*:* However, this whole algorithm runs until an optimized solution is achieved, or certain criterion is met. For Example: minimum pattern recognition or classification error or maximum number of generations.

2.5. RELATED WORK

This section describes the work carried out related to the immune system in computer science field. In discussion, AIS is being differentiated from other machine learning techniques. In **[GR94],** Gilbert and Routen created a content addressable auto-associative memory system, on the basis of immune network theory. The purpose for this system was image recognition. This was not a stable model, because it was not able to remember patterns. The system created by Gilbert and Route considers the immune system as a connectionist model in which local nodes (B cells) interact to get new experiences or to identify past situations. In their approach, they did not focus on B cells and antibodies but, focused only on the parts those were important to present their interaction. However, in **[HC96]** Hunt and Cooke focused on not only B cells and antibodies but, also on the genetic algorithm mechanism by which these antibodies are produced.

[FJSP93], showed immune system evolution and operation using Genetic Algorithm. **[HC96]** showed the same operation using computer program. It considered gene selection, proliferation and mutation in production of antibody. Bersini and Varela in **[BV90]** applied this approach in many engineering problems such as Travelling Salesman Problem, Optimization of a control function for the cart-pole balancing problem, etc. In **[EE95],** it was used the same approach to solve machine learning problems such as information extraction and classification of data etc.

AIS is a system that provides unsupervised learning, which is noise tolerant and is self-organized and that does not require any negative example. These kinds of systems are combination of learning classifier systems and neural networks, based on information retrieval and machine induction. So, these systems prove useful where neural networks and learning classifier systems do not work separately **[FPP86][H86][HC96]**. For example, neural networks are trained for specific examples, but AIS are inherently generalized systems. Learning classifier systems find difficulty to separate global and local optimum solutions, here AIS proves very helpful. There is a large number of applications of AIS such as in machine learning, autonomous robotics navigation **[LJ02][HC96]**. AIS is used in character recognition and data analysis. AIS is distributed, dynamic, robust, adaptive and diverse in nature having various applications in computer network security. Clonal Selection Algorithm is used as a conjunction with negative selection algorithm because of its learning capabilities. Another application of clonal selection is to solve multi-modal optimization tasks because it has the capacity to recognize a set of binary characters presented in a Hamming shape-space. **[LJ02][FPP86].**

In this dissertation, AIS has been adapted to solve the DQPG problem discussed in chapter 1. DQPG using AIS is discussed in the next chapter.

CHAPTER 3

DQPG USING AIS

Distributed query plan generation is an intricate problem as discussed in Chapter 1. DQPG was addressed using genetic algorithm in **[VSV10][VSV10]**. In this chapter, an attempt has been made to solve DQPG problem using Artificial Immune System (AIS) described in Chapter 2. AIS based DQPG algorithms are presented here to compute near optimal distributed query plans for a given distributed query. There should be a mapping between original algorithm and AIS. The population of antibodies are the query plans in DQPG. Both antibodies and query plans terms have been used interchangeably. The fitness function used in DQPG is treated as antigens. Using this fitness function, fitness of antibody query plans are computed and query plans with good fitness are used to make clones in DQPG. AIS is similar to the GA discussed in evolutionary techniques with one difference of no crossover exists in AIS. AIS based DQPG algorithm is used to generate distributed query plans based on the property of closeness [VSV10][VSV11]. In this chapter, we propose two AIS based DQPG models DQPG$_{AIS}$-I and DQPG$_{AIS}$-II used to generate distributed query plans incorporating the property of original algorithm (CLONALG).

3.1. DQPG$_{AIS}$-I

AIS given by [LN02], as discussed in Chapter 2, is being adapted corresponding to the DQPG problem. DQPG algorithm based on AIS is shown in Figure 3.1. This model is different from the original model in terms of two parameters have been explained in section 3.3 and section 3.4.

3.1.1 The DQPG$_{AIS}$ -1 Algorithm

```
Input: RS:  Relation-Site matrix,
P:  Size of antibodies population,
$G_P$:  Pre-specified number of generations,
R:  Number of relations to be used in query plan,
S:  Number of sites containing these relations in a query plan,
β:  Clone rate,
n:   Top query plans with least QPC value to be selected to compute clones,
Output: Top-k Query Plans
Method:
Step 1: QP = QueryPlan (RS, P)
Step 2: Compute fitness of each antibody query plan in QP
        $QPC = \sum_{i=1}^{M} \frac{Si}{N}\left(1 - \frac{Si}{N}\right)$
Step 3:Repeat
    Step 3.1:   Choose top 'n' antibody query plans with least QPC value.
    Step 3.2: Calculate total number of clones of selected top n query plans using
        $C_n = \sum_{i=1}^{n} \beta * P / i$
    Step 3.3: Compute clones for each selected query plan.
        $Clone_i = C_i * C_n$
        Where, $C_i = \frac{(Z - QPC_i)}{Z1}$;  $Z_1 = Z_1 + (Z - QPC_i)$ and  $Z = \frac{(R-1)}{R}$
    Step 3.4: Mutated Clones = Mutation (P, n, Cn, QPC, Cloned Query Plans);
    Step 3.5: Compute fitness of Mutated Clones using step2.
    Step 3.6: Population = P+ Mutated Clones;
    Step 3.7: P=Top P Query plans of Population
Until Generation=Gp;
$C_n$: Total number of clones;   $Clone_i$: Clones for ith selected query plan.
```

Figure 3. 1 DQPG$_{AIS}$-I algorithm.

Roulette Wheel Selection Operator

In Roulette Wheel selection, individuals (Query Plans) are given a selection probability P_{QP} proportional to individual fitness [ATA12]. An algorithm which illustrate the Roulette Wheel Selection is given in Figure 3.2.

$$\boldsymbol{P_{QP}} = \frac{\mathbf{1}}{\boldsymbol{P}-\mathbf{1}}\left(\mathbf{1} - \frac{\boldsymbol{Fitness_i}}{\sum_{j\in Population} \boldsymbol{Fitness_j}}\right)$$

```
For all individuals of population
Sum= Sum + fitness of current individual
End For
For all population individuals
Probability = Sum of probabilities + (fitness / Sum)
Sum of Probabilities=+ Probability
End For
Number = Random between 0 and 1
For all population individuals
If Number > Probability but less than next Probability
Then you have been selected
End For
```

Figure 3.2 Roulette Wheel Selection algorithm [ATA12].

Step 1: First we initialize the population of antibodies query plan. Here we define the method taking Relation-Site matrix and yields population of size P. This population improves over the generations. Each query from population can be represented as $QP = \{\boldsymbol{X_1}, \boldsymbol{X_2}, \boldsymbol{X_3}, \boldsymbol{X_4}, \boldsymbol{X_5}, \boldsymbol{X_6}\}$**.**

Step 2: QPC of each antibody query plan is calculated using the fitness function**.** The Query Plan with least QPC is the fittest Query Plan. This step correlates to the affinity between antigen and antibodies.

***Step 3*:** For each Generation, repeat the following steps.

Step 3.1: *S*elect Top-k query plans based on their calculated fitness value.

Step 3.2: *F*or given top-k query plans, calculate total clones based on their fitness using $C_n = \sum_{i=1}^{n} \beta * P / i$ [BS12].

Step 3.3: Using Roulette Wheel Fitness proportionate function, we wish to calculate clones of each selected antibody query plan of total clones value. Where, $Clone_i \alpha C_i$ implies that larger is the value of C_i, more number of clones will be generated for i^{th} query plan. Again use roulette wheel selection function to compute clones of these selected 'n' query plans.

Step 3.4: Mutation is now applied on the generated clones based on their mutation rate. Mutation rate is computed using QPC * Number of sites used in query plan and which is inversely proportional to the fitness of the antibody query plan.

Step 3.5: Compute fitness of mutated query plans using step 2.

Step 3.6: Add mutated query plans in present population 'P' and sort them in ascending order on the basis of QPC value.

Step 3.7: Choose best 'P' query plans from the combine population based on fitness value for next generation.

Step 3 will be repeated until some stopping criterion is met. Some stopping criteria may include maximum number of generation reached, optimum solution of the problem etc. and returns Top-k Query Plans of high fitness. In $DQPG_{AIS}$-1 algorithm, the stopping criterion used is the maximum number of generations and the aim of this algorithm is to get Top-k query plans from population of antibody query plans.

3.1.2. DQPG$_{AIS}$-1 Example

The application of AIS based DQPG algorithm to solve distributed query plan generation problem is exemplified. For this purpose, consider a distributed database system with six sites namely, S1, S2, S3, S4, S5 and S6. Consider a distributed query accessing six relations namely, R1, R2, R3, R4, R5 and R6.

```
Select A1, A2, A3
From R1, R2, R3, R4, R5, R6
Where R1.A1=R2.A1 and R3.A2=R4.A2 and R5.A3=R6.A3
```

Site	*S1*	*S2*	*S3*	*S4*	*S5*	*S6*
Relations						
R1	1	0	1	0	1	1
R2	0	0	1	0	1	0
R3	0	0	1	1	0	0
R4	1	0	1	0	1	1
R5	0	1	0	1	0	0
R6	1	1	0	0	0	1

Figure 3.3 Relation-site Matrix.

The relations used in the distributed query are accessed from the relation-site matrix, which is shown in Figure 3.3. Relation-site matrix is the combination of the relations and sites, where relations in matrix are the relations used in the distributed query and sites represents where these relations are present. In the relation-site matrix, the value 1 and 0 in the cells indicate the presence and absence of a relation respectively on the corresponding host sites.

Input: *RS = 6 X 6;* G_P *=2; R = 6;* P *=10; β = 0.99; S=6; n=0.4*P*

Output: *Top 4 query plans.*

Step1: Generate Initial Population P =10 of antibodies using Relation-Site matrix as given in Table 3.1.

QPN	***Query Plan***	***QPC***
X1	[5,6,4,4,1,2]	0.7778
X2	[5,1,2,6,4,4]	0.7778
X3	[3,1,1,6,3,2]	0.7222
X4	[5,4,4,6,4,4]	0.5000
X5	[3,2,2,2,3,1]	0.6111
X6	[1,2,4,5,2,4]	0.7222
X7	[6,6,2,2,3,4]	0.7222
X8	[3,3,1,4,3,4]	0.6111
X9	[1,3,6,5,3,1]	0.7222
X10	[3,4,4,2,2,4]	0.6111

Table 3.1 Query Plans population with QPC.

Step 2: Compute fitness of each query plan using QPC function as given in Table 3.1.

Step 3: For Generation =1, repeat the following steps:

Step 3.1: Top-4 query plans from Table 3.1 are given in Table 3.2

QPN	***Query Plans***	***QPC***
X4	[5,4,4,6,4,4]	0.5000
X5	[3,2,2,2,3,1]	0.6111
X8	[3,3,1,4,3,4]	0.6111
X10	[3,4,4,2,2,4]	0.6111

Table 3.2 Top-4 query plans.

Step 3.2: Use Table 3.2 to compute Cn. For this purpose, Let us consider α=0.9;   β=0.99; Such that β=α*β. Thus, $Cn = 19$.

Step 3.3: For Top-4 query plans from Table 3.2. Compute Z i.e. 0.83 for R=6. Now compute $Z1$ using Z, as given in Table 3.3.

QPN	***QPC***	***Z -QPC_i***
1	0.5000	0.3300
2	0.6111	0.2189
3	0.6111	0.2189
4	0.6111	0.2189
Total ($Z1$) = 0.9867		

Table 3.3 Sum of Fitness.

Using Z1, compute C_i and $Clone_i$

C_i	$Clone_i$
0.3345	7

0.2219	4
0.2219	4
0.2219	4

Table 3.4 Clone computation using probability distribution.

Clones of Top-4 query plans are generated based on $Clone_i$ values, as given in Table 3.5.

QPN	***Clones of antibody Query Plans***
1	[5,4,4,6,4,4]
2	[5,4,4,6,4,4]
3	[5,4,4,6,4,4]
4	[5,4,4,6,4,4]
5	[5,4,4,6,4,4]
6	[5,4,4,6,4,4]
7	[5,4,4,6,4,4]
8	[3,2,2,2,3,1]
9	[3,2,2,2,3,1]
10	[3,2,2,2,3,1]
11	[3,2,2,2,3,1]
12	[3,3,1,4,3,4]
13	[3,3,1,4,3,4]
14	[3,3,1,4,3,4]
15	[3,3,1,4,3,4]
16	[3,4,4,2,2,4]
17	[3,4,4,2,2,4]
18	[3,4,4,2,2,4]
19	[3,4,4,2,2,4]

Table 3. 5 Clones of Top-4 selected query plans.

Step 3.4: Mutated query plans are given in Table 3.6.

QPN	**Query Plan**	**Mutated Query Plan**	**QPC**
1	[5,4,4,6,4,4]	[5,4,3,6,4,2]	0.778
2	[5,4,4,6,4,4]	[5,1,2,6,4,4]	0.778
3	[5,4,4,6,4,4]	[5,4,4,6,3,1]	0.778
4	[5,4,4,6,4,4]	[1,4,4,2,4,4]	0.500
5	[5,4,4,6,4,4]	[5,1,4,6,4,3]	0.778
6	[5,4,4,6,4,4]	[5,4,5,6,2,4]	0.722
7	[5,4,4,6,4,4]	[5,2,4,6,5,4]	0.722
8	[3,2,2,2,3,1]	[3,2,1,4,3,1]	0.722

9	[3,2,2,2,3,1]	[3,2,2,2,1,4]	0.667
10	[3,2,2,2,3,1]	[3,1,2,2,4,1]	0.722
11	[3,2,2,2,3,1]	[3,2,3,2,3,2]	0.500
12	[3,3,1,4,3,4]	[1,2,1,4,3,4]	0.722
13	[3,3,1,4,3,4]	[3,3,2,1,3,4]	0.667
14	[3,3,1,4,3,4]	[3,3,1,4,5,6]	0.778
15	[3,3,1,4,3,4]	[3,5,1,4,6,4]	0.778
16	[3,4,4,2,2,4]	[3,3,4,2,1,4]	0.722
17	[3,4,4,2,2,4]	[1,4,4,5,2,4]	0.667
18	[3,4,4,2,2,4]	[3,4,6,2,1,4]	0.778
19	[3,4,4,2,2,4]	[3,4,4,1,5,4]	0.667

Table 3. 6 Mutated Query Plans with their QPC.

Step 3.5: Compute fitness of mutated query plans as shown in Table 3.6.

Step 3.6: Add Mutated Clones to the existing population P, as shown below in Table 3.7.

QPN	Query Plan	QPC
1	[5,4,3,6,4,2]	0.7778
2	[5,1,2,6,4,4]	0.7778
3	[5,4,4,6,3,1]	0.7778
4	[1,4,4,2,4,4]	0.5000
5	[5,1,4,6,4,3]	0.7778
6	[5,4,5,6,2,4]	0.7222
7	[5,2,4,6,5,4]	0.7222
8	[3,2,1,4,3,1]	0.7222
9	[3,2,2,2,1,4]	0.6667
10	[3,1,2,2,4,1]	0.7222
11	[3,2,3,2,3,2]	0.5000
12	[1,2,1,4,3,4]	0.7222
13	[3,3,2,1,3,4]	0.6667
14	[3,3,1,4,5,6]	0.7778
15	[3,5,1,4,6,4]	0.7778
16	[3,3,4,2,1,4]	0.7222
17	[1,4,4,5,2,4]	0.6667

18	[3,4,6,2,1,4]	0.7778
19	[3,4,4,1,5,4]	0.6667
20	[5,6,4,4,1,2]	0.7778
21	[5,1,2,6,4,4]	0.7778
22	[3,1,1,6,3,2]	0.7222
23	[5,4,4,6,4,4]	0.5000
24	[3,2,2,2,3,1]	0.6111
25	[1,2,4,5,2,4]	0.7222
26	[6,6,2,2,3,4]	0.7222
27	[3,3,1,4,3,4]	0.6111
28	[1,3,6,5,3,1]	0.7222
29	[3,4,4,2,2,4]	0.6111

Table 3.7 Combination of Mutated clones and existing population.

Step 3.7: From Table 3.7 choose best P=10 query plans, as shown in Table 3.8.

QPN	Query Plan	QPC
X1	[1,4,4,2,4,4]	0.5000
X2	[3,2,3,2,3,2]	0.5000
X3	[5,4,4,6,4,4]	0.5000
X4	[3,2,2,2,3,1]	0.6111
X5	[3,3,1,4,3,4]	0.6111
X6	[3,4,4,2,2,4]	0.6111
X7	[3,2,2,2,1,4]	0.6667
X8	[3,3,2,1,3,4]	0.6667
X9	[1,4,4,5,2,4]	0.6667
X10	[3,4,4,1,5,4]	0.6667

Table 3.8 Population for next generation.

For Generation=2

Step 3.1: Top-4 plans from Table 3.8 are shown in Table 3.9.

QPN	Query Plan	QPC
X1	[1,4,4,2,4,4]	0.5000
X2	[3,2,3,2,3,2]	0.5000
X3	[5,4,4,6,4,4]	0.5000
X4	[3,2,2,2,3,1]	0.6111

Table 3.9 Selected Top-4 query plan for 2nd Generation.

Step 3.2: Use Table 3.9 to compute Cn. For this purpose, Suppose α=0.9 and β=0.99. Where β can be given byβ=α*β.Thus, $Cn = 19$.

Step 3.3: For Top-4 query plans from Table 3.9. Compute Z i.e. 0.83 for R=6. Now, use Z to compute $Z1$ as shown in Table 3.10.

QPN	***QPC***	***Z -QPC_i***
1	0.5000	0.3300
2	0.5000	0.3300
3	0.5000	0.3300
4	0.6111	0.2189
Total (Z1) = 1.2189		

Table 3.10 Z1 value computation using Roulette Wheel for 2nd Generation.

Use *Z1*, to compute C_i and $Clone_i$

C_i	$Clone_i$
0.2707	5
0.2707	5
0.2707	5
0.1800	4

Table 3.11 Clone proportion computation of Top-4 in 2nd Generation.

Clones of Top-4 query plans are generated based on $Clone_i$ values as given in Table 3.12.

QPN	***Selected Query Plans Clones***
1	[1,4,4,2,4,4]
2	[1,4,4,2,4,4]
3	[1,4,4,2,4,4]
4	[1,4,4,2,4,4]
5	[1,4,4,2,4,4]
6	[3,2,3,2,3,2]
7	[3,2,3,2,3,2]
8	[3,2,3,2,3,2]
9	[3,2,3,2,3,2]

10	[3,2,3,2,3,2]
11	[5,4,4,6,4,4]
12	[5,4,4,6,4,4]
13	[5,4,4,6,4,4]
14	[5,4,4,6,4,4]
15	[5,4,4,6,4,4]
16	[3,2,2,2,3,1]
17	[3,2,2,2,3,1]
18	[3,2,2,2,3,1]
19	[3,2,2,2,3,1]

Table 3.12 Clones of Top-4 query plans in 2nd Generation.

Step 3.4: Mutated query plans of Table 3.12 are shown in Table 3.13.

QPN	***Query Plan***	***Mutated Query Plan***	***QPC***
1	[1,4,4,2,4,4]	[3,4,4,2,4,5]	0.6667
2	[1,4,4,2,4,4]	[1,4,6,2,4,5]	0.7778
3	[1,4,4,2,4,4]	[5,4,4,3,4,4]	0.5000
4	[1,4,4,2,4,4]	[1,3,5,2,4,4]	0.7778
5	[1,4,4,2,4,4]	[1,4,4,2,6,5]	0.7778
6	[3,2,3,2,3,2]	[1,2,3,1,3,2]	0.6667
7	[3,2,3,2,3,2]	[3,2,4,5,3,2]	0.7222
8	[3,2,3,2,3,2]	[1,2,4,2,3,2]	0.6667
9	[3,2,3,2,3,2]	[5,2,3,2,6,2]	0.6667
10	[3,2,3,2,3,2]	[5,2,3,2,1,2]	0.6667
11	[5,4,4,6,4,4]	[5,3,4,6,5,4]	0.7222
12	[5,4,4,6,4,4]	[5,5,6,6,4,4]	0.6667
13	[5,4,4,6,4,4]	[5,4,4,1,2,4]	0.6667
14	[5,4,4,6,4,4]	[6,4,5,6,4,4]	0.6111
15	[5,4,4,6,4,4]	[5,4,4,1,5,4]	0.6111
16	[3,2,2,2,3,1]	[1,2,3,2,3,1]	0.6667
17	[3,2,2,2,3,1]	[3,2,3,2,3,5]	0.6111
18	[3,2,2,2,3,1]	[4,2,2,6,3,1]	0.7778
19	[3,2,2,2,3,1]	[3,2,2,1,3,6]	0.7222

Table 3.13 Mutated Query Plans with respective computed QPC in 2nd Generation.

Step 3.5: Compute fitness of mutated query plans are shown in Table 3.13.

Step 3.6: Add Mutated Clones to the existing population P as shown in Table 3.14.

QPN	***Query Plan***	***QPC***
1	[3,4,4,2,4,5]	0.6667
2	[1,4,6,2,4,5]	0.7778
3	[5,4,4,3,4,4]	0.5000
4	[1,3,5,2,4,4]	0.7778
5	[1,4,4,2,6,5]	0.7778
6	[1,2,3,1,3,2]	0.6667
7	[3,2,4,5,3,2]	0.7222
8	[1,2,4,2,3,2]	0.6667
9	[5,2,3,2,6,2]	0.6667
10	[5,2,3,2,1,2]	0.6667
11	[5,3,4,6,5,4]	0.7222
12	[5,5,6,6,4,4]	0.6667
13	[5,4,4,1,2,4]	0.6667
14	[6,4,5,6,4,4]	0.6111
15	[5,4,4,1,5,4]	0.6111
16	[1,2,3,2,3,1]	0.6667
17	[3,2,3,2,3,5]	0.6111
18	[4,2,2,6,3,1]	0.7778
19	[3,2,2,1,3,6]	0.7222
20	[5,4,4,5,4,4]	0.444
21	[5,4,1,6,4,4]	0.611
22	[5,4,4,3,4,2]	0.611
23	[5,4,4,1,4,4]	0.500
24	[5,4,4,5,4,4]	0.444
25	[5,1,4,6,4,4]	0.611
26	[3,1,2,2,3,2]	0.611
27	[3,2,1,2,3,2]	0.667
28	[5,2,4,6,4,4]	0.667
29	[5,4,1,3,4,4]	0.667

Table 3.14 Combination of Mutated clones and existing population.

Step 3.7: From Table 3.14 choose best P=10 query plans as shown in Table 3.15.

QPN	***Query Plan***	***QPC***
X1	[5,4,4,5,4,4]	0.4444
X2	[5,4,4,5,4,4]	0.4444
X3	[5,4,4,3,4,4]	0.5000
X4	[5,4,4,1,4,4]	0.5000
X5	[6,4,5,6,4,4]	0.6111
X6	[5,4,4,1,5,4]	0.6111
X7	[3,2,3,2,3,5]	0.6111
X8	[5,4,1,6,4,4]	0.6111
X9	[5,4,4,3,4,2]	0.6111
X10	[5,4,1,6,4,4]	0.6111

Table 3.15 Population for next generation.

The above steps are repeated for a pre-specified number of generations. Thereafter, the Top-4 query plans are produced as output.

3.2. DQPG$_{AIS}$-II

Another adaptation of AIS has been used to address the DQPG problem. This adaptation, referred to as DQPG$_{AIS}$-II is given in Figure 3.4.

3.2.1. DQPG$_{AIS}$-II Algorithm

```
Input: RS: Relation-Site matrix,
P: Size of antibodies population,
$P_m$: Probability of mutation,
$G_P$: Pre-specified number of generations,
R: Number of relations to be used in query plan,
S: Number of sites is containing these relations,
β: Clone rate,
n: Top query plans with least QPC value to be selected for clone computation
Output: Top-k Query Plans
Method:
Step 1: QP = QueryPlan (RS, P)
Step 2: Compute fitness of each antibody query plan in QP

        $QPC = \sum_{i=1}^{M} \frac{Si}{N}\left(1 - \frac{Si}{N}\right)$

Step 3: Repeat
        Step 3.1: Choose top 'n' antibody query plans with least QPC value.
        Step 3.2: Calculate total number of clones of selected top n query plans
        using
            $C_n = C_n + \left(\frac{(\beta * P)}{i}\right)$
        Step 3.3: Compute number of clones for each selected query plan.
            $C_i = \frac{(Z - QPC_i)}{Z1}$
            Where, $Z_1 = Z_1 + (Z - QPC_i)$; $Z = \frac{(R-1)}{R}$;
        Step 3.4: Mutated Clones = Mutation (P, Pm, n, Cn, QPC, Cloned Query Plans);
        Step 3.5: Compute fitness of mutated clones using step 2.
        Step 3.6: Population = P+ Mutated Clones;
        Step 3.7: P=Top 'P' Query plans of Population
Until Generation=Gp;
```

Figure 3.4 DQPG$_{AIS}$-II algorithm.

Steps *1, 2, 3, 3.1 and 3.2* are same as in DQPG$_{AIS}$-I discussed in section 3.1.1.

Step 3.3: Based on $C_i = \frac{(Z-QPC_i)}{Z1}$, a roulette wheel is constructed using it to compute clones of each selected antibody query plan. Randomly generate number of clones of each selected i^{th} query plan equals to the number of calculated clones is explained with example in section 3.2.2.

Step 3.4: In DQPG$_{AIS}$-II, the mutation rate is taken as constant and query plans are mutated using Roulette Wheel. It is based on the concept of algorithm proposed in [LJ02], of inverse relation between fitness and mutation of antibodies.

Step *3.5, 3.6 and 3.7 are the same as discussed in section 3.3.1.*

The step 3 given in Figure 3.3 will be repeated for a pre-specified number of generations. Thereafter, the Top-k query plans are produced as output.

3.2.2. DQPG$_{AIS}$-II Example

Input: *RS = 6 X 6; G_P =2; R = 6;* P *=10; β = 0.99; S=6; n=0.4*P*

Output: *Top-4 query plans.*

Step1: Generate 10 antibody query plans from RS matrix as shown in Table 3.16.

Step 2: Compute fitness of each antibody query plans as given in Table 3.16.

QPN	***Query Plan***	***QPC***
X1	[5,6,4,4,1,2]	0.7778
X2	[5,1,2,6,4,4]	0.7778
X3	[3,1,1,6,3,2]	0.7222
X4	[5,4,4,6,4,4]	0.5000
X5	[3,2,2,2,3,1]	0.6111
X6	[1,2,4,5,2,4]	0.7222
X7	[6,6,2,2,3,4]	0.7222

X8	[3,3,1,4,3,4]	0.6111
X9	[1,3,6,5,3,1]	0.7222
X10	[3,4,4,2,2,4]	0.6111

Table 3.16 Antibody Query Plans population with respective QPC.

Step 3: For Generation=1, repeat the following steps:

Step 3.1: Choose Top-4 query plans from population P as shown in Table 3.17.

QPN	***Query Plans***	***QPC***
X4	[5,4,4,6,4,4]	0.5000
X5	[3,2,2,2,3,1]	0.6111
X8	[3,3,1,4,3,4]	0.6111
X10	[3,4,4,2,2,4]	0.6111

Table 3.17 Top-4 query plans.

Step3.2: Compute total number of clones. For α=0.9, β=0.99, where β=α*β, Cn =19.

Step3.3: For Top-4 query plans from Table 3.17. Compute Z i.e. 0.83 for R=6. Compute $Z1$ using Z as given in Table 3.18.

QPN	***QPC***	***Z -QPC_i***
1	0.5000	0.3300
2	0.6111	0.2189
3	0.6111	0.2189
4	0.6111	0.2189
Total ($Z1$) = 0.9867		

Table 3.18 Sum of fitness using Roulette Wheel.

Compute $C_i$using Z1 as shown in Table 3.19.

C_i	0.3345	0.2219	0.2219	0.2219

Table 3.19 Probability of each query plans.

Using Roulette Wheel, as shown in Figure 3.5 and Table 3.20, clones of antibody query plans are generated, as given in Table 3.21.

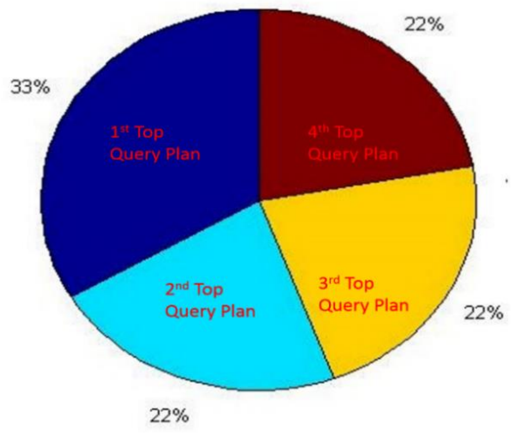


Figure 3. 5 Top-4 query plans for clone generation with their probability.

0.2219	0.4438	0.6657	1.0000

Table 3.20 Cumulative Probability.

Random number	Clones of selected query plans
0.8147	[5,4,4,6,4,4]
0.9058	[5,4,4,6,4,4]
0.1270	[3,4,4,2,2,4]
0.9134	[5,4,4,6,4,4]
0.6324	[3,2,2,2,3,1]
0.0975	[3,4,4,2,2,4]
0.2785	[3,4,4,2,2,4]
0.5469	[3,3,1,4,3,4]
0.9575	[5,4,4,6,4,4]
0.9649	[5,4,4,6,4,4]
0.1576	[3,4,4,2,2,4]
0.9706	[5,4,4,6,4,4]
0.6787	[3,2,2,2,3,1]
0.7577	[3,2,2,2,3,1]
0.7431	[3,2,2,2,3,1]
0.3922	[3,3,1,4,3,4]
0.6555	[3,2,2,2,3,1]
0.1712	[3,4,4,2,2,4]
0.7060	[3,2,2,2,3,1]

Table 3.20 Clones of top-4 selected query plans based on its probability.

Step3.4: For mutation, Roulette Wheel as given in Figure 3.22, is used to determine the amount of mutation to be carried out in an antibody query plan as shown in Figure 3.24.

QPC	***Probability***
0.5000/2.3333	0.2143
0.6111/2.3333	0.2619
0.6111/2.3333	0.2619
0.6111/2.3333	0.2619
Sum	1.0000

Table 3.21 Probability of selected query plan.

0.2143	0.4762	0.7381	1.000

Table 3.22 Cumulative probability of best 4 query plans.

QPN	***Random Number***	***Query Plan***	***Mutated Query Plan***	***QPC***
1	0.0318	[5,4,4,6,4,4]	[5,4,4,6,4,4]	0.500
2	0.2769	[3,2,2,2,3,1]	[3,2,2,2,3,2]	0.444
3	0.0462	[5,4,4,6,4,4]	[5,4,4,3,4,4]	0.500
4	0.0971	[5,4,4,6,4,4]	[5,4,2,6,4,4]	0.667
5	0.8235	[3,4,4,2,2,4]	[3,4,4,2,2,4]	0.611
6	0.6948	[3,3,1,4,3,4]	[3,5,1,4,3,4]	0.778
7	0.3171	[3,2,2,2,3,1]	[3,4,2,2,3,1]	0.778
8	0.9502	[3,4,4,2,2,4]	[3,4,4,2,6,4]	0.667
9	0.0344	[5,4,4,6,4,4]	[5,2,4,6,4,4]	0.667
10	0.4387	[3,2,2,2,3,1]	[3,1,2,2,3,1]	0.667
11	0.3816	[3,2,2,2,3,1]	[3,2,2,4,3,1]	0.778
12	0.7655	[3,4,4,2,2,4]	[3,3,4,2,2,4]	0.667
13	0.7952	[3,4,4,2,2,4]	[3,4,6,2,2,4]	0.778
14	0.1869	[5,4,4,6,4,4]	[5,4,4,5,4,4]	0.444
15	0.4898	[3,3,1,4,3,4]	[3,2,1,4,3,4]	0.778
16	0.4456	[3,2,2,2,3,1]	[3,1,2,2,3,1]	0.667
17	0.6463	[3,3,1,4,3,4]	[3,1,1,4,3,4]	0.667
18	0.7094	[3,3,1,4,3,4]	[3,2,1,4,3,4]	0.778
19	0.7547	[3,4,4,2,2,4]	[3,1,4,2,2,4]	0.778

Table 3.23 Mutated clones with their fitness.

Step3.5: Calculate fitness of mutated query plans as shown in Table 3.24.

Step3.6: Add mutated clones to the existing population P as shown in Table 3.25.

QPN	*Query Plan*	*QPC*
1	[5,4,4,6,4,4]	0.500
2	[3,2,2,2,3,2]	0.444
3	[5,4,4,3,4,4]	0.500
4	[5,4,2,6,4,4]	0.667
5	[3,4,4,2,2,4]	0.611
6	[3,5,1,4,3,4]	0.778
7	[3,4,2,2,3,1]	0.778
8	[3,4,4,2,6,4]	0.667
9	[5,2,4,6,4,4]	0.667
10	[3,1,2,2,3,1]	0.667
11	[3,2,2,4,3,1]	0.778
12	[3,3,4,2,2,4]	0.667
13	[3,4,6,2,2,4]	0.778
14	[5,4,4,5,4,4]	0.444
15	[3,2,1,4,3,4]	0.778
16	[3,1,2,2,3,1]	0.667
17	[3,1,1,4,3,4]	0.667
18	[3,2,1,4,3,4]	0.778
19	[3,1,4,2,2,4]	0.778
20	[5,6,4,4,1,2]	0.778
21	[5,1,2,6,4,4]	0.778
22	[3,1,1,6,3,2]	0.722
23	[5,4,4,6,4,4]	0.500
24	[3,2,2,2,3,1]	0.611
25	[1,2,4,5,2,4]	0.722
26	[6,6,2,2,3,4]	0.722
27	[3,3,1,4,3,4]	0.611
28	[1,3,6,5,3,1]	0.722
29	[3,4,4,2,2,4]	0.611

Table 3.24 Combination of Mutated clones and existing population.

Step 3.7: Choose top 'P' antibody query plans for new population from Table 3.25. The new population for next generation is shown below in Table 3.26.

QPN	***Query Plan***	***QPC***
X1	[5,4,4,6,4,4]	0.500
X2	[3,2,2,2,3,2]	0.444
X3	[5,4,4,3,4,4]	0.500
X4	[5,4,4,5,4,4]	0.444
X5	[5,4,4,6,4,4]	0.500
X6	[3,4,4,2,2,4]	0.611
X7	[3,2,2,2,3,1]	0.611
X8	[3,3,1,4,3,4]	0.611
X9	[3,4,4,2,2,4]	0.611
X10	[5,4,2,6,4,4]	0.667

Table 3.25 Population for next generation.

For Generation=2, repeat the following steps:

Step3.1: Select top-4 query plans are shown below in Table 3.27.

QPN	**Query Plan**	**QPC**
X2	[3,2,2,2,3,2]	0.444
X4	[5,4,4,5,4,4]	0.444
X1	[5,4,4,6,4,4]	0.500
X3	[5,4,4,3,4,4]	0.500

Table 3.26 Selected Top-4 query plan for 2nd Generation.

Step3.2: Compute total number of clones. For α=0.9, β=0.99 where β=α*β, Cn =19.

Step3.3: For top four query plans from Table 3.27. Compute Z i.e. 0.83 for R=6. Now compute $Z1$ using Z as shown in Table 3.28.

QPN	***QPC***	$(Z - QPC_i)$
1	0.444	0.386
2	0.444	0.386
3	0.500	0.330
4	0.500	0.330
Total (Z1) = 1.432		

Table 3.27 Sum of fitness using Roulette Wheel.

Compute $C_i$using Z1 as shown in Table 3.29.

C_i	0.27	0.27	0.23	0.23

Table 3.28 Clone computation Probability of top-4 query plan.

Roulette Wheel of Top-4 query plans for clones computation has been shown in Figure 3.5.

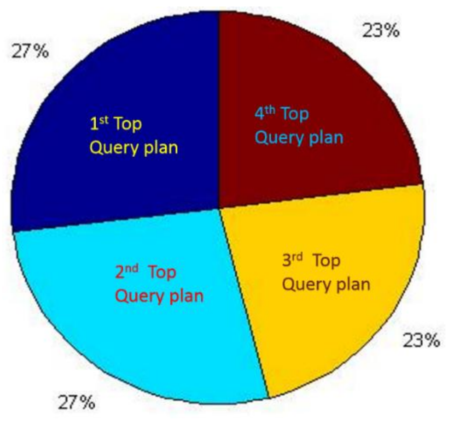


Figure 3.6 Top 4 query plans for clone generation with their probability.

Calculate probability of each selected antibody query plan to generate clones using Roulette Wheel is shown in Table 3.30.

0.27	0.54	0.77	1.00

Table 3.29 Each query plans proportion based on fitness.

Generate random numbers equal to the total number of clones and use roulette wheel to compute clones of chosen antibody query plans as shown in Table 3.31.

Random number	***Clones of selected antibodies***
0.9595	[5,4,4,3,4,4]
0.6557	[5,4,4,6,4,4]
0.0357	[3,2,2,2,3,2]
0.8491	[5,4,4,3,4,4]
0.9340	[5,4,4,3,4,4]
0.6787	[5,4,4,6,4,4]
0.7577	[5,4,4,6,4,4]
0.7431	[5,4,4,6,4,4]
0.3922	[5,4,4,5,4,4]
0.6555	[5,4,4,6,4,4]
0.1712	[3,2,2,2,3,2]
0.7060	[5,4,4,6,4,4]
0.0318	[3,2,2,2,3,2]
0.2769	[5,4,4,5,4,4]

0.0462	[3,2,2,2,3,2]
0.0971	[3,2,2,2,3,2]
0.8235	[5,4,4,3,4,4]
0.6948	[5,4,4,6,4,4]
0.3171	[5,4,4,5,4,4]

Table 3.30 Clones of top 'n' selected query plans based on its probability.

Step3.4: Table 3.32 and Table 3.33 show Mutated clones using roulette wheel algorithm.

QPC	***Probability***
0.444/1.888	0.235
0.444/1.888	0.235
0.500/1.888	0.265
0.500/1.888	0.265
Sum	1.0000

Table 3.31 Sum of Probability of selected query plans.

0.235	0.470	0.735	1.000

Table 3.32 Roulette Wheel for selected Query Plans for mutation.

Step 3.5: Computed fitness of mutated clones using step 2 are shown in Table 3.34.

QPN	***Random Number***	***Query Plan***	***Mutated Query Plan***	***QPC***
1	0.9502	[5,4,4,3,4,4]	[5,4,4,6,4,4]	0.500
2	0.0344	[3,2,2,2,3,2]	[3,2,2,2,3,2]	0.444
3	0.4387	[5,4,4,5,4,4]	[5,4,4,6,4,4]	0.500
4	0.3816	[5,4,4,5,4,4]	[5,4,2,5,4,4]	0.611
5	0.7655	[5,4,4,3,4,4]	[3,4,4,3,4,4]	0.444
6	0.7952	[5,4,4,3,4,4]	[5,4,1,3,4,4]	0.667
7	0.1869	[3,2,2,2,3,2]	[3,4,2,2,3,2]	0.611
8	0.4898	[5,4,4,6,4,4]	[3,4,4,6,4,4]	0.500
9	0.4456	[5,4,4,5,4,4]	[5,2,4,5,4,4]	0.611
10	0.6463	[5,4,4,6,4,4]	[5,4,4,5,4,4]	0.444
11	0.7094	[5,4,4,6,4,4]	[5,4,1,6,4,4]	0.611

12	0.7547	[5,4,4,3,4,4]	[5,4,4,3,4,2]	0.611
13	0.2760	[5,4,4,5,4,4]	[5,4,4,1,4,4]	0.500
14	0.6797	[5,4,4,6,4,4]	[5,4,4,5,4,4]	0.444
15	0.6551	[5,4,4,6,4,4]	[5,1,4,6,4,4]	0.611
16	0.1626	[3,2,2,2,3,2]	[3,1,2,2,3,2]	0.611
17	0.1190	[3,2,2,2,3,2]	[3,2,1,2,3,2]	0.667
18	0.4984	[5,4,4,6,4,4]	[5,2,4,6,4,4]	0.667
19	0.9597	[5,4,4,3,4,4]	[5,4,1,3,4,4]	0.667

Table 3.33 Mutated clones with their respective fitness.

Step 3.6: Add mutated clones Cn to the existing population P as given in Table 3.35.

QPN	***Query Plan***	***QPC***
1	[5,4,4,6,4,4]	0.500
2	[3,2,2,2,3,2]	0.444
3	[5,4,4,3,4,4]	0.500
4	[5,4,4,5,4,4]	0.444
5	[5,4,4,6,4,4]	0.500
6	[3,4,4,2,2,4]	0.611
7	[3,2,2,2,3,1]	0.611
8	[3,3,1,4,3,4]	0.611
9	[3,4,4,2,2,4]	0.611
10	[5,4,2,6,4,4]	0.667
11	[5,4,4,6,4,4]	0.500
12	[3,2,2,2,3,2]	0.444
13	[5,4,4,6,4,4]	0.500
14	[5,4,2,5,4,4]	0.611
15	[3,4,4,3,4,4]	0.444
16	[5,4,1,3,4,4]	0.667
17	[3,4,2,2,3,2]	0.611
18	[3,4,4,6,4,4]	0.500
19	[5,2,4,5,4,4]	0.611
20	[5,4,4,5,4,4]	0.444

21	[5,4,1,6,4,4]	0.611
22	[5,4,4,3,4,2]	0.611
23	[5,4,4,1,4,4]	0.500
24	[5,4,4,5,4,4]	0.444
25	[5,1,4,6,4,4]	0.611
26	[3,1,2,2,3,2]	0.611
27	[3,2,1,2,3,2]	0.667
28	[5,2,4,6,4,4]	0.667

Table 3.34 Combination of Mutated clones and existing population.

Step 3.7: Choose Top P=10 antibody query plans for new population from Table 3.35. The new population for next generation is shown below in Table 3.36.

QPN (New)	Query Plan	QPC
X1	[5,4,4,6,4,4]	0.500
X2	[3,2,2,2,3,2]	0.444
X3	[5,4,4,3,4,4]	0.500
X4	[5,4,4,5,4,4]	0.444
X5	[5,4,4,6,4,4]	0.500
X6	[3,4,4,2,2,4]	0.611
X7	[3,2,2,2,3,1]	0.611
X8	[3,3,1,4,3,4]	0.611
X9	[3,4,4,2,2,4]	0.611
X10	[5,4,2,6,4,4]	0.667

Table 3.35 Population for next generation.

The above steps are repeated for a pre-specified number of generations. Thereafter, the Top-k query plans are produced as output.

3.3. COMPARISON OF THE PROPOSED MODELS

The proposed models DQPG$_{AIS}$-I and DQPG$_{AIS}$-II are variants of algorithm CLONALG [LJ02]. These two models give similar output, top-k query plans, but in different ways. The basic difference between the two models are in the parameters given below.

Mutation: Mutation is inversely proportional to the fitness of antibodies. The purpose of inverse relation between fitness of antibody query plan and mutation is that higher the fitness of the query plan, lesser would be the changes in the query plan. Thus, a query plan with higher fitness would be better to be passed to the next generation and vice versa. In $DQPG_{AIS}$-I, mutation rate is calculated by product of fitness value and the number of sites used in the query plan. On the other hand, generated clones of antibodies in $DQPG_{AIS}$-II are mutated with a constant rate using Roulette Wheel selection.

Clone Generation: $DQPG_{AIS}$-I and $DQPG_{AIS}$-II use different measures to compute the number of clones. The clones are thereafter selected using Roulette Wheel Selection.

3.4. EXPERIMENTAL RESULTS

GA based DQPG algorithm ($DQPG_{GA}$) given in [VSV10][VSV11], $DQPG_{AIS}$-I and $DQPG_{AIS}$-II are implemented using MATLAB R2009a. Comparison of DQPG-I with $DQPG_{GA}$ is discussed next.

3.4.1. $DQPG_{AIS}$-I vs $DQPG_{GA}$

First, graphs showing Average QPC of Top-10 query plans over 400 generations were plotted for 12, 16, 20, 24 and 28 relations and are shown in Figure 3.7, Figure 3.8, Figure 3.9, Figure 3.10 and Figure 3.11 respectively. The parameters considered to plot the graphs are mutation rate (0.05) and crossover (0.8) for GA, clone rate (0.99, 0.98) and best query plans for clone generation from the whole population (0.4, 0.6) in case of $DQPG_{AIS}$-I. It is observed from the graphs that $DQPG_{AIS}$-I gives better results over $DQPG_{GA}$. As the number of generation increases, query plans with lower Average QPC is produced.

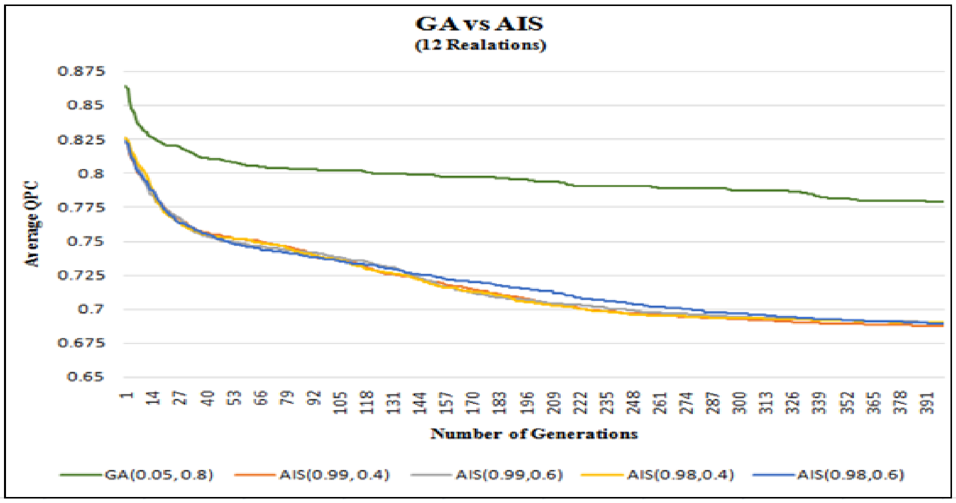


Figure 3.7 GA vs AIS for 12 Relations.

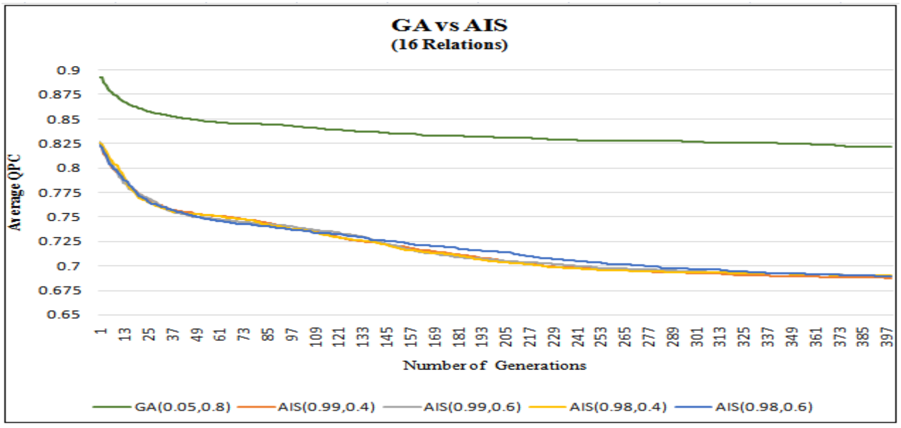


Figure 3.8 GA vs AIS for 16 Relations.

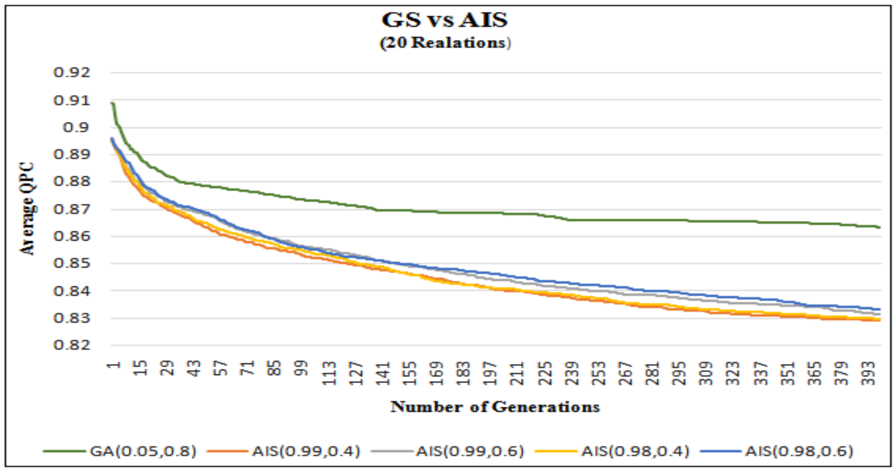


Figure 3.9 GA vs AIS for 20 Relations.

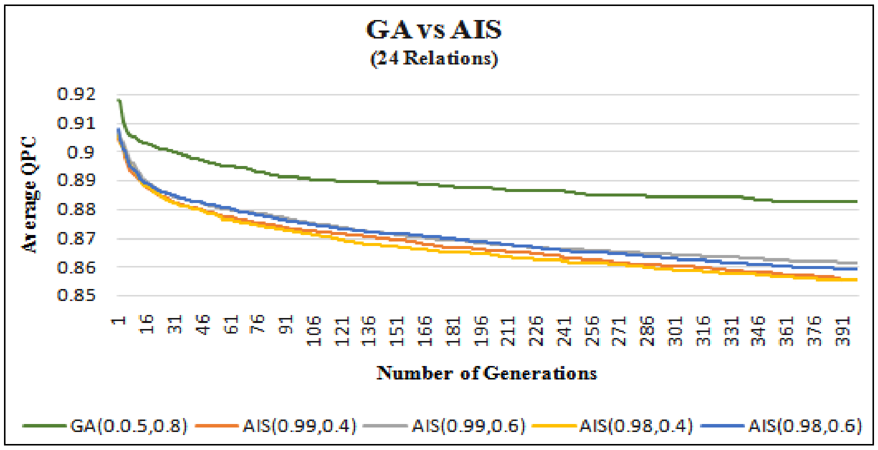


Figure 3.10 GA vs AIS for 24 Relations.

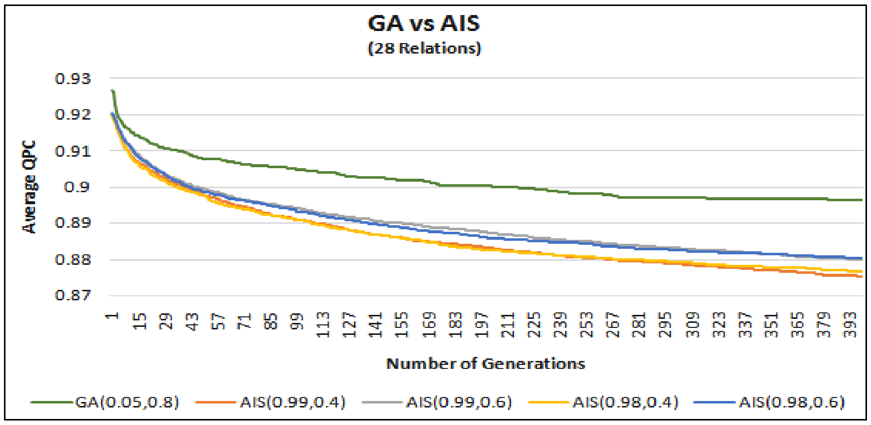


Figure 3.11 GA vs AIS for 28 Relations.

Next, graphs showing Average QPC vs Top-K query plans produced by DQPG$_{AIS}$-I and DQPG$_{GA}$, after 400 generations for 12, 16, 20 and 2 relations are plotted and are shown in Figure 3.12, Figure 3.13, Figure 3.14 and Figure 3.15 respectively. It can be observed from the graphs that DQPG$_{AIS}$-I. in comparison to DQPG$_{GA}$, is able to generate query plans having comparatively lower Average QPC.

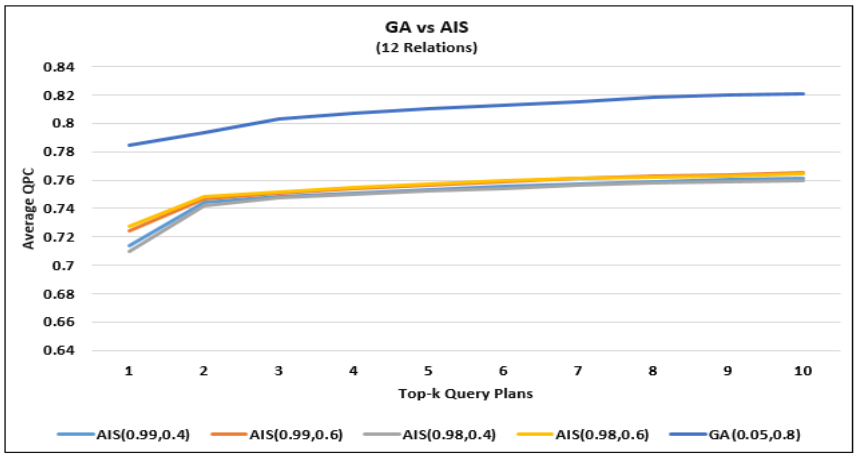


Figure 3.12 GA vs AIS Top-k Query Plans for 12 Relations.

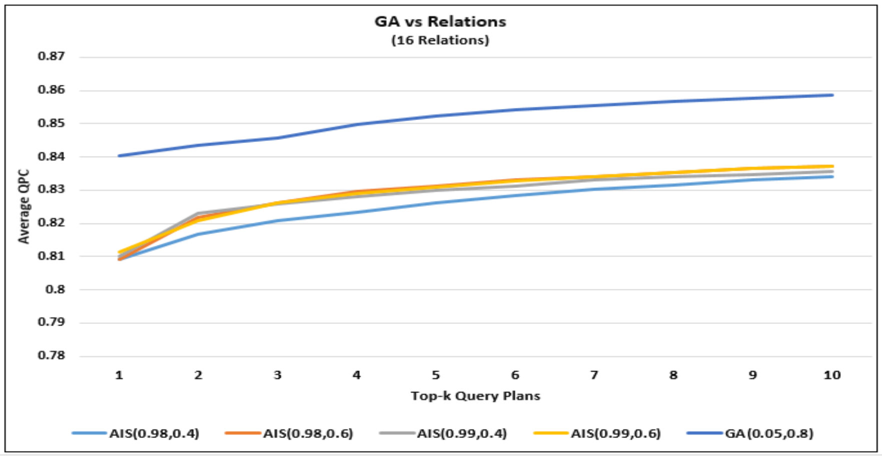


Figure 3.13 GA vs AIS Top-k Query Plans for 16 Relations.

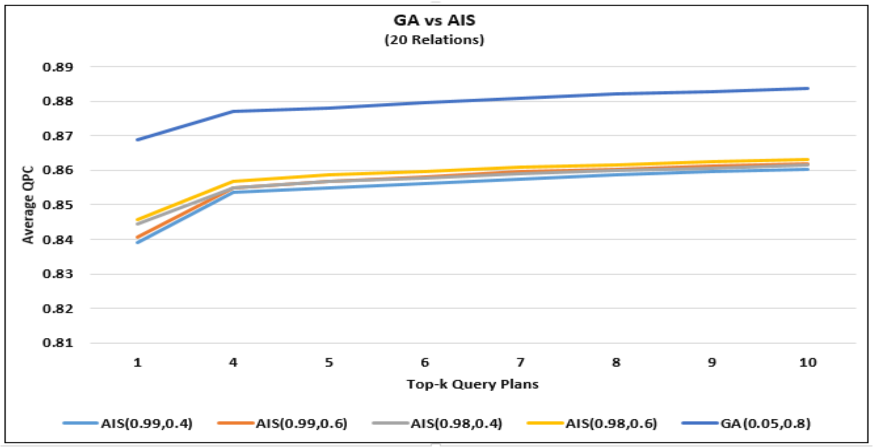


Figure 3.14 GA vs AIS Top-k Query Plans for 20 Relations.

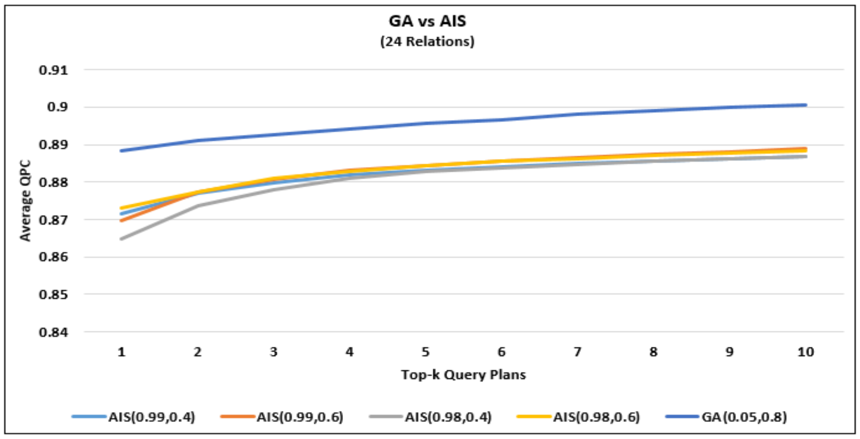


Figure 3.15 GA vs AIS Top-k Query Plans for 24 Relations.

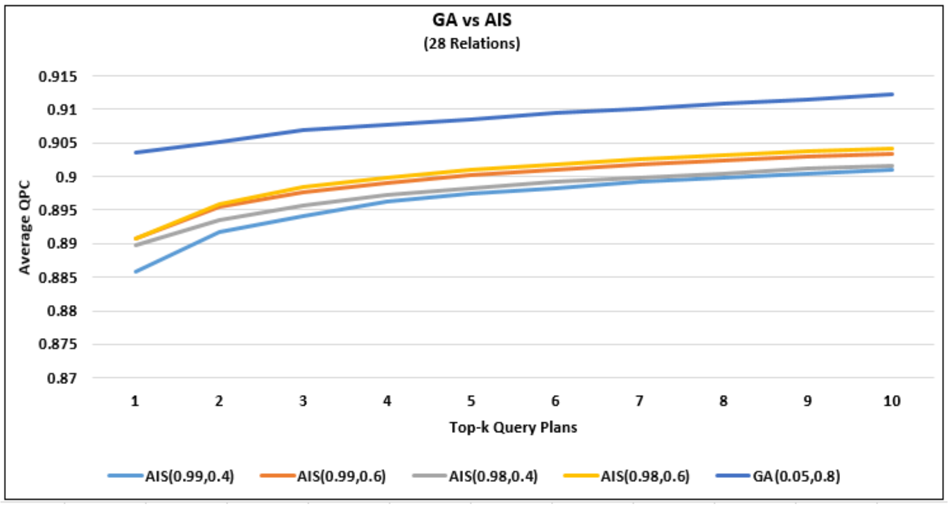


Figure 3.16 GA vs AIS Top-k Query Plans for 28 Relations.

Next, experimental comparisons of DQPG_{AIS}-II and DQPG_{GA} are carried out and is discussed next.

3.4.2. DQPGAIS-II vs DQPG_{GA}

First, graphs showing Average QPC of Top-k query plans over 400 generations were plotted for 12, 16, 20, and 24 relations and are shown in Figure 3.17, Figure 3.18, Figure 3.19, and Figure 3.20 respectively. The graphs were plotted for mutation rate (0.05) and crossover (0.8) for GA. The parameters considered for DQPG_{AIS}-II are clone generation

(0.4, 0.6), best query plans of whole population used in clone generation (0.99, 0.98) and mutation rate (0.05, 0.01). It is observed from the graphs that DQPG$_{AIS}$-II gives better results over DQPG$_{GA}$. As the number of generation increases, query plans with lower Average QPC is produced.

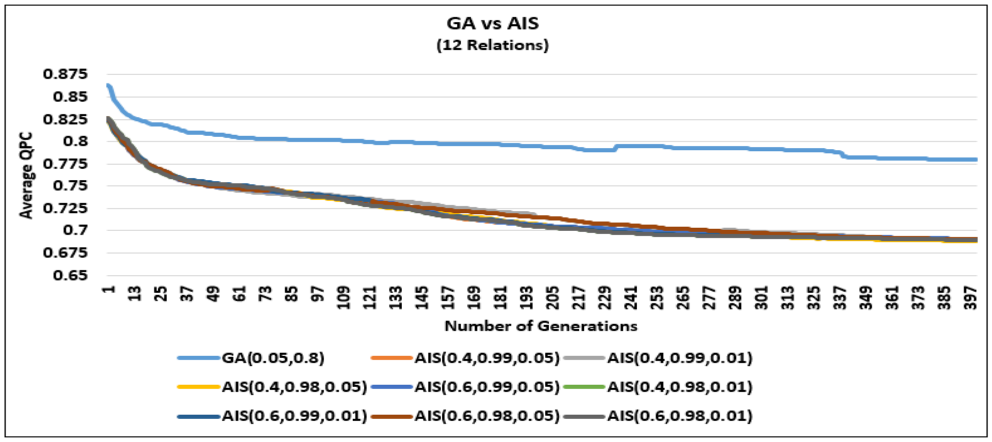


Figure 3.17 GA vs AIS for 12 Relations.

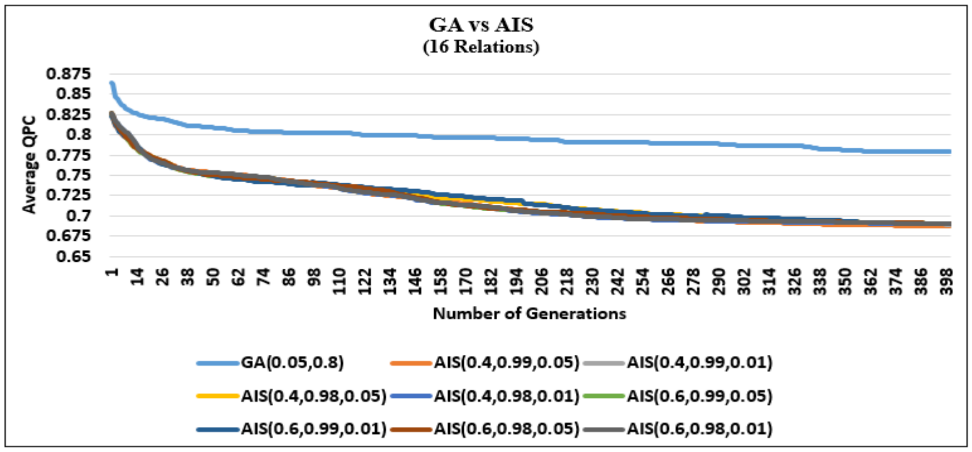


Figure 3.18 GA vs AIS for 16 Relations.

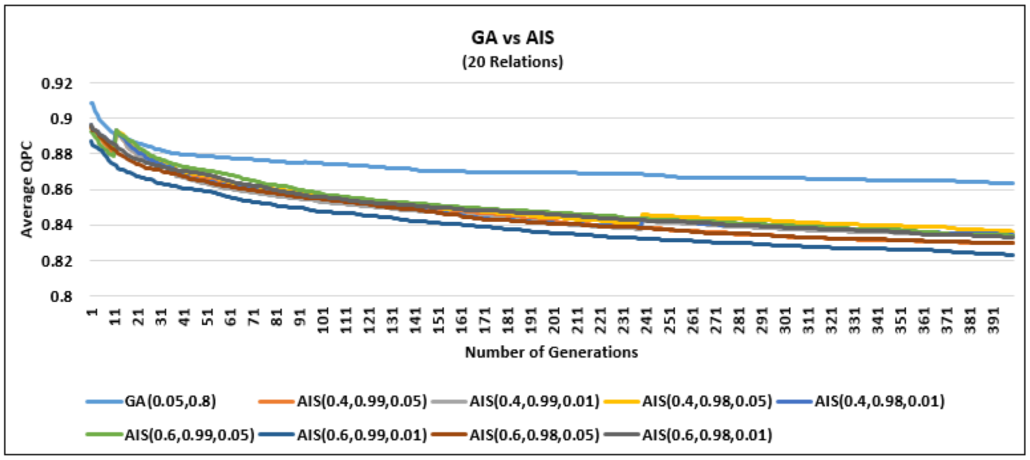


Figure 3.19 GA vs AIS for 20 Relations.

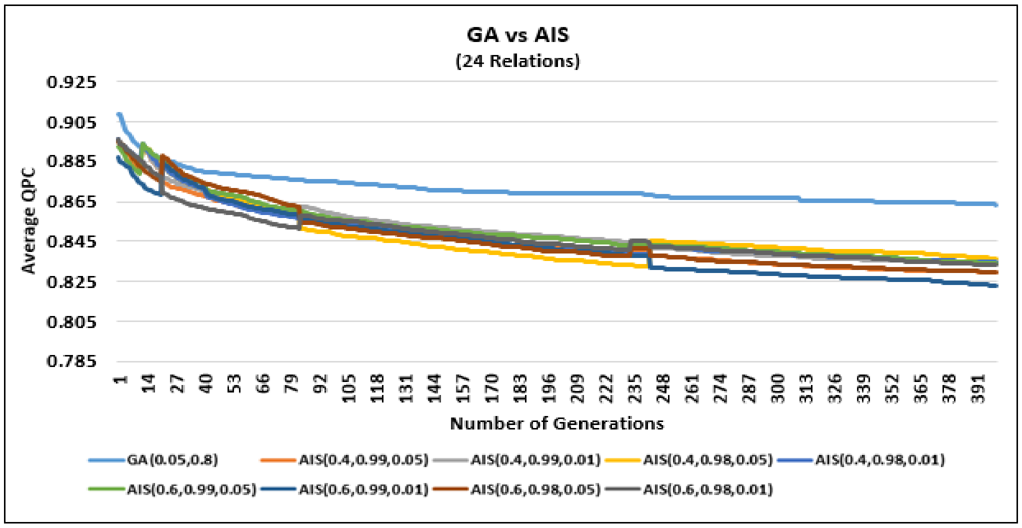


Figure 3.20 GA vs AIS for 24 Relations.

Next, graphs showing Average QPC vs Top-K query plans produced by $DQPG_{AIS}$-II and $DQPG_{GA}$, after 400 generations for 12, 16, 20 and 24 relations are plotted and are shown in Figure 3.21, Figure 3.22, Figure 3.23 and Figure 3.24 respectively. It can be observed from the graphs that $DQPG_{AIS}$-II, in comparison to $DQPG_{GA}$, is able to generate query plans having comparatively lower Average QPC.

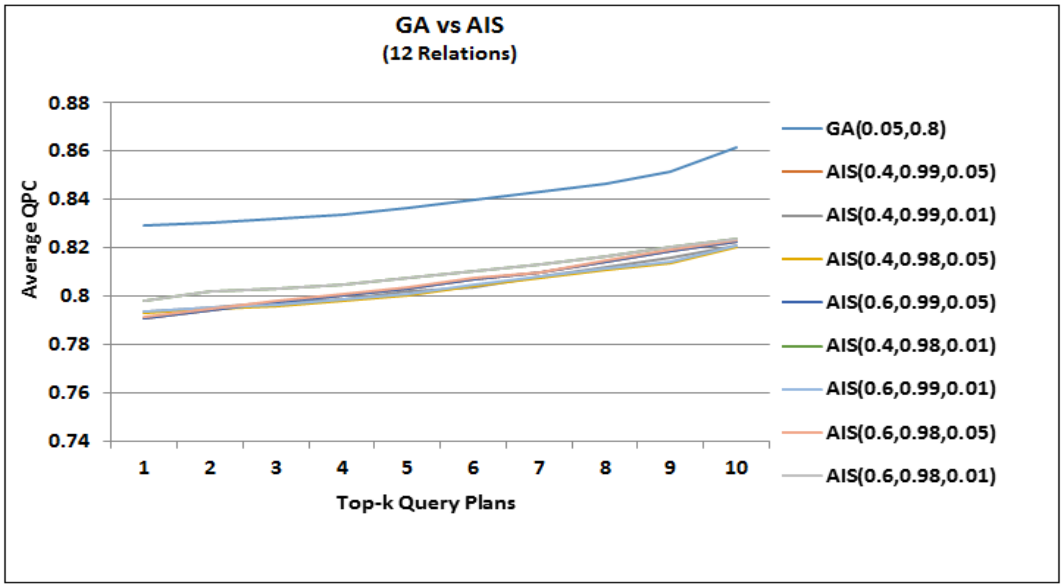


Figure 3.21 GA vs AIS Top-k Query Plans for 12 Relations.

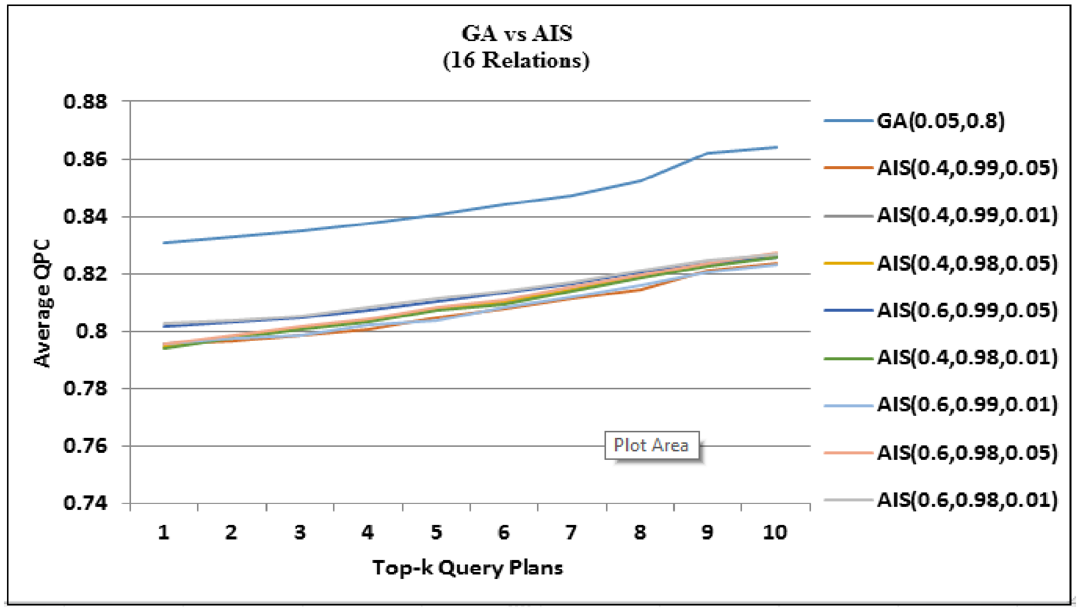


Figure 3.22 GA vs AIS Top-k Query Plans for 16 Relations.

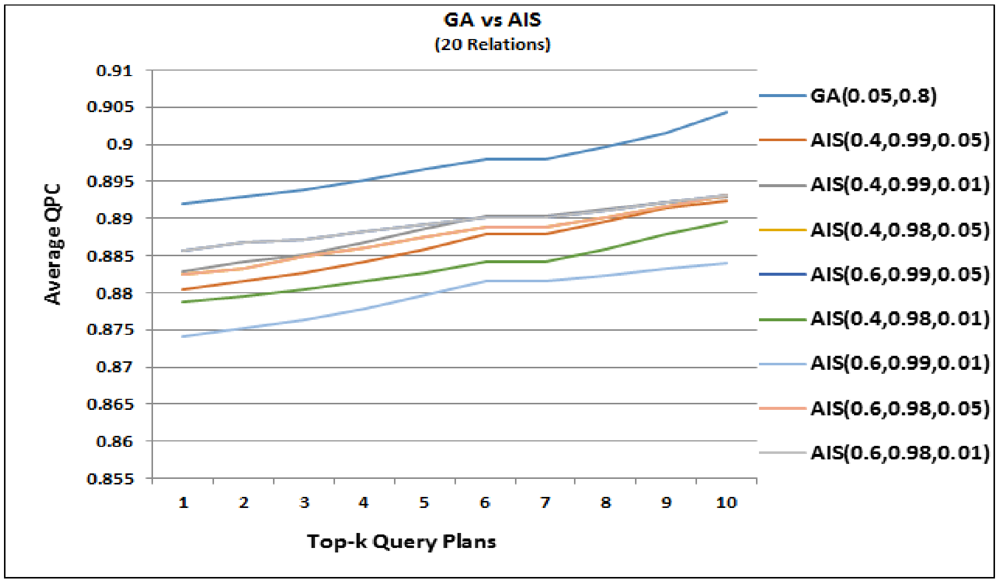


Figure 3.23 GA vs AIS Top-k Query Plans for 20 Relations.

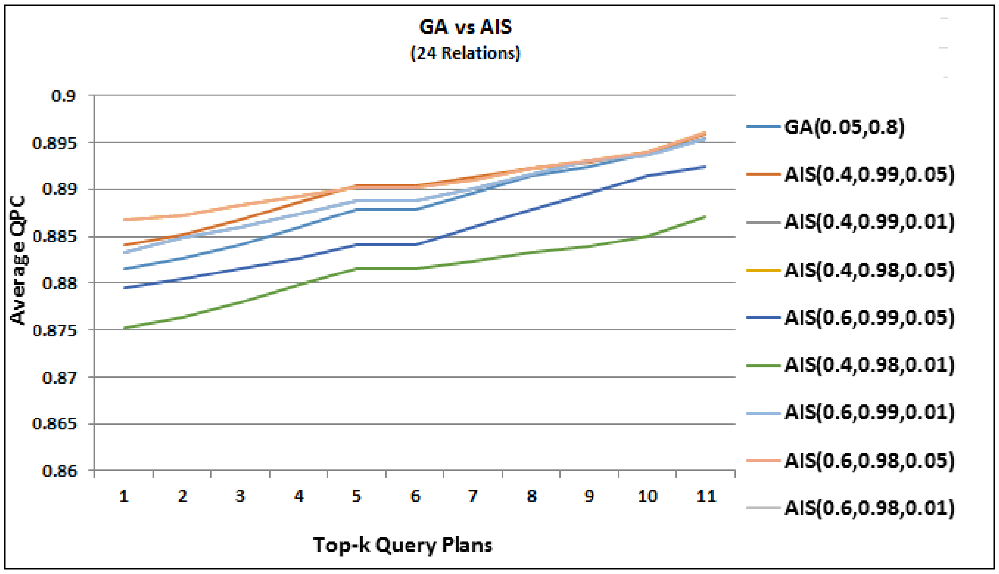


Figure 3.24 GA vs AIS Top-k Query Plans for 24 Relations.

CHAPTER 4

CONCLUSION

In this dissertation, an attempt has been made to address the DQPG problem given in [VSV10, VSV11]. In this regard, two AIS based DQPG models $DQPG_{AIS}$-I and $DQPG_{AIS}$-II are proposed that generates Top-K query Plans for a given distributed query. In $DQPG_{AIS}$-I, mutation rate is computed by product of fitness value and the number of sites used in the query plan. On the other hand, generated clones of antibody query plans in $DQPG_{AIS}$-II are mutated with a constant rate using Roulette Wheel selection. $DQPG_{AIS}$-I and $DQPG_{AIS}$-II use different measures to compute the number of clones. Further, experimental based comparison of the two proposed models $DQPG_{AIS}$-I and $DQPG_{AIS}$-II with $DQPG_{GA}$ showed that both the proposed models are able to generate query plans having comparatively lower average QPC. The query plans, so generated would lead to efficient processing of distributed queries.

REFERENCES

[A89] Alan S.Perelson. Immune network theory. Immunological Reviews,(10):5-36, 1989.

[AN87] Ada, G. L. & Nossal, G. J. V. (1987), “The Clonal Selection Theory”, Scientific American, 257(2), pp. 50-57.

[ATA12] O.Abdoun, C.Tajani, J.Abouchabaka ‘Analyzing the performance of mutation operators to solve the traveling salesman problem’Int. J.Emerg. Sci., 2 (1) (2012), pp. 61-77

[BC81] BERNSTEIN, P.A, AND CHIU, DW, “Using semi-joins to solve relational queries,” J ACM28, 1 (Jan.1981), 25--40.

[BS12] S. Binitha, S.Siva Sathya, “A survey of bio inspired optimization algorithms,” Int. J. Soft Comput. Eng. (IJSCE), 2231-2307, 2 (2) (2012).

[BV90] H. Bersini and F. Varela 1990. Hints for adaptive problem solving gleaned from immune networks. Proceedings of the First Conference on Parallel Problem Solving from Nature, 343–354.

[CHF+10] Bin Chen, Fengru Huang, Yu Fang, Zhou Huang, and Hui Lin. An approach for heterogeneous and loosely coupled geospatial data distributed computing. Computers and Geosciences, 36(7):839 - 847, 2010.

[CP84] Stefano Ceri and Giuseppe Pelagatti. Distributed Databases: Principles and systems. McGraw-Hill Book Company, 1984. 25.

[CT94] C.J. Gibert and T. W. Routen. Associative memory in an immune-based system. In Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94), pages 852–857, Seattle, July 31-August 4 1994.

[CY92] Chen, M. S., and Yu, P. S., Interleaving a join sequence with semi joins in distributed query processing. IEEE Transaction on Parallel and distributed System 5, pp 611-621, 1992.

[CY93] Chen, M. S., and Yu, P. S., Combining joining and semi join operations for distributed query processing. IEEE Transaction on knowledge and Data Engineering 5, pp 534-534, 1993.

[CZ99] De Castro, L. N. & Von Zuben, F. J. (1999), "Artificial Immune Systems: Part I – Basic Theory and Applications", Technical Report – RT DCA 01/99, p. 95.

[CZ00] De Castro, L. N. & Von Zuben, F. J. (2000), "The Clonal Selection Algorithm with Engineering Applications", GECCO'00 – Workshop Proceedings, pp. 36-37.

[D95] C.J. Date, "An Introduction to Database Systems, Addison Wesley," Reading MA, 1995.

[E74] CODD, E.F. Recent investigations in relational data base systems. Proc. IFIP Congress 1974, North-Holland Pub. Co., Amsterdam, pp. 1017-1021.

[EE95] John E. Hunt and Ennise E. Cooke. An adaptive, distributed learning system, based on the immune system. In Proceedings of the IEEE International Conference on Systems, Man and Cybernatics, pages 2494-2499, 1995.

[EMG99] E. Bonabeau, M. Dorigo, and G. Theraulaz. Swarm Intelligence: From Natural to Artificial System. Oxford University Press, New York, 1999.

[EN10] Elmasari, Navathe, Fundamental of Database system, sixth edition, Pearson Education, Inc. 2010.

[ESW78] Robert Epstein , Michael Stonebraker , Eugene Wong, Distributed query processing in a relational data base system, Proceedings of the 1978 ACM SIGMOD international conference on management of data, May 31-June 02, 1978, Austin, Texas [doi>10.1145/509252.509292].

[FJY+13] Fister, I.J., Yang, X-S., Fister, I., Brest, J., Fister, D. (2013e). 'A brief review of nature-inspired algorithms for optimization'. Electrotechnical Review. 80, 3, 116-122

[FM08] Floreano, D., Mattiusi, C., Bio-Inspired Artificial Intelligence: Theories, Methods and Technologies, MIT Press, 2008.

[FPP86] J. D. Farmer, N. H. Packard and A. S. Perelson 1986. 'The immune system, adaptation and machine learning'. Physica, 22D, 187–204.

[G85] Grefenstette, j.J., Proceedings of 1st International Conference on Genetic Algorithms and their application, Hillsdale, NJ: Lawrence Erlbaum, pp. 160-168, 1985.

[G98] M. Gregory, "Genetic algorithm optimization of distributed database queries," in Proc. ICEC, 1998, pp. 271–276.

[GR94] C. J. Gilbert and T. W. Routen 1994. Associative memory in an immune-based system. Proceedings of AAAI'94, 2, 852–857.

[H75] Holland, J. H. (1975), Adaptation in Natural and Artificial Systems, MIT Press.

[H86] G. W. Hoffmann 1986. A neural network model based on the analogy with the immune system. Journal of Theoretical Biology, 122, 33–67.

[HC96] Hunt, J. E. & Cooke, D. E. (1996), "Learning Using an Artificial Immune System", Journal of Network and Computer Applications, 19, pp. 189-212.

[HHC+9] Husbands, P., Harvey, I., Cliff, D., Miller, G., "Artificial Evolution: A new path for artificial intelligence", Brain and Cognition-34, pp. 130-159, 1997.

[K00] D. KOSSMAN, "The state of the art in distributed query processing"., ACM Comput. Surv. 32, 4 (Dec. 2000). 422–46.

[L06] L.N. Decastro, "Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications", Chapman and Hall/CRC, 2006.

[LJ02] L.N. Decastro and J. Timmis, "Artificial Immune Systems: A Novel Paradigm to Pattern Recognition," University of Paisley, UK, pp. 67-84, 2002.

[M94] M. Millonas. Swarms, Phase Transitions, and Collective Intelligence. Addison-Wesley Publishing Company, Reading (1994).

[M99] M. Melanie, 'An Introduction to Genetic Algorithms', Massachusetts: MIT Press, 1999.

[MDP+1] Mishra, A. K., Das M.N., Panda, T.C., Swarm Intelligence Optimization: Editorial Survey, International Journal of Emerging Technology and Advanced Engineering, 2013.

[MHH] B.M. Monjurul Alom, Frans Henskens and Michael Hannaford, "Query processing and optimization in Distributed Database System," International Journal of Computer Science and Network Security (IJCSNS)., Sep2009.

[ML94] Srinivas. M and Patnaik. L, "Adaptive probabilities of crossover and mutation in genetic algorithms," IEEE Transactions on System, Man and Cybernetics, vol.24, no.4, pp.656–667, 1994

[MTM11] Mishra, K.K., Tiwari, S., Misra, A.K.: A bio inspired algorithm for solving optimization problems. In: 2011 2nd International Conference on Computer and Communication Technology (ICCCT), September 15-17, pp. 653–659 (2011).

[N93] Nossal, G. J. V. (1993), "The Molecular and Cellular Basis of Affinity Maturation in the Antibody Response", Cell, 68, pp. 1-2.

[NG94] Nossal, G. J. V. (1994), "Negative Selection of Lymphocytes", Cell, 76, pp. 229-239.

[OV91] Ozsu, M.T., Valduriez, P., Distributed Database system: where are we now? IEEE Computer, Vol. 4, No. 8, pp68-78, August 1991.

[OV11] Ozsu, M.T., Valduriez, P., Principles Distributed Databased System, Third Edition, Springer, 2011.

[OZ97] Ozsoyoglu, Z.M., and Zhou, N., Distributed query processing in Broadcasting local area network. In Proc. 20th Hawaii Int. Conf. on system Sciences, pp 419-429, 1987

[PC91] Phlilips, A.B., Chiu, D.W., Using Semi-joins to solve Relational Queries, Journal of the Associated for computing machinery, vo. 28, pp.25-40, 1991.

[RM71] S. Rho, S. T. March, "Optimizing join queries: A genetic algorithm Approach". Annals of Operations Research, 71, 199–228.

[R90] Richard G. Weinand. Somatic mutation, affinity maturation and antibody repertoire: A computer Model.Journal of Theoretical Biology, 143(3)343-382, 1990.

[R12] Ray, C., Distributed Database System, Pearson education India, 2012.

[SAL+94] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-Nonself Discrimination in a Computer. In Proceedings of IEEE Symposium on Research in Security and Privacy, pages202-212, Oakland, CA, 16–18 May 1994.

[SAS+09] Das, S., Biswas, A., Dasgupta, S., Abraham, A. (2009). Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications, Volume 203/2009 of Studies in Computational Intelligence. Springer Berlin/Heidelberg, 23-55.

[SBRA93] S. Forrest, B. Javornik, R. Smith, and A.S. Perelson. Using genetic algorithms to explore pattern recognition in the immune system. Evolutionary Computation, 1(3):191–211, 1993.

[SBM98] S. Salza , G. Barone and T. Morzy "A distributed algorithm for global query optimization in multi database systems", International Conference on Advances in Database and Information Systems, pp.95 -106 1998.

[SC92] P. E. Seiden and F. Celada 1992. A model for simulating cognate recognition and response in the immune system. Journal of Theoretical Biology, 158, 329–357.

[SC10] Sevinc E, Cosar A, "An Evolutionary Genetic Algorithm for Optimization of Distributed Database Queries", The Computer journal, 2010.

[SG98] Arun Swami , Anoop Gupta, Optimization of large join queries, Proceedings of the 1988 ACM SIGMOD international conference on Management of data, p.8-17, June 01-03, 1988, Chicago, Illinois, United States [doi>10.1145/50202.50203].

[SL90] Amit P. Sheth , James A. Larson, Federated database systems for managing distributed, heterogeneous, and autonomous databases, ACM Computing Surveys (CSUR), v.22 n.3, p.183-236, Sept. 1990 [doi>10.1145/96602.96604].

[TT90] Gomer Thomas , Glenn R. Thompson , Chin-Wan Chung , Edward Barkmeyer , Fred Carter , Marjorie Templeton , Stephen Fox , Berl Hartman, Heterogeneous distributed database systems for production use, ACM Computing Surveys (CSUR), v.22 n.3, p.237-266, Sept. 1990.

[UD05] U. Aickelin, D. Dasgupta Artificial immune systems tutorial. Burke, G. Kendall (Eds.), Search methodologies—introductory tutorials in optimization and decision support techniques, Kluwer (2005), pp. 375–399.

[VBS+11] Chifu, V.R. , Pop, C.B. , Salomie, I. , Dinsoreanu, M. , Niculici, A.N. , Suia,

D. S. (2011). 'Bio-inspired methods for selecting the optimal web service composition: bees or cuckoos intelligence?'. International Journal of Business Intelligence and Data Mining. 6, 4, 321-344.

[VSV10] Vijay Kumar, T. V., Singh, V., & Verma, A. K. (2010). Generating distributed query processing plans using genetic algorithm. In DSDE 2010 - International Conference on Data Storage and Data Engineering (pp. 173–177).

[VSV11] Vijay Kumar, T.V., Singh, V., Verma, A.K.: Distributed Query Processing Plans Generation using Genetic Algorithm. International Journal of Computer Theory and Engineering 3(1), 38–45 (2011).

[WY91] Eugene Wong, Karel Youssefi, Decomposition—a strategy for query processing, ACM Transactions on Database Systems (TODS), vol.1 no.3, p.223-241, Sept. 1976 [doi: 10.1145/320473.320479].

[XY10] M. Xifeng, F.Yuanyuan, "Distributed Database System Query Optimization Algorithm Research", IEEE Intl. Conf. on Computer Science and Information Technology", Vol.8, pp 657- 660, 2010.

[YC84] Yu C.T , Chang C.C., "Distributed query processing," ACM Computing Surveys, vol. 16, pp 399-433, 1984.

[YZH+0] Y. Zhu, Z. Tang, H. Dai, S. Gao. Cooperation artificial immune system with application to travelling salesman problem. ICIC Express Letters, 2 (2) (2008), pp. 143–148.

[ZHW05] Zhu, Q. and P.-Å. Larson. (1996). Global Query Processing and Optimization in the CORDS Multidatabase System. In Proc. of 9th Int'l Conf. on Paral and Distr. Comp. Syst., pp 640–648.

BIBLIOGRAPHY

1. Alan S. Perelson. Immune network theory. Immunological Reviews,(10):5-36, 1989.
2. Ada, G. L. & Nossal, G. J. V. (1987), “The Clonal Selection Theory”, Scientific American, 257(2), pp. 50-57.
3. BERNSTEIN, P.A, AND CHIU, DW, “Using semi-joins to solve relational queries,” J ACM28, 1 (Jan.1981), 25--40.
4. H. Bersini and F. Varela 1990. Hints for adaptive problem solving gleaned from immune networks. Proceedings of the First Conference on Parallel Problem Solving from Nature, 343–354.
5. Bin Chen, Fengru Huang, Yu Fang, Zhou Huang, and Hui Lin. An approach for heterogeneous and loosely coupled geospatial data distributed computing. Computers and Geosciences, 36(7):839 - 847, 2010.
6. Stefano Ceri and Giuseppe Pelagatti. Distributed Databases: Principles and systems. McGraw-Hill Book Company, 1984. 25.
7. C.J. Gibert and T. W. Routen. Associative memory in an immune-based system. In Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94), pages 852–857, Seattle, July 31-August 4 1994.
8. Chen, M. S., and Yu, P. S., Interleaving a join sequence with semi joins in distributed query processing. IEEE Transaction on Parallel and distributed System 5, pp 611-621, 1992.

9. Chen, M. S., and Yu, P. S., Combining joining and semi join operations for distributed query processing. IEEE Transaction on knowledge and Data Engineering 5, pp 534-534, 1993.
10. De Castro, L. N. & Von Zuben, F. J. (1999), “Artificial Immune Systems: Part I – Basic Theory and Applications”, Technical Report – RT DCA 01/99, p. 95.
11. De Castro, L. N. & Von Zuben, F. J. (2000), “The Clonal Selection Algorithm with Engineering Applications”, GECCO’00 – Workshop Proceedings, pp. 36-37.
12. C.J. Date, “An Introduction to Database Systems, Addison Wesley,” Reading MA, 1995.
13. CODD, E.F. Recent investigations in relational data base systems. Proc. IFIP Congress 1974, North-Holland Pub. Co., Amsterdam, pp. 1017-1021.
14. John E. Hunt and Ennise E. Cooke. An adaptive, distributed learning system, based on the immune system. In Proceedings of the IEEE International Conference on Systems, Man and Cybernatics, pages 2494-2499, 1995.
15. E. Bonabeau, M. Dorigo, and G. Theraulaz. Swarm Intelligence: From Natural to Artificial System. Oxford University Press, New York, 1999.
16. Elmasari, Navathe, Fundamental of Database system, sixth edition, Pearson Education, Inc. 2010.
17. Robert Epstein , Michael Stonebraker , Eugene Wong, Distributed query processing in a relational data base system, Proceedings of the 1978 ACM SIGMOD international conference on management of data, May 31-June 02, 1978, Austin, Texas [doi>10.1145/509252.509292].

18. Fister, I.J., Yang, X-S., Fister, I., Brest, J., Fister, D. (2013e). 'A brief review of nature-inspired algorithms for optimization'. Electrotechnical Review. 80, 3, 116-122.

19. Floreano, D., Mattiusi, C., Bio-Inspired Artificial Intelligence: Theories, Methods and Technologies, MIT Press, 2008.

20. J. D. Farmer, N. H. Packard and A. S. Perelson 1986. 'The immune system, adaptation and machine learning'. Physica, 22D, 187–204.

21. Grefenstette, J., Proceedings of 1st International Conference on Genetic Algorithms and their application, Hillsdale, NJ: Lawrence Erlbaum, pp. 160-168, 1985.

22. M. Gregory, "Genetic algorithm optimization of distributed database queries," in Proc. ICEC, 1998, pp. 271–276.

23. C. J. Gilbert and T. W. Routen 1994. Associative memory in an immune-based system. Proceedings of AAAI'94, 2, 852–857.

24. Holland, J. H. (1975), Adaptation in Natural and Artificial Systems, MIT Press.

25. G. W. Hoffmann 1986. A neural network model based on the analogy with the immune system. Journal of Theoretical Biology, 122, 33–67.

26. Hunt, J. E. & Cooke, D. E. (1996), "Learning Using an Artificial Immune System", Journal of Network and Computer Applications, 19, pp. 189-212.

27. Husbands, P., Harvey, I., Cliff, D., Miller, G., "Artificial Evolution: A new path for artificial intelligence", Brain and Cognition-34, pp. 130-159, 1997.

28. D. KOSSMAN, "The state of the art in distributed query processing"., ACM Comput. Surv. 32, 4 (Dec. 2000). 422–46.

29. L.N. Decastro and J. Timmis, “Artificial Immune Systems: A Novel Paradigm to Pattern Recognition”, University of Paisley, UK, pp. 67-84, 2002.
30. M. Millonas. Swarms, Phase Transitions, and Collective Intelligence. Addison-Wesley Publishing Company, Reading (1994).
31. M. Melanie, ‘An Introduction to Genetic Algorithms’, Massachusetts: MIT Press, 1999.
32. Mishra, A. K., Das M.N., Panda, T.C., Swarm Intelligence Optimization: Editorial Survey, International Journal of Emerging Technology and Advanced Engineering, 2013.
33. B.M. Monjurul Alom, Frans Henskens and Michael Hannaford, “Query processing and optimization in Distributed Database System,” International Journal of Computer Science and Network Security (IJCSNS)., Sep2009.
34. Srinivas. M and Patnaik. L, "Adaptive probabilities of crossover and mutation in genetic algorithms," IEEE Transactions on System, Man and Cybernetics, vol.24, no.4, pp.656–667, 1994.
35. Mishra, K.K., Tiwari, S., Misra, A.K.: A bio inspired algorithm for solving optimization problems. In: 2011 2nd International Conference on Computer and Communication Technology (ICCCT), September 15-17, pp. 653–659 (2011).
36. Nossal, G. J. V. (1993), “The Molecular and Cellular Basis of Affinity Maturation in the Antibody Response”, Cell, 68, pp. 1-2.
37. Nossal, G. J. V. (1994), “Negative Selection of Lymphocytes”, Cell, 76, pp. 229-239.

38. Ozsu, M.T., Valduriez, P., Distributed Database system: where are we now? IEEE Computer, Vol. 4, No. 8, pp68-78, August 1991.
39. Ozsu, M.T., Valduriez, P., Principles Distributed Databased System, Third Edition, Springer, 2011.
40. Ozsoyoglu, Z.M., and Zhou, N., Distributed query processing in Broadcasting local area network. In Proc. 20th Hawaii Int. Conf. on system Sciences, pp 419-429, 1987
41. Phlilips, A.B., Chiu, D.W., Using Semi-joins to solve Relational Queries, Journal of the Associated for computing machinery, vo. 28, pp.25-40, 1991.
42. S. Rho, S. T. March, "Optimizing join queries: A genetic algorithm Approach". Annals of Operations Research, 71, 199–228.
43. Richard G. Weinand.Somatic mutation, affinity maturation and antibody repertoire: A computer Model.Journal of Theoretical Biology, 143(3)343-382, 1990.
44. Ray, C., Distributed Database System, Pearson education India, 2012.
45. S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-Nonself Discrimination in a Computer. In Proceedings of IEEE Symposium on Research in Security and Privacy, pages202-212, Oakland, CA, 16–18 May 1994.
46. Das, S., Biswas, A., Dasgupta, S., Abraham, A. (2009). Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications, Volume 203/2009 of Studies in Computational Intelligence. Springer Berlin/Heidelberg, 23-55
47. S. Forrest, B. Javornik, R. Smith, and A.S. Perelson. Using genetic algorithms to explore pattern recognition in the immune system. Evolutionary Computation, 1(3):191–211, 1993.

48. S. Salza , G. Barone and T. Morzy "A distributed algorithm for global query optimization in multi database systems", International Conference on Advances in Database and Information Systems, pp.95 -106 1998

49. P. E. Seiden and F. Celada 1992. A model for simulating cognate recognition and response in the immune system. Journal of Theoretical Biology, 158, 329–357.

50. Sevinc E, Cosar A, "An Evolutionary Genetic Algorithm for Optimization of Distributed Database Queries", The Computer journal, 2010.

51. Arun Swami , Anoop Gupta, Optimization of large join queries, Proceedings of the 1988 ACM SIGMOD international conference on Management of data, p.8-17, June 01-03, 1988, Chicago, Illinois, United States [doi>10.1145/50202.50203].

52. Amit P. Sheth , James A. Larson, Federated database systems for managing distributed, heterogeneous, and autonomous databases, ACM Computing Surveys (CSUR), v.22 n.3, p.183-236, Sept. 1990 [doi>10.1145/96602.96604].

53. Gomer Thomas , Glenn R. Thompson , Chin-Wan Chung , Edward Barkmeyer , Fred Carter , Marjorie Templeton , Stephen Fox , Berl Hartman, Heterogeneous distributed database systems for production use, ACM Computing Surveys (CSUR), v.22 n.3, p.237-266, Sept. 1990.

54. U. Aickelin, D. Dasgupta Artificial immune systems tutorial. Burke, G. Kendall (Eds.), Search methodologies—introductory tutorials in optimization and decision support techniques, Kluwer (2005), pp. 375–399.

55. Chifu, V.R. , Pop, C.B. , Salomie, I. , Dinsoreanu, M. , Niculici, A.N. , Suia, D.S. (2011). 'Bio-inspired methods for selecting the optimal web service composition:

bees or cuckoos intelligence?’. International Journal of Business Intelligence and Data Mining. 6, 4, 321-344.

56. Vijay Kumar, T.V., Singh, V., Verma, A.K.: Distributed Query Processing Plans Generation using Genetic Algorithm. International Journal of Computer Theory and Engineering 3(1), 38–45 (2011).

57. Eugene Wong , Karel Youssefi, Decomposition—a strategy for query processing, ACM Transactions on Database Systems (TODS), vol.1 no.3, p.223-241, Sept. 1976 [doi: 10.1145/320473.320479].