Implementation of n-Queens Puzzle using Meta-Heuristic algorithm (Cuckoo Search)

A

Dissertation

submitted

in partial fulfillment

for the award of the Degree *of*

Master of Technology

in Department of Computer Science & Engineering

(with specialization in *Computer Science & Engineering*)

Supervisor:
Dr.Bright Keswani
Associate Professor and Head
Department of Computer Applications

Submitted By:
Ram Gopal Sharma
M.Tech CSE (Part-Time)
Enrolment No. SGVU101595090

Department of Computer Science & Engineering

Suresh Gyan Vihar University

Mahal, Jagatpura, Jaipur

2013

Candidate’s Declaration

I hereby declare that the work, which is being presented in the dissertation, entitled “Implementation of n-Queens Puzzle using Meta-heuristic Algorithm (Cuckoo Search)” in partial fulfilment for the award of Degree of “**Master of Technology**” with specialization in Computer Science & Engineering and submitted to the Department of Computer Science & Engineering, Suresh Gyan Vihar University is a record of my own investigations carried under the Guidance of **Dr. Bright Keswani**, Associate Professor and Head, Department of Computer Applications.

I have not submitted the matter presented in this dissertation anywhere for the award of any other Degree.

(Ram Gopal Sharma)

Enrolment No.: SGVU101595090

Counter Singed by

Dr. Bright Keswani

Associate Professor and Head

Department of Computer Applications

Suresh Gyan Vihar University, Jaipur

DETAILS OF CANDIDATE, SUPERVISOR AND EXAMINER

Name of Candidate: Ram Gopal Sharma **Roll No.** 112716

Deptt. of Study: Computer Science & Engineering

Enrolment No: SGVU101595090

Thesis Title: Implementation of n-Queens Puzzle using Meta-heuristic Algorithm (Cuckoo Search)

Supervisor and Examiner Recommended **(with Office Address including Contact Numbers, email ID)**		
Supervisor		
Internal Examiner		
1	**2**	**3**

Signature with Date

Programme Coordinator **Dean / Principal**

Abstract

This dissertation uses the concept of cuckoo search algorithm for the n-queens puzzle. This algorithm follows an important behaviour of cuckoo bird. Cuckoo search is the Meta-heuristic algorithms. We can use Meta-heuristic algorithms for combinatorial problems. The nature-inspired Meta-heuristic algorithms used in many optimization problems such as n-queens puzzle and travelling salesman problem i.e. NP-hard problems. The n-queen puzzle is the problem of keeping n queens on chessboard (n column, n row) so that keep at most 1 queen in each column, row and diagonal.

In our work, first search the position of queen in chessboard using Cuckoo search with L'evy flights, then proposed the algorithm. Then, we will compare the results of proposed algorithm with the results of backtracking algorithm and we see, our proposed algorithm is better than backtracking algorithm. We choose to develop Cuckoo Search application in Java programming language. We used JDK version 7 and NetBeans IDE version 6.8.

Acknowledgement

It's great pleasure to me to submit the dissertation report for the award of the Degree of M.Tech. I specially thankful to my esteemed guide **Dr. Bright Keswani**, Associate Professor and Head, Department of Computer Applications, Suresh Gyan Vihar University, Jaipur for his full support and direction throughout the course of our work. His dedication, perfection and persistence have been a perpetual source of stimulus for me.

I would also thankful for the contribution of **Mrs. Savita Shivani,** Program Coordinator (M.Tech, CSE), Suresh Gyan Vihar University, Jaipur for her full support and help during the course.

I would also thankful for the contribution of all the faculty members of the department for their gracious help during the course.

I humbly extend our sense of gratitude to administrator of this University for providing us their valuable help and time with a congenial working environment.

Ram Gopal Sharma

Roll No- 112716

Enrolment No- SGVU101595090

TABLE OF CONTENTS

List of Tables

List of Figures

List of Symbols and Abbreviations

PSO	Particle Swarm Optimization
CS	Cuckoo Search
CSP	Constraint Satisfaction Problem
NP	Non-Deterministic Polynomial-Time
DE	Differential Evolution
ABC	Artificial Bee Colony
NSP	Nurse Scheduling Problem
TSP	Travelling Salesman Problem
API	Application Programming Interface
JDK	Java Development Kit
IDE	Integrated Development Environment

Chapter 1

INTRODUCTION

1.1 INTRODUCTION

This dissertation uses the concept of cuckoo search algorithm for the n-queens puzzle. This algorithm follows an important behaviour of cuckoo bird. Cuckoo search is the Meta-heuristic algorithms Meta-heuristic algorithm describes computational methods that optimize a problem to improve a new solution with the previous solution iteratively. Meta-heuristic do some or no assumptions about the problem. Meta-heuristic gives good solution for the problem. Meta-heuristic algorithm used in many optimization problems.

Meta-heuristic algorithms inspired by nature such as, Particle Swarm Optimization, Firefly Algorithm etc are increasingly grown. These Meta-heuristic algorithms have been used in many optimization problems such as the n-queens puzzle i.e. NP-hard problems.

We can use Meta-heuristic algorithms for combinatorial optimization such as the travelling salesman problem. In travelling salesman problem the solutions increases more than exponentially as increasing the problem size.

Meta-heuristic algorithm also used for large size combinatorial problems. Meta-heuristics copy the best feature in nature such as biological systems. There are two important features.

1. Selection of the fittest
2. Adaptation to the environment.

For implementation, these features can be translated into important features of the modern meta-heuristics i.e. intensification and diversification. Intensification means to find the best solutions among the current best solutions and using diversification, the algorithm can search the best solution more space efficiently.

X.S. Yang and S. Deb developed an algorithm named **Cuckoo Search algorithm** in 2009 based on Meta-heuristic algorithm. This algorithm based on brood parasitism of cuckoo bird. Reproduction method seen in species is called brood parasitism. There are three types of brood parasitism.

1. Intraspecific
2. Interspecific
3. Nest Takeover

Some cuckoo bird laid their eggs in other nest. They may remove the other's egg to increase the hatching probability of own eggs. This is called nest takeover. Some host birds don't behave friendly against intruders and engage indirect conflict with them. In such situation host bird will throw those align eggs away or make a new nest elsewhere i.e.

increases their reproductivity. Generally, the host eggs hatch slightly after the cuckoo eggs. Whenever the first cuckoo egg is hatched, the first action it will take is to throw the host eggs outside from the nest; this action increases the food of cuckoo provided by its host bird

The aim of our work is to develop an algorithm based on cuckoo search algorithm. In our work, first search the position of queen in chessboard using Cuckoo search with L'evy flights, then proposed the algorithm. Then, we will compare the results of proposed algorithm with the results of backtracking algorithm and we see, our proposed algorithm is better than backtracking algorithm. The main objective of our work is to reduce the time complexity of n-queen puzzle. We choose to develop Cuckoo Search application in Java programming language. We used JDK version 7 and NetBeans IDE version 6.8.

1.2 Related works

In the past years, so many papers have been published in the field of combinatorial optimization. We can use Meta-heuristic algorithms [3] for combinatorial optimization such as n-queen puzzle. X.S. Yang and S. Deb [1, 4] developed an algorithm named **Cuckoo Search algorithm** in 2009 based on Meta-heuristic algorithm. Isra N. Alkallak [6] developed A Hybrid Algorithm from Cuckoo Search Method with N-Queens Problem in 2012. X.S. Yang and S. Deb developed Engineering Optimization [4] by Cuckoo Search in 2010.

In this dissertation, a CS algorithm with improved parameter is used to generate initial population and their fitness is calculated by using objective function. At the end, we find the solution of n-Queens puzzle in less time as compared to backtracking algorithm.

1.3 History

We use n-Queens puzzle to illustrate cuckoo search algorithm. It is an NP-hard problem. The n-queen puzzle is the problem of keeping n queens on chessboard (n column, n row) so that keep at most 1 queen in each column, row and diagonal.

Max Bezzel proposed n-queen puzzle as eight queen puzzle in 1848. Many mathematicians have worked on this puzzle and generalize it's as n-queens problem. Franz Nauck solves the eight queen puzzle in 1850 and generalize puzzle into n-queens puzzle. S. Günther developed a way of finding solutions by using determinants in 1874, and further J.W.L. Glaisher improve this method.

In 1972, Edsger Dijkstra used this to describe the structured programming. He published a complete description of a depth-first backtracking algorithm.

To solve the eight queen puzzle, we use various techniques such as genetic algorithm, constraint programming and logic programming etc. We solve this problem recursively using backtracking algorithm but this does not give the good solution i.e. time complexity is high as compared to other algorithms.

In general method, to find the solutions of 8-queen puzzle, we have to generate the permutations of the numbers 1 to 8 and use the elements of each permutation as indices to place a queen on each row and rejects those position with diagonal attack. The backtracking algorithm with few improvements on the permutation method

a) Construct the search tree by considering one row of the board at a time.
b) Eliminating most non-solution board positions at a very early stage in their construction.
c) It rejects diagonal attacks even on incomplete chessboard it examines only 15,720 possible queen placements.

We can solve this problem using Constraint programming very effectively.

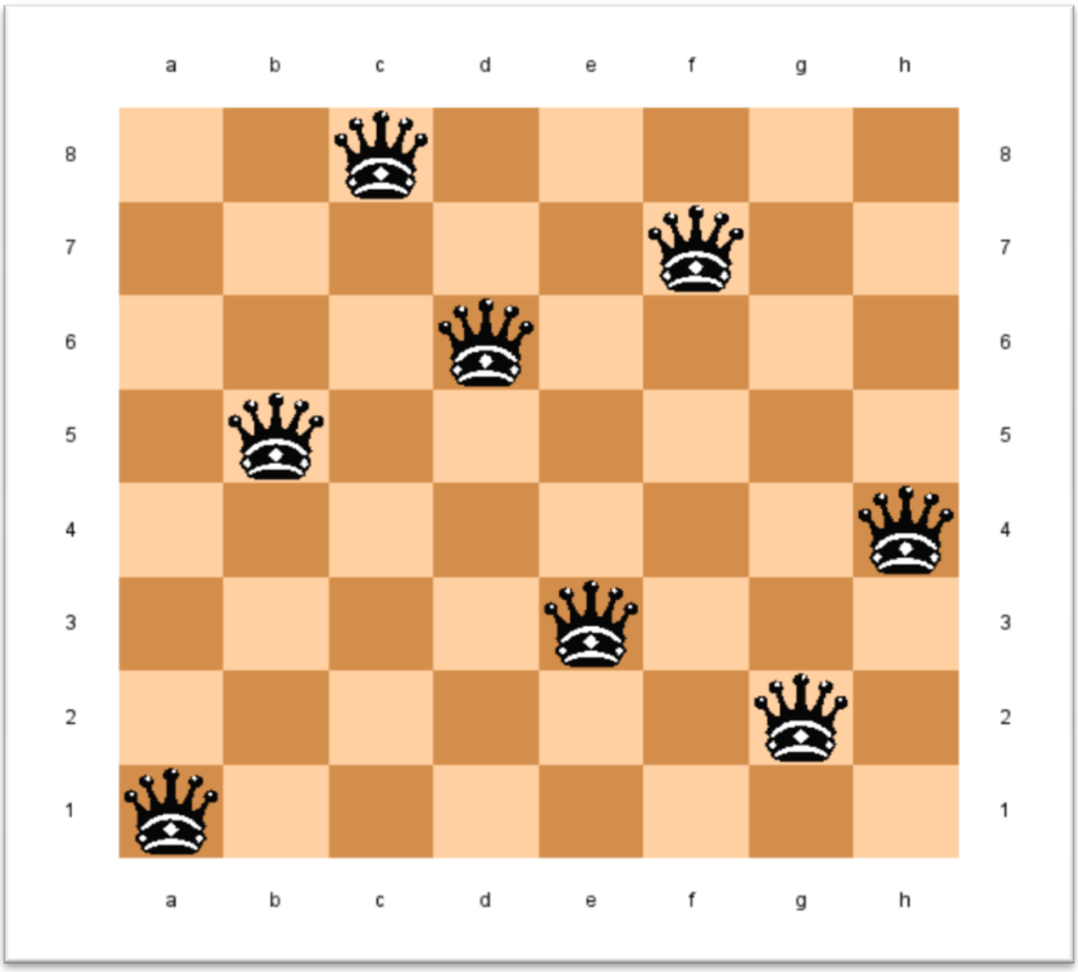


Figure 1.1 8-Queens Puzzle

Another way to solve this puzzle is an iterative repair algorithm. In this algorithm, we start with all queens on the board i.e. with one queen per column. Then counts the number of attacks and uses a heuristic h to determine how to correct the placement of the queens.

But iterative repair algorithm and backtracking algorithm described above, does not guarantee a solution.

1.4 Problem Statement

N-Queens puzzle is an NP-hard problem. The n-queen puzzle is the problem of keeping n queens on chessboard (n column, n row) so that keep at most 1 queen in each column, row and diagonal.

First of all, we have to model the n-Queen puzzle as a CSP problem. A finite CSP (Constraint Satisfaction Problem) consists

1. A set of variables associated with finite domains
2. A set of constraints restricting the values that the variables can simultaneously take.

A solution to a CSP is an assignment of a value from its domain to every variable, in such a way that every constraint is satisfied.

We must find a value for each of the variables that satisfy these constraints. Relation between local collections of variables is called constraint.

Each variable having a finite set of values called the domain of the variable. The Domain of variable Xi is written Di.

A solution to the n-queens puzzle will be any assignment of values to the Variables Q1 ...QN that satisfies all of the constraints. We can represent queen position for ith queen as Q[i] =j, where i represent column value for ith queen and j represent row value for that queen.

- Variables {Q1, Q2,Q3,......Qn } represent the queens,
- Domains $Q_i \in \{1, 2, 3, ...n\}$ $\forall i \in \{1, 2, 3,....n\}$
- Constraints
    1. $Q\,i \neq Q\,j$ $\forall i,j \in \{1, 2,.....n\}$, $i \neq j$ … condition for rows,
    2. $|\,Qi - Q\,j\,| \neq |\,i - j|$ $\forall$ i, j $\in$ {1,2,.....n} ,i $\neq$ j … diagonals,

1.5 Objective

The main objective of the work is to find new techniques to reduce the complexity of NP hard problems and find out optimized solution for those problems. NP-hard problem is a class of problem those are "at least as hard as the hardest problems in NP". NP-hard problems may be

1. Decision problems
2. Search problems
3. Optimization problems.

In our work, first search the position of queen in chessboard using Cuckoo search with L'evy flights, then proposed the algorithm. Then, we will compare the results of proposed algorithm with the results of backtracking algorithm and we see, our proposed algorithm is better than backtracking algorithm.

Optimization refers to the selection of a best solution from set of available solutions. In Optimization problem, we have to minimize or maximize by a systematically choosing input values from a given set and computing the value of the function. We can say optimization means to find the best solution among available solutions using some objective function given a defined domain.

In our proposed work, we use two objective functions and calculate the fitness value of each of the solution using these objective functions.

1.6 Applications

Meta-heuristic algorithm describes computational methods that optimize a problem to improve a new solution with the previous solution iteratively. Meta-heuristic do some or no assumptions about the problem. Meta-heuristic gives good solution for the problem. Meta-heuristic algorithm used in many optimization problems.

Meta-heuristic algorithms inspired by nature such as, Particle Swarm Optimization, Firefly Algorithm etc are increasingly grown. These Meta-heuristic algorithms have been used in many optimization problems, including NP-hard problems such as the n-queens puzzle.

We can use Meta-heuristic algorithms for combinatorial optimization such as the travelling salesman problem. In travelling salesman problem the solutions increases more than exponentially as increasing the problem size.

Meta-heuristic algorithm also used for large size combinatorial problems. Meta-heuristics copy the best feature in nature such as biological systems. There are two important features.

1. Selection of the fittest
2. Adaptation to the environment.

For implementation, these features can be translated into important features of the modern meta-heuristics i.e. intensification and diversification. Intensification means to find the best solutions among the current best solutions and using diversification, the algorithm can search the best solution more space efficiently.

The applications of Cuckoo Search have seen into optimization. The cuckoo search algorithm used to solve

a) The nurse scheduling problem.
b) Data fusion in wireless sensor networks.
c) Knapsack problems
d) Generate independent test paths for structural software testing and test data generation.
e) Travelling Salesman problem
f) N-queens puzzle etc.

If we compare the cuckoo search with DE, ABC and PSO, then we have seen that CS and DE algorithms provide more robust results than ABC and PSO. A detailed study of various structural optimization problems suggests that cuckoo search gives better results than other algorithms. Cuckoo search is more suitable for large problems. To train neural networks, we can also use the Cuckoo search algorithm. Cuckoo search is also use for embedded system design and design optimization.

i. Nurse Scheduling Problem:

In Nurse scheduling problem (NSP), we have to determine a work schedule for nurses that is both reasonable and efficient. This is an example of complex problem due to its many constraints and combinations. It is an example of constraint programming in which many difficulties encountered.

Because of its high number of constraints and the many solutions, this problem can be solved by using a heuristic properly such as genetic algorithms or local search. This scheduling problem appears to be NP-hard. The Cuckoo Search algorithm has been used to solve this problem efficiently.

ii. Traveling Salesman Problem:

In travelling salesman problem (TSP), we have given

a) A list of cities
b) Their pair wise distances,

The goal is to find the shortest possible path that visits each city exactly once and returns to the origin city.

The TSP has several applications such as

a) Planning
b) Logistics
c) The manufacture of microchips.

We can use TSP as a sub-problem in many applications, such as DNA sequencing. In these applications, the term city represents customers, DNA fragments etc and the term distance represents travelling times or cost. In many areas, additional constraints such as time windows or limited resources make the problem considerably harder.

iii. Knapsack Problem:

In knapsack problem, we have given a set of items, each having a weight and a value, we have to determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. The problem also used in resource allocation where there are financial constraints.

iv. n-Queens Puzzle:

The n-queen puzzle is the problem of keeping n queens on chessboard (n column, n row) so that keep at most 1 queen in each column, row and diagonal.

Chapter 2

Literature Review

2.1 Cuckoo Breeding Behaviour

Cuckoo Search algorithm is based on the behaviour Cuckoos bird has many characteristics which differentiate them from other birds such as their beautiful sounds and their aggressive reproduction strategy. Ani and Guira cuckoos lay their eggs in communal nests by removing other’s eggs to increase the hatching probability of their own eggs. Many species engage the brood parasitism by laying their eggs in the nests of other host birds. Cuckoos engage brood parasitism. Brood parasitism's are of three types

a) Intraspecific brood parasitism
b) Cooperative breeding
c) Nest takeover.

Some host birds don't behave friendly against intruders and engage indirect conflict with them. In such situation host bird will throw those align eggs away or make a new nest elsewhere i.e. increases their reproductivity. Cuckoos are also specialized in the mimicry in pattern and colour of the eggs of some host birds.

Generally, the host eggs hatch slightly after the cuckoo eggs. Whenever the first cuckoo egg is hatched, the first action it will take is to throw the host eggs outside from the nest; this action increases the food of cuckoo provided by its host bird i.e. when the cuckoo chicks have hatched, they lift any other eggs they find in the nest onto their backs and then throw them overboard. Hatching early means that cuckoo chicks can oust other bird's eggs, so that they get all the food their foster parents bring home.

2.2 L´evy Flights

A random walk having step-lengths that is heavy-tailed is called L'evy flights. In L'evy flight step length have a probability distribution. The steps made in isotropic random directions, when a walk in a space dimensions greater than one; L´evy flight named at the mathematician Paul Pierre L´evy.

Benoît Mandelbrot, who used L'evy flight for specific definition of the step sizes distribution. Benoît Mandelbrot used the term **Cauchy flight,** when step sizes having a Cauchy distribution, and used the term **Rayleigh flight,** when step sizes having a normal distribution

Many animals and insects have some flight behaviour i.e. L´evy flights behaviour. Reynolds and Frye shows that fruit flies or Drosophila melanogaster; explore their landscape using a series of straight flight paths punctuated by a sudden 90^o turn, leading to an L´evy-flight-style intermittent scale free search pattern.

Graphical representation of L´evy flights is given below. Figure 2.1 shows 1000 steps of an L´evy flight in 2 dimensions. The origin of the motion is at [0, 0]. In this figure the angular direction is distributed uniformly and the step size is distributed according to L´evy distribution.

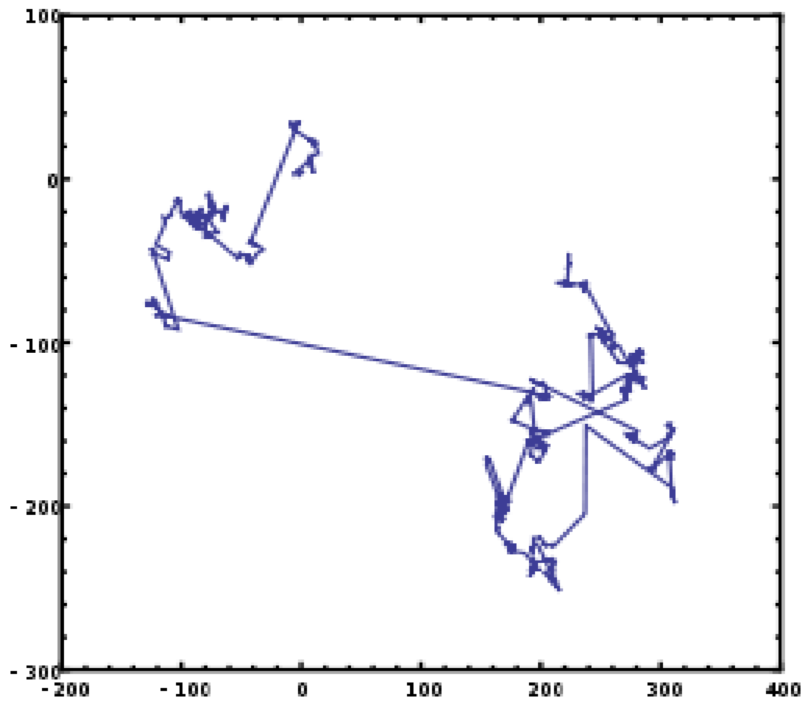


Figure 2.1 Lev'y Flight

2.3 Cuckoo Search algorithm

X.S. Yang and S. Deb developed an algorithm named **Cuckoo Search algorithm** in 2009 based on Meta-heuristic algorithm. This algorithm based on brood parasitism of cuckoo bird. Reproduction method seen in species is called brood parasitism. Some host birds don't behave friendly against intruders and engage indirect conflict with them. In such situation host bird will throw those align eggs away or make a new nest elsewhere i.e. increases their reproductivity. Cuckoos are also specialized in the mimicry in pattern and colour of the eggs of some host birds.

For optimization problems, we can apply cuckoo search algorithm and we seen that it can perform well as compared to other metaheuristic algorithms in different applications.

CS is similar to GA and PSO and it is population based algorithm. Due to step length is large and heavy tailed, the randomization is more efficient. In CS, the number of parameters to be tuned is less as compared to GA and PSO, so it is more suitable for many optimization problems. In CS, each nest can represent a set of solutions.

To describe Cuckoo Search, we use the following three rules

a) The best nests with high quality of eggs will carry over to the next generations.
b) Each cuckoo lays one egg at a time, and dumps its egg in randomly chosen nest.

c) The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $P_a \in [0, 1]$.

So, the host bird can either give up the nest or throw the egg away, and make a new nest. Generally, rule c can be approximated by the fraction P_a of the n nests is replaced by new nests.

In maximization problem, the fitness of a solution can be proportional to the value of the objective function. Generally, we can use the following representations

a) Each egg in a nest represents a solution
b) Cuckoo egg represent a new solution
c) Use the new and good solutions to replace a not so good solution in the nests.

Using above three rules and representation, Cuckoo Search algorithm can be described as shown below.

<u>Pseudo Code:</u>

```
i.     Begin
ii.    Objective function f(x), $x = (x_{1 \ldots} x_d)^T$
iii.   Generate initial population of n host nests $x_i$ (i = 1, 2... n)
iv.    While (t < MaxGeneration) or (stop criterion)
v.     Get a cuckoo randomly by L´evy flights
vi.    Evaluate its quality/fitness $F_i$
vii.   Choose a nest among n (say, j) randomly
viii.  If ($F_i > F_j$),
ix.    Replace j by the new solution;
x.     End
xi.    Fractions (pa) of worse nests are abandoned and new ones are built;
xii.   Keep the best solutions (or nests with quality solutions);
xiii.  Rank the solutions and find the current best
xiv.   End while
xv.    Postprocess results and visualization
xvi.   End
```

2.4 Backtracking algorithm

There are several approaches derived to solve n-queen problem such as recursion, depth first search, backtracking etc. But from all of them we are using backtracking to make comparison with cuckoo search, because backtracking if more efficient from all of them.

Backtracking is a method to find some or all solutions for some computational problem that makes candidates to the solutions incrementally, and give up each partial candidate *c* whenever it determines that *c* cannot reach to a valid solution.

To solve constraint satisfaction problems, Backtracking is a good choice, such as Sudoku, crosswords and other problems. Backtracking is most appropriate method for knapsack problem, parsing. Backtracking is used for combinatorial optimization problems. Backtracking is also the basis of logic programming languages.

Pseudo code:

```
(i)   RECURSIVENQUEENS ($Q$ [1... $n$], $r$)
(ii)  If $r = n+1$
(iii) Print $Q$
(iv)  Else
(v)   For $j \leftarrow 1$ to $n$
(vi)  Legal TRUE
```

```
(vii) For $i \leftarrow 1$ to $r - 1$
(viii)        If ($Q[i] = j$) or ($Q[i] = r - j + i$) or ($Q[i] = r + i - j$)
(ix) Legal FALSE
(x)  If legal
(xi) $Q[r] \leftarrow j$
(xii) RECURSIVENQUEENS ($Q$ [1 ... $n$], $r$ +1)
```

We can understand the execution of the backtracking algorithm with the help of recursion tree. In recursion tree, the root belongs to the original invocation of the algorithm and the edges in the recursion tree belong to the recursive calls. If we move from root to any node down in the tree i.e. path shows the partial solution to the *n*-Queens problem. If there is already a queen on every row then the leaves correspond to partial solutions cannot be extended.

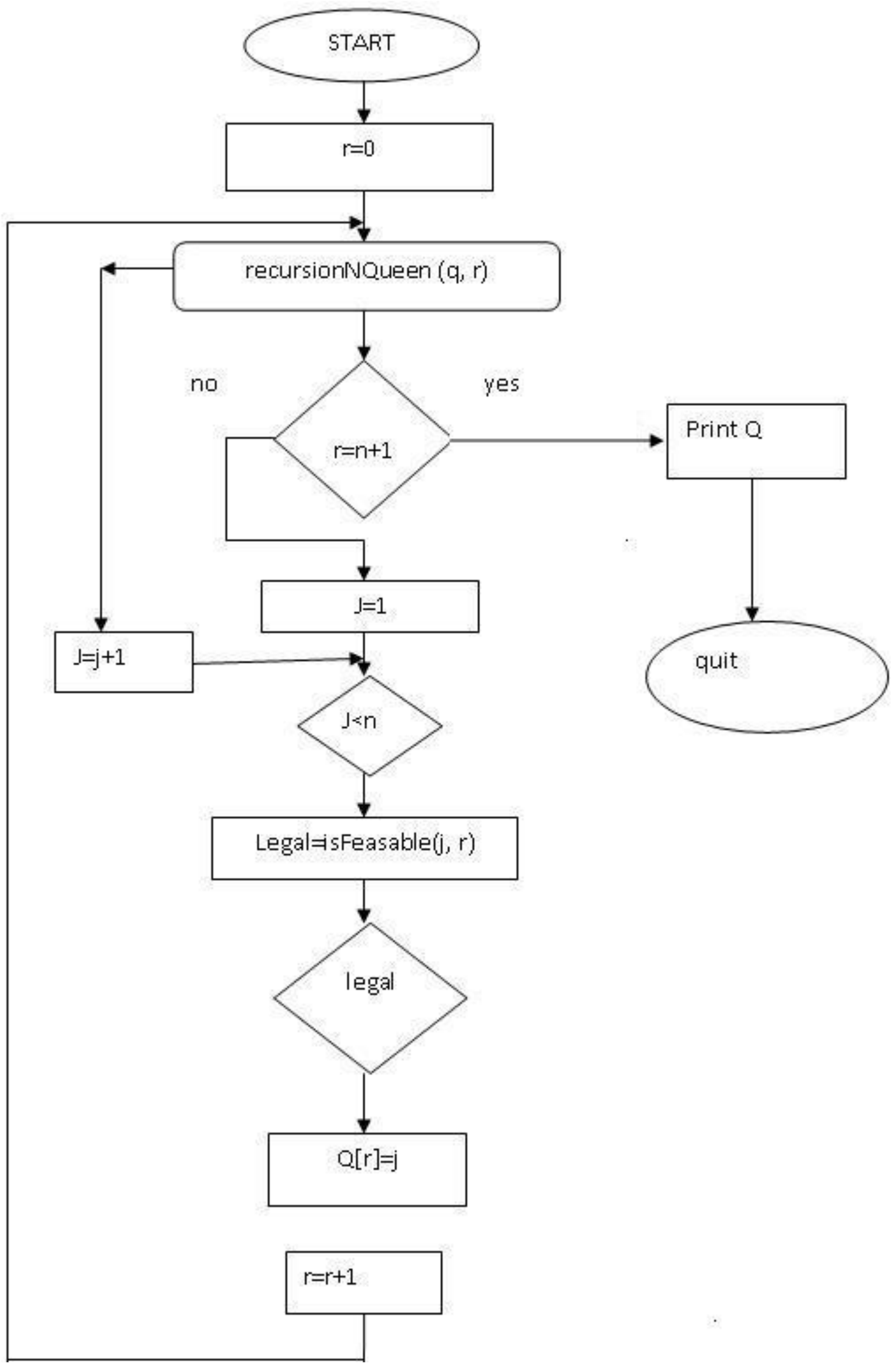


Figure 2.2 Flow Chart of Backtracking

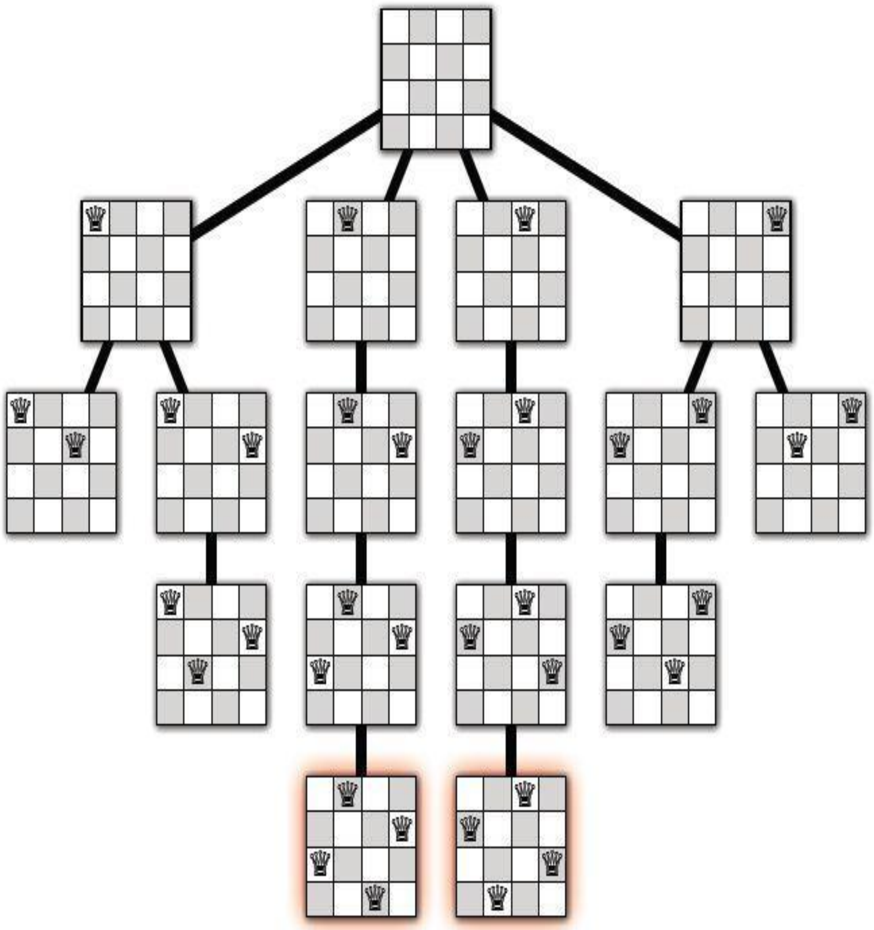


Figure 2.3 Recursion Tree for the 4-Queeens Puzzle

Chapter 3

Proposed Work

3.1 Proposed Algorithm

Heuristic or meta-heuristic search process is subjected to generating new solutions after each step. Initial population generated in initialization step is considered as parent node for next generation. In heuristic search techniques generation of new solutions is most considerable step, because most of the time and computation power spends in generating new solutions. If we are able to find out efficient way to generate new solutions, we can reduce computation time and computation power magically.

In implementation of n-queens problem with cuckoo search, we generate new solutions for current best parent node by using following steps;

1. Firstly, for each queen select one random value of row by performing L'evy flight.
2. Calculate the fitness value of each of the solutions using equation 1 and 2.
3. If new value of row for that queen is better than previous best than select it for new generation.
4. Repeat the process until goal node is found or no new generation left

Above steps are followed for each iteration of cuckoo search process. After iteration, we get the result better than previous result. This result is considered as parent for next generation.

Implementation of Cuckoo Search in n-queens puzzle involves some assumption:

1. Positions of chess board represent new solution.
2. Each queen is considered as egg of Cuckoos.
3. Blocks in chess board represent nest of host bird.

Functions used in Cuckoo Search are:

3.1.1 Initial Population:

In initial population generation n queens are placed on chess board randomly. To enhance the efficiency of the search, the initial population consists of n queen numbered from 1 to n. As we know that only one queen can be placed in one column hence Q[i] queen is placed in i^{th} column which is fixed in whole search process for all i.

Now our task is to find row for each queen. We can represent queen position for ith queen as Q[i] =j, where i represent column value for ith queen and j represent row value for that queen.

In cuckoo search we assume each queen as eggs of cuckoo and chess board position is considered as nest of other birds to put the eggs of cuckoo (queen). Also assume that only one egg can be put in one nest.

Initially put the eggs randomly in nests than find the fitness of the nest to put the eggs in best nest.

3.1.2 Objective Function:

Objective function is also known as heuristic function. In optimization problem in heuristic search or meta-heuristic search our goal is to minimize or maximize the value of objective function. This function is used to find out the fitness of candidate solution. There are different heuristic functions for different problems.

In n-queens problem objective function is number of attacks in all over chess board. And our task is two find out the position of chess board in which number of attacks will be exactly zero. Attacks are calculated by calculating number of queens in same row, column or diagonal. Because columns are fixed for each queen i.e. no queen will be in same column, hence we should test only for rows and diagonal. Condition for rows and diagonal attacks are given below:

a) $Q\,i \neq Q\,j \quad \forall i,j \in \{1, 2, ...n\},\ i \neq j$ ....(1) condition for rows,

b) $|\,Qi - Q\,j\,| \neq |\,i - j| \quad \forall\ i, j \in \{1,2,.....n\}\ , i \neq j$ ....(2) condition for diagonals,

Heuristic h = number of 'attacks'

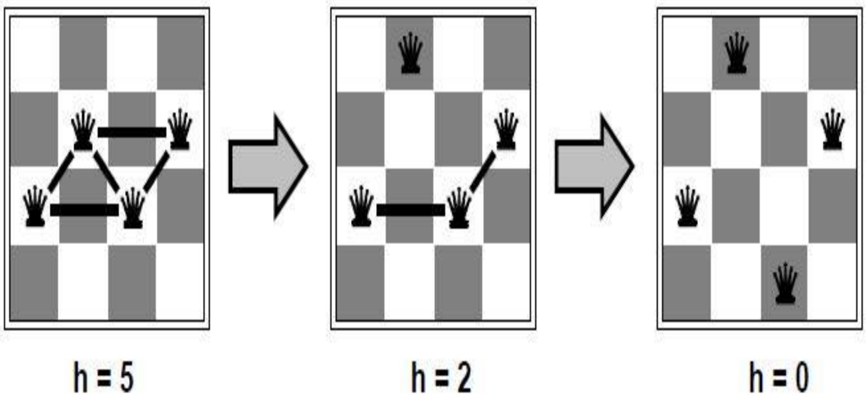


Figure 3.1 Heuristic h

3.2 Methodology used in the work

Both the algorithms are implemented in java version 7 and NetBeans IDE 6.8. We have developed our software system for the CS algorithm. Firstly, in order to make algorithm execute faster, we developed our application with object-oriented concept. With object-oriented concept, software scalability and maintenance is much easier. It provides all the information needed to replace the object without affecting the other code. So, if we want to implement new logic for different optimization problems, it will take substantially less time. On the other side, with object-oriented approach, we can identify the source of errors easily.

We choose to develop CS application in JAVA programming languages because of its many advantages over C, C++ and other modern programming languages. We used newest JDK (Java Development Kit) version 7 and NetBeans IDE (Integrated Development Environment) version 6.8.

3.2.1 Java:

Java is an example of object-oriented programming language. Java has in-built API (Application Programming Interface i.e. it can handle user and graphics interfaces so that it is used to create different applications or applets. Java has a set of API's. Java is a platform independent. Java also has a many standard libraries that can be used for doing mathematics.

The syntax of Java is much similar as C++ and C. Java does not have pointers, this is the major difference. By using Java, we must write object oriented code. If we have familiar with the syntax of C and C++ then it is useful to understand the syntax of Java.

In Java we can easily distinguish between different applications i.e.

a) The programs that perform the same functions as those written in other programming languages.

b) The programs that can be embedded in a Web page and accessed over the Internet.

When a java program is compiled, a byte code is generated and it can be read and executed by any platform i.e. those platform can run Java. Our main focus is how to write java applications.

Sun's JDK

The JDK (Java Development Kit) includes a standard set of classes and these classes provide the basic functions of Java. The JDK includes several packages. The groups of classes called package that share common definitions such as the language package (java.lang). This package contains the lowest level of required classes.

Packages also share a name space, the scope of internal definitions within the package. Different packages can contain objects of the same name; the fully qualified object name includes the package name for uniqueness. A package can be imported into a class for simplify coding, which removes the requirement to fully name packaged objects.

Some included packages examples are:

➢ Java.lang	Object, Math, Thread, String, System
➢ Javax.swing	Font, Button, Window, Menu, Border
➢ Java.applet	AudioClip, Applet
➢ Java.io File	InputStream, StreamTokenizer, File, OutputStream

- Java.net URL, Socket, ContentHandler

Java naming conventions

In Java, All keywords and names are case sensitive.

FILES

In java, files are saving as .java extension and the file name must be as the class name within the file i.e. if we create a class name Circle in a source code then the file is saving as Circle.java. If we compile the java file then the compiled classes are stored in a file i.e. Circle. class.

CLASSES

We must class named be nouns. In class name, the first letter of each word in should be capitalized.

Example: Circle.

METHODS

Method name should be typically a verb. In method name, the first letter should be lowercase. In subsequent word, the first letter should be capitalized.

Example: getCircletName ().

VARIABLES

A variable name in java should be short but descriptive and avoiding the common variable names like k and l etc.

Example: empTaxNumber.

CONSTANTS

A constant in java should be in all upper case names and to separate internal words use underscores.

Example: CIR_BVN_TRE.

Objects

In Object Oriented programming, objects are very important i.e. the main focus on objects. The objects are instance of class. An object is a collection of data and functions called methods to change the data. Same kinds of objects are in the same class. A class is a group of objects that have same relationship and properties. A class is defined by class keyword. We can create one or more from a class.

Example: define a class circle

```
public class Circle

{

  double x, r, y;

}
```

This class does not have methods, but it can be used to define a circle by its position, and radius.

Constructors

Constructor is a method that automatically create instance of class. The name of constructor is same name as the class. At least one constructor is in a class. We initialize a new object belongs to the class using constructor.

The constructor Circle creates an object of the Circle class by specifying two parameters:

a) The initial points
b) Radius of a circle.

We simply say that circle is the constructor for the circle class.

We can use the following java properties given below:

Variable Declaration:

All variables types must be declared. The primitive types are

byte	8,
short	16
int	32
long	64 bit integer variables respectively
float	32
double	64-bit floating point variables
Boolean	true or false.

- From any method, instance variables and methods can be accessed in the class. The x in the argument list of the above constructor refers to the local value of the parameter which is set when circle is called. T**his** keyword is used to refer to those variables that are defined for the entire class in contrast to those defined locally within a method and those that are arguments to a method.

  For example- this.y refers to the variable y which is defined just after the first line of the class definition. We also use multiple constructors.

- A new programmer defined type is class. Each class defines methods and data to change the data. When objects are created from that class, Fields in the class are template for the instance variables that are created i.e. when an object is initialized from the class, a new set of instance variables is created each time.
- Using the **dot operator,** the members of a class are accessed by referring to an object created from that class. For example, suppose that a class Circle contains the instance variable X and a method area. If an object of this class is named c, then the instance variable in c would be accessed as c.X and the method accessed as c.area.
- For terminate individual statements, a semicolon is used.

Comments

Three comment styles use in Java.

- **//** is used for a single line comment in starting in line and it can be used anywhere in the program.
- Begin with **/*** is used for multiple line comments and end with ***/**; it is also used for commenting out a portion of the text on a line.
- Text enclosed within **/** ... */** serves to generate documentation using the javadoc command.

Java Class Definition

Classes are the encapsulated definition of variables and methods to operate on those properties. For creating actual examples of the class we used the class definition as a model. A Class definition includes different components.

The behaviour of the class is depending on the attributes of the class and defined as method. We store the attribute values as variables in two ways

- Specific instance of the class.
- Variables shared by all instances of the class.

Following components having in class definition:

- Package name is name of the package where this class belongs.
- To specify how external access to this class is managed access modifier Keyword is used. Access modifier Keyword may be public or private.
- A mandatory keyword is Class keyword.
- Variables defined outside of a method and available to all methods in the class is called Instance variables.
- Class variables are defined with the static keyword, a single instance of the variable is created which is shared by all instances of the class. Instance variables are created when the class is loaded initially and can be set and accessed even before new instances of the class are created.
- Local Variables defined inside a method. The variable scope is inside the method where it is declared.
- Instance methods are functions that operate on instances of the class.
- Class methods are functions that operate on class variables of the class.

- Constructors are methods that have the same name as the class and are automatically called to create new instances of the class.

METHODS

We defined each method within a class and method is equivalent to a subroutine, a function, procedure in other programming languages.
The following terms include in the definition of method:

- Access modifier Keyword is used to specify how external access to this method is managed. Access modifier Keyword may be

(i) public:	accessible from other classes
(ii) private:	not accessible outside this class
(iii) protected:	accessible from sub-classes of this class
(iv) Default:	accessible from other classes in the same package

- A mandatory keyword for class methods is Static keyword.
- The data type returned by this method is Return type. If the method does not return a value, the data type void must be used.
- The name of the method is Method name.
- Arguments are a comma separated list of data types and names passed as parameters to this method.
- Method body is Java statements that provide the functionality of this method.

If a method does not return a value, then it specifies as void otherwise the method must end with a 'return' statement to provide the value to the calling method. A method may have several exit points.

Main method

Executable applications must have a "main" method defined. The entry point to the application is main method. Other classes do not need a main method but may have one defined for use as a testing entry point.

The main method always written as:

```
public static void main (String [] args)
 {...}
```

PACKAGES

Packages in Java are used to both group related classes and it is to ensure that namespace conflicts should be minimised.

Each class resides in a package. If the class is a member of the default package that means the class does not include a package specifier. Each Java class has the fully qualified class name consists of the package name and the class name and these concatenated with dot notation.

Example:

Java.lang.Object class is the fully qualified name of the Object class. The java.lang package is included in every java class by default.

For package naming, the general accepted convention is to use the author's internet domain name as the initial components of the package name and we write the initial portion of the package name as uppercased.

Example:

By Microsoft packages are created for using with their domain name as microsoft.com; also have a package name as COM.microsoft. If Microsoft were to create a class called Employee for their salary subsystem, the fully qualified class name would be com.Microsoft.salary.Employee. Sun reserved some package names such as

- java
- javax
- sun

CLASS FILES AT RUNTIME

For both UNIX and Windows systems, the root directory for the class hierarchy must also be identified in the environment variable CLASSPATH. Using Java, this environment variable can contain a list of directories to be searched at runtime for the initial directory containing the class hierarchy. Compiled Java classes must appear in a directory tree structure that corresponds to their package name at runtime. The compiler can be instructed to create the directory hierarchy by using the command line option –d as given below

```
SET CLASSPATH=C:\Course
javac –d . Source.java
```

If Source.java contains the following code:

```
package circes;
```

```
public class Circle {
...
```

the compiled Source.class file must appear in the file C:\Course\circes\Circle.class.

Compiled class files **must** be placed in a directory that matches their package name.

Microsoft’s licence system, Employee class must be stored for a Windows operating system, in

“ ..\com\microsoft\licence\Employee.class” or, for a UNIX system, the directory

“ ../com/microsoft/licence/Employee.class”

To execute this class, use this command:

```
java circes.Circle
```

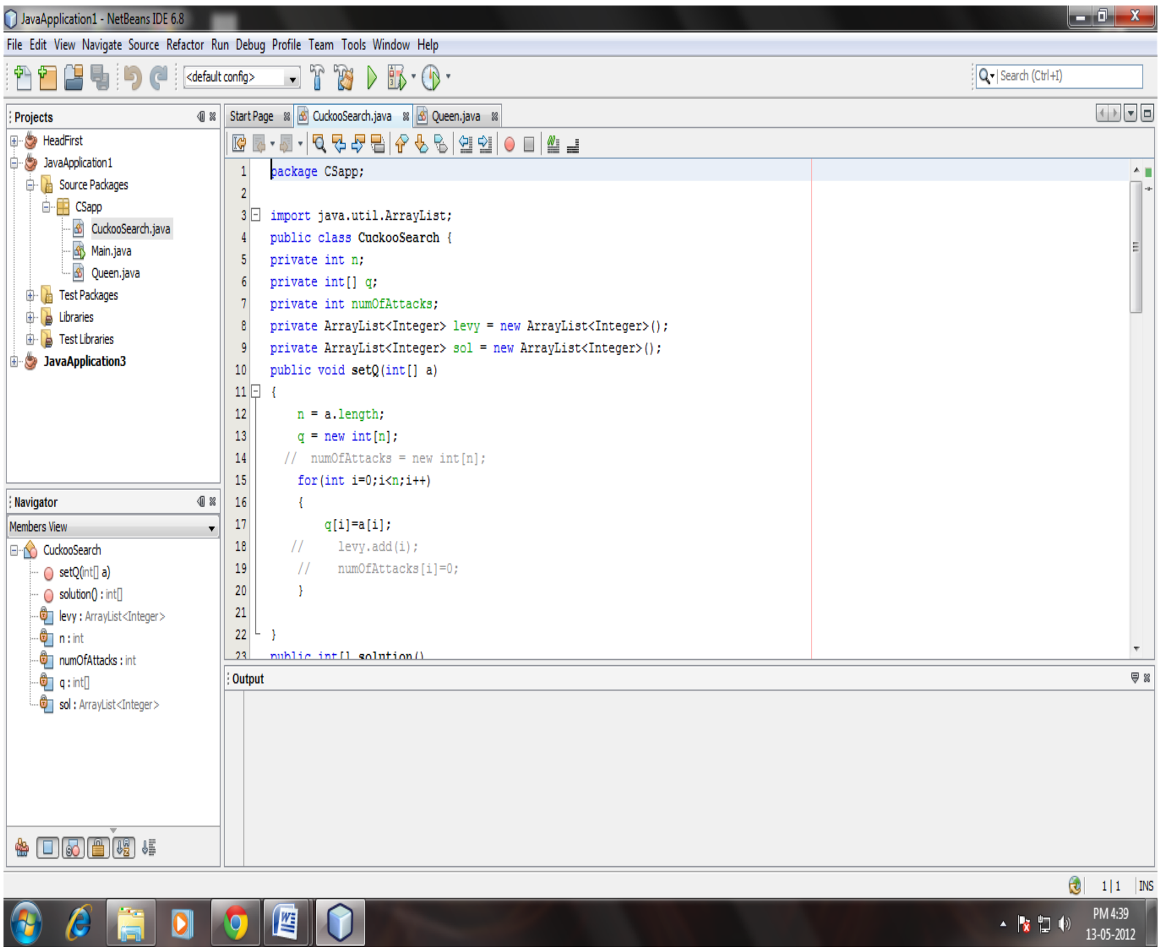
Note that since CLASSPATH has been set already, this command can be invoked from any directory on your system.

3.2.2 NetBeans:

NetBeans IDE is the original free Java IDE (Integrated Development Environment). NetBeans IDE provides support for several languages such as Java, C/C++, PHP, JavaScript etc.) and frameworks.

NetBeans provides an integrated development environment to develop application with PHP, Java, JavaScript, Python and others.

The NetBeans IDE is written in Java and can run on Windows, Mac OS, Linux, Solaris and other platforms supporting a compatible JVM. A pre-existing JVM or a JDK is not required.



```
package CSapp;

import java.util.ArrayList;
public class CuckooSearch {
private int n;
private int[] q;
private int numOfAttacks;
private ArrayList<Integer> levy = new ArrayList<Integer>();
private ArrayList<Integer> sol = new ArrayList<Integer>();
public void setQ(int[] a)
{
    n = a.length;
    q = new int[n];
  //  numOfAttacks = new int[n];
    for(int i=0;i<n;i++)
    {
        q[i]=a[i];
   //     levy.add(i);
    //    numOfAttacks[i]=0;
    }

}
public int[] solution()
```

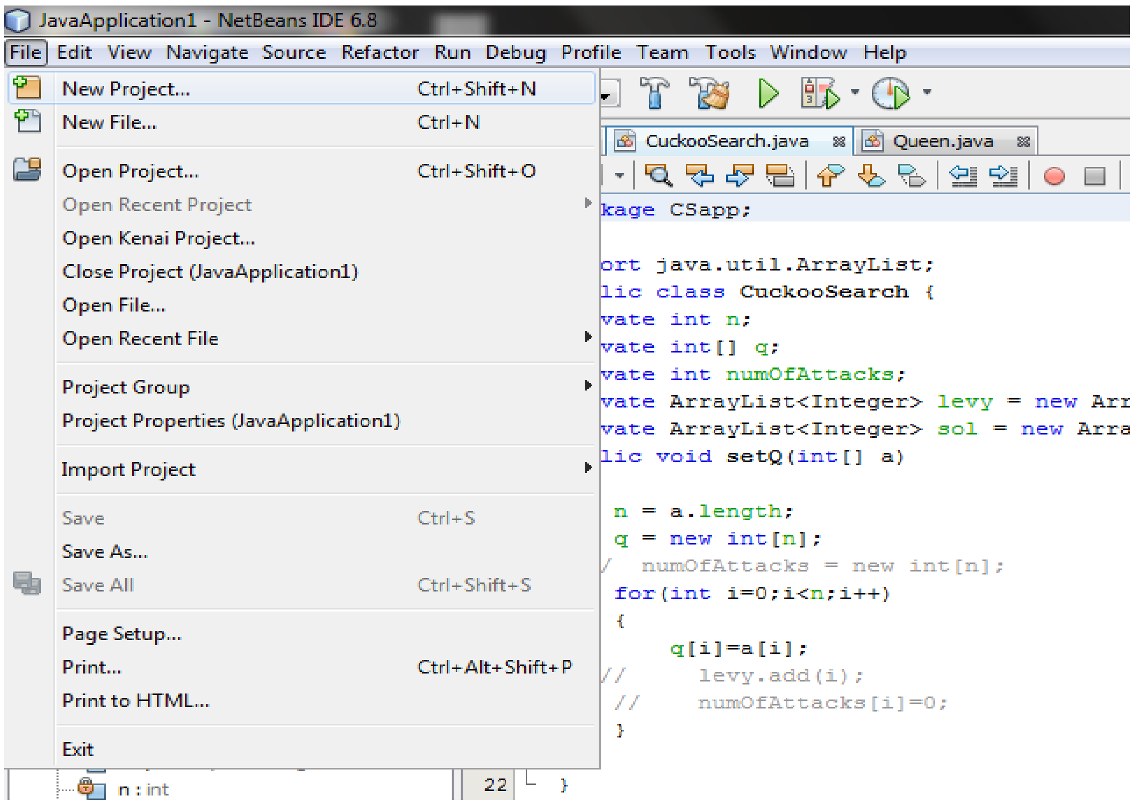
Figure 3.2 Snapshots of NetBeans

The NetBeans platform provides a set of modular software components called modules; by using this we can develop different applications. Applications based on the NetBeans platform including the NetBeans IDE can be extended by third party developers.

NetBeans is an open-source project dedicated to providing rock solid software development products (the NetBeans IDE and the NetBeans Platform) that address the needs of developers, users and the businesses, which rely on NetBeans as a basis for their products. Sun Microsystems made NetBeans as open source in June 2000. Sun Microsystems remains as project sponsor until January 2010, when Sun Microsystems became a subsidiary of Oracle.

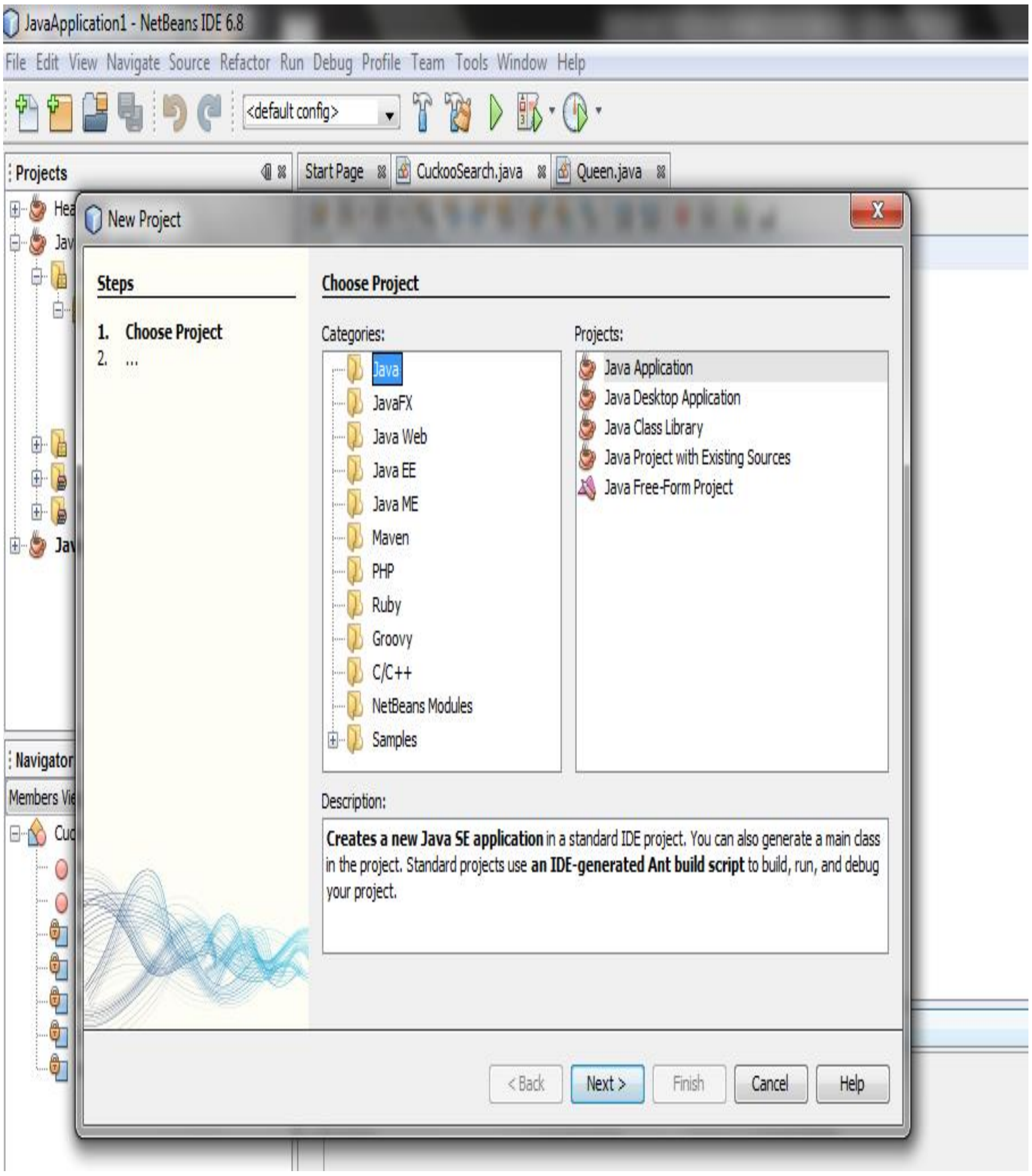
The NetBeans IDE and NetBeans Platform are two base products and these are free for commercial and non-commercial use. The source code for NetBeans IDE and NetBeans Platform is available to anyone to reuse, within the terms of use. The legal section contains information regarding licensing, privacy policy, copyright issues, and terms of use. we can found complete product information in the Products section.

<u>**Working with NetBeans:**</u>


JavaApplication1 - NetBeans IDE 6.8
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
New Project... Ctrl+Shift+N
New File... Ctrl+N
Open Project... Ctrl+Shift+O
Open Recent Project
Open Kenai Project...
Close Project (JavaApplication1)
Open File...
Open Recent File
Project Group
Project Properties (JavaApplication1)
Import Project
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Page Setup...
Print... Ctrl+Alt+Shift+P
Print to HTML...
Exit
CuckooSearch.java
Queen.java
n : int
22

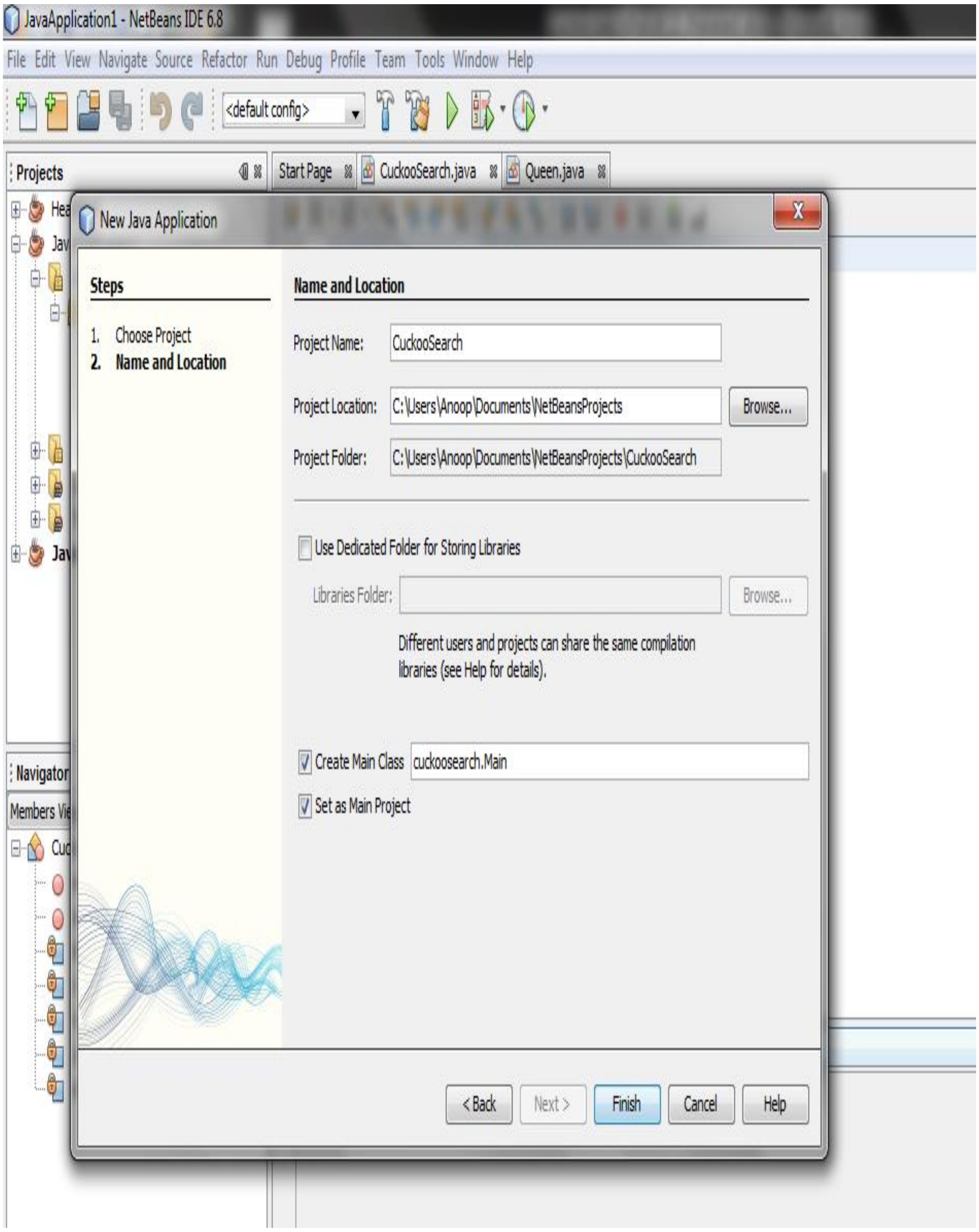
```
kage CSapp;

ort java.util.ArrayList;
lic class CuckooSearch {
vate int n;
vate int[] q;
vate int numOfAttacks;
vate ArrayList<Integer> levy = new Arr
vate ArrayList<Integer> sol = new Arra
lic void setQ(int[] a)

 n = a.length;
 q = new int[n];
/  numOfAttacks = new int[n];
 for(int i=0;i<n;i++)
 {
     q[i]=a[i];
//     levy.add(i);
//     numOfAttacks[i]=0;
 }
 }
```

Figure 3.3 Open new Project in NetBeans



Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

Figure 3.4 Select Categories in NetBeans



Name and Location

Project Name: CuckooSearch

Project Location: C:\Users\Anoop\Documents\NetBeansProjects

Project Folder: C:\Users\Anoop\Documents\NetBeansProjects\CuckooSearch

☐ Use Dedicated Folder for Storing Libraries

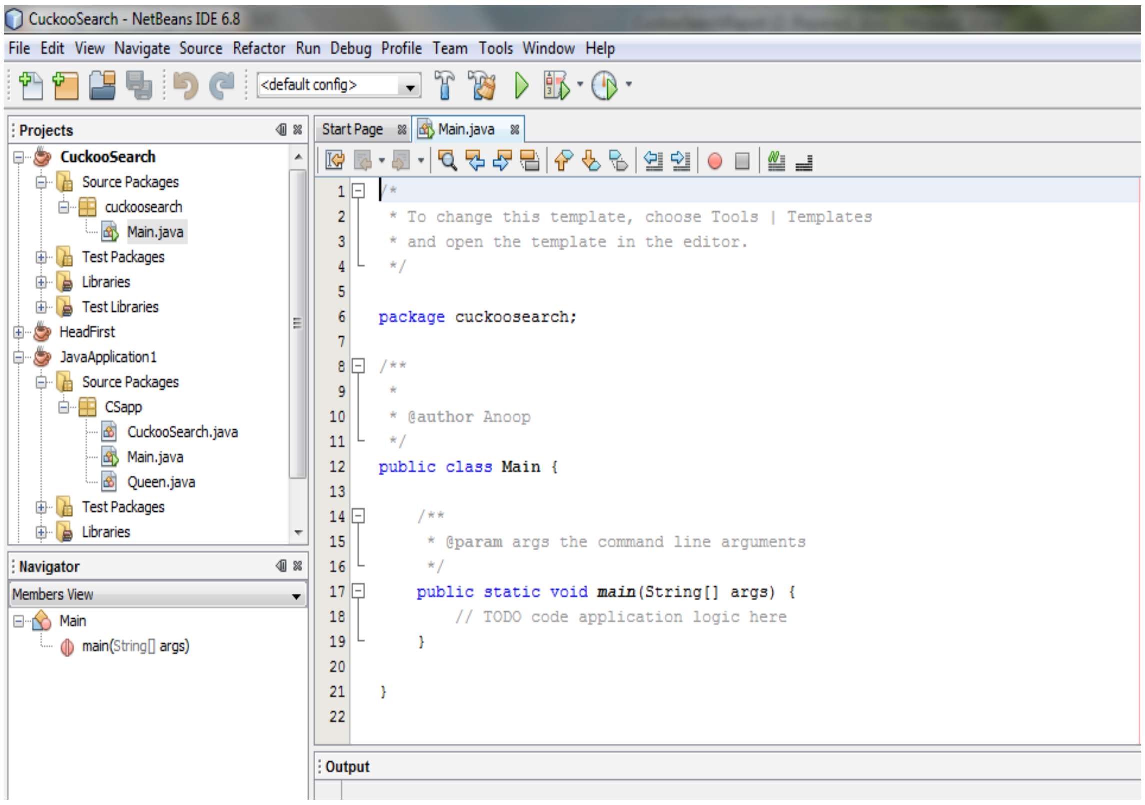
Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☑ Create Main Class cuckoosearch.Main

☑ Set as Main Project

Figure 3.5 Create new Project in NetBeans



```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package cuckoosearch;

/**
 *
 * @author Anoop
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

Figure 3.6 Start writing Java code

3.2.3 CODING CONCEPT

We have used Java and NetBeans for coding our n-queens problem using Meta-heuristic Cuckoo Search Technique. Algorithm starts with the generation of initial population. In initial population generation n queens are placed on chess board randomly. To enhance the efficiency of the search, the initial population consists of n queen numbered from 1 to n. As we know that only one queen can be placed in one column hence Q[i] queen is placed in i^{th} column which is fixed in whole search process for all i.

Now our task is to find row for each queen. We can represent queen position for i^{th} queen as Q[i]=j, where i represent column value for i^{th} queen and j represent row value for that queen.

In cuckoo search we assume each queen as eggs of cuckoo and chess board position is considered as nest of other birds to put the eggs of cuckoo (queen). Also assume that only one egg can be put in one nest.

Initially put the eggs randomly in nests than find the fitness of the nest to put the eggs in best nest.

Now our task is to check the fitness of initial population by using objective function, (which is no. of hits in whole chess board for this problem). Now new solution is generated. Heuristic or meta-heuristic search process is subjected to generating new solutions after each step. Initial population generated in initialization step is considered as parent node for next generation. In heuristic search techniques generation of new solutions is most considerable step, because most of the time and computation power is spent in generating new solutions. If we are able to find out efficient way to generate new solutions, we can reduce computation time and computation power magically.

In implementation of n-queens problem with cuckoo search, we generate new solutions for current best parent node by using following steps;

- Firstly, for each queen select one random value of row by performing L’evy flight.
- If new value of row for that queen is better than previous best than select it for new generation.
- Repeat the process until goal node is found or no new generation left

These steps are followed each iteration of cuckoo search processes. After iteration we get the result better than previous result. This result is considered as parent for next generation. For finding the solution for n-queen problem the no. of hits should be made zero which is achieved by performing iterations consecutively.

Chapter 4

Results & Discussion

4.1 COMPUTATIONAL RESULTS:

Some test results of n-queen problem using cuckoo search algorithm given below with computation time. After those results we will compare results of backtracking and cuckoo search.

Sample Outputs using cuckoo search:

(i) enter the size of chess board **50**

```
   Q0 = {0,10}   Q1 = {1,12}   Q2 = {2,14}   Q3 = {3,16}   Q4 = {4,18}   Q5 = {5,20}
Q6 = {6,22}   Q7 = {7,24}   Q8 = {8,26}   Q9 = {9,28}   Q10 = {10,30}   Q11 = {11,32}
Q12 = {12,34}   Q13 = {13,36}   Q14 = {14,38}   Q15 = {15,40}   Q16 = {16,42}   Q17 =
{17,44}   Q18 = {18,46}   Q19 = {19,48}   Q20 = {20,1}   Q21 = {21,3}   Q22 = {22,5}
Q23 = {23,7}   Q24 = {24,9}   Q25 = {25,11}   Q26 = {26,13}   Q27 = {27,15}   Q28 =
{28,17}   Q29 = {29,19}   Q30 = {30,21}   Q31 = {31,23}   Q32 = {32,25}   Q33 = {33,27}
Q34 = {34,29}   Q35 = {35,31}   Q36 = {36,33}   Q37 = {37,35}   Q38 = {38,37}   Q39 =
{39,39}   Q40 = {40,41}   Q41 = {41,43}   Q42 = {42,45}   Q43 = {43,47}   Q44 = {44,49}
Q45 = {45,1}   Q46 = {46,3}   Q47 = {47,5}   Q48 = {48,7}   Q49 = {49,9}
   Num of attacks = 10
   Q0 = {0,10}   Q1 = {1,12}   Q2 = {2,14}   Q3 = {3,32}   Q4 = {4,18}   Q5 = {5,43}
Q6 = {6,22}   Q7 = {7,16}   Q8 = {8,26}   Q9 = {9,28}   Q10 = {10,30}   Q11 = {11,13}
Q12 = {12,2}   Q13 = {13,34}   Q14 = {14,36}   Q15 = {15,38}   Q16 = {16,40}   Q17 =
{17,0}   Q18 = {18,46}   Q19 = {19,49}   Q20 = {20,24}   Q21 = {21,6}   Q22 = {22,4}
Q23 = {23,48}   Q24 = {24,8}   Q25 = {25,11}   Q26 = {26,7}   Q27 = {27,15}   Q28 =
{28,17}   Q29 = {29,44}   Q30 = {30,21}   Q31 = {31,23}   Q32 = {32,25}   Q33 = {33,27}
Q34 = {34,29}   Q35 = {35,31}   Q36 = {36,33}   Q37 = {37,1}   Q38 = {38,37}   Q39 =
{39,39}   Q40 = {40,41}   Q41 = {41,47}   Q42 = {42,45}   Q43 = {43,19}   Q44 = {44,42}
Q45 = {45,20}   Q46 = {46,3}   Q47 = {47,5}   Q48 = {48,35}   Q49 = {49,9}
   Num of attacks = 0
   iteration 12
   BUILD SUCCESSFUL (total time: 1 second)
```

(ii)

```
enter the size of chess board 100
 Q0 = {0,4}  Q1 = {1,6}  Q2 = {2,8}  Q3 = {3,10}  Q4 = {4,12}  Q5 =
{5,14}  Q6 = {6,16}  Q7 = {7,18}  Q8 = {8,20}  Q9 = {9,22}  Q10 = {10,24}
Q11 = {11,26}  Q12 = {12,28}  Q13 = {13,30}  Q14 = {14,32}  Q15 =
{15,34}  Q16 = {16,36}  Q17 = {17,38}  Q18 = {18,40}  Q19 = {19,42}  Q20
= {20,44}  Q21 = {21,46}  Q22 = {22,48}  Q23 = {23,50}  Q24 = {24,52}
Q25 = {25,54}  Q26 = {26,56}  Q27 = {27,58}  Q28 = {28,60}  Q29 =
{29,62}  Q30 = {30,64}  Q31 = {31,66}  Q32 = {32,68}  Q33 = {33,70}  Q34
= {34,72}  Q35 = {35,74}  Q36 = {36,76}  Q37 = {37,78}  Q38 = {38,80}
Q39 = {39,82}  Q40 = {40,84}  Q41 = {41,86}  Q42 = {42,88}  Q43 =
{43,90}  Q44 = {44,92}  Q45 = {45,94}  Q46 = {46,96}  Q47 = {47,98}  Q48
= {48,1}  Q49 = {49,3}  Q50 = {50,5}  Q51 = {51,7}  Q52 = {52,9}  Q53 =
{53,11}  Q54 = {54,13}  Q55 = {55,15}  Q56 = {56,17}  Q57 = {57,19}  Q58
= {58,21}  Q59 = {59,23}  Q60 = {60,25}  Q61 = {61,27}  Q62 = {62,29}
Q63 = {63,31}  Q64 = {64,33}  Q65 = {65,35}  Q66 = {66,37}  Q67 =
{67,39}  Q68 = {68,41}  Q69 = {69,43}  Q70 = {70,45}  Q71 = {71,47}  Q72
= {72,49}  Q73 = {73,51}  Q74 = {74,53}  Q75 = {75,55}  Q76 = {76,57}
Q77 = {77,59}  Q78 = {78,61}  Q79 = {79,63}  Q80 = {80,65}  Q81 =
{81,67}  Q82 = {82,69}  Q83 = {83,71}  Q84 = {84,73}  Q85 = {85,75}  Q86
= {86,77}  Q87 = {87,79}  Q88 = {88,81}  Q89 = {89,83}  Q90 = {90,85}
Q91 = {91,87}  Q92 = {92,89}  Q93 = {93,91}  Q94 = {94,93}  Q95 =
{95,95}  Q96 = {96,97}  Q97 = {97,99}  Q98 = {98,1}  Q99 = {99,3}
Num of attacks = 35
 Q0 = {0,4}  Q1 = {1,6}  Q2 = {2,8}  Q3 = {3,10}  Q4 = {4,88}  Q5 =
{5,51}  Q6 = {6,16}  Q7 = {7,18}  Q8 = {8,20}  Q9 = {9,12}  Q10 = {10,24}
Q11 = {11,26}  Q12 = {12,28}  Q13 = {13,30}  Q14 = {14,13}  Q15 =
{15,78}  Q16 = {16,34}  Q17 = {17,36}  Q18 = {18,87}  Q19 = {19,40}  Q20
= {20,42}  Q21 = {21,44}  Q22 = {22,46}  Q23 = {23,48}  Q24 = {24,50}
Q25 = {25,52}  Q26 = {26,54}  Q27 = {27,56}  Q28 = {28,58}  Q29 =
{29,60}  Q30 = {30,90}  Q31 = {31,64}  Q32 = {32,66}  Q33 = {33,68}  Q34
= {34,70}  Q35 = {35,72}  Q36 = {36,74}  Q37 = {37,76}  Q38 = {38,93}
Q39 = {39,80}  Q40 = {40,82}  Q41 = {41,84}  Q42 = {42,86}  Q43 = {43,2}
Q44 = {44,22}  Q45 = {45,92}  Q46 = {46,94}  Q47 = {47,96}  Q48 =
{48,98}  Q49 = {49,62}  Q50 = {50,5}  Q51 = {51,7}  Q52 = {52,9}  Q53 =
{53,11}   Q54 = {54,0}   Q55 = {55,15}   Q56 = {56,17}   Q57 = {57,19}   Q58
= {58,21}   Q59 = {59,23}   Q60 = {60,25}   Q61 = {61,27}   Q62 = {62,29}
Q63 = {63,31}   Q64 = {64,33}   Q65 = {65,35}   Q66 = {66,37}   Q67 =
{67,39}   Q68 = {68,41}   Q69 = {69,43}   Q70 = {70,45}   Q71 = {71,47}   Q72
= {72,49}   Q73 = {73,14}   Q74 = {74,53}   Q75 = {75,55}   Q76 = {76,57}
Q77 = {77,59}   Q78 = {78,61}   Q79 = {79,63}   Q80 = {80,65}   Q81 =
{81,67}   Q82 = {82,69}   Q83 = {83,71}   Q84 = {84,73}   Q85 = {85,75}   Q86
= {86,77}   Q87 = {87,79}   Q88 = {88,81}   Q89 = {89,83}   Q90 = {90,85}
Q91 = {91,38}   Q92 = {92,89}   Q93 = {93,91}   Q94 = {94,32}   Q95 =
{95,95}   Q96 = {96,97}   Q97 = {97,99}   Q98 = {98,1}   Q99 = {99,3}
Num of attacks = 0
iteration 3
BUILD SUCCESSFUL (total time: 4 seconds)
```

(iii) enter the size of chess board **150**

```
Q0 = {0,29}        Q1 = {1,31}    Q2 = {2,33}    Q3 = {3,35}    Q4 = {4,37}
Q5 = {5,39}        Q6 = {6,41}    Q7 = {7,43}    Q8 = {8,45}    Q9 = {9,47}
Q10 = {10,49}      Q11 = {11,51}Q12 = {12,53}Q13 = {13,55}Q14 = {14,57}
Q15 = {15,59}      Q16 = {16,61}Q17 = {17,63}Q18 = {18,65}Q19 = {19,67}
Q20 = {20,69}      Q21 = {21,71}Q22 = {22,73}Q23 = {23,75}Q24 = {24,77}
Q25 = {25,79}      Q26 = {26,81}Q27 = {27,83}Q28 = {28,85}Q29 = {29,87}
Q30 = {30,89}      Q31 = {31,91}Q32 = {32,93}Q33 = {33,95}Q34 = {34,97}
Q35 = {35,99}      Q36 = {36,101}Q37 = {37,103}Q38 = {38,105}
Q39 = {39,107}     Q40 = {40,109}      Q41 = {41,111}      Q42 = {42,113}
Q43 = {43,115}     Q44 = {44,117}      Q45 = {45,119}      Q46 = {46,121}
Q47 = {47,123}     Q48 = {48,125}      Q49 = {49,127}      Q50 = {50,129}
Q51 = {51,131}     Q52 = {52,133}      Q53 = {53,135}      Q54 = {54,137}
Q55 = {55,139}     Q56 = {56,141}      Q57 = {57,143}      Q58 = {58,145}
Q59 = {59,147}     Q60 = {60,149}      Q61 = {61,0}        Q62 = {62,2}
Q63 = {63,4}       Q64 = {64,6}        Q65 = {65,8}        Q66 = {66,10}
Q67 = {67,12}      Q68 = {68,14}       Q69 = {69,16}       Q70 = {70,18}
Q71 = {71,20}      Q72 = {72,22}       Q73 = {73,24}       Q74 = {74,26}
Q75 = {75,28}      Q76 = {76,30}       Q77 = {77,32}       Q78 = {78,34}
Q79 = {79,36}      Q80 = {80,38}       Q81 = {81,40}       Q82 = {82,42}
```

```
Q83 = {83,44}      Q84 = {84,46}      Q85 = {85,48}      Q86 = {86,50}
Q87 = {87,52}      Q88 = {88,54}      Q89 = {89,56}      Q90 = {90,58}
Q91 = {91,60}      Q92 = {92,62}      Q93 = {93,64}      Q94 = {94,66}
Q95 = {95,68}      Q96 = {96,70}      Q97 = {97,72}      Q98 = {98,74}
Q99 = {99,76}      Q100 = {100,78}    Q101 = {101,80}    Q102 = {102,82}
Q103 = {103,84}    Q104 = {104,86}    Q105 = {105,88}    Q106 = {106,90}
Q107 = {107,92}    Q108 = {108,94}    Q109 = {109,96}    Q110 = {110,98}
Q111 = {111,100} Q112 = {112,102}    Q113 = {113,104}    Q114 = {114,106}
Q115 = {115,108} Q116 = {116,110}    Q117 = {117,112}    Q118 = {118,114}
Q119 = {119,116} Q120 = {120,118}    Q121 = {121,120}    Q122 = {122,122}
Q123 = {123,124} Q124 = {124,126}    Q125 = {125,128}    Q126 = {126,130}
Q127 = {127,132} Q128 = {128,134}    Q129 = {129,136}    Q130 = {130,138}
Q131 = {131,140} Q132 = {132,142}    Q133 = {133,144}    Q134 = {134,146}
Q135 = {135,148} Q136 = {136,0}      Q137 = {137,2}      Q138 = {138,4}
Q139 = {139,6}     Q140 = {140,8}      Q141 = {141,10}     Q142 = {142,12}
Q143 = {143,14}    Q144 = {144,16}     Q145 = {145,18}     Q146 = {146,20}
Q147 = {147,22}    Q148 = {148,24}     Q149 = {149,26}
No. of Attacks=28
Q0 = {0,29}          Q1 = {1,31}    Q2 = {2,33}    Q3 = {3,35}    Q4 = {4,37}
Q5 = {5,39}          Q6 = {6,41}    Q7 = {7,43}    Q8 = {8,45}    Q9 = {9,47}
Q10 = {10,49}        Q11 = {11,51}Q12 = {12,53}Q13 = {13,55}Q14 = {14,57}
Q15 = {15,59}        Q16 = {16,61}Q17 = {17,63}Q18 = {18,65}Q19 = {19,67}
Q20 = {20,69}        Q21 = {21,71}Q22 = {22,73}Q23 = {23,113}Q24 = {24,77}
Q25 = {25,79}        Q26 = {26,81}Q27 = {27,83}Q28 = {28,85}Q29 = {29,87}
Q30 = {30,82}        Q31 = {31,91}Q32 = {32,131}Q33 = {33,95}Q34 = {34,97}
Q35 = {35,99}        Q36 = {36,101}Q37 = {37,103}Q38 = {38,105}Q39 = {39,16}
Q40 = {40,109}       Q41 = {41,111}Q42 = {42,140}Q43 = {43,115}
Q44 = {44,117}Q45 = {45,119}    Q46 = {46,121}Q47 = {47,123}
Q48 = {48,22}Q49 = {49,127}Q50 = {50,129}    Q51 = {51,75}
Q52 = {52,133}Q53 = {53,135}Q54 = {54,137}Q55 = {55,139}
Q56 = {56,141}Q57 = {57,143}Q58 = {58,145}Q59 = {59,147}
Q60 = {60,149}Q61 = {61,0}Q62 = {62,142}Q63 = {63,1}Q64 = {64,74}
  Q65 = {65,7}Q66 = {66,9}Q67 = {67,93}Q68 = {68,13}Q69 = {69,15}
  Q70 = {70,17}Q71 = {71,8}Q72 = {72,21}Q73 = {73,3}Q74 = {74,25}
```

```
Q75 = {75,28}Q76 = {76,30}Q77 = {77,32}Q78 = {78,19}Q79 = {79,36}
Q80 = {80,38}Q81 = {81,40}Q82 = {82,42}Q83 = {83,44}Q84 = {84,46}
Q85 = {85,107}Q86 = {86,5}Q87 = {87,27}Q88 = {88,23}Q89 = {89,11}
Q90 = {90,18}Q91 = {91,56}Q92 = {92,58}Q93 = {93,60}Q94 = {94,62}
Q95 = {95,64}Q96 = {96,66}Q97 = {97,68}Q98 = {98,70}Q99 = {99,72}
Q100 = {100,78}Q101 = {101,80}Q102 = {102,125}Q103 = {103,84}
Q104 = {104,86}Q105 = {105,88}Q106 = {106,90}Q107 = {107,92}
Q108 = {108,94}Q109 = {109,96}Q110 = {110,98}Q111 = {111,100}
Q112 = {112,102}Q113 = {113,104}Q114 = {114,106}Q115 = {115,108}
Q116 = {116,110}Q117 = {117,112}Q118 = {118,114}Q119 = {119,116}
Q120 = {120,118}Q121 = {121,120}Q122 = {122,122}Q123 = {123,124}
Q124 = {124,126}Q125 = {125,128}Q126 = {126,130}Q127 = {127,132}
Q128 = {128,134}Q129 = {129,136}Q130 = {130,138}Q131 = {131,148}
Q132 = {132,52}Q133 = {133,144}Q134 = {134,146}Q135 = {135,34}
Q136 = {136,50}Q137 = {137,2}Q138 = {138,4}Q139 = {139,6}
Q140 = {140,76}Q141 = {141,10}Q142 = {142,12}Q143 = {143,14}
Q144 = {144,54}Q145 = {145,89}Q146 = {146,20}Q147 = {147,48}
Q148 = {148,24}Q149 = {149,26}
No. of Attacks=0
iteration 9
BUILD SUCCESSFUL (total time: 17 seconds)
```

Sample Outputs using Backtracking

(i) n = **10**

```
q0={0,0}      q1={1,2}      q2={2,5}      q3={3,7}      q4={4,9}
q5={5,4}      q6={6,8}      q7={7,1}      q8={8,3}      q9={9,6}
BUILD SUCCESSFUL (total time: 1 second)
```

(ii) n = **15**

```
q0={0,0}      q1={1,2}      q2={2,4}      q3={3,1}      q4={4,9}
q5={5,11}     q6={6,13}     q7={7,3}      q8={8,12}     q9={9,8}
q10={10,5}    q11={11,14}   q12={12,6}    q13={13,10}   q14={14,7}
BUILD SUCCESSFUL (total time: 1 second)
```

(iii) n = **25**

```
q0={0,0}      q1={1,2}      q2={2,4}      q3={3,1}      q4={4,3}
q5={5,8}      q6={6,10}     q7={7,12}     q8={8,14}     q9={9,18}
q10={10,20}   q11={11,23}   q12={12,19}   q13={13,24}   q14={14,22}
q15={15,5}    q16={16,7}    q17={17,9}    q18={18,6}    q19={19,13}
q20={20,15}   q21={21,17}   q22={22,11}   q23={23,16}   q24={24,21}
BUILD SUCCESSFUL (total time: 1 second)
```

(iv) n = 28

```
q0={0,0}      q1={1,2}      q2={2,4}      q3={3,1}      q4={4,3}
q5={5,8}      q6={6,10}     q7={7,12}     q8={8,14}     q9={9,16}
q10={10,22}   q11={11,24}   q12={12,21}   q13={13,27}   q14={14,25}
q15={15,23}   q16={16,26}   q17={17,6}    q18={18,11}   q19={19,15}
q20={20,17}   q21={21,7}    q22={22,9}    q23={23,13}   q24={24,19}
q25= {25, 5}  q26= {26, 20} q27= {27, 18}
BUILD SUCCESSFUL (total time: 6 seconds)
```

(v) n = **30**

```
q0={0,0}    q1={1,2}    q2={2,4}    q3={3,1}    q4={4,3}    q5={5,8}    q6={6,10}
q7={7,12}    q8={8,14}    q9={9,6}    q10={10,22}    q11={11,25}    q12={12,27}
q13={13,24}      q14={14,21}      q15={15,23}      q16={16,29}      q17={17,26}
q18={18,28}   q19={19,15}   q20={20,11}   q21={21,9}   q22={22,7}   q23={23,5}
```

```
q24={24,17}       q25={25,19}       q26={26,16}       q27={27,13}       q28={28,20}
q29={29,18}
```

BUILD SUCCESSFUL **(total time: 1 minute 30 seconds)**

4.2 COMPARISON OF RESULTS

In our work we use Cuckoo search algorithm with L'evy flights with some constraints, then, we will compare the results of proposed algorithm with the results of backtracking algorithm and we see our proposed algorithm is better than backtracking algorithm.. Both the algorithms are implemented in identical system under same running condition. A number of experiments have been carried out by giving different inputs and applying Cuckoo Search algorithm and Backtracking algorithm. The table 4.1 below shows the comparison of running time between both the algorithms.

Table 4.1: Comparison of Cuckoo Search algorithm and Backtracking algorithm

S.NO.	Number of Queens (n)	Backtracking	Cuckoo Search
1.	10	1	1
2.	15	1	1
3.	25	1	1
4.	28	6	1
5.	30	90	1
6.	50	-	1
7.	80	-	2
8.	100	-	4
9.	150	-	17
10.	200	-	23

The results obtained using CUCKOO SEARCH algorithm are clearly better than those obtained from BACKTRACKING. In backtracking algorithm we gets the result till n is 29 efficiently and when n is 30 it gives the result in reasonable time but n becomes higher than 30 backtracking algorithm becomes hang due to lot of backtrack. While on the other hand cuckoo search gives result efficiently till 300 (tested till 300 even may more).

Graphical representation of comparison of results shown below; in this representation number of queens on the chess board represents on x-axis and y-axis shows the computational time required to solve the puzzle.

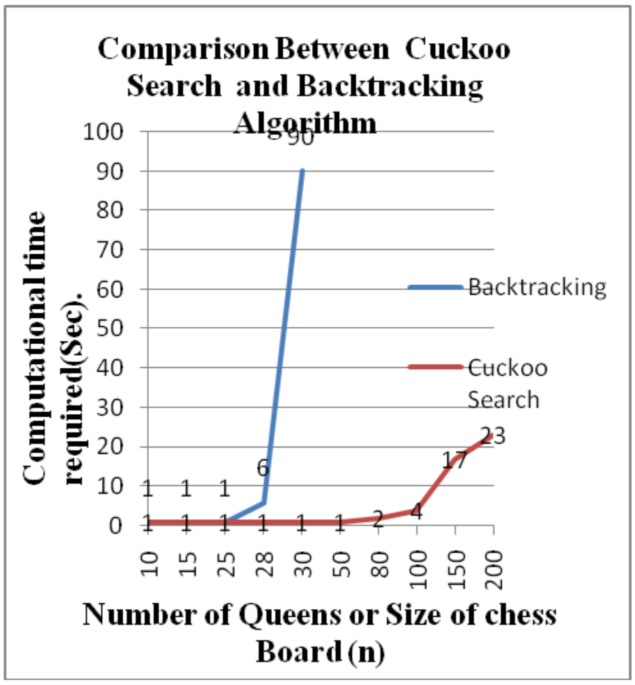


Figure 4.1 The running time Comparison of Cuckoo Search Algorithm and Backtracking algorithm

Figure 4.1 shows the running time comparison of backtracking algorithm and cuckoo search algorithm. This difference is due to in backtracking there are too much steps that have to backtrack in search of required result with respect to the size of n. And when n is small enough that does not be seen by user due to high computation power of hardware, but if n is large it requires more steps in backtracking.

While in cuckoo search all the queens are placed in the chess board one queen in each column. Now attacks are only in rows and diagonals. We minimize the attacks by selecting best row for each queen. It can give result instantly for large value of n.

Chapter 5

Conclusion

CONCLUSION

This dissertation uses the concept of cuckoo search algorithm for the n-queens puzzle. We have developed a new meta-heuristic algorithm, based on the cuckoo species breeding behaviour. In our work we use Cuckoo search algorithm with L'evy flights with some constraints, then, we will compare the results of proposed algorithm with the results of backtracking algorithm and we see our proposed algorithm is better than backtracking algorithm. Comparison with backtracking is just a small example for the project. Even it is proved in researches that cuckoo search gives better result over other heuristic or meta-heuristic search techniques.

If we compare the cuckoo search with DE, ABC and PSO, then we have seen that CS and DE algorithms provide more robust results than ABC and PSO. A detailed study of various structural optimization problems suggests that cuckoo search gives better results than other algorithms. Cuckoo search is more suitable for large problems. To train neural networks, we can also use the Cuckoo search algorithm. Cuckoo search is also use for embedded system design and design optimization.

At last, we can say Meta-heuristic algorithm is a good choice to solve the NP hard problem such as N-queen puzzle efficiently. Using Meta-heuristic algorithm the time complexity is less as compared to other algorithm and run very efficiently.

Chapter 6

REFERENCES

References

[1]. X.S. Yang, and Suash Deb., “Cuckoo search via L´evy flights”, in: Proceeding. of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), December 2009, IEEE Publications, pp. 210-214 (2009).

[**2].** E., Tsang., A Glimpse of Constraint Satisfaction, 1999, Journal Artificial Intelligence review, Kluwer Academic Publishers, vol. 13, pp. 215-217.

[3]. X.S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2008, pp. 128.

[**4].** X.S. Yang., and Suash Deb, Engineering Optimization by Cuckoo Search, Int. J. of Mathematical Modeling and Numerical Optimization, Vol. 1, No. 4, 2010, pp. 330–343.

[5]. P. Civicioglu and E. Besdok, A conception comparison of the cuckoo search, particle swarm optimization, differential evolution and artificial bee colony algorithms, Artificial Intelligence Review, DOI 10.1007/s10462-011-92760, 6 July (2011).

[**6].** Isra N. Alkallak, A Hybrid Algorithm from Cuckoo Search Method with N-Queens Problem, *Raf. J. of Comp. & Math’s. , Vol. 9, No. 2, 2012.*

[7]. Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran, Fundamental of Computer algorithms, Galgotia Publications Pvt. Ltd.,2007.

[8]. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms, Second edition, PHI,2002.

Chapter 7

APPENDICES

7.1List of Publications:

[1] Ram Gopal Sharma, Bright Keswani, **Implementation of n-Queens Puzzle using Meta-heuristic Algorithm (Cuckoo Search),** published in the International Journal of Latest Trends in Engineering and Technology(IJLTET), Vol. 2, Issue 3, May 2013, pp. 343-347.

7.2 Program Code

Java Program for backtracking:

Program: 1

```
package javabacktracking;

public class Main                              // creating main class to run the code//
{
   public static void main(String[] args) {
      NQueen obj=new NQueen();
      int n;
      System.out.println("enter the size of chess board");
      try{
      BufferedReader ip = new BufferedReader(new InputStreamReader(System.in));
      n = Integer.parseInt(ip.readLine());
      if(n<=3)
      {
         System.out.println("invalid input");
         System.exit(0);
      }
      }
      catch(IOException e)
      {
      System.out.println("IOException: " + e);
      }
      obj.nQ(n);
      obj.solution(n);
    }
}
```

Program: 2

```
package javabacktracking;
public class NQueen{
     int q[];
     public void nQ(int n)
     {
         q=new int[n];
        for(int i=0;i<n;i++)
         {
             q[i]=-1;
         }
     }
    public void solution(int n)
    {

        int i=0;
        while(i<n)
        {
            int j = q[i]+1;
            boolean pos = false;
            while(j<n&&pos==false)
            {
                q[i]=j;
                pos=isFeasable(i);
                if(pos==false)
                {
                    j++;
                }
            }
            if(pos==false)
            {
                q[i]=-1;
                i--;
            }
            else i++;
        }
        for(int k=0;k<n;k++)
            System.out.println("q"+k+"={"+k+","+q[k]+"}");
    }
    private boolean isFeasable(int x)
    {
        int hit=0;
        for(int a=0;a<x;a++)
        {

                if(q[a]==q[x]||(q[a]-q[x])*(q[a]-q[x])==(a-x)*(a-x))
                {
                    return false;
                }


        }
        return true;
    }
}
```

Java Program for cuckoo search

Program: 1

```
package CSapp;                                  //make class Queen under package CSapp

import java.io.*;
//import java.util.ArrayList;
public class Queen {

   private int n;
   public int[] setQueen()
   {
      int temp;
      int q[] = null;
      System.out.println("enter the size of chess board");
      try{
      BufferedReader ip = new BufferedReader(new InputStreamReader(System.in));
      n = Integer.parseInt(ip.readLine());
      if(n<=3)
      {
         System.out.println("invalid input");
         System.exit(0);
      }
      }
      catch(IOException e)
      {
      System.out.println("IOException: " + e);
      }
      q = new int[n];
      q[0]=(int)(Math.random()*(n/4));
      for(int i=1;i<n;i++)
      {
         if(q[i-1]+2<n)
         q[i]=q[i-1]+2;
         else if(q[1]%2==0)
```

```
            {
               q[i]=1;
                }
            else
            {
                q[i]=0;
            }
        }
        return q;
    }
    public void printQueen(int[] x)                                    // method for print output
    {
        for(int i=0;i<n;i++){
        System.out.println("Q"+i+" = {"+i+","+x[i]+"}");
        }
    }
    public static int hitCount(int[] q)
    {
        int hit=0 ;
        for(int i=0;i<q.length-1;i++)
        {
            for(int j=i+1;j<q.length;j++)
            {
                if((q[i]==q[j]||(q[i]-q[j])*(q[i]-q[j])==(i-j)*(i-j)))
                {
                    hit=hit+1;
                }
            }
        }
        return hit;
    }
}
```

Program: 2

```
/**
make another class CuckooSearch under the same package CSapp
in this class we implemented the code of cuckoo search algorithm
**/

package CSapp;

import java.util.ArrayList;
public class CuckooSearch {
private int n;
private int[] q;
private int numOfAttacks;
private ArrayList<Integer> levy = new ArrayList<Integer>();
private ArrayList<Integer> sol = new ArrayList<Integer>();
public void setQ(int[] a)
{
   n = a.length;
   q = new int[n];

   for(int i=0;i<n;i++)
   {
      q[i]=a[i];
   }

}
public int[] solution()
{
   int hit= Queen.hitCount(q);
   for(int i=0;i<n;i++)

 {
      levy.add(i);
```

```
    }

for(int m=0;m<n;m++)
    {
        int egg = m;//levy.get(i);
        for(int u = 0;u<n;u++)
            sol.add(u);
        numOfAttacks=0;
      for(int u = 0;u<n;u++)
        {
        if((egg!=u)&&(q[egg]==q[u]||(q[egg]-q[u])*(q[egg]-q[u])==(egg-u)*(egg-u)))
              numOfAttacks++;
        }
        while(sol.isEmpty()==false&&numOfAttacks!=0)
        {
            int j = (int)(Math.random()*(sol.size()));
            int nest = sol.get(j);
            sol.remove(j);
            int tempY=q[egg];
            q[egg]=nest;
            int tempHit=Queen.hitCount(q);
            if(hit>=tempHit)
            {
                hit=tempHit;
                 numOfAttacks=0;
               for(int u = 0;u<n;u++)
        {
          if((egg!=u)&&(q[egg]==q[u]||(q[egg]-q[u])*(q[egg]-q[u])==(egg-u)*(egg-u)))
    numOfAttacks++;
        }
            }
            else
                q[egg]=tempY;
        }
    }
    return q;
}
}
```

Program: 3

```
package CSapp;
/**
 *
 * @author Anoop
 */
public class Main {
    public static void main(String[] args) {
        Queen obj = new Queen();
        int[] pos = obj.setQueen();
        obj.printQueen(pos);
        int totalHit = 0;
        totalHit = Queen.hitCount(pos);
        System.out.println(totalHit);
        CuckooSearch objCS = new CuckooSearch();
        int x=0;
        int tempPos[];
        int tempHit;
        objCS.setQ(pos);
        while(totalHit!=0&&x<50)
       {


            pos = objCS.solution();
            totalHit = Queen.hitCount(pos);
            x++;
        }
        obj.printQueen(pos);
        System.out.println(Queen.hitCount(pos));
        System.out.println("iteration " +x);
    }
}
```