A NEW EVOLUTIONARY ALGORITHM BASED ON BERNOULLI'S PRINCIPLE

MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF DEGREE OF

Master of Technology

In

Information Systems

Submitted By:

ANKUR SHRIVAS

(2K12/ISY/06)

Under the Guidance

of

Dr. O. P. Verma

(Prof. and Head, Department of IT)

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(2012-2014)

CERTIFICATE

This is to certify that **Ankur Shrivas (2K12/ISY/06)** has carried out the major project titled **"A new evolutionary algorithm based on Bernoulli's Principle"** in partial fulfilment of the requirements for the award of Master of Technology degree in Information Systems by **Delhi Technological University**.

The major project is a bonafide piece of work carried out and completed under my supervision and guidance during the academic session **2012-2014**. To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Dr. O. P. Verma
Professor and Head
Department of Information Technology
Delhi Technological University
Delhi-110042

ACKNOWLEDGEMENT

I take the opportunity to express my sincere gratitude to my project mentor Dr. O.P.Verma, Prof. and Head of Department, Department of Information Technology, Delhi Technological University, Delhi for providing valuable guidance and constant encouragement throughout the project .It is my pleasure to record my sincere thanks to him for his constructive criticism and insight without which the project would not have shaped as it has.

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

Ankur Shrivas
Roll No. 2K12/ISY/06
M.Tech (Information Systems)
E-mail: ankur.shrivas15@gmail.com

ABSTRACT

In this study a new type of nature inspired Evolutionary algorithm is proposed. The proposed algorithm is based on the Bernoulli's Principle. Bernoulli's principle states that, in a steady flow, the sum of all forms of mechanical energy in a fluid along a streamline is the same at all points on that streamline. Bernoulli's principle can be derived from the principle of conservation of energy. During the streamline flow, the sum of kinetic energy and potential energy remain constant. The proposed algorithm is based on the Bernoulli's Principle only and used the Bernoulli's principle for the optimization purpose. In the proposed algorithm initially particles are assigned positions and velocities. Using these positions and velocities, the fitness function is calculated which is used to find the global best of the system and on every iteration the position and velocity of the particle is updated on the basis of this global best. This algorithm has been applied to various functions to determine the efficiency and the obtained results confirmed the high performance of the algorithm.

Table of Contents

Figures and Tables

Chapter 1

INTRODUCTION

1.1 Overview

Many optimization algorithms which have been developed proved a great source of research. We have seen a growing tendency or interest towards the optimization algorithms. These algorithms solved many complex computational problems related to various fields like image processing, pattern recognition, control objectives etc. In the field of economics also these algorithms proved very useful in predicting the economic behaviour of the market. Various evolutionary algorithms like particle swarm optimization [1], Ant colony optimization [3] have been used to solve different type of problems. These algorithms use different principle in their optimization algorithm. For example, Particle swarm optimization [1] is based on the movement of organisms in a bird flock or fish school. The movements of the particles are guided by their own best known position in the search-space as well as the entire swarm's best known position. Thus, each particle's movement is influenced by its local best known position but, is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. When improved positions are being discovered these will then come to guide the movements of the swarm. This is expected to move the swarm toward the best solutions. The process is repeated and by doing so it is hoped, but not guaranteed, that a satisfactory solution will eventually be discovered. Similarly, Ant Colony Optimization [3] is based on the behaviour of ants seeking a path between their colony and a source of food. In the natural world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food. Thus, when one ant finds a good path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads to all the ants following a single path. The idea of the ant colony algorithm is to mimic this behaviour with "simulated ants" walking around the graph representing the problem to solve.

Other than the movement of the living organisms like ants or swarms, some other optimization algorithm based on various principles are also proposed like Gravitational search algorithm, Wind driven optimization etc. Gravitational search algorithm is based on the law of gravity. This algorithm is based on the Newtonian gravity: ‘‘Every particle in the universe attracts every other particle with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them”. In the proposed algorithm, agents are considered as objects and their performance is measured by their masses. All these objects attract each other by the gravity force, and this force causes a global movement of all objects towards the objects with heavier masses. Hence, masses cooperate using a direct form of communication, through gravitational force. Similarly, Wind driven optimization is inspired from the earth’s atmosphere, where wind blows in an attempt to equalize horizontal imbalances in the air pressure. Since temperature differences lead to variations in air density and air pressure at different locations, horizontal differences in air pressure cause the air to move from high pressure regions to low pressure regions.

1.2 Motivation

A single optimization algorithm cannot solve all the problems related to various fields. Because one optimization algorithm gives better result for some problems but the same algorithm doesn’t provide better results for some other problems. So there always remain a scope and a challenge for new optimization algorithm. Getting inspired from the nature we have introduced a new Evolutionary algorithm in this study which is based on the Bernoulli’s Principle. The flow of liquid particles during the streamline flow is determined through the Bernoulli’s Principle. Bernoulli’s Principle states that for an incompressible liquid, an increase in the speed of the fluid occurs with a decrease in pressure or in the fluid’s potential energy. This Principle is utilised to propose a new algorithm where the particles follows the same mode of guidance as liquid particles followed the Bernoulli’s Principle during streamline.

Chapter 2

LITERATURE REVIEW

2.1 OPTIMIZATION

In mathematics, computer science and management science a mathematical optimization is the selection of a best element from some set of available alternatives.

In the simplest case, an optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function. More generally, optimization includes finding "best available" values of some objective function given a defined domain, including a variety of different types of objective functions and different types of domains.

2.2 OPTIMIZATION PROBLLEM:

An optimization problem can be represented in the following way:

Given: a function $f : A \longrightarrow \mathbf{R}$ from some set A to the real numbers.

Sought: an element x_0 in A such that $f(x_0) \leq f(x)$ for all x in A ("minimization") or such that $f(x_0) \geq f(x)$ for all x in A ("maximization").

Such a formulation is called an optimization problem.

2.2.1 STEPS TO BE FOLLOWED IN SOLVING OPTIMIZATION PROBLEM

Attempting to solve the real world problems via optimization algorithm requires the cyclic performance of the four steps. The main steps are :

1) The observation and study of the real-world situation associated with a practical problem.
2) The abstraction of problem by the construction of mathematical model, that is, described in terms of preliminary fixed parameters and variables, these are to be determined such that the model performs in an acceptable manner.

3) The solution of a resulting pure mathematical problem that requires an analytical or numerical parameter dependent solution.
4) The evaluation of the solution and its practical implementation.

It may be necessary to adjust the parameters and refine the model, which will result in a new mathematical problem to be solved and evaluated. It may be required to perform the modelling cycle a number of times, before an acceptable solution is obtained. The formulation of an appropriate and consistent optimization problem (or model) is probably the most important.

The flow chart to describe the working procedure to solve an optimization problem:

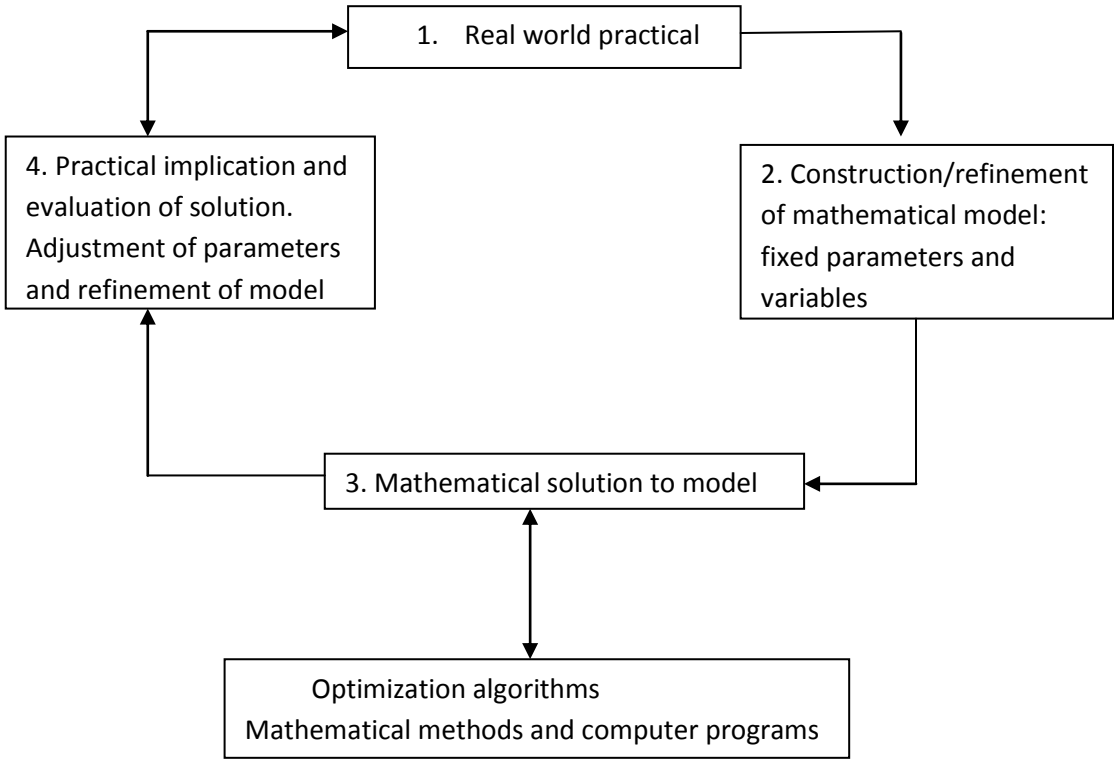

Fig.2.1. Flowchart of Mathematic modelling process to solve optimization problem.

2.3 OPTIMIZATION ALGORITHM

Nature is a wonderful source of inspiration for developing optimization techniques that can tackle difficult problems in science and engineering. Since the early 1970s, various nature inspired optimization algorithms have emerged starting with the Genetic Algorithm (GA) [21], some of which have proven to be very efficient global optimization methods. Along with the Genetic Algorithm, Particle Swarm Optimization (PSO) [1], Ant Colony Optimization (ACO) [3], Differential Evolution [14], and many others have been proposed and successfully implemented. However, because each algorithm possesses strengths and weaknesses, there is no single method within the family of nature-inspired numerical optimization algorithms that stands out as the best for solving all types of problems, a fact which was mathematically proven by Wolpert et al.

2.3.1 EVOLUTIONARY ALGORITHM

An Evolutionary Algorithm uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the quality of the solutions. Evolution of the population then takes place after the repeated application of the above operators.

THE PROCESS:

1. Generate the initial population of individuals randomly – first Generation.
2. Evaluate the fitness of each individual in that population
3. Repeat on this generation until termination (time limit, sufficient fitness achieved, etc.):
    1. Select the best-fit individuals for reproduction - parents
    2. Breed new individuals through crossover and mutation operations to give birth to offspring.
    3. Evaluate the individual fitness of new individuals
    4. Replace least-fit population with new individual.

Evolutionary algorithms often perform well approximating solutions to all types of problems because they ideally do not make any assumption about the underlying fitness landscape;
This generality is shown by successes in fields as diverse as engineering, art, biology, economics, marketing, genetics, operations, research, robotics, social sciences, physics, politics and chemistry. Evolutionary algorithm poses a number of features that can help to position them within the family of generate and test methods:

1. Evolutionary algorithms are population based i.e. they process a whole collection of candidate solutions simultaneously.
2. Evolutionary algorithms mostly use recombination to mix information of more candidate solution to a new one
3. Evolutionary algorithms are stochastic.

Evolutionary Algorithms have a number of components, procedure or operators that must be specified in order to define a particular Evolutionary Algorithm. The most important components are:

1. Representation(definition of individuals)
2. Evaluation function (or fitness function).
3. Population
4. Parent search mechanism.
5. Variation operators, recombination and mutation.
6. Survivor selection mechanism (Replacement).

Each of these components must be specified in order to define a particular Evolutionary Algorithm. To obtain a running algorithm the initialisation procedure and a termination condition must also be defined.

Components of Evolutionary Algorithm

1. Representation

Objects forming possible solutions within the original problem context are referred as phenotypes, their encoding, the individuals within the Evolutionary Algorithm are called genotypes. The first design step is commonly called Representation, as it amounts to specifying a mapping from the phenotypes onto a set of genotypes that are said to represent these phenotypes.

2. **Evaluation function**

The role of evaluation function is to represent the requirements to adapt to. It forms the basis of selection, and thereby it facilitates improvements. It is a function or procedure that assigns a quality measure to genotypes. The evaluation function is commonly called the Fitness function.

3. **Population**

The role of population is to hold possible solutions. A population is a multiset of genotypes. Defining a population can be as simple as specifying how many individuals are in it, that is, setting the population size.

4. **Parent search mechanism**

The role of parent selection or matching selection is to distinguish among individuals based on their quality, in particular, to allow the better individuals to become parents of the next generation. Together with survivor selection mechanism, parent selection is responsible for pushing quality improvements.

5. **Variation Operators**

Its role is to create new individuals from old ones. It performs the generate step.

Mutation

A unary variation operator is commonly called mutation. A mutation operator is always stochastic. Its output depends on the outcomes of a series of random choices.

Recombination

A binary variation operator is called as Recombination or crossover. Similar to mutation it is stochastic operator.

6. **Survivor selection mechanism**

The role of survivor selection mechanism or environmental selection is to distinguish among individuals based on their quality. As opposed to parent selection which is typically stochastic, survivor selection is often deterministic.

Initialization

The first population is seeded by randomly generated individuals. In principle, problem specific heuristics can be used in this step aiming at an initial population with higher fitness.

Termination condition

If the problem has a known optimal fitness level, probably coming from a known optimum of the given objective function, then reaching this level should be used as stopping condition. However Evolutionary Algorithms are stochastic and mostly there are no guarantees to reach an optimum, hence this condition might never get satisfied and the algorithm may never stop. This requires that this condition is extended with one that certainly stops the algorithm. Commonly used options for this purpose are:

1. The total number of fitness evaluations reaches a given limit.
2. The population diversity drops under a given threshold.
3. The maximally allowed CPU time elapses.
4. For a given period of time, the fitness improvements remains under a threshold value.

2.3.2 EXAMPLES OF EVOLUTIONARY ALGORITHM

1. Particle swarm optimization

In computer science, particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. PSO optimizes a problem by having a population of candidate solutions, dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position but, is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

The algorithm was simplified and it was observed to be performing optimization. PSO is a metaheuristic [2] as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, metaheuristics such as PSO do not guarantee an optimal solution is ever found.

The Algorithm

A basic variant of the PSO algorithm works by having a population (called a swarm) of candidate solutions. These particles are moved around in the search-space according to a few simple formulae. The movements of the particles are guided by their own best known position in the search-space as well as the entire swarm's best known position. When improved positions are being discovered these will then come to guide the movements of the swarm. The process is repeated and by doing so it is hoped, but not guaranteed, that a satisfactory solution will eventually be discovered.

For example:

$$f: \mathbb{R}^n \to \mathbb{R}$$

This is the cost function which must be minimized. The function takes a candidate solution as argument in the form of a vector of real numbers and produces a real number as output which indicates the objective function value of the given candidate solution. The gradient of f is not known. The goal is to find a solution $\mathbf{a}$ for which $f(\mathbf{a}) \leq f(\mathbf{b})$ for all $\mathbf{b}$ in the search-space, which would mean $\mathbf{a}$ is the global minimum. Maximization can be performed by considering the function $h = -f$ instead.

Let S be the number of particles in the swarm, each having a position $\mathbf{x}_i \in \mathbb{R}^n$ in the search-space and a velocity $\mathbf{v}_i \in \mathbb{R}^n$. Let $\mathbf{p}_i$ be the best known position of particle i and let $\mathbf{g}$ be the best known position of the entire swarm. A basic PSO algorithm is then

- For each particle $i = 1, ..., S$ do:
  - Initialize the particle's position within the boundaries of the search-space.
  - Initialize the particle's best known position to its initial position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
  - If ($f(\mathbf{p}_i) < f(\mathbf{g})$) update the swarm's best known position: $\mathbf{g} \leftarrow \mathbf{p}_i$
  - Initialize the particle's velocity: $\mathbf{v}_i$
- Until a termination criterion is met e.g. number of iterations performed, or a solution with adequate objective function value is found, repeat:
  - For each particle $i = 1, ..., S$ do:
    - Pick random numbers: $r_p, r_g \sim U(0,1)$
    - For each dimension $d = 1, ..., n$ do:
      - Update the particle's velocity:
      - $\mathbf{v}_{i,d} \leftarrow \omega\, \mathbf{v}_{i,d} + \varphi_p\, r_p\, (\mathbf{p}_{i,d} - \mathbf{x}_{i,d}) + \varphi_g\, r_g\, (\mathbf{g}_d - \mathbf{x}_{i,d})$.

- Update the particle's position: $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$
- If ($f(\mathbf{x}_i) < f(\mathbf{p}_i)$) do:
  - Update the particle's best known position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
  - If ($f(\mathbf{p}_i) < f(\mathbf{g})$) update the swarm's best known position: $\mathbf{g} \leftarrow \mathbf{p}_i$
- Now **g** holds the best found solution.

The parameters ω, φ_p, and φ_g are selected by the practitioner and control the behaviour and efficacy of the PSO method. The choice of PSO parameters can have a large impact on optimization performance. Selecting PSO parameters that yield good performance has therefore been the subject of much research.

2. Ant colony optimization

In computer science and operation research, the ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. This algorithm is a member of the ant colony algorithms family, in swarm intelligence methods, and it constitutes some metaheurisitic optimizations. The algorithm was aiming to search for an optimal path in a graph, based on the behaviour of ants seeking a path between their colony and a source of food. The original idea has since diversified to solve a wider class of numerical problems, and as a result, several problems have emerged, drawing on various aspects of the behaviour of ants. In the natural world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food.

Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones. Pheromone evaporation also has the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained.

Thus, when one ant finds a good (i.e., short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads to all the ants'

following a single path. The idea of the ant colony algorithm is to mimic this behaviour with "simulated ants" walking around the graph representing the problem to solve.

The Algortihm

An ant is a simple computational agent in the ant colony optimization algorithm. It iteratively constructs a solution for the problem at hand. The intermediate solutions are referred to as solution states. At each iteration of the algorithm, each ant moves from a state x to state y, corresponding to a more complete intermediate solution. Thus, each ant k computes a set $A_k(x)$ of feasible expansions to its current state in each iteration, and moves to one of these in probability. For ant k, the probability p_{xy}^k of moving from state x to state y depends on the combination of two values, viz., the *attractiveness* η_{xy} of the move, as computed by some heuristic indicating the *a priori* desirability of that move and the *trail level* τ_{xy} of the move, indicating how proficient it has been in the past to make that particular move.

The *trail level* represents a posteriori indication of the desirability of that move. Trails are updated usually when all ants have completed their solution, increasing or decreasing the level of trails corresponding to moves that were part of "good" or "bad" solutions, respectively.

In general, the kth ant moves from state x to state y with probability

$$p_{xy}^k = \frac{(\tau_{xy}^\alpha)(\eta_{xy}^\beta)}{\sum_{y \in \text{allowed}_y}(\tau_{xy}^\alpha)(\eta_{xy}^\beta)}$$

Where,

- τ_{xy} is the amount of pheromone deposited for transition from state x to y, $0 \leq \alpha$ is a parameter to control the influence of τ_{xy}, η_{xy} is the desirability of state transition xy (*a priori* knowledge, typically $1/d_{xy}$, where d is the distance) and $\beta \geq 1$ is a parameter to control the influence of η_{xy}. τ_{xy} And η_{xy} represent the attractiveness and trail level for the other possible state transitions.

Pheromone update

When all the ants have completed a solution, the trails are updated by $\tau_{xy} \leftarrow (1-\rho)\tau_{xy} + \sum_{k} \Delta\tau_{xy}^{k}$

Where, τ_{xy} is the amount of pheromone deposited for a state transition xy, ρ is the *pheromone evaporation coefficient* and $\Delta\tau_{xy}^{k}$ is the amount of pheromone deposited by kth ant.

2.3.3. OTHER OPTIMIZATION ALGORITHM

Other than the evolutionary algorithm various other optimization algorithms based on the various natural principles which are responsible for various phenomena of the atmosphere were proposed. For example Gravitational search algorithm which is based on the principle of gravitational force of the earth was proposed. Similarly an algorithm known as Wind driven algorithm which is based on the movement of the wind in the atmosphere was also proposed. Thus, other than the algorithm which are based on the movement of living organisms and insects like swarm, ants, bee, firefly etc, there are also some other algorithms which are based on the nature's principle. These algorithms also proved their efficiency in various practical problems like Wind driven algorithm has been applied to various problems like Linear antenna array synthesis [31] [32], Double-sided artificial magnetic conductor synthesis[31], E-shaped patched antenna synthesis and provided good results. So these algorithms also provided a great and wide area of research. In the same direction we also proposed an algorithm which is based on the principle of Bernoulli's Principle. A little overview of the earlier proposed algorithm which is based on nature's principle is provided.

1. **Wind Driven Optimization**

In essence, WDO is a population based iterative heuristic global optimization technique for multi-dimensional and multi-modal problems with the potential to implement constraints on the search domain similar to PSO, although this potential is not explicitly demonstrated in this manuscript. The inspiration for WDO comes from atmospheric motion in which the trajectory of an infinitesimally small air parcel can be described via Newton's second law of motion. The inspiration for WDO comes from the earth's atmosphere, where wind blows in

an attempt to equalize horizontal imbalances in the air pressure [31] [32]. The term "wind" actually refers to horizontal air motion, particularly in the lowest layer of the earth's atmosphere called the troposphere. The troposphere extends from the surface of the earth's crust up to approximately 18 km in altitude, where the thickness may vary based on latitude [31]. Due to the earth's gravitational field, , the mass of the atmosphere applies a force on the earth's crust, where the air pressure can simply be defined by the force exerted per unit area [11] [12]. The troposphere layer contains more than 75% of the atmosphere's mass, and most weather activities, such as wind, occur within it. The radiation from the sun that reaches earth causes heating both on the surface of the earth and the atmosphere itself. However, the amount of localized heating varies depending on various factors such as latitude, the amount of cloud coverage in the region and whether the area is a body of water or soil. In addition, the spherical shape of the earth allows for the illumination of only half of the earth's surface at any given time, resulting in a daily fluctuation of the amount of energy falling on a particular location. Due to variations in solar energy reaching different locations on the earth's surface, the temperature can fluctuate significantly among regions. Areas with high temperatures have rising warm air, and regions with low temperatures have sinking cold air, which causes the air density to decrease in high temperature areas and to increase in low temperature areas. Since temperature differences lead to variations in air density and air pressure at different locations, horizontal differences in air pressure cause the air to move from high pressure regions to low pressure regions [11] [12].

More specifically, the wind blows in the direction from a high pressure zone to a low-pressure zone at a velocity proportional to the pressure gradient force. Thus, a negative sign is utilized in the equation below to indicate the descending direction in the gradient.

In the WDO abstraction, an air parcel is considered to be dimensionless, so that the WDO implementation is not complicated by separate coordinates for each corner of the cuboid. Thus, in the implementation of the WDO scheme, the air parcels are assumed to be dimensionless and weightless, which simplifies the equations while preserving the accuracy of the physical interpretation.

In the abstraction of the wind, it is also assumed that the atmosphere is homogenous and that a hydrostatic balance exists. Utilizing the fact that the equations derived in atmospheric dynamics are in a rectangular coordinate system and considering that the horizontal movement of air is stronger compared to its vertical movement, the wind can be treated as a

horizontal motion only, which is due entirely to the horizontal pressure variation [38]. On the other hand, the WDO algorithm will operate on an n-dimensional search space, so the three dimensional atmospheric dynamic equations must be re-mapped to handle multi-dimensional optimization problems.

$$u_{new} = (1-\alpha)u_{cur} - gx_{cur} + (\mathrm{RT}\,|\frac{1}{i}-1|\,(x_{opt} - x_{cur})) + (\frac{cu_{cur}^{other\,\dim}}{i}) \qquad (1)$$

At each iteration, the velocity and the position of all air parcels need to be updated. Once the new velocity is calculated the position can be updated.

$$x_{new} = x_{cur} + (u_{new}\Delta \mathrm{t})\,. \qquad (2)$$

The flowchart describing the working procedure of the Wind driven algorithm is given in fig 2.2.

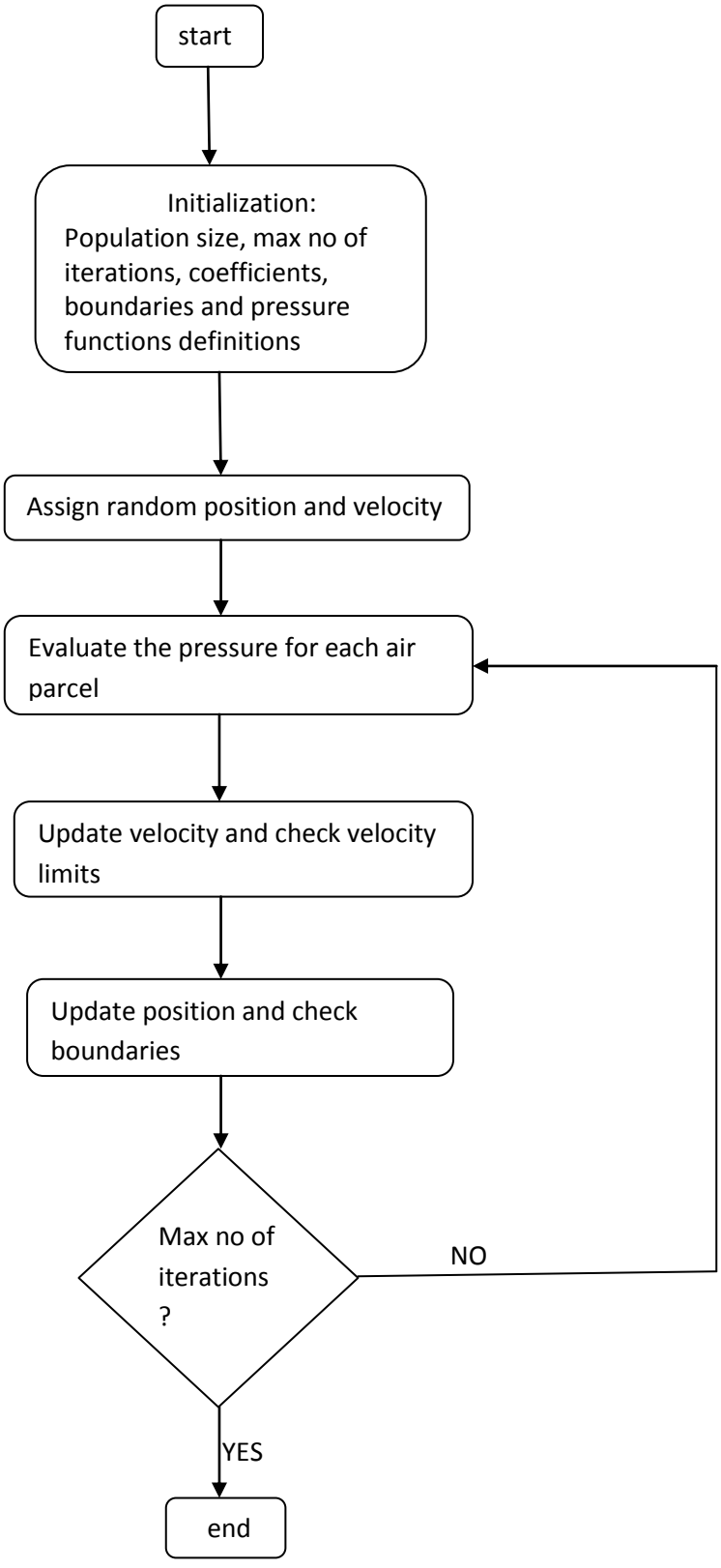

Fig. 2.2 Flowchart regarding working procedure of WDO.

Gravitational Search Algorithm

A new optimization algorithm based on the law of gravity, namely Gravitational Search Algorithm (GSA) [30] is proposed. This algorithm is based on the Newtonian gravity: ‘‘Every particle in the universe attracts every other particle with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them. The different steps of the proposed algorithm are the followings:

(a) Search space identification.

(b) Randomized initialization.

(c) Fitness evaluation of agents.

(d) Update G(t), best(t), worst(t) and M i (t) for i = 1,2,...,N.

(e) Calculation of the total force in different directions.

(f) Calculation of acceleration and velocity.

(g) Updating agents’ position.

(h) Repeat steps c to g until the stop criteria is reached.

(i) End.

In other words, each mass presents a solution, and the algorithm is navigated by properly adjusting the gravitational and inertia masses. By lapse of time, we expect that masses be attracted by the heaviest mass. This mass will present an optimum solution in the search space.

The GSA could be considered as an isolated system of masses. It is like a small artificial world of masses obeying the Newtonian laws of gravitation and motion. More precisely, masses obey the following laws:

Law of gravity: each particle attracts every other particle and the gravitational force between two particles is directlyproportional to the product of their masses and inversely proportional to the distance between them, R. We use here R instead of R 2 , because according to our experiment results, R provides better results than R 2 in all experimental cases.

Law of motion: the current velocity of any mass is equal to the sum of the fraction of its previous velocity and the variation in the velocity. Variation in the velocity or acceleration of any mass is equal to the force acted on the system divided by mass of inertia.

The flowchart describing the working procedure of the algorithm is given in the fig 2.3.

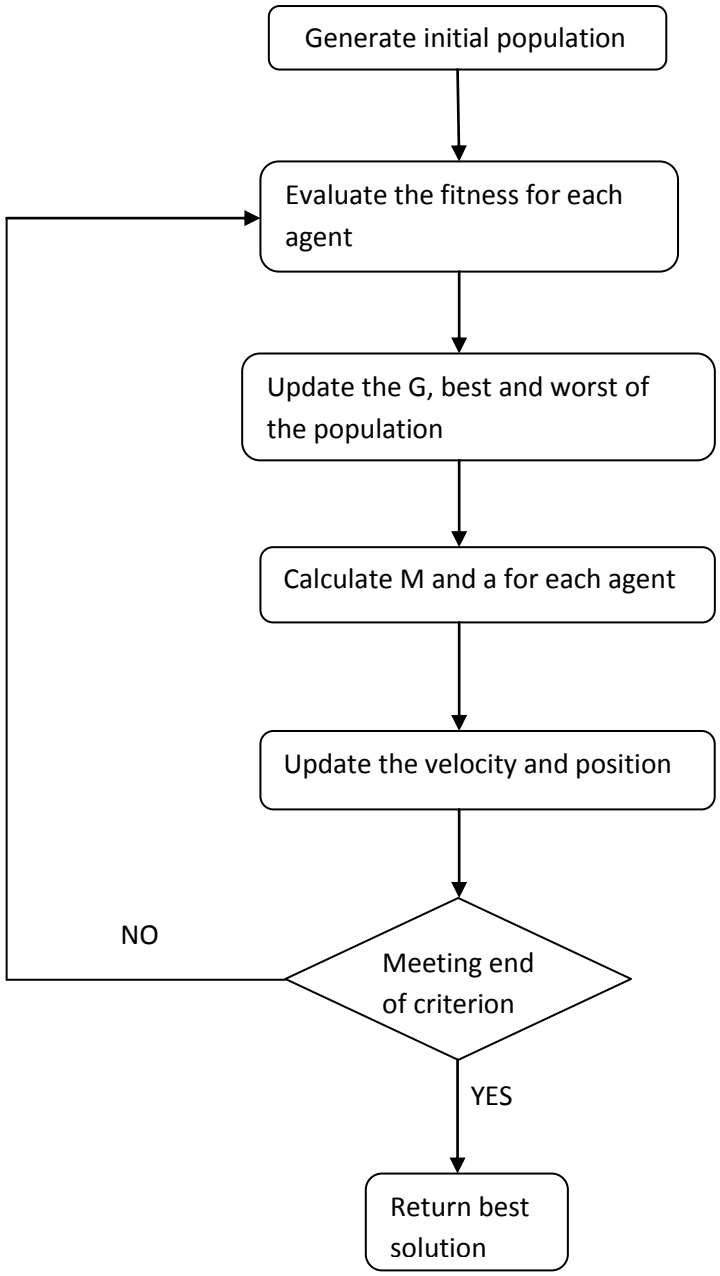

Fig. 2.3 Flowchart regarding working procedure of WDO.

Chapter 3

PROPOSED METHODOLOGY

In this study a new type of nature inspired optimization algorithm is proposed which is based on the Bernoulli's Principle. The proposed algorithm follows the Bernoulli's principle which governs streamline flow of liquid particles. The flow of liquid particles during the streamline flow is determined through the Bernoulli's Principle. Bernoulli's Principle states that for an incompressible liquid, an increase in the speed of the fluid occurs simultaneously with a decrease in pressure or in the fluid's potential energy. In this algorithm particles are assigned random velocities and the positions. The positions of these particles then updated at every iteration to attain the global minima of the function.

3.1 Bernoulli's Principle

This principle was given by Daniel Bernoulli. Bernoulli's Principle [11-12] is based on the principle of conservation of energy. In fluid dynamics, Bernoulli's Principle states that an increase in the speed of the fluid depends upon the pressure and the fluid's potential energy. Bernoulli's Principle states that in a steady flow the sum of all forms of energy viz. Kinetic energy and potential energy remains constant. Thus the speed of the fluid varies with an increase or decrease in both its dynamic pressure and kinetic energy, and a decrease in its static pressure and potential energy.

Fluid particles react only to the pressure difference and to their own weight. During streamline flow, if the fluid particles moving with an increase in the speed then it must be flowing from a region of high pressure to a region of low pressure and if the particle is moving with a decrease in the speed then the direction of flow is from a region of low pressure to a region of high pressure. Also the highest speed of the fluid particles observed where the pressure is lowest along the streamline while it will remain low where the pressure is highest. The Fig.2.4 represents the Bernoulli's Principle mathematically.

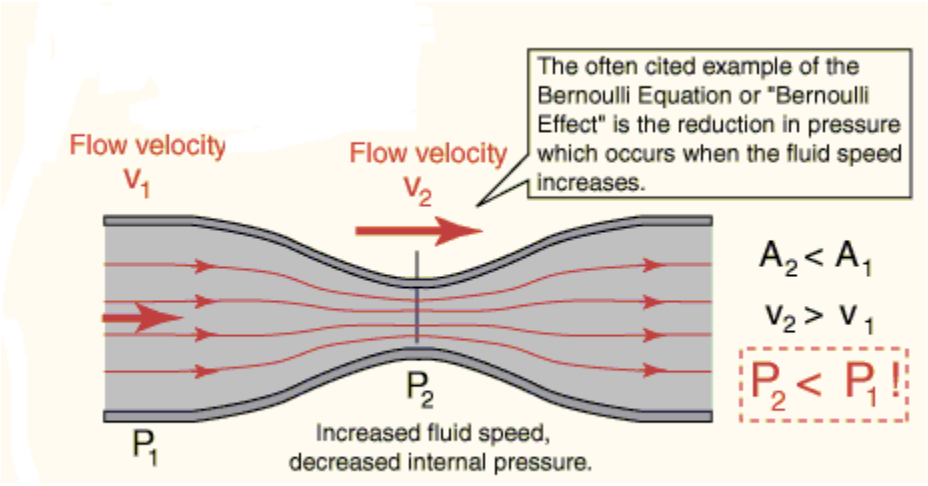

Fig. 2.4 Flow of liquid following Bernoulli's Principle

According to Bernoulli's Principle,

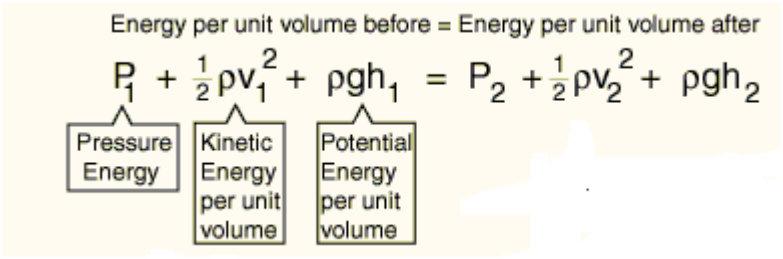

Fig. 2.5 Standard equation describing Bernoulli's Principle

The symbols have their standard representation in the Fig(2.5).

3.2 Incompressible flow equation

In most flows of liquids, the density of a fluid particle can be considered to be constant. Therefore these fluids are known as Incompressible fluids and their flow is considered as incompressible flow. Bernoulli's Principle is valid only for incompressible flow. The equation which applies to the streamline flow known as Bernoulli's equation is

$$\frac{v^2}{2} + gh + \frac{p}{\rho} = k \qquad (3)$$

Where v = velocity with which the liquid particle is moving

g = gravitational constant

h = elevation with respect to the reference plane

p = pressure at a particular point

ρ = density of liquid particle.

k = constant.

This equation suggests that the sum of the energy at every point remains constant. The first term corresponds to the kinetic energy of the particle at any point of time while the second one represents the potential energy component of the particle. The third and the last term represent the pressure energy.

3.3 Proposed Algorithm

In this section a new optimization algorithm is introduced based on Bernoulli's Principle mentioned above. The algorithm follows the Bernoulli's Principle for the optimization of particles. Bernoulli's Principle is followed by liquid particles during the streamline flow of liquid. In the streamline flow of liquid, the flow of liquid particles is observed from a region of high pressure towards the region of low pressure and the speed of liquid particles also changes with the variation in pressure gradient. But the sum of all forms of energy remains constant at every point of time, considering this fact it can be concluded that the sum of energy of a particle at the current position is equal to the sum of energy of the particle at the optimum position.

$$\frac{v_{opt}^2}{2} + gx_{opt} + \frac{p_{opt}}{\rho} = \frac{v_{cur}^2}{2} + gx_{cur} + \frac{p_{cur}}{\rho} \tag{4}$$

Where, v_{opt} = velocity of the particle at the optimum position

x_{opt} = optimum position of the particle.

v_{cur} = velocity of the particle at the current position

x_{cur} = current position of the particle.

p_{cur} = pressure at the current position.

p_{opt} = pressure at the optimum position.

ρ = density of the particle.

g = gravitational constant.

The density of the particle is considered as a constant factor. Also the potential energy related to the gravitational force acts as a vertical force in the physical three dimensional atmosphere, but when it is mapped to N-dimensional space it becomes an attractive force that pull towards the origin of the coordinate system.

By rearranging the terms:

$$\frac{v_{opt}^2}{2} - \frac{v_{cur}^2}{2} = g(x_{cur} - x_{opt}) + \frac{p_{cur}}{\rho} - \frac{p_{opt}}{\rho} \quad (5)$$

$$\frac{(v_{opt}^2 - v_{cur}^2)}{2} = g(x_{cur} - x_{opt}) + \frac{1}{\rho}(p_{cur} - p_{opt}) \quad (6)$$

Relation with Kinematics:

The following well known equations of kinematics are utilised in the proposed algorithm to illustrate the motion of particle. These equations are:

$$v = u + at \quad (7)$$

$$s = ut + \frac{1}{2}at^2 \quad (8)$$

$$v^2 = u^2 + 2as \quad (9)$$

Where v = final velocity of the particle

u = initial velocity of the particle

a = acceleration of the particle with which the particle is moving.

t = time taken by the particle to cover the distance.

s = distance covered by the particle.

The kinematics describes the motion of a particle. In the proposed algorithm also the liquid particle follows the same equations of motion. Therefore the movement of a particle from one position to another can be described on the basis of these equations. Considering here that the particle is moving from current position to the optimum position, the equations can be applied.

$$v_{opt}^2 = v_{cur}^2 + 2a(x_{opt} - x_{cur}) \quad (10)$$

$$v_{opt}^2 - v_{cur}^2 = 2a(x_{opt} - x_{cur}) \quad (11)$$

Here $s = (x_{opt} - x_{cur})$ = displacement covered by the particle.

Putting the value of equation (6) in equation (4) we will get

$$a(x_{opt} - x_{cur}) = g(x_{cur} - x_{opt}) + \frac{1}{\rho}(p_{cur} - p_{opt}) \quad (12)$$

$$a = -g + \frac{1}{\rho}(p_{cur} - p_{opt}) \quad (13)$$

Using equation(5) and considering, t=1

$$v_{opt} = v_{cur} + a \quad (14)$$

Putting the value of a from the equation (11)

$$v_{opt} = v_{cur} - g + \frac{1}{\rho}(p_{cur} - p_{opt}) \quad (15)$$

Once the new velocity is calculated, the new position can be updated of that particle using the equation :

$$x_{new} = x_{cur} + (v_{opt}\Delta t) \quad (16)$$

Where x_{new} = updated position of the particle.

x_{cur} = current position of the particle.

v_{opt} = updated velocity.

The flow chart related to the working procedure of the algorithm is described in the figure(2.6). In this we have described the working of the algorithm step by step. In first step it initializes various parameters. In the next step it assigns random velocities and positions to the particles. For each particle then fitness function is calculated and based on the results of fitness function velocity and position are updated of the particle.

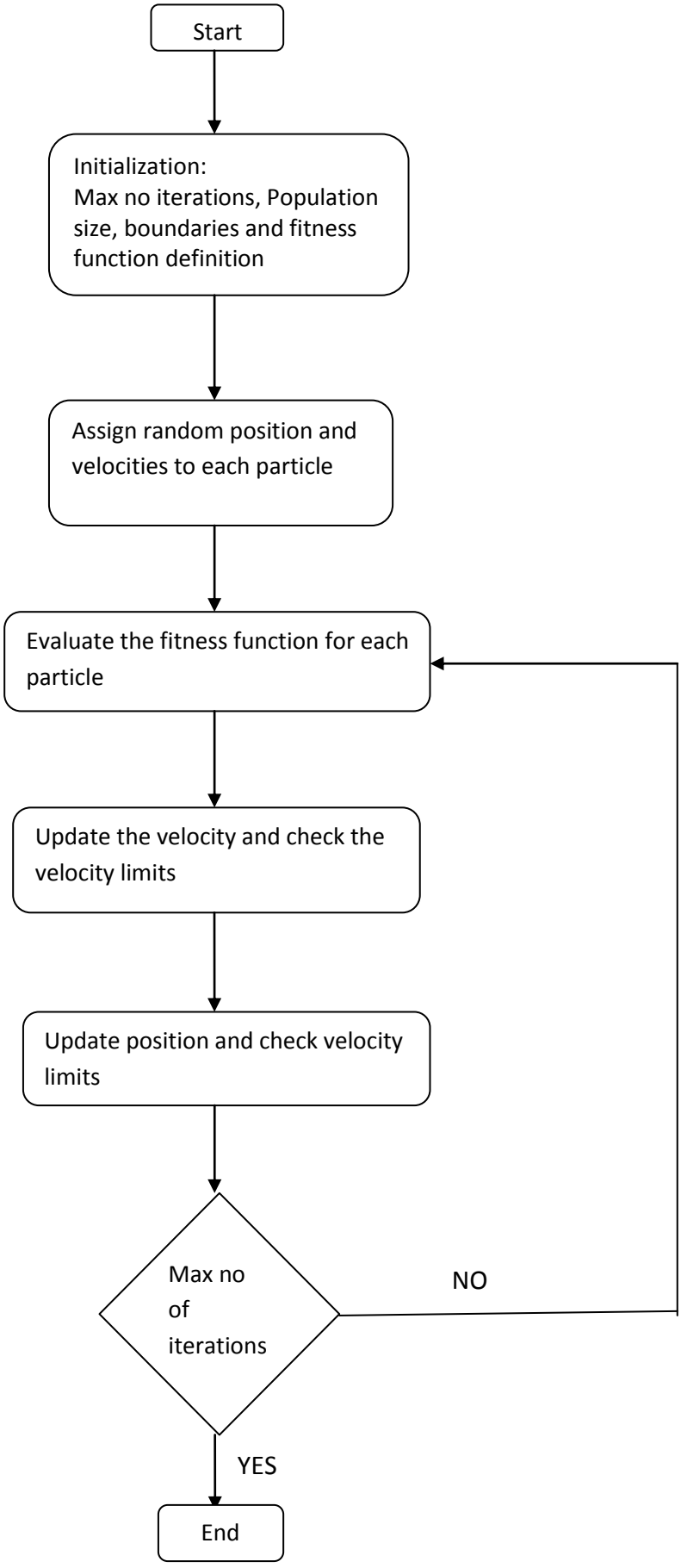

Fig.2.6 Flowchart of the proposed algorithm.

Chapter 4

EXPERIMENTAL RESULTS

4.1 Benchmark Functions

To evaluate the proposed algorithm the proposed algorithm is applied to various benchmark functions to check the efficiency of the algorithm. Some benchmark functions on which the algorithm applied are:

1. **Ackley Function:**

$$f(\mathrm{x}) = -\mathrm{a}\exp(-\mathrm{b}\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}) - \exp(\frac{1}{d}\sum_{i=1}^{d}\cos(\mathrm{c}x_i)) + \mathrm{a} + \exp(1)$$

Global minima: $f(\mathrm{x}) = 0$ at x = (0,...., 0).

2. **Sphere Function:**

$$f(\mathrm{x}) = \sum_{i=1}^{d} x_i^2$$

Global minima: $f(\mathrm{x}) = 0$ at x = (0,...., 0).

3. **Easom Function:**

$$f(\mathrm{x}) = -\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2 - (x_2-\pi)^2)$$

Global minima: $f(\mathrm{x}) = -1$ at x = (π, π).

4. **Bohachevsky Function:**

$$f_1(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2)$$

Global minima: $f_j(x) = 0$ at x = (0,0) for all =1,2,3.

5. **Branin Function:**

$$f(x) = \mathrm{a}(x_2 - \mathrm{b}\,x_1^2 + \mathrm{c}\,x_1 - \mathrm{r})^2 + s(1-\mathrm{t})\cos(x_1) + \mathrm{s}$$

Global minima: $f(\mathrm{x}) = 0.397887$ at x = (-π, 12.275), (π, 2.275) and (9.42478, 2.475).

6. Booth Function

$$f_j(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

Global minima: $f(\mathrm{x}) = 0$ at x = (1, 3).

7. Michalewicz Function

$$f(x) = -\sum_{i=1}^{d} \sin(x_i) \sin^{2m}(\frac{ix_i^2}{\pi})$$

Global minima: $f(\mathrm{x}) = -1.803$ at x = (2.20, 1.57).

8. Shubert Function

$$f(x) = (\sum_{i=1}^{5} i\cos((i+1)x_1 + i))(\sum_{i=1}^{5} i\cos((i+1)x_2 + i))$$

Global minima: $f(\mathrm{x}) = -186.7309$

9. Rosenbrock Function

$$f(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

Global minima: $f(\mathrm{x}) = 0$ at x= (1, 1).

10. Beale Function

$$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$$

Global minima: $f(\mathrm{x}) = 0$ at x= (3, 0.5).

In order to test the performance of the proposed algorithm, the algorithm is experimented with some benchmark functions. Each of the functions selected have different properties, which should allow us to draw conclusions about the performance of the proposed algorithm with respect to different problem topologies. The values for g, ρ,v_{max} and the numerical optimization performance in terms of average minima of the function are listed in Table 1.

Table 1. Parameters used in the test functions and corresponding proposed algorithm Performance.

S.No.	Functions	**g**	**ρ**	v_{max}	**Average minima**
1.	Sphere	0.02	0.001	2.00	0.00000
2.	Ackley	0.02	0.001	0.25	8.88178e-016
3.	Easom	0.02	0.001	0.54	-0.99999
4.	Booth	0.02	0.001	0.51	1.53673076810789e-009
5.	Bohachevsky	0.02	0.001	0.10	0.00000
6.	Branin	0.02	0.001	0.10	0.398346655
7.	Rosenbrock	0.02	0.001	0.30	0.00000
8.	Shubert	0.02	0.001	0.71	-186.0399470
9.	Michalewicz	0.50	0.100	0.57	-1.804082137
10.	Beale	0.02	0.001	0.29	9.75552e-018

Table 2. A detail comparison of the proposed algorithm with other evolutionary Algorithms

Functions	PSO	Firefly Algorithm	Bacterial Foraging Algorithm	Proposed Algorithm
Sphere	**5.9273e-017**	**2.6951e-005**	**7.8730e-006**	**0.00000**
Ackley	**1.0e-010**	**0.00647158**	**0.0066**	**8.88178e-016**
Rosenbrock	**0.00000**	**0.00000**	**1.5393e-004**	**0.00000**
Bohachevsky	**4.4408e-016**	**0.4131883**	**0.0010**	**0.00000**
Beale	**1.7565e-014**	**0.52334490**	**2.7221e-005**	**9.75552e-018**
Michalewicz	**-1.9326**	**-1.37**	**-1.9285**	**-1.804082137**
Booth	2.2792e-010	**1.80000038**	**5.1540e-005**	**1.53673e-009**
Easom	-1.0000	**-0.0127796**	-1.0000	**-0.99999**
Shubert	**-48.5498739**	-186.73	-186.7093	**-186.0399470**
Branin	0.3979	7.7827046	0.3979	0.398346655

4.2 Results and discussions:

In the Table.2 there is a comparison provided of the proposed algorithm with other evolutionary algorithms. The proposed result is providing better results in many of the cases. In sphere, Ackley, Rosenbrock, Beale, Bohachevsky and Michalewicz functions, the proposed algorithm is performing better than PSO, Firefly and Bacterial foraging algorithm. But in the case of booth although PSO is providing the best results but the proposed algorithm is performing better than firefly and bacterial foraging algorithm. Likewise in the case of Easom function the proposed algorithm is performing better than firefly algorithm but not better than PSO and bacterial foraging. But overall the proposed algorithm is providing the better result as compared to other evolutionary algorithms. As it has also been proved that one single algorithm cannot satisfy all the test functions because of their different functionalities. Therefore the performance of the proposed algorithm is better and the obtained results also confirmed the high performance of the proposed algorithm.

Chapter 5

CONCLUSION

In recent years, various Heuristic optimization methods have been developed. Some of these algorithms are inspired by swarm behaviours in nature. In this study, a new type of nature-inspired global optimization algorithm based on the Bernoulli's Principle is proposed. The equations for implementing the proposed algorithm were derived from the Bernoulli's Principle which governs the motion of the liquid particles during streamline flow. Similarly, the proposed algorithm also governs the particle to reach at an optimal position by considering the pressure energy term of the Bernoulli's Principle as the fitness function. The proposed algorithm calculates the fitness function of the particles and based on the value of the fitness function, the global best is decided. The global best is the main factor which is responsible for the movement of the particles because now every particle will follow this global best for their movement. The proposed algorithm doesn't consider the local best for the movement of the particle, only global best is responsible for the movement of particles and based on the global best positions are updated then, at every iteration. To illustrate the simplicity and robustness of the proposed algorithm, the algorithm has been applied to various functions. A detailed comparison of the proposed algorithm with Particle swarm optimization is also done in the study. In some of the functions the proposed algorithm provides better results compared to the particle swarm optimization and Firefly algorithm which proved the efficiency of the proposed algorithm.

Chapter 6

REFERENCES

[1] KennedyJ., Eberhart, R., "Particle swarm optimization", Neural Networks, 1995. Proceedings., IEEE International Conference on Vol. 4 , pp. 1942 – 1948.

[2] X. S. Yang, "Nature inspired metahuristic algorithm, Luniver press, 2008, pp. 81-95.

[3] Z. Baojiang, L. Shiyong, Ant colony optimization algorithm and its application to neuro-fuzzy controller design, Journal of Systems Engineering and Electronics 18 (2007) 603–610.

[4] A. Badr, A. Fahmy, A proof of convergence for ant algorithms, Information Sciences 160 (2004) 267–279.

[5] F.V.D. Bergh, A.P. Engelbrecht, A study of particle swarm optimization particle trajectories, Information Sciences 176 (2006) 937–971.

[6] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics – Part B 26 (1) (1996) 29–41.

[7] W. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Information Sciences 178 (2008) 3096–3109.

[8] R.A. Formato, Central force optimization: a new nature inspired computational framework for multidimensional search and optimization, Studies in Computational Intelligence 129 (2008) 221–238.

[9] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.

[10] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (1999) 82–102.

[11] D. Holliday, R. Resnick, J. Walker, Fundamentals of physics, John Wiley and Sons, 1993.

[12] H.C. Verma, Concepts of physics, 2011.

[13] M. Dorigo and T. Stutzle, Ant Colony Optimization. Cambridge, MA,USA: MIT Press, 2004.

[14] K. Price, R. M. Storn, and J. A. Lampinen, Differential Evolution:A Practical Approach to Global Optimization. Berlin, Germany: Springer-Verlag, 2005.

[15] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics – Part B 26 (1) (1996) 29–41.

[16] O. Cordon, S. Damas, J. Santamarı, A fast and accurate approach for 3D image registration using the scatter search evolutionary algorithm, Pattern Recognition Letters 27 (2006) 1191–1200.

[17] I. Ellabib, P. Calamai, O. Basir, Exchange strategies for multiple ant colony system, Information Sciences 177 (2007) 1248–1264.

[18] J.D. Farmer, N.H. Packard, A.S. Perelson, The immune system, adaptation and machine learning, Physica D 2 (1986) 187–204.

[19] R.A. Formato, Central force optimization: a new metaheuristic with applications in applied electromagnetics, Progress in Electromagnetics Research 77 (2007) 425–491.

[20] C. Hamzaçebi, Improving genetic algorithms' performance by local search for continuous function optimization, Applied Mathematics and Computation 196 (1) (2008) 309–317.

[21] R.L. Haupt, E. Haupt, Practical Genetic Algorithms, second ed., John Wiley & Sons, 2004.

[22] C. Hamzaçebi, Improving genetic algorithms' performance by local search for continuous function optimization, Applied Mathematics and Computation 196 (1) (2008) 309–317.

[23] D.H. Kim, A. Abraham, J.H. Cho, A hybrid genetic algorithm and bacterial foraging approach for global optimization, Information Sciences 177 (2007) 3918–3937.

[24] Y.L. Lin, W.D. Chang, J.G. Hsieh, A particle swarm optimization approach to nonlinear rational filter modeling, Expert Systems with Applications 34

(2008) 1194–1199.

[25] H. Nezamabadi-pour, S. Saryazdi, E. Rashedi, Edge detection using ant algorithms, Soft Computing 10 (2006) 623–628.

[26] X. Tan, B. Bhanu, Fingerprint matching by genetic algorithms, Pattern Recognition 39 (2006) 465–477.

[27] K.S. Tang, K.F. Man, S. Kwong, Q. He, Genetic algorithms and their applications, IEEE Signal Processing Magazine 13 (6) (1996) 22–37.

[28] P.K. Tripathi, S. Bandyopadhyay, S.K. Pal, Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients, Information Sciences 177 (2007) 5033–5049.

[29] T.H. Kim, I. Maruta, T. Sugie, Robust PID controller tuning based on the constrained particle swarm optimization, Automatica 44 (2008) 1104–1110.

[30] Esmat Rashedi, Hossein Nezamabadi-pour, Saeid Saryazdi Gravitational Search Algorithm Department of Electrical Engineering, University of Kerman, Kerman, Iran.

[31] Zikri Bayraktar, Muge Komurcu, Douglas H. Werner: Wind Driven Optimization (WDO): A Novel Nature-Inspired Optimization Algorithm and its Application to Electromagnetics, Department of Electrical Engineering, Department of Meteorology, The Pennsylvania State University.

[32] Zikri Bayraktar, Muge Komurcu, Jeremy A. Bossard, and Douglas H. Werner: The Wind Driven Optimization Technique and its Application in Electromagnetics, IEEE Transactions on Antennas and Propagation, 2013.