NEW DEVELOPMENTS IN COGNITIVE OPTIMIZATION

by

PARVATI ARUNA KANDALA

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2012

To my father K. Ch. A. V. S. N. Murthy,

and mother K. Annapurna

ACKNOWLEDGEMENTS

I would like to convey my deep gratitude to professor Dr. Alan Bowling for his constant guidance and encourangement throughout my research. I am very grateful to him for his trust and strong belief on me which helped in the completion of my research. I would like to thank Dr. B. P. Wang and Dr. Manfred Huber for serving in my thesis committee. I also extend thanks to Dr. Jay Rosenberger for his guidance during my research.

It is my pleasure to thank each and every member of The Robotics, Biomechanics, and Dynamic Systems Laboratory who tolerated, helped me by providing valuble suggestions. Special thanks to Debi Barton for her help through out my study.

I would like to thank one of my close friends, Yashwanth M. Swamy for his moral support and endless help. I would also like to thank my roomates Shravani Dwarakapalli, Anitha Josephine Royappan, Santana Bala for making my life peaceful and lively. My special thanks to Srider Comerica, Harin Pagidimunthala, Shravan Neela, Mahesh Varrey, Vivek Reddy for being with me as a family and all my friends here as well in India for standing by me during my tough times.

I owe my love and gratitude to my parents, my sister Lavanya, my brother Chaitanya and brother-in-law Rathnakar Samavedam for their endless love, encourangement and boosting up my confidence without whom this wouldn't have been

possible. Sincere thanks to Kamalakar Ganti for being extremely supportive and evaluating my performance during my research.

March 6, 2012

ABSTRACT

NEW DEVELOPMENTS IN COGNITIVE OPTIMIZATION

PARVATI ARUNA KANDALA, M.S.

The University of Texas at Arlington, 2012

Supervising Professor: Alan Bowling

This work presents an approach to solve function optimization problems using cognitive optimization. A cognitive search algorithm which is developed mimics human cognition in structure and characteristics. This algorithm is developed based on the key aspects of a cognitive system, namely, the use of prior knowledge, communication, and self-organization. The processes have the knowledge of how to solve the optimization problems from traditional techniques such as bracketing, gradient search and interval-halving methods. This search algorithm shows that, by passing messages, the processes communicate, share information, and self-organize around the solution. A matlab program is coded to perform the tests on difficult problems which are not solved by the traditional bracketing and gradient search techniques alone. This work will be applied on complex functions such as non-convex, multiple local optima and discontinuous functions. The results for the different cases considered explore the structure and characteristics of cognitive optimization, which provide guidelines for the development of general cognitive systems for smart devices.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The main objective of this research is to develop a search algorithm which mimics human cognition in its structure and function. Function optimization is used as a means for demonstrating this structure. This chapter introduces the various definitions of what cognition means, of a cognitive model and its working, and of the aspects of cognition considered and a cognitive approach to optimization.

1.1 Cognition

Cognition is an area within psychology that describes how humans acquire, store, transform and use knowledge [3]. In simple words, cognition is defined as the acquisition of knowledge [4]. It refers to knowing and thinking. It also involves perception, awareness, judgement, the understanding of emotions, memory and learning [5]. The definition of **cognition** given by different authors is listed below.

1. "In my view $\cdots$, it is better to reserve the term **cognition** for the manipulation of declarative knowledge " [6, 7].
2. " $\cdots$ **Cognition** refers to all the processes by which the sensory input is transformed, reduced, elaborated, stored, recovered, and used [including] terms as sensation, perception, imagery, retention, recall, problem solving and thinking " [6, 8].

3. "**Cognition** is the collection of mental processes and activities used in perceiving, remembering, thinking and understanding, as well as the act of using those processes " [6, 9].
4. "The use and handling of knowledge. Those who stress the role of **cognition** in perception underline the importance of knowledge - based processes in the making sense of the 'neurally coded' signals from the eye and other sensory organs. It seems that man is different from other animals very largely because of the far greater richness of his cognitive processes. Associated with memory of individual events and sophisticated generalizations, they allow subtle analogies and explanations - and ability to draw pictures and speak and write" $\cdots$ [6, 10].

1.2 Human Cognition and Optimality

Human cognition is the study of how the human brain thinks. It has become the major study area in the field of artificial intelligence to emulate the high-level human capability [11]. It is far beyond that of machine cognition. It has the ability to evaluate an ideal result depending on the outcomes. Therefore, the selection of actions or perception of things further is based on the output of the resulted actions. The inclusive knowledge helps for better outcomes of the options available for the system's progress. Von Neumann's concept of cardinal utility function [12] suggests and supports that the theory of optimization is the key aspect of cognition. In simpler words, the ability to optimize is a key aspect of cognition [13]. Human cognition is highly complicated and involves a complex structure. Intelligent systems or the working models (cognitive architectures) are developed to the range where behavior similar to human thoughts is demonstrated. A number of models were proposed that

describe how they might function [14]. One such cognitive model is considered in this research [15].

1.3 Model of Cognition

The cognitive model attempts to recreate the way the brain carries out a particular task such as learning or making decisions. In this work, cognition is separated into four broad functional categories, namely, perception, reason, decision and action. The functionality of each area is discussed below.

Perception Collection of information about the world through available sensors.

Reason Drawing conclusions about the world from the data collected.

Decision Depending on the conclusions, a course of action is determined.

Action Suitable action is carried out upon the taken decision.

The result of the particular action taken is provided by the system's perception sensors and the cycle repeats accordingly. Each functional area shown in Fig. 1.1 is composed of a set of processes that operate on the information passed in between. The proposed research focuses primarily on the communication between processes. All processes communicate with each other by passing messages in order to share information. The arrow marks represent the possible routes of communication. The figure shows all possible ways of communication between the areas as there is no specific path to follow in order to share the information.

1.3.1 Cognitive Characteristic Model

The principle characteristic of this model is to break down the functional area into a finite set of elements or processes. This approach involves deconstructing a complex problem into simple processes whose inner workings should be as simple as

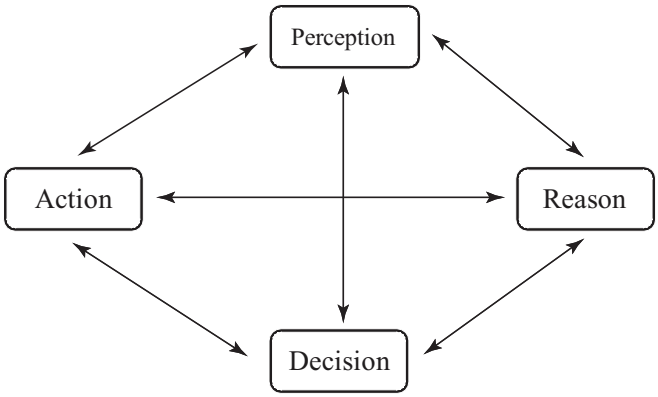


Figure 1.1. Model of Cognition.

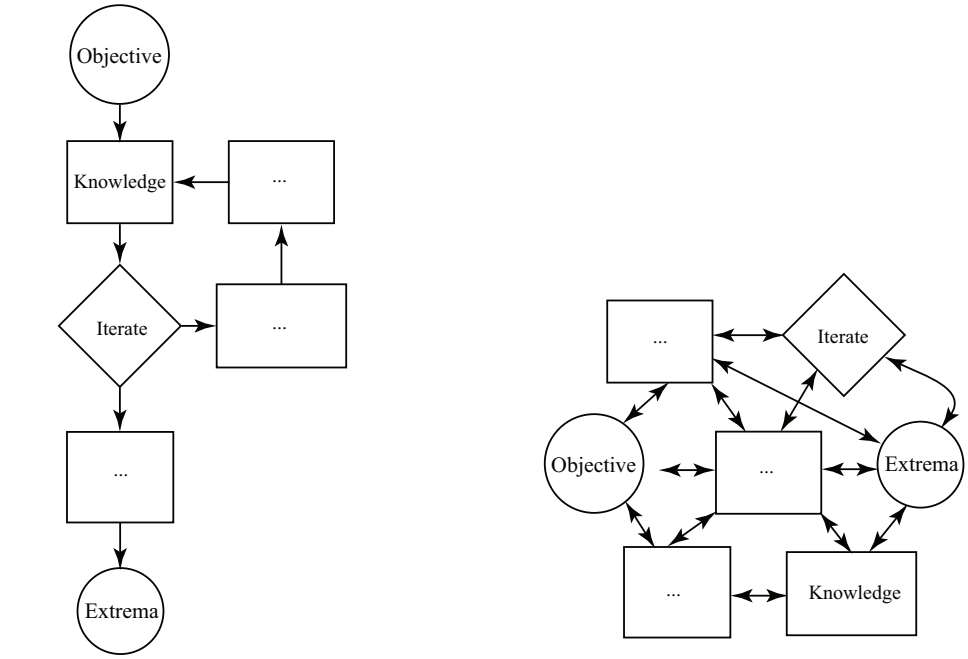
possible, which therefore simplifies the development of a cognitive system. Having processes which function differently is a key element in a cognitive system that generates something like *imagination*, to fulfill this purpose [16]. Thus, the cognitive approach uses this finite set of processes considered as inherited knowledge, which communicate in order to self-organize around a particular action or solution. All processes have freedom to interact with each other in this system. There is no higher hierarchy involved. Although there are several different aspects of cognition, only the following are considered here.

Aspects of Cognition

1. The use of inherited knowledge to arrive at a solution or an action,
2. The use of communication between different system components to facilitate organization,
3. Synergism or self-organization [15].

1.3.2 Cognitive Organizational Model

The organization of the processes is synergistic in nature. Fig. 1.2 shows the difference in the structures of Sequential and Non-sequential system



(a) Traditional Optimization, Non-Synergistic.

(b) Cognitive or Synergetic Optimization.

Figure 1.2. Sequential Vs Non-Sequential.

Sequential System A sequential system relies on explicit definitions of logic paths between inputs and outputs. As shown in Fig. 1.2 (a), every possible path between the input and output must be encoded. This system leads to complex structures due to an enumerated set of logic paths. This is associated with rule-based systems.

Non-Sequential system

These diagrams are more likely seen in psychology literature, describing how sections of the brain map to different functions and process information in or-

der to self-organize around a particular action [17, 18]. These systems differ from the standard representation as there is no pre-defined pathway between the inputs and outputs. Fig. 1.2(b) shows a cognitive system. Messages passing through the processes follow different paths, in effect reorganizing them. This self-organization is a key aspect of cognition. The number of logic paths is formed by the number of permutations of the processes. This undirected communication generates numerous combinations of logic paths, allowing the system to handle the large number of inputs and outputs associated with human cognition without encoding every single logic path.

1.4 Evaluation Metrics

The goal here is to develop a search algorithm to the extent that it displays characteristics, behavior, and performance similar to human cognition using function optimization. The Function optimization problem was chosen because its parameters are clearly defined and a clear definition of a solution exists in order to analyze the guidelines for the further development of cognitive systems. Human cognition has the ability to choose a better result depending on the outcomes of that time and can handle a wide range of tasks which drives the use of two performance metrics in the development of the cognitive search, namely *convergence rate* and *feasibility*. The idea is to increase the rate of convergence as well as the feasible problem set that the algorithm can address. Here, the attempt is to show that cognitive optimization can provide a viable approach to function optimization, but not to attempt *the most efficient, complete, comprehensive, or competitive search algorithm possible*.

CHAPTER 2

BACKGROUND

Cognitive Optimization can be categorized as a non-traditional optimization technique. These approaches work over wide range of problem sets and are more likely to find global optima, unlike traditional techniques.

2.1 Traditional Vs Cognitive Optimization

Traditional optimization methods provide powerful tools for locating and characterizing local extrema. But they are limited to finding local optimal points in continuous and differentiable functions. These methods are analytical and make use of the techniques of mathematical formalisms, geometry, statistics, calculus, geometry in locating the optimum points. These techniques have limited scope in practical applications.

Non-traditional optimization techniques primarily overcome some of the limitations of the classical techniques and have greater chances of finding global optima. These algorithms work on discontinuous functions over discontinuous domains and deal with complex engineering optimization problems.

Local & Global Maxima: A local maximum value of a function is the highest value of a function within a very small interval. Similarly, a global maximum is the highest value of the function within the entire domain.

Well-known examples of non-traditional optimization are referred to as evolutionary approaches such as genetic algorithms. Ant colony optimization,

Particle swarm optimization, Social cognitive optimization are different non-traditional optimization techniques.

2.1.1 Genetic Algorithm

Genetic Algorithm (GA) belongs to the class of Evolutionary Algorithms. They mimic the principles of natural genetics and natural selection to constitute search optimization procedures [19]. They are generally applied in many fields as in engineering, economics, chemistry, bioinformatics, computer science, etc.

In a GA, a population of strings (called chromosomes or genomes) define a candidate solution to an optimization problem.

GA's work in these following steps:

1. Initialisation: The initial population is generated randomly. The population depends on the type of the problem.
2. Selection: A proportion of the existing population is selected to breed a new generation. In the current population, individual solutions are selected through a fitness-based process.
3. After selection, if the termination criterion is met, the best solution is returned, else the second generation is selected through genetic operators (reproduction, crossover, mutation) to these individuals.
4. Reproduction: The selected individuals, parents are set for breeding which result in child solution using the crossover and mutation methods which represent characteristics similar to that of parent solution[20, 21].
5. Termination: This process continues till the termination conditions are reached. Common termination conditions are:

- A solution is found that satisfies minimum criteria

- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations likely no longer produce better results
- Manual inspection
- Combinations of the above [22]

The Genetic Algorithm flow chart is shown in Fig. 2.1.

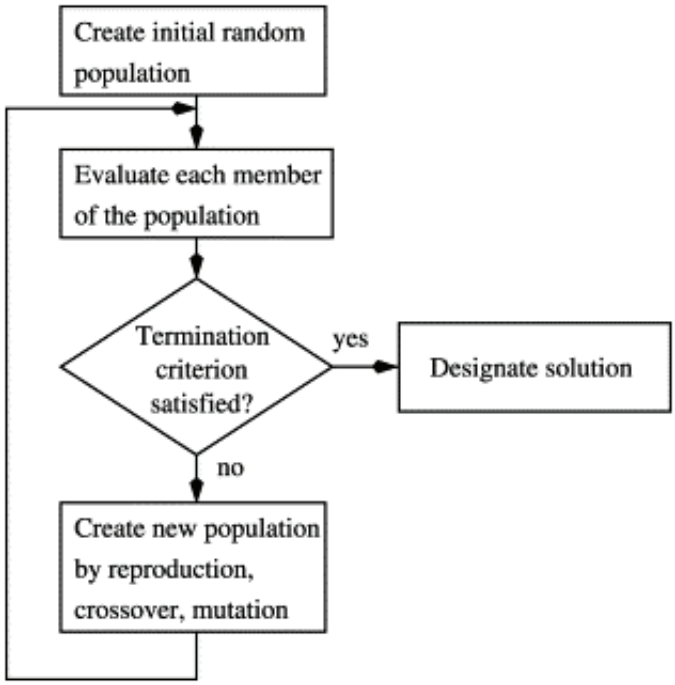


Figure 2.1. Genetic Algorithm Flow chart [1].

2.1.2 Ant colony optimization

Ant colony optimization (ACO) is a population based metaheuristic and the first approximate technique which was inspired by the collective behavior of social insect colonies [23, 24] designed for combinatorial optimization problems. ACO derives inspiration from the foraging behavior of some ant species. This algorithm was developed on aiming the optimal path, based on the behavior of ants seeking a path between their nests and food sources through graphs [25]. The solution construction process is stochastic and is biased by a pheromone model. [26, 27]. Ant colony optimization is applied in diversified fields such as, multi-objective functions, parallel implementations, continuous optimization, dynamic optimization problems, stochastic optimization problems, industrial problems, constraint satisfaction, bioinformatics problems etc.[28]

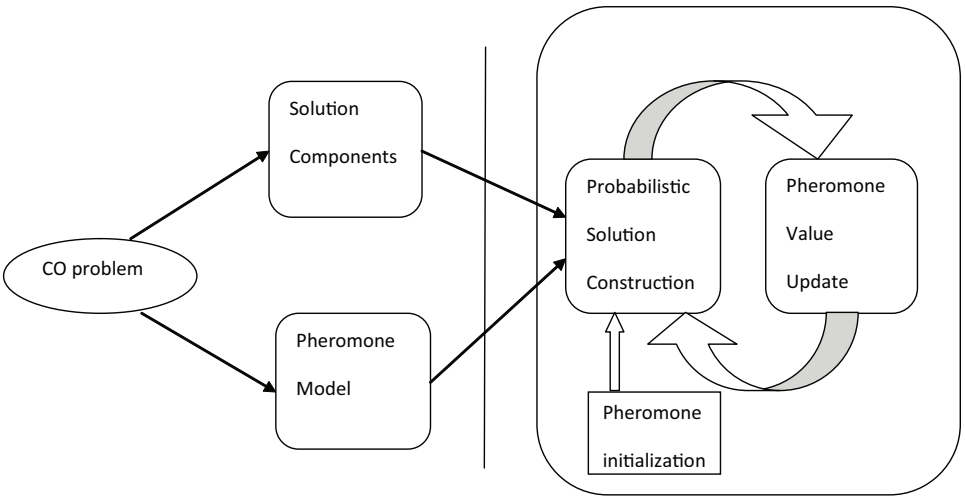


Figure 2.2. The ACO Frame Work [2].

2.1.3 Particle swarm optimization

Particle swarm optimization (PSO) is a population based, stochastic approach, self-adaptive search optimization technique which was first introduced by Kennedy and Eberhart [29, 30]. PSO belongs to the class of swarm intelligence techniques that are used to solve continuous and discrete optimization problems. The simulation of social behavior of the bird flock, fish schooling etc are the motivations for the development. Though PSO shares many similarities with that of GA, the former has no evolutionary operators such as crossover and mutation. In PSO simple software agents, the potential solutions called particles, move in the search space of an optimization problem. The position of a particle represents a candidate solution to the optimization problem at hand. Each particle searches for better positions in the search space by changing its velocity (accelerating) according to its rules towards *pBest* and *nBest* locations (local versions of PSO). *pBest* is the particle's location at which the best fitness so far has been achieved and *nBest* is the neighbor's best location at which the best fitness in a neighborhood so far has been achived. Acceleration is weighted by a random term, with separate random numbers being generated for acceleration towards *pBest* and *nBest* locations each particle [31, 32].

Advantages

- Insensitive to scaling of design variables
- Simple implementation
- Easily parallelized for concurrent processing
- Derivative free
- Very few algorithm parameters
- Very efficient global algorthimic search

Disadvantages

- Slow convergence in refined search stage (weak local search ability) [33].

The applications of PSO are in different fields, such as multi-objective optimization, medicine and emergent system identification, electronics and training neural networks etc [34].

2.1.4 Social cognitive optimization

Social Cognitive Optimization (SCO) is a simple optimization model based on the observational learning mechanism in human social cognition [35, 36]. The individuals who simulate the human cognition are social cognitive agents. Each agent possesses a personal memory and a set of simple search rules. Rules of operation are encoded in each agent. The agents cooperate with their peers through the social sharing of information from an external library, that all the agents have access to, instead of from the personal memory of other agents [15, 35]. This algorithm implements a more sophisticated form of interaction. It also converges faster and in fewer function evaluations than the genetic algorithm or particle swarm [37].

2.1.5 Simulated Annealing

Simulated Annealing is one of the non-traditional search algorithms which rely on random sampling. It is a stochastic optimization procedure [38]. The search procedure here is that, the algorithm mimics the cooling phenomenon of molten metals. It operates by simulating the cooling of a physical system whose possible energies correspond to the values of the objective function being minimized [39]. These are simple and highly efficient in resolving the global optimization of a given function in large discrete search space. The main goal is to find a good solution in a fixed amount of time, rather than finding the best solution [40].

2.2 Literature

In the proposed research, the cognitive model is structured with a finite number of simple processes which are unique and deterministic in nature.

A similar kind of work is done by Rodney A. Brooks, MIT Professor and member of M.I.T's Artificial Intelligence Lab who developed the subsumption architecture for controlling mobile robots in 1986. The idea is the deconstruction of complicated intelligent behavior into many simple behavior modules. Each module does a simple, low-level task and all these simple action modules put together can be organized into more complex behaviors.

A layering methodology in increasing levels of competence is built to operate the robot control system. In simple words, the approach is to incrementally build up the capabilities of the intelligent system [41]. Each layer is made of simple and connected processors called Augmented Finite State Machines. Inputs can be supressed and outputs can be inhibitated internally. Each layer acts asynchronously ; The structure can be considered hierarchical because the higher levels subsume the function of the lower levels. This architecture is robust and flexible.

In the cognitive model, the processes interact or communicate with each other by sharing the information through message passing, whereas in the subsumption architecture, the basic idea is the emergent behavior, which means cognition emerges as a result of different layers of architecture acting together.

The key attribute in the cognitive model is communication between processes in all possible paths, replicating non-sequential design. This implies that there is no particular order for the processes to follow. They reorganize by themselves to obtain output. The logical paths are not pre-defined which makes the structure non-deterministic as well as flexible.

2.3 Preliminary Work

In order to avoid complications and uncertainites raised by choosing a complex problem, a simple problem is choosen. Problem Statement :

$$\max_x f(x) := 1 - x^2 \qquad \frac{df(x)}{dx} = -2x \tag{2.1}$$

The optimal solution is $(x, f(x)) = (0, 1)$. The first derivative equals zero at this solution, which will be used as the optimality condition for the cognitive search.

2.3.1 Inherited knowledge

Bracketing and gradient search algorithms [42] are used as inherited knowledge in developing the search algorithm.These techniques search in different ways.

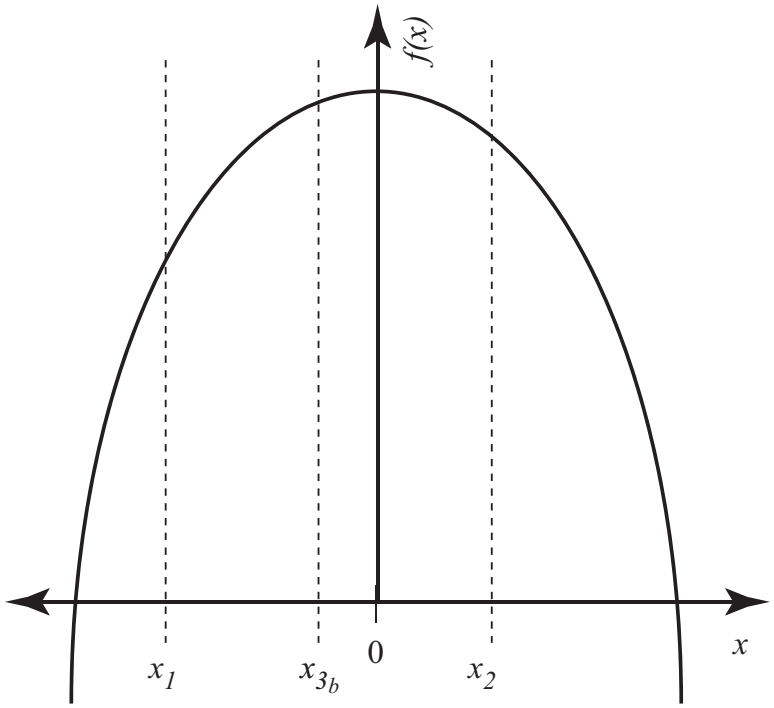


Figure 2.3. Bracketing Search.

Bracketing Search The bracketing search jumps discontinuously around the function's domain. The bracketing search starts with two initial guesses that define a bracket or region that includes an extremum. This is true if the first derivatives of the initial guesses have opposite signs. The goal is to continually reduce the region within the bracket until it surrounds the extremum to within a given tolerance. This process is shown in Fig. 2.3 where x_1 and x_2 are the initial guesses and $x_{3_b} = (x_1 + x_2)/2$ is the midpoint of the bracket. The first derivative at x_{3_b} is calculated and if its sign equals that of x_1, then x_{3_b} replaces it in the bracket; The new bracket is defined between x_{3_b} and x_2. The next step would be to consider $x_{4_b} = (x_2 + x_{3_b})/2$ and so on until the region within the bracket becomes small enough to conclude that the extremum has been found. This bracketing search is different than the one traditionally used in root finding [42].

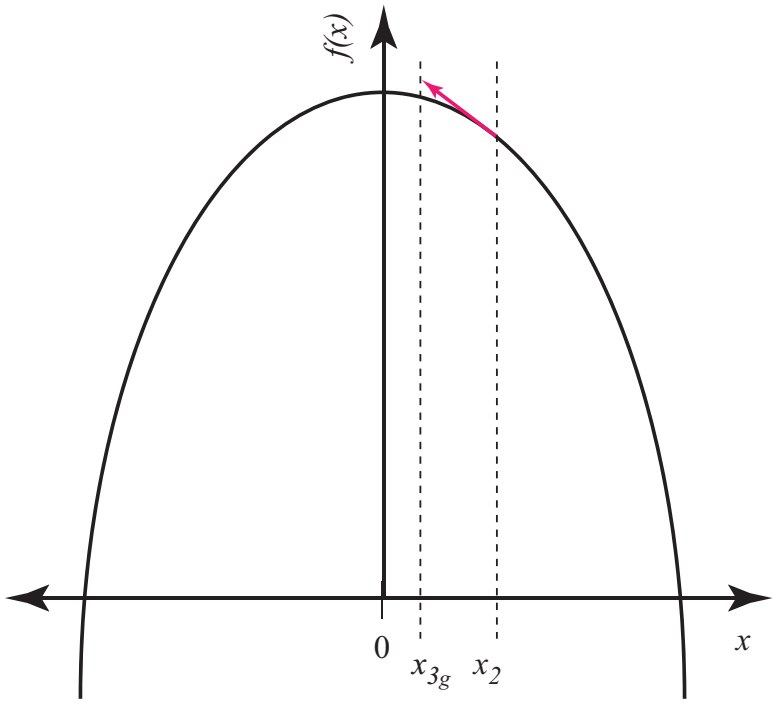


Figure 2.4. Gradient Search.

Gradient Search The gradient search uses the gradient, or first derivative, to find a candidate x which is closer to the maximum. The gradient gives the direction in which the function is most increasing, as shown in Fig. 2.4. If the initial guess is x_2, the next value to check is:

$$x_{3_g} \;=\; x_2 \;+\; s\;\nabla_x f(x_2) \tag{2.2}$$

where $\nabla_x f(x_2)$ is the gradient of $f(x)$ evaluated at x_2, and s is an arbitrary number which moves x_{3_g} away from the initial guess x_2 in the direction of the gradient. *The performance of the gradient approach is highly dependent on the choice of s.* The gradient at x_{3_g} is used to find x_{4_g} and so on [42].

2.3.2 Communication

Bracketing and Gradient techniques are used to formulate different cognitive systems below, which can best utilize the communication between the processes.

Marginal Cognitive System These bracketing and gradient search algorithms can be combined to form the marginally cognitive system shown in Fig. 2.5a. The searches run independently and a new `Compare` process returns the solution with the largest function value.

Performance Comparison of the independent solutions must wait until both algorithms have finished, and thus the search converges at the rate of the slowest one. It is possible to stop the search when the first search ends, but this is not effective since the other search might find a solution nearer the optimum.

Simple Cognitive System Fig. 2.5b shows the algorithm where in order to improve the performance, the communication in between can be increased by removing the optimality check from the individual searches. The bracketing and gradient searches are reduced to single iteration processes for simplification.

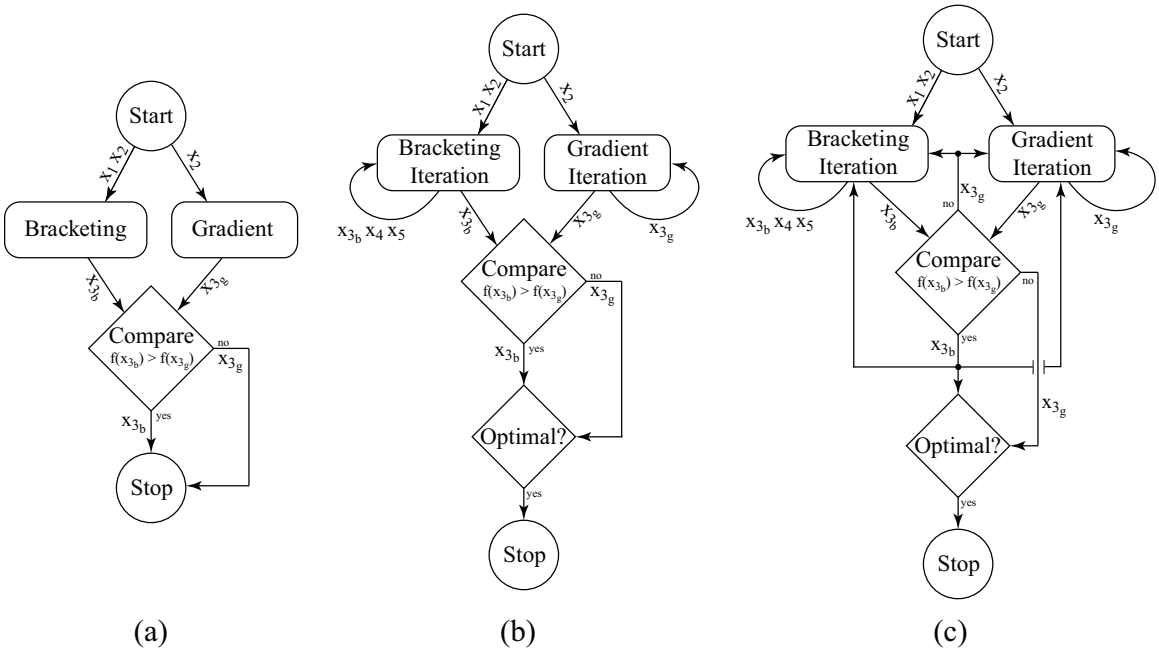


Figure 2.5. (a) Marginal, (b) Simple, and (c) Extensively communicative cognitive searches.

The variables x_4 and x_5 define the new bracket after each iteration. The result of each iteration must be fed back into the algorithm so that it may continue the search. In psychology these are referred to as re-entrant processes, because they accept their own output as input; re-entrant processes are an important characteristic of cognition[17]. The result of each iteration is compared and the largest value is checked for optimality. If the solution is optimal, the cognitive search terminates.

Performance This search converges at the rate of the fastest search algorithm, an improvement over Fig. 2.5a which converged at the rate of the slowest search algorithm.

Extensively Communicative Cognitive System In order to facilitate better communication between processes and improve the convergence rate, the algorithm is modified. This is simple for the gradient search which can begin its next iteration at the value provided by `Compare`. Since the concept behind the bracketing search is to reduce the bracket's size, the incoming information is also used to reduce the bracket. The details of this alteration are not discussed in detail, but the gist is that the bracketing process attempts to move the current bracket to center around the new best result. The bracketing processes in Fig. 2.5b and Fig. 2.5c are slightly different.

Performance The system in Fig. 2.5c can find the optimal solution in fewer iterations than either the gradient or bracketing searches can alone. When other performance measures are considered, such as the number of function evaluations and the run time, the cognitive approach lies in the middle.

2.3.3 Synergism or Self-Organization

Fig. 2.6 shows the originator of the value returned from `Compare` at each iteration of the cognitive search. It continually switches between bracketing and gradient searches, except at the label `Both` which indicates when the difference between x_{3_b} and x_{3_g} is small enough that $f(x_{3_b})$ and $f(x_{3_g})$ are numerically indistinguishable; however, the first derivatives, evaluated in `Optimal?`, are distinguishable. No discernible pattern to this switching exists, implying that the cognitive search automatically adjusts to converge to a solution. *This implies that the cognitive search is synergistic and self-organizes around the solution procedure that quickly finds the optimum.* All examples discussed were run on A DELL Precision T3400 with a 266GHz Dual Core processor and 3.25 GB of RAM with $S = 0.1$ and a optimality tolerance of

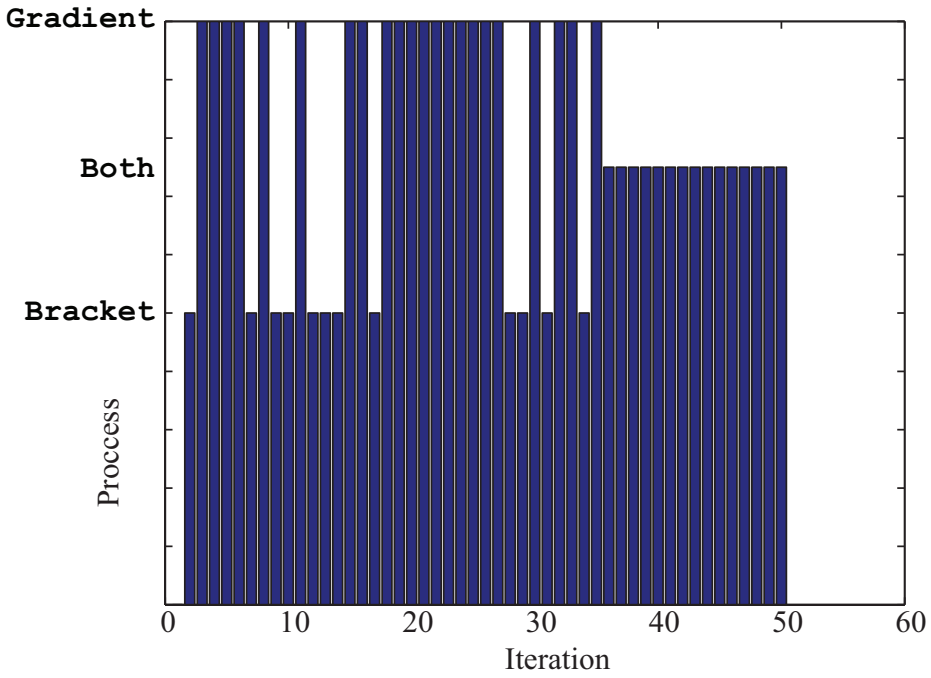


Figure 2.6. Process Originating the Value Returned by `Compare` at each Iteration.

2.22×10^{-14}. The optimizations in Table 2.1 started with initial values of $x_1 = -632$ and $x_2 = 700$. Since a simple, continuous function is used here, its derivative can be used to check optimality.

2.4 Increasing the Feasible Problem set

The cognitive search algorithm developed [15] could solve problems with multiple local maximas, discontinuous functions and also non-convex problems which are not addressed by standalone bracketing and gradient algorithms.The guidelines deduced from this work are listed in Table 2.2 below [15].

Table 2.1. Bracket, Gradient, and Cognitive Search Performance

Method	Steps	Function evals $f(x)$	Function evals $\frac{df(x)}{dx}$	time (sec)
Gradient	194	196	390	1.15×10^{-2}
Bracketing	62	64	64	4.71×10^{-3}
Cognitive	49	128	213	6.74×10^{-3}

Table 2.2. Guidelines for Cognitive System Development

	Guidelines
1	Processes should implement different methods to search for a solution
2	The component processes should be as simple and deterministic as possible
3	Processes should be formulated to fully utilize shared information
4	Processes should broadcast information as often as possible
5	Multiple messages should be passed and operated on simultaneously
6	Communication should be re-entrant
7	One or more processes must evaluate the system's progress

2.5 Limitations

The limitations of this work are as follows :

1. The nature of the processes in the preliminary work allowed the algorithm to be implemented sequentially.
2. Extensive use of the inherited knowledge, beyond the composition of processess was provided; specifically, the regions to search to find extrema is based on the knowledge of the function.

The search algorithm developed in this thesis addresses the limitations of the previous work.

CHAPTER 3

COGNITIVE SEARCH ALGORITHM

The search algorithm is developed from the general guidelines listed in Table 2.2 which are recognized as a part of preliminary work. The developed algorithm overcomes the limitations of the previous work and improves the system's performance further. Here, the goal is to improve the search algorithm by adding more aspects that mimic human cognition.

3.1 Proposed Cognitive Search Algorithm

The developed cognitive search algorithm is shown below in Fig. 3.1.

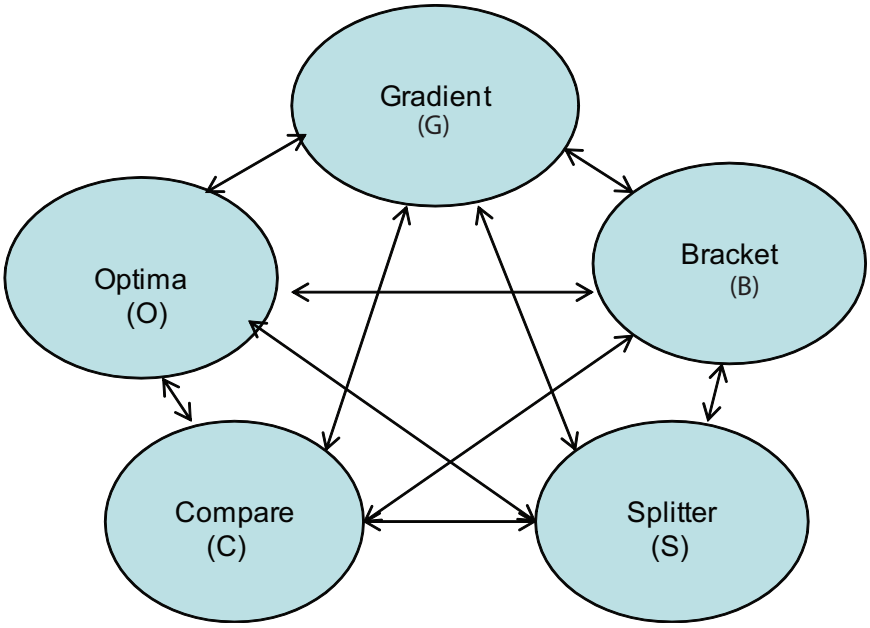


Figure 3.1. Flow chart for developed cognitive algorithm.

3.1.1 Non-sequential, Non-deterministic Implementation

New processes were added to the proposed algorithm and original ones were modified, as seen in Fig. 3.1. The processes can run asynchronously such that each process operates independently of other processes and in parallel. But due to time limitations, the implementation was made sequential and working made similar to parallel processing. This algorithm can address complex optimization functions and is more likely to locate global optima.

M is the formatted message string. B, C, G, O & S are Bracketing, Compare, Gradient, Optima & Splitter processes respectively.

```
        ⋮
while (chk==0)
  [S]=splitter(M);
  [B]=brac(M);
  [G]=grad (M);
  [C]= compare(M);
  [O,chk]= optima(M);

  M=[ S; B; G; C; O ];
end
        ⋮
```

(a)

```
        ⋮
while (chk==0)
  [G]=grad (M);
  [O,chk]= optima(M);
  [B]=brac(M);
  [C]= compare(M);
  [S]=splitter(M);

  M=[ G; O; B; C; S ];
end
        ⋮
```

(b)

Figure 3.2. Two Different Cognitive Optimization Matlab Pseudo-Codes, (a) SBGCO and (b) GOBCS process calling sequences.

Pseudo codes in Fig. 3.2a and 3.2b represent two different calling sequences of the processes. Irrespective of the arrangement of processes, the algorithm yields the same result.

Communication

- Processes communicate by passing messages. Since these messages do not follow any specific path, they can be combined and sequenced in numerous ways.
- Processes communicate extensively and use the information from other processes. This undirected communication results in a non-sequential search algorithm.

Messages

- In this work, the messages are formatted only for single variable functions. All messages have a standard format. M is the input message string.

$$\left\{ x_{guess},\ x_{lo},\ x_{up},\ f(x_{guess}),\ f(x_{lo}),\ f(x_{up}),\ \frac{df(x_{guess})}{dx},\ \frac{df(x_{lo})}{dx},\ \frac{df(x_{up})}{dx},\ tag \right\} \tag{3.1}$$

  where x_{guess} is the guess, x_{lo} and x_{up} are the upper and lower bounds for the bracket, $f(x_i)$ is the function value at x_i, and $\frac{df(x_i)}{dx}$ is the derivative/gradient at x_i. The numerical representation of messages' characteristic is tag:
  - $tag = 1$ repesents the messages containing extrema within x_{lo} and x_{up}.
  - $tag = 0$ represents the messages which may or may not contain extrema between x_{lo} and x_{up} .
  - $tag = 2$ represents the messages with highest function value calculated at x_{guess}.
- Every process acts on the initial message, x_{lo} and x_{up} that are user inputs and lists a set of output. All the processes take the message list, M in Fig. 3.2, as input, work on them and generate output messages. An output message is an altered form of the input which has been operated on by each process.

- The processes have knowledge of the part of the message to be acted upon. Every function/process may not utilize every element of a message. The incoming messages may be altered, used to produce new messages, or discarded.
- The output message list is the input for the next iteration. This continues until the extreme is achieved.

3.1.2 Processes' Functionality

This new type of search algorithm shown in Fig. 3.1 has 5 processes (considered inherited knowledge). These processes are simple, deterministic, and function differently.

1. **Bracketing & Gradient search** These are the traditional optimization search techniques discussed in Sec. 2.3.1.
2. **Splitter** This works on an interval halving method [42]. The discarded region from the bracketing search technique is the input for the splitter. The interval is successively divided into four equal length sections consisting of five points. These five points are the two boundary points and the points between them at the distance of one-fourth, two-fourths, and three-fourths between the boundary points respectively. At these five points, considering every two points as a region, if the width tolerance ≥ 0.01, the first derivatives are evaluated. For each region considered, if the first derivatives have opposite signs, they are tagged as tag=1 or otherwise tag=0.
3. **Compare** In this process, as the name suggests, the input messages in the list are compared . The process outputs a message with highest/largest function value.

4. **Optima** This process evaluates the system's performance. The messages sent are checked for the optimality tolerance and the algorithm terminates if the necessary condition is achieved.

3.1.3 Messages in each process

Each process works on a particular message in the list. These are indicated by *tags*.

Bracketing acts on messages with $tag = 1$ in the message list and outputs two messages with new brackets one, containing extrema, and the other having the discarded bracket, tagged 1 and 0, respectively.

Gradient works on messages with $tag = 2$ in the message list. The gradient value is calculated at the x_{guess} .

Splitter operates on messages with $tag = 0$ in the entire list and outputs four messages which are tagged 1 or 0 depending on the signs of their first derivatives.

Compare works only if there are two or more messages in the list. This process outputs a message having highest function value at x_{guess} which is tagged 2.

Optima acts as a stopping condition for the algorithm. This process checks for messages with $tag = 2$ and verifies if $\frac{df(x_guess)}{dx}$ has reached the tolerance value and terminates the algorithm.

Table 3.1 shows the message tags each function acts on.

Table 3.1. Message Processed

Function	Messages processed	No.of messages	output messages
Gradient	tag = 2	1 or Ø	tag = 1
Optima	tag = 2	1 or Ø	tag = 2
Compare	$No.of messages \geq 2$	1 or Ø	tag = 2
Bracketing	tag = 1	2 or Ø	tag = 1, tag= 0
Splitter	tag = 0	4 or Ø	tag = 1, tag = 0

3.2 Faster Convergence and Feasible Problem Set

The efficiency of the developed cognitive algorithm is evaluated by perfomance metrics as discussed in Sec. 1.4.

Convergence rate

In the developed cognitive algorithm, Fig. 3.1, all the processes are simplified to the point where the bracketing or gradient cannot solve the problem alone. The architecture allows all the processes to cooperate in finding the extremum. Each iteration of the cognitive search involves evaluation of all five: bracketing, gradient, splitter, compare, and optima processes. The number of iterations is considered as one of the performance metrics.

Feasibility

Optimization problems that are considered difficult for the bracketing and gradient searches are attemped by the developed search algorithm. The developed algorithm is less likely to get stuck in local maxima, that is it has a better chance of finding global maxima in all the cases.

The following are the reasons for the algorithm to locate global maxima in most of the cases :

- The **Splitter** process works on the discarded brackets from **Bracketing** and selects the best bracket if any. In this way, the algorithm checks the discarded brackets/regions in every iteration and it is likely that *most of the function domain is searched*.
- It is prefered to calculate **Gradient** only at the highest function value's x_{guess} which is returned by **Compare** on checking the entire message list, rather than on every single x_{guess} in the message list. This also reduces the overall run time.
- **Optima** operates on the message returned by **Compare** (highest function value's corresponding x_{guess} in the entire message list) and terminates the search if $\frac{df(x_{guess})}{dx}$ has reached optimality tolerance.

Thus, this search algorithm has increased the number of problems that can be solved beyond the intersection of the ones that the individual bracketing, gradient, and splitter (modified interval-halving) searches can address with most of the function domain searched and better rate of convergence. This algorithm is run on various cases and the results are discussed in the next chapter.

CHAPTER 4

RESULTS & DISCUSSIONS

4.1 Introduction

The developed cognitive search algorithm resulted in the desired cognitive behavior when a message passing structure was implemented. Cognitive search involves a large number of processes and more information to be processed, so it is expected to have longer runtime and function evaluations. Yet the cognitive approach is well suited for parallel and asynchronous operations, which can significantly reduce the overall runtimes. Thus, cognitive search is not used as a measure of performance. For this reason, the number of iterations is used as the performance metric. This algorithm allows more extensive search of the function domain and is likely to find the global or more optimal solution, in a reasonable number of iterations and the ability to address a broad set of dissimilar problems was also achieved. That is, the increase in the feasible problem set using inherited knowledge, communication, and self-organization are achieved.

In the following sections, the cognitive search algorithm is run on non-convex, multiple local maxima and discontinuous functions and the results of all cases are presented below.

4.2 Assumptions

For the cases considered,

- In Eq. 2.2, S=0.1 is an arbitrary guess taken for gradient algorithm.

- Optimality tolerance of 10^{-5} for continuous functions and 2^{-52} for discontinuous functions is assumed.
- $\emptyset$ indicates an empty element.
- The cases were run on A DELL Precision T3400 with a 2.49 GHz Dual Core processor with 3.25 GB of RAM using uncomplied Matlab.

4.3 Case 1(a) : Multiple Local Maxima function (MLM)

For a better understanding of the developed cognitive algorithm, a multiple local maxima function with **Splitter, Bracket, Gradient, Compare,** and **Optima** as the order of arrangement and with initial guesses $x_{lo}, x_{up} = (-5, 5)$ is shown below.

Properties : Continuous, Nonlinear, Unconstrained, Nonconvex, Multiple local Optima, Univariable.

Problem Statement :

$$\max_x f(x) = \frac{-1}{14}(x+4)(x+1)(x-1)(x-3) - 0.5$$

$$df(x)/dx = -(4x^3 + 3x^2 - 26x - 1)/14 \tag{4.1}$$

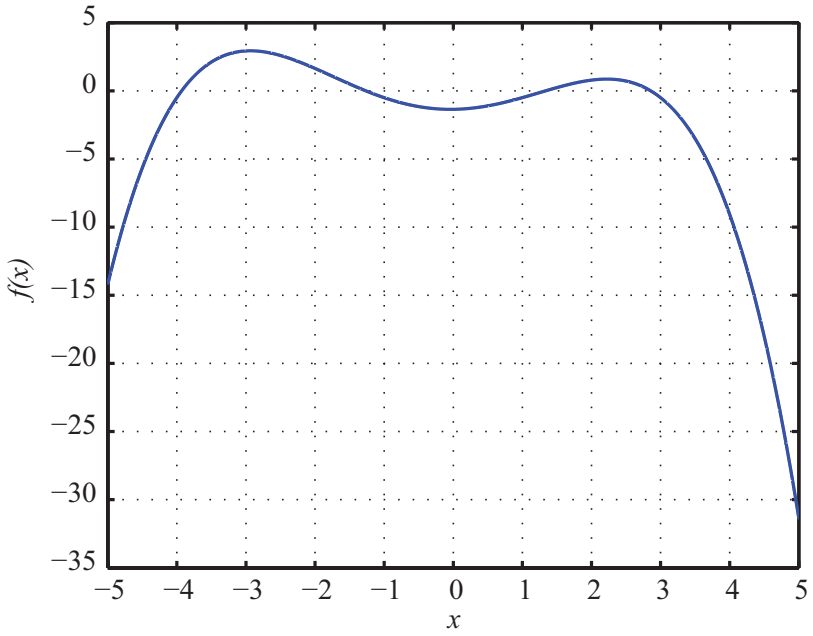


Figure 4.1. Multiple Local Maxima for case 1(a).

- This function has multiple local maxima at $(x, f(x)) = (2.22, 0.86), (-2.94, 2.94)$

- The optimality condition for the cognitive search is when the first derivative equals zero at this solution.

4.3.1 Messages in each iteration

The message list over 3 iterations is shown in Table 4.1.

- Iter 0 is the initial message string with the user defined inputs which is basic input to all the processes.
- Iter 1 is the output of the previous iteration and input to next iteration

  $R1$ & $R2$ are the messages from **bracketing** process
- In Iter 2,

  $R1$ - $R4$ are the messages from **splitter** process resulted from $R2$ of Iter 1

  $R5$ - $R6$ are the messages from **bracketing** process resulted from $R1$ of Iter 1

  $R7$ is the message from **compare** process
- In Iter 3 ,

  $R1$ - $R16$ are the messages from **splitter** resulted from $R1$, $R3$, $R4$, $R6$

  $R17$ - $R20$ are the messages from **bracketing** resulted from $R2$, $R5$

  $R21$ is the message from **gradient** resulted from $R6$

  $R22$ is the message from **compare** process

Table 4.1. Messages list for SBGCO

Iter	Row	$\left\{ x_{guess},\ x_{lo},\ x_{up},\ f(x_{guess}),\ f(x_{lo}),\ f(x_{up}),\ \frac{df(x_{guess})}{dx},\ \frac{df(x_{lo})}{dx},\ \frac{df(x_{up})}{dx},\ tag \right\}$									
0	$R1$	5.00	−5.00	5.00	−31.3	−14.2	−31.3	−31.7	21.1	−31.7	1
1	$R1$	0.00	0.00	5.00	−1.35	Ø	Ø	0.07	0.07	−31.7	1
	$R2$	Ø	−5.00	0.00	Ø	Ø	Ø	Ø	Ø	Ø	0
2	$R1$	Ø	−5.00	−3.75	Ø	Ø	Ø	Ø	21.1	5.16	0
	$R2$	−3.12	−3.75	−2.50	2.85	Ø	Ø	Ø	5.16	−1.44	1
	$R3$	Ø	−2.50	−1.25	Ø	Ø	Ø	Ø	−1.44	−2.02	0
	$R4$	Ø	−1.25	0.00	Ø	Ø	Ø	Ø	−2.02	0.07	0
	$R5$	2.50	0.00	2.50	0.71	Ø	Ø	−1.08	0.07	−1.08	1
	$R6$	Ø	2.50	5.00	Ø	Ø	Ø	Ø	Ø	Ø	0
	$R7$	0.00	0.00	5.00	−1.35	Ø	Ø	0.07	0.07	−31.7	2
3	$R1$	Ø	−5.00	−4.68	Ø	Ø	Ø	Ø	21.1	16.0	0
	$R2$	Ø	−4.68	−4.37	Ø	Ø	Ø	Ø	16.0	11.7	0
	$R3$	Ø	−4.37	−4.06	Ø	Ø	Ø	Ø	11.7	8.14	0
	$R4$	Ø	−4.06	−3.75	Ø	Ø	Ø	Ø	8.14	5.16	0
	$R5$	Ø	−2.50	−2.18	Ø	Ø	Ø	Ø	−1.44	−2.02	0
	$R6$	Ø	−2.18	−1.87	Ø	Ø	Ø	Ø	−2.02	−2.28	0
	$R7$	Ø	−1.87	−1.56	Ø	Ø	Ø	Ø	−2.28	−2.26	0
	$R8$	Ø	−1.56	−1.25	Ø	Ø	Ø	Ø	−2.26	−2.02	0
	$R9$	Ø	−1.25	−0.93	Ø	Ø	Ø	Ø	−2.02	−1.62	0
	$R10$	Ø	−0.93	−0.62	Ø	Ø	Ø	Ø	−1.62	−1.10	0
	$R11$	Ø	−0.62	−0.31	Ø	Ø	Ø	Ø	−1.10	−0.52	0
	$R12$	Ø	−0.31	0.00	Ø	Ø	Ø	Ø	−0.52	0.07	0
	$R13$	Ø	2.50	3.12	Ø	Ø	Ø	Ø	−1.08	−4.93	0
	$R14$	Ø	3.12	3.75	Ø	Ø	Ø	Ø	−4.93	−11.0	0
	$R15$	Ø	3.75	4.37	Ø	Ø	Ø	Ø	−11.0	−19.8	0
	$R16$	Ø	4.37	5.00	Ø	Ø	Ø	Ø	−19.8	−31.7	0
	$R17$	−3.12	−3.12	−2.50	2.85	Ø	Ø	0.89	0.89	−1.44	1
	$R18$	Ø	−3.75	−3.12	Ø	Ø	Ø	Ø	Ø	Ø	0
	$R19$	1.25	1.25	2.50	−0.13	Ø	Ø	1.50	1.50	−1.08	1
	$R20$	Ø	0.00	1.25	Ø	Ø	Ø	Ø	Ø	Ø	0
	$R21$	0.00	0.00	0.00	−1.35	0.00	0.00	0.08	0.00	0.00	1
	$R22$	−3.12	−3.75	−2.50	2.85	Ø	Ø	Ø	5.16	−1.44	2

4.3.2 Results of case 1(a)

Table 4.2 shows the individual performance of bracketing and gradient algorithms for the multiple local maxima function.

Table 4.2. Comparison of Bracket and Gradient Algorithms for case 1(a)

Initial bracket	Method	Solution $x,\ f(x)$	Iters	No.eval $f(x)$	No. eval $\frac{df(x)}{dx}$	Run time (sec)
$(-5, 5)$	Bracketing	$2.2237, 0.8613$	16	18	18	0.00261
	Gradient	$2.2237, 0.8613$	32	34	66	0.00426
$(-10, 10)$	Bracketing	$2.2237, 0.8613$	17	19	19	0.00131
	Gradient	$2.2237, 0.8613$	33	35	68	0.00214
$(-20, 10)$	Bracketing	$-2.9354, 2.9377$	22	24	24	0.00169
	Gradient	$2.2237, 0.8613$	33	35	68	0.00227
$(-100, 100)$	Bracketing	$2.2237, 0.8613$	25	27	27	0.00200
	Gradient	$2.2237, 0.8613$	51	53	104	0.00343

The following Table 4.3 shows the results for case 1(a) with different arrangements of processes where S-Splitter, B-Bracket, G-Gradient, C-Compare, O-Optima.

Table 4.3. MLM function with different arrangements for case 1(a)

Arrangement	Tolerance	Extrema
SBGCO	0.00001	-2.9353
BGOSC	0.00001	-2.9353
GOBCS	0.00001	-2.9353
CSGBO	0.00001	-2.9353
OCBSG	0.00001	-2.9353

The following Table 4.4 shows the results for case 1(a) with different initial values. In all cases the global optimum was found.

Table 4.4. MLM function with different initial values for case 1(a)

Initial bracket	Iters	Arrangement	No.of messages	No.eval $f(x)$	No. eval $\frac{df(x)}{dx}$	Run time (sec)
$(-5, 5)$	20	SBGCO	2991	234	5764	0.10843
	20	BGOSC	2775	234	5332	0.10821
	20	GOBCS	2283	78	4426	0.11567
	20	CSGBO	2283	81	4433	0.10644
	20	OCBSG	2775	213	5273	0.10875
$(-10, 10)$	21	SBGCO	5293	256	10346	0.20795
	21	BGOSC	4737	256	9234	0.18268
	21	GOBCS	3995	82	7842	0.14651
	21	CSGBO	3995	85	7849	0.15481
	21	OCBSG	4737	234	9172	0.17526
$(-20, 10)$	23	SBGCO	11395	301	22505	0.65993
	23	BGOSC	11179	301	22073	0.62084
	23	GOBCS	9127	88	18092	0.55028
	23	CSGBO	9127	110	18588	0.54835
	23	OCBSG	10839	277	21325	0.60602
$(-100, 100)$	20	SBGCO	75719	323	151222	34.432
	20	BGOSC	72007	232	143798	33.768
	20	GOBCS	58591	76	117044	26.835
	20	CSGBO	59177	95	118211	28.421
	20	OCBSG	68939	211	137603	32.407

4.4 Case 1(b) : Multiple Local Maxima function (MLM)

Properties : Continuous, Nonlinear, Unconstrained, Nonconvex, Multiple local Optima, Univariable

Problem Statement :

$$\begin{aligned} \max_x \ f(x) &= x(x+10)(x+5)(x-10)(x-5)^2 \\ df(x)/dx &= -6x^5 + 25x^4 + 500x^3 - 1875x^2 - 5000x + 12500 \end{aligned} \tag{4.2}$$

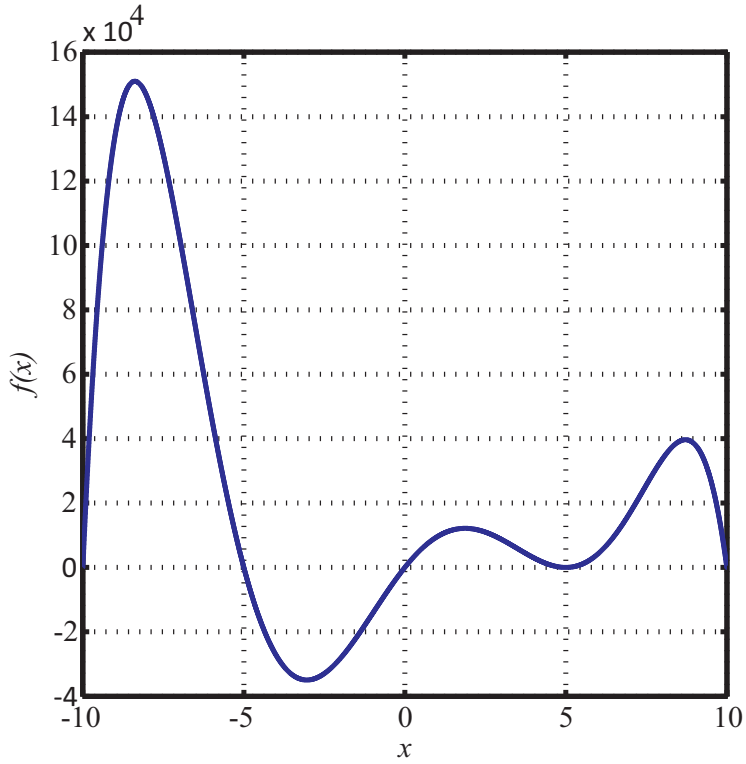


Figure 4.2. Multiple Local Maxima for case 1(b).

Multiple local maxima's

$$(x, f(x)) = (-8.3836, 150985.3) \quad (1.8748, 12145.9) \quad (8.7166, 39671.0)$$

- The optimality condition for the cognitive search is when the first derivative equals zero at this solution.

4.4.1 Results of case 1(b)

Table 4.5 shows the individual performance of bracketing and gradient algorithms for multiple local maxima function. The following Table 4.6 shows the

Table 4.5. Comparison of Bracket and Gradient Algorithms for case 1(b)

Initial bracket	Method	Solution $x,\ f(x)$	Iters	No.eval $f(x)$	No. eval $\frac{df(x)}{dx}$	Run time (sec)
$(-10, 10)$	Bracketing	$5, 0$	2	4	4	0.00701
	Gradient	$5, 0$	1	3	4	0.00180
$(-50, 50)$	Bracketing	$8.7166, 39671$	25	27	27	0.00189
	Gradient	$5, 0$	9	11	20	0.00072
$(-100, 100)$	Bracketing	$8.7166, 39671$	26	28	28	0.00195
	Gradient	$5, 0$	19	21	40	0.00135

results for case 1(b) with different arrangements of processes where S-Splitter, B-Bracket, G-Gradient, C-Compare, O-Optima.

Table 4.6. MLM function with different arrangements for case 1(b)

Arrangement	Tolerance	Extrema
BOSGC	0.00001	-8.3836
GBSOC	0.00001	-8.3836
CBSOG	0.00001	-8.3836
OGBCS	0.00001	-8.3836
SOBCG	0.00001	-8.3836

The following Table 4.7 shows the results for case 1(b) with different initial values. The global solution is found in all cases.

Table 4.7. MLM function with different initial values for case 1(b)

Initial bracket	Iters	Arrangement	No.of messages	No.eval $f(x)$	No. eval $\frac{df(x)}{dx}$	Run time (sec)
$(-10, 10)$	35	BOSCG	3209	634	5792	0.13936
	35	GBSOC	1781	107	3337	0.14844
	35	CBSOG	3075	570	4953	0.15524
	35	OGBCS	1781	104	3332	0.09955
	35	SOBCG	3209	600	5692	0.16565
$(-50, 50)$	40	BOSCG	40287	899	79695	18.524
	40	GBSOC	36511	197	72699	17.447
	40	CBSOG	40205	468	79800	18.472
	40	OGBCS	36511	192	72688	17.487
	40	SOBCG	45901	894	90768	19.993
$(-100, 100)$	41	BOSCG	65837	941	130753	40.039
	41	GBSOC	59727	201	119123	37.254
	41	CBSOG	65753	259	134365	39.667
	41	OGBCS	59727	196	119112	36.711
	41	SOBCG	78561	971	156009	47.816

Performance

To determine the general behavior of the developed algorithm, many trial cases were run where initial guesses with large ranges of width were considered.

1. Results from Table 4.2 and Table 4.4 show that for the considered multiple maxima function, the developed cognitive system converges to a solution in fewer iterations than the individual performances of bracketing and gradient algorithms for the case of higher bracket width and lies in the middle with lower bracket widths.

2. Table 4.3 and Table 4.6 show the self-organization property of cognition in converging to a better solution. Also, irrespective of the arrangement of the processes, non-sequential behavior of the processes is implied.
3. Though the run time for the cognitive search algorithm is longer, it converges to a global solution in all the cases whereas the bracketing and gradient searches fail.

Conclusions

1. Table 4.2 and Table 4.5 show that the individual algorithms fail to find the global maximum for the given initial guesses in most of the cases.
2. Table 4.3 and Table 4.6 conclude that irrespective of the arrangement of the processes, for the assumed tolerance value, the algorithm finds a global solution.
3. The results in Table 4.4 and Table 4.7 show that for different arrangements, there is a slight deviation in run time and also with the increase in the size of the bracket, the run time increases as:
    - The regions to search are more when compared to lower bracket widths.
    - The number of function evaluations are more than in standalone algorithms.
    - The number of messages to operate on is multiplied in the splitter process geometrically.

4.5 Case 2 : Discontinuous function

Properties : Discontinuous, Nonlinear, Unconstrained, Nonconvex, Local Optima, Univariable.

Problem Statement :

$$\begin{aligned} f(x) &= \frac{2}{x^2 - x} \\ df(x)/dx &= -2(2x-1)/(x^2-x)^2 \end{aligned} \tag{4.3}$$

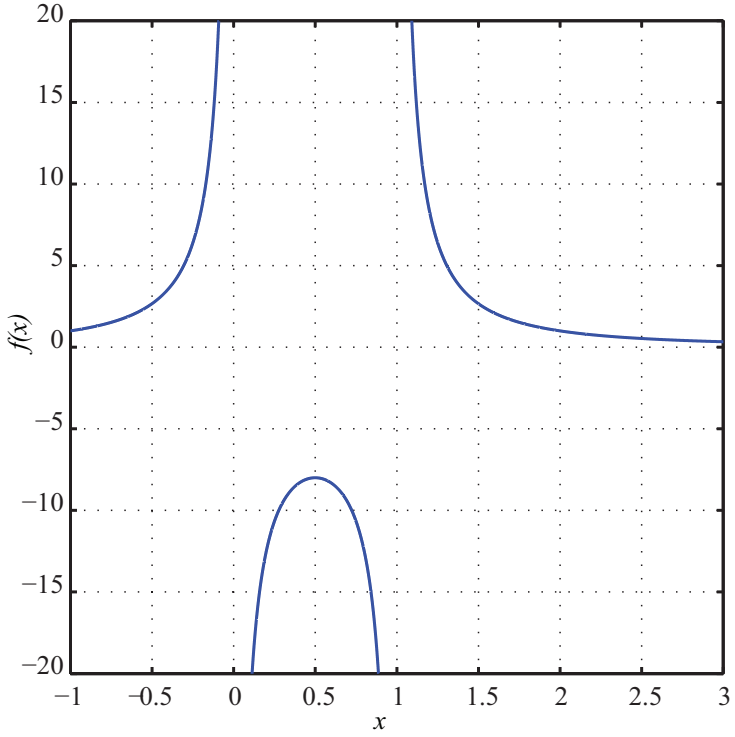


Figure 4.3. Discontinuous function.

The extremum is at $(x, f(x)) = (0.5, -8)$

4.5.1 Results of case 2

Table 4.8 shows the individual performance of bracketing and gradient algorithms for discontinuous function. The gradient is blank because it never converged to a solution.

Table 4.8. Comparison of Bracket and Gradient Algorithms

Initial bracket	Method	Solution $x,\ f(x)$	Iters	No.eval $f(x)$	No. eval $\frac{df(x)}{dx}$	Run time (sec)
$(-1, 3)$	Bracketing	$0.5, -8$	3	5	5	0.00726
	Gradient					
$(-10, 10)$	Bracketing	$0.5, -8$	55	57	57	0.01064
	Gradient					
$(-100, 100)$	Bracketing	$0.5, -8$	58	60	60	0.00200
	Gradient					

The following Table 4.9 shows the results of a discontinuous function with different arrangement of processes where S-Splitter, B-Bracket, G-Gradient, C-Compare, O-Optima.

Table 4.9. Discontinuous function with different arrangements

Arrangement	Tolerance	Extrema
BOSCG	eps	0.5
SBGCO	eps	0.5
GCBOS	eps	0.5
CGOSB	eps	0.5
OCBSG	eps	0.5

The following Table 4.10 shows the results for a discontinuous function with different initial values.

Table 4.10. Discontinuous function with different initial values

Initial bracket	Iters	Arrangement	No.of messages	No.eval $f(x)$	No. eval $\frac{df(x)}{dx}$	Run time (sec)
$(-1, 3)$	3	BOSCG	33	8	60	0.0153
	3	SBGCO	33	9	61	0.0154
	3	GCBOS	33	9	61	0.0157
	3	CGOSB	33	8	60	0.0156
	3	OCBSG	33	7	59	0.0151
$(-5, 5)$	56	BOSCG	5779	1545	9699	0.1904
	56	SBGCO	5779	1546	9700	0.1897
	56	GCBOS	2587	168	4798	0.1069
	56	CGOSB	2587	167	4797	0.1095
	56	OCBSG	5779	1544	9698	0.1938
$(-10, 10)$	57	BOSCG	7811	1601	13701	0.2811
	57	SBGCO	7811	1602	13702	0.2532
	57	GCBOS	3957	171	7531	0.1448
	57	CGOSB	3957	170	7530	0.1386
	57	OCBSG	7811	1600	13700	0.2615
$(-100, 100)$	61	BOSCG	78751	1835	155323	434.228
	61	SBGCO	78751	1836	155324	33.968
	61	GCBOS	58589	183	116767	26.449
	61	CGOSB	58589	182	116766	27.149
	61	OCBSG	78751	1834	155322	33.597

Performance

1. From Table 4.8 and Table 4.10, it is seen that though the number of iterations of the cognitive search algorithm is greater than or equal to that of bracketing, the former retains the self-organization property.

2. Table 4.9 shows that, irrespective of the arrangement of the processes, the algorithm converges to a global solution implying the non-sequential behavior of the processes.

Conclusions

1. Table 4.8 shows that the gradient algorithm completely fails as it repeatedly travels up to the asymptotes of the function.
2. The results in Table 4.9 conclude that irrespective of the arrangement of the processes, for the assumed tolerance value the algorithm guarantees a global solution.
3. The results in Table 4.10 show that for different arrangements, there is a slight deviation in run time and also with the increase in the size of the bracket, the run time increases because the region to search is more when compared to lower bracket widths.

4.6 Summary

Thus, these results above show the desired performance in terms of a feasible problem set and rate of convergence as discussed in Sec. 1.4 and Sec. 3.2. The development of the cognitive algorithm exhibits characteristics similar to human cognition in structure and performance.

4.7 Guidelines

The guidelines deduced from the work done are :

- Processes do not work on every single message sent from other process.

  Processes have knowledge of the message to be acted upon. Rather than calculating **Gradient** at every single x_{guess} in the message list, it is calculated only at the highest function value's x_{guess}. This inhibits unnecessary messages in the list which reduced the over all run time drastically.

CHAPTER 5

CONCLUSIONS

In this paper, a cognitive algorithm is developed in the context of function optimization which mimics human cognition in its structure and performance. This is important because optimization has been identified as one of the key functions of cognition. The algorithm has the prior knowledge of how to solve problems from traditional optimization techniques. All the processes are simplified to the point where none can solve the problem alone. Each process included is significant and deterministic. These processes use inherited knowledge to communicate to get to a solution in a form of formatted message lists. The distribution of the messages to all the processes provided the synergistic approach to the algorithm. This message-passing approach leads to the non-sequential form of the processes. The feasibility of the problem set that the algorithm can address is achieved. The work provided resulted in an output with no randomness involved. It was shown that the developed cognitive algorithm is capable of addressing difficult optimization problems that are not addressed by traditional optimization techniques. This search algorithm located global solutions for non-convex multiple local maxima functions.

APPENDIX A

MATLAB CODE

A.1 Matlab code for MLM function case1(a)

```
function [opti,value,tend,iter]= cosgb(guess1,guess2)
tic;iter=0;
global B G S P O M Mo;
B=[];G=[];S=[];O=[];M=[];P=[];Mo=[];
format short g
xlo=guess1;fxlo= feval(guess1);dfxlo=dfeval(guess1);
xup=guess2;fxup=feval(guess2); dfxup=dfeval(guess2);
xguess=xup; % xguess should be either up or low
fxguess=feval(xguess); dfxguess=dfeval(xguess);
if (dfxlo>=0 && dfxup<=0)
tag=1;
else
tag=0;
end
M= [xguess,xlo,xup,fxguess,fxlo,fxup,dfxguess,dfxlo,dfxup,tag]
chk=0;
while (chk==0)
Mo=[];
[S]=spliter(M);
[O,chk]= optima(M);
if chk ~= 0
opti =O(1)
value=O(4)
break
```

```
end
[B]=brac(M);
[C]= compare(M);
[G]=grad (M);
Mo
iter=iter+1;
M=Mo;
end
tend=toc
iter
return

function fx = feval(x)
fx = -1/14*(x+4)*(x+1)*(x-1)*(x-3)-0.5; % function with 2 maxima's
%fx=2/(x^2-x); % discontinuous function
% fx=-x*(x+10)*(x+5)*(x-10)*(x-5)^2; % 3 maxima's function
return;
function dfx = dfeval(x)
dfx = (1/14)*( 1 + 26*x - 3*x^2 - 4*x^3);
%dfx=(-4*x+2)/(x^2-x)^2;
%dfx=-6*x^5+25*x^4+500*x^3-1875*x^2-5000*x+12500;
return;
function [O,chk]=optima(M)
global Mo;
O=NaN(1,10);
```

```
for i=1:size(M,1)
If (M(i,10)==2)
h=abs(M(i,7));
if (h<=0.00001)
[O]= M(i,:) ; chk=1;
A= (size(Mo,1));
Mo(A+1,:)=O(1,:);
break;
else
O=[]; chk=0;
end
else
O=[]; chk=0;
end
end
return

function [C]= compare(M)
global Mo;
C=NaN(1,10);
if (size(M,1)>=2)
[funcC, I ]=max(M(:,4));
[C]= M(I,:);
C(10)=2;
A= (size(Mo,1));
```

```
Mo(A+1,:)=C(1,:);
else
C=[];
end
return

function [B] = brac(M)
global Mo;
B1=NaN(1,10); B2=NaN(1,10);
for i=1:size(M,1)
if (M(i,10)==1)
B1(1)=(M(i,2)+M(i,3))*0.5;
B1(4)=feval(B1(1)); % func val at guess
B1(7)=dfeval(B1(1)); % diff @ xguess
if (sign(B1(7))==sign(M(i,8)))
B1(2)=B1(1); B1(3)=M(i,3); B1(10)=1;
B1(8)=dfeval(B1(2));B1(9)=dfeval(B1(3));
B2(2)=M(i,2); B2(3)=B1(1);B2(10)=0;
elseif (sign(B1(7))==sign(M(i,9)))
B1(2)=M(i,2);B1(3)= B1(1);B1(10)=1;
B1(8)=dfeval(B1(2));B1(9)=dfeval(B1(3));
B2(2)=B1(1);B2(3)=M(i,3);B2(10)=0;
else
end
B=[B1;B2];
```

```
A= (size(Mo,1));
Mo(A+1,:)=B(1,:); Mo(A+2,:)=B(2,:);
else
B=[];
end
end
return

function [G]= grad(M)
global Mo;
G=NaN(1,10);
for i=1:size(M,1)
if (M(i,10)==2)
G(1)=M(i,1)+0.1*M(i,7); G(4)=feval(G(1)); G(7)=dfeval(G(1)); G(10)= 1;
A= (size(Mo,1));
Mo(A+1,:)=G(1,:);
else
G=[];
end
end
return

function [S]=spliter(M)
global Mo;
S1=NaN(1,10);S2=NaN(1,10);S3=NaN(1,10); S4=NaN(1,10);
```

```
for i=1:size(M,1)
if ((M(i,10)==0)&& (M(i,3)-M(i,2))>=0.01)
P=(M(i,:));
m=(P(3)+P(2))*0.5; L= P(3)-P(2);
a=P(2)+0.25*L; b=P(3)-0.25*L;
S1(2)=P(2);S1(3)=a;S1(8)=dfeval(S1(2));S1(9)=dfeval(S1(3));
if (S1(8)>0 && S1(9)<0) S1(10)= 1;
S1(1)= (S1(2)+S1(3))*0.5; S1(4)=feval(S1(1));S1(7)=dfeval(S1(1));
else
S1(10)=0;
end
S2(2)=a; S2(3)=m; S2(8)=dfeval(S2(2));S2(9)=dfeval(S2(3));
if (S2(8)>0 && S2(9)<0)S2(10)= 1;
S2(1)= (S2(2)+S2(3))*0.5; S2(4)=feval(S2(1));S2(7)=dfeval(S2(1));
else
S2(10)=0;
end
S3(2)=m; S3(3)=b; S3(8)=dfeval(S3(2));S3(9)=dfeval(S3(3));
if (S3(8)>0 && S3(9)<0)
S3(10)= 1; S3(1)= (S3(2)+S3(3))*0.5;
  S3(4)=feval(S3(1));S3(7)=dfeval(S3(1));
else
S3(10)=0;
end
S4(2)=b;S4(3)=P(3); S4(8)=dfeval(S4(2));S4(9)=dfeval(S4(3));
```

```
if (S4(8)>0 && S4(9)<0)
S4(10)= 1; S4(1)= (S4(2)+S4(3))*0.5;
S4(4)=feval(S4(1));S4(7)=dfeval(S4(1));
else
S4(10)=0;
end
S=[S1;S2;S3;S4];
A= (size(Mo,1));
Mo(A+1,:)=S(1,:); Mo(A+2,:)=S(2,:);
Mo(A+3,:)=S(3,:) ; Mo(A+4,:)=S(4,:);
else
S=[];
end
end
return
```

REFERENCES

[1] G. Renner and a. Ekart, "Genetic algorithms in computer aided design," *CAD Computer Aided Design*, vol. 35, pp. 709–726, 2003.

[2] B. C, "Ant colony optimization : Introduction and hybridizations," in *Proceedings of the International Conference on Hybrid Intelligent Systems(HIS)*, Sep 2007, pp. 24–29.

[3] M. W. Matlin, *Cognition*. John Wiley & sons, 2008.

[4] S. K. Reed, *Cognition theory and applications*. Thomson Learning, 2007.

[5] A. F. Ashman and R. N.F.Conway, *An Introduction to Cognitive education: theory and applications*. London;New York : Routledge, 2002.

[6] R. Pfeifer, "Cognition-perspectives from autonomous agents," *Robotics and Autonomous Systems*, vol. 15, pp. 47–70, 1995.

[7] D. McFarland, "Defining motivation and cognition in animals," *International studies in the philosophy of science*, vol. 5, pp. 153–170, 1991, p.160.

[8] U.Neisser, *Cognitive Psychology*. Appleton-Century Crofts,NewYork, 1967.

[9] M.H.Ashcroft, *Human memory and Cognition*, 2nd ed. Harper Collins College Publishers, 1967.

[10] R.L.Gregory, *The Oxford Companion to the mind*. Oxford University Press,Oxford, 1987.

[11] J. Youngdo and A. Younghwa, "Intelligent system modeling for human cognition," in *5 th International conference on Convergence and Hybrid Information Technology,ICHIT*, vol. 206 CCIS, 2011, pp. 529–536.

[12] J. Neumann and O. Morgenstern, *The theory of Games and Economic behavior*. Princeton NJ : Princeton U.Press, 1953.

[13] P. J. Werbos, "Using adp to understand and replicate brain intelligence: the next level design," in *Proceedings of IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learing*, 2007.

[14] J. R. Busemeyer and A. Diederich, *Cognitive Modeling*. SAGE publications, 2010.

[15] A. Bowling and F. Makedon, "Cognitive optimization in asistive living system development," in *proceedings of Applied Bionics and Biomechanics*, 2010.

[16] P. J. Werbos, "Intelligence in the brain:a theory of how it works and how to build it," *Neural Networks*, pp. 200–212, 2009.

[17] T. E., *In:The Dynamical Systems Approach to Cognition*. World Scientific Publishing Co.., 2003, vol. 10.

[18] T. W and D. JP, *The Dynamical Systems Approach to Cognition (Studies of Nonlinear Phenomena in Life Science)*. World Scientific, 2003, vol. 10.

[19] K. Deb, *Optimization for Engineering Design: Algorithms and Examples*. Prentice-Hall of India Private Limited, 2005.

[20] A. Eiben, "Genetic algorithms with multi-parent recombination," in *Proceedings of the International Conference on Evolutionary Computation*, 1994, pp. 78–87.

[21] C.-K. Ting, "On the mean convergence time of multi-parent genetic algorithms without selection," *Advances in Artificial Life*, pp. 403–412, 2005.

[22] G. DE, *Genetic Algorithms in search,Optimization & Machine Learning*. Addison-Wesley Publishing Company,Inc, 1989.

[23] G. W, "Study on immunized ant colony optimization," in *Proceedings of the International Conference on Natural Computation (ICNC)*, August 2007, pp. 792–796.

[24] D. M and B. C, "Ant colony optimization theory: A survey," *Theoritical Computer Science*, vol. 344, pp. 243–278, 2005.

[25] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proceedings of ECAL91-European Conference On Artifical Life*, 1991, pp. 134–142.

[26] M. Dorigo, "Ant colony optimization," *Scholarpedia*, vol. 2, no. 3, p. 1461, 2007.

[27] V. Maniezzo, "Ant colony optimization," in *Proceedings of the Third Metaheuristics International Conference*, 1999, pp. 299–303.

[28] B. M. Dorigo.M and S. .T, "Ant colony optimization," in *Proceedings of the IEEE computational Intelligence Magazine*, ser. 4, vol. 1, 2006, pp. 28–39.

[29] J. Kennedy and R.C.Eberhart, "Particle swarm optimization," in *Proceedings of the International Conference on Neural Networks*, 1995, pp. 1942–1948.

[30] C. Zeng, "A particle swarm optimization algorithm with rich social cognition," in *Natural Computation*, 2009.

[31] X. Hu, Y. Shi, and R. Eberhart, "Recent advances in particle swarm," in *In the proceedings of IEEE congress on Evolutionary Computation*, 2004.

[32] M. Dorigo, M. A. M. D. Oca, and A. Engelbrecht, "Particle swarm optimization," *Scholarpedia*, vol. 3, no. 11, p. 1486, 2008.

[33] J. F. Schutte, "The particle swarm optimization algorithm," ufl, Tech. Rep., 2005.

[34] M.-P. Song and G. C. Gu, "Particle swarm optimization," in *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, August 2004, pp. 26–29.

[35] X.-F. Xie, W.-J. Zhang, and Z.-L. Yang, "Social cognitive optimization for nonlinear programming problem," in *Proceedings of the first International Conference on Machine Learning and Cybernetics*, November 2000, pp. 4–5.

[36] L. Ma, R. xi Wang, and Y. ping Chen, "The social cognitive optimization algorithm: Modifiability and application," 2010, appeared in ICEEE.

[37] X.-F. Xie and W.-J. Zhang, "Solving engineering design problems by social cognitive optimization," in *Genetic and Evolutionary Computation Conference (GECCO)*, 2004, pp. 261–262.

[38] S.Kirkpatrick, C.D.Gelatt, and M.P.Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[39] P. Salamon, P. sibani, and R. Frost, *Facts,Conjectures,and improvements for simulated annealing*. Society for industrial and Applied Mathematics, 2002.

[40] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Statistical Science*, vol. 8, pp. 10–15, 1993.

[41] R. A. Brooks, "Intelligence without representation," *Artificial Intelligence*, vol. 47, pp. 139–159, 1991.

[42] H. Lieberman, *Introduction to Operations Research*. The McGraw-Hill Companies, 2001.

BIOGRAPHICAL STATEMENT

Parvati Aruna Kandala was born in Hyderabad, India on 8 th August 1988. She received her Bachelor's degree in Mechanical Engineering in the year 2009 from Jawaharlal Nehru Technological University, Hyderabad, India. Her interests in Design and Robotics made her choose Mechanical Engineering for her Master's in Fall-2009 at the University of Texas at Arlington, Texas. Her Master's degree in Mechancial Engineering from The University of Texas at Arlington in 2012 has fulfilled her wish in attaining a higher degree in advanced technology and engineering.