# Solving Optimization Problem using Compact Genetic algorithm

**Neelu Sahu, Prateek kumar Singh**

*Abstract*—**An optimization problem is the problem of finding the *best* solution from all feasible solutions. Optimization problems can be divided into two categories depending on whether the variables are continuous or discrete. An optimization problem with discrete variables is known as a combinatorial optimization problem. In this paper we introduce the compact genetic algorithm (CGA) to solve an optimization problem. The cGA represent the population as a probability distribution over the set of solution. These paper show only research work for solving optimization problem using compact genetic algorithm.**

***Index Terms*—compact genetic algorithm, optimization problem.**

## I. INTRODUCTION

An optimization problem is the problem of finding the *best* solution from all feasible solutions. Optimization problems can be divided into two categories depending on whether the variables are continuous or discrete. An optimization problem with discrete variables is known as a combinatorial optimization problem. In a combinatorial optimization problem, we are looking for an object such as an integer, permutation or graph from a finite set. Optimization problem is handled by heuristic methods which provide reasonable solution of the problem [6]. Many of the techniques for this problem, such as Abraham et al and Braun et al presented three basic heuristics implied by Nature for namely Genetic Algorithm (GA), Simulated Annealing (SA) and Tabu Search (TS), and heuristics derived by a combination of their three algorithms. GA and SA are powerful stochastic optimization methods, which are inspired form the nature. This paper starts with the description of various compact genetic algorithm and GA operators in
Section 2. Section 3 gives the outline of the compact genetic algorithm. In section 4, we introduce optimization and
Discuss how compact genetic algorithm can be used to achieve optimization. The discussion ends with a conclusion and future trend.

## II. GENETIC ALGORITHM

GA is an efficient searching tool that was invented by John Holland [19].The genetic algorithm has great application for Optimization of complicated problems particularly in where Isn't adequate information about search space? Although, considering that, genetic algorithm isn't guarantee best possible solution, but normally, would provide optimum or partly optimum solution by suitable approximate at short time. For solving any problem by genetic algorithm, eight components must be defined :

**Representation (definition of individual):**
Represents each chromosome in the real world. A chromosome is a set of parameters which define a proposed solution to the problem that the genetic algorithm is trying to solve.

**Fitness function:**
These function shows the fitness of each chromosome. It is used to evaluate the chromosome and also controls the genetic operators.

**Population:**
The role of the population is to hold possible solution.

**Parent selection mechanism:**
The role of parent selection is to distinguish among individuals based on their quality, in particular, to allow the better individuals to become parents of the next generation.

**Reproduction:**
The reproduction operator is based on the Darwinian notion of "survival of the fittest". Individuals taking part in successive generations are obtained through a reproduction process or evolution operation. Individual strings are copied into a mating pool according to their respective fitness values. The higher the fitness values of the strings, the higher the probability of contributing one or more offspring in the next generation.

**Crossover operators:**
Recombination operator selects two or more chromosomes and then produces two new children from them. It aims at mixing up genetic information coming from different chromosomes to make a new individual.

**Mutation operators:**
Mutation operator selects one chromosome and then produces one new child from it by a slight change over the parent.

**Survivor selection mechanism:**
The role of survivor selection is to distinguish among individuals based on their quality. This mechanism survives the individual among the passing from one generation to the next generation.

**Termination Condition:**
The condition to ending the running of genetic algorithm.

## III. OPTIMIZATION PROBLEM

An optimization problem is the problem of finding the *best* solution from all feasible solutions. Optimization

***Manuscript received Aug, 2014*.**
***Prateek kumar Singh**, MTech (CSE), Department of Computer Science and Engineering, RGPV/LNCT Jabalpur,India*
***Neelu Sahu**,,M.E (C.T.A)Department of Computer Science and Engineering, SSCET/ CSVTU, Bhilai, India*

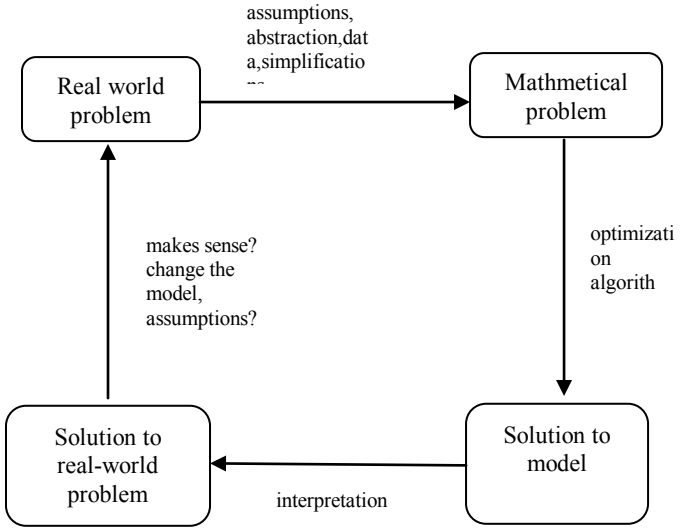


ISSN: 2278 – 7798 

problems can be divided into two categories depending on whether the variables are continuous or discrete. An optimization problem with discrete variables is known as a combinatorial optimization problem. In a combinatorial optimization problem, we are looking for an object such as an integer, permutation or graph from a finite (or possibly countable infinite) set.

- **Continuous optimization problem**
- **Combinatorial optimization problem**

**A schematic view of modeling/optimization process**




**What is a model:**

Model: A schematic description of a system, theory, or phenomenon that accounts for its known or inferred properties and maybe used for further study of its characteristics.

Mathematical models: are abstract models describe the mathematical relationships among elements in a system.

**What is discreet optimization?**

Discrete Optimization is a field of applied mathematics, combining techniques from combinatory and graph theory, linear programming, theory of algorithms, to solve optimization problems over discrete structures.

## IV. COMPACT GENETIC ALGORITHM

The Compact Genetic Algorithm (CGA) proposed by Harik, Lobo and Goldberg represents the population as a probability distribution over the set of solutions; thus, the whole population needs not to be stored. At each generation, CGA samples individuals according to the probabilities specified in the probability vector. The individuals are evaluated and the probability vector is updated towards the better individual. The CGA mimics the order-one behavior of Simple Genetic Algorithm (SGA) with uniform crossover using a small amount of memory, and achieves comparable quality with approximately the same number of fitness evaluations as the SGA. First step, the probability vector is initialized with 0.5. Each dimension in the vector represents the probability of each bit happened to be one [5]. Two candidate solutions are sampled from this vector. After evaluating, the winner and loser are specified. The winner is 11100101 and the loser is 10001100. The probability vector is updated according to the winner. The different bit between the winner and loser guides the probability to come closer to the better solution. Therefore, each dimension in the probability vector is updated toward the better solution by adding or subtracting the probability with an updating step size (1!). In a different bit, we add probability when the winner is 1, and subtract the probability when the winner is 0. E.g. updating step size is 0.1, the probability vector becomes as in step 4. The process of the CGA is repeated until the probability vector has converged. The concept of the CGA is simple and it has been proved that it performs like the SGA with population!, when the updating step size in the CGA is 1 ! [7]. the CGA reduces the size and power requirements of the system by representing the population as a probability vector rather than a collection of bit strings. The compact(cGA) as an estimation of distribution algorithm (EDA) that generates offspring population according to the estimated probabilistic model of parent population instead of using traditional recombination and mutation operators [3]. The cGA initializes a probability (distribution) vector (PV) over the set of solutions and two solutions are randomly generated by using this PV. The generated solutions are ranked based on their fitness values. The cGA represents the population as a PV over a set of solutions and operationally mimics the order-one behaviour of simple GA (sGA) with uniform crossover using a small amount of memory. When confronted with easy problems (e.g., continuous-unimodal problems involving lower order BBs), the cGA achieves the performance of the SGA (with the uniform crossover) in terms of the number of fitness evaluations. That the cGA may not be effective in solving real-world problems. In order to obtain better solutions to such difficult problems, the cGA should exert a higher selection pressure. This, in turn, increases the survival probability of higher order BBs, thereby preventing loss of the best solution found so far [12]. In other words, higher selection pressure may play the role of memory. Therefore, it can take care of a finite number of decision errors and some linkage information of genes. Selection pressure of the cGA can be increased by creating a larger tournament size in a simple manner. Goldberg and Miller analyzed the growth and of a particular gene in the population as a one-dimensional random walk. As the GA progresses, genes fight with their competitors, and their number in the population can go up or down, depending on whether the GA makes good or bad decision. These decisions are made implicitly by the GA when selection takes place. The next section explores the effects of this decision making.

**Selection**

Selection gives to more copies to better individuals. But it does not always do so for better genes. This is because genes are always evaluated within the context of a larger individual. For example, consider the onemax problem (that counting ones). Suppose individual a competes with individual b.

Individual chromosome fitness

a. 1011 3

b. 0101 2

When these two individuals compete, individuals a will win. However, at the level of the gene, a decision error is made on the second position. That is, selection incorrectly prefers the



ISSN: 2278 – 7798 

schema 0 to 1. The role of the population is to buffer against a
finite number of such decision errors. Imagine the following selection scheme: pick two individuals randomly from the population and keep two copies of the better one. This scheme is equivalent to a steady-state binary tournament selection. In a population of size n, the proportion of the winning alleles will increase by 1/n. For instance, in the previous example the proportion of 1's will increase by 1/n at gene position 1 and 3, and the proportion of 0‘s will also increase by 1/n at gene position2. At gene position 4, the proportion will remain the same. This thought experiment suggests that an update rule increasing a gene‘s proportion by 1/n simulates small steps in the action of a GA with a population of size n. The next section explores how the generation of individuals from probability distributions mimics the effects of crossover.

**Crossover**

The roll of crossover in the GA is to combine bits and pieces from fit solutions. A repeated application of most commonly used crossover operators eventually leads to a decorrelation of the population‘s genes. In this decorrelation state, the population is more compactly represented as a probability vector. Thus the generation of individuals from this vector can be seen as a shortcut to the eventual aim of crossov

## V. PROPOSED METHODOLOGY

For solving any problem we have to first consider task and resources into matrix form.

Pseudo code of the cGA

***Compact GA*** (n, N, fitness)
P= allocate vector of n real number;
*For i: =1 to n*
*do p [i]:= 0.5;*
*t=0;*
Generate two solutions from probability vector
*a:= **generate** p[i]; b := **generate** p[i];*
Compete both solutions
if (fitness (a)> fitness (b)) then
*W = a;*
*Else*
*L = b;*
*t= t+1;*
*(*Where W is winner and L is loser*)*
Update the probability vector
d=1/n;
*For i: = 1 to n do*
*If( W [i] >L[i] )then*
*p[i]:=p[i] +d;*
*else*
*p[i]:= p[i] -d;*

*Check if the probability vector has converged.*

*Go to Step2, if it is not satisfied.*

*The probability vector represents the final solution.*

cGA manage its population as a probability vector is PV, Probability vector is initialized with parameter 0.5 to represent a randomly generated population. In each generation (i.e. iteration), generate the individuals from the probability vector and find out the best one and then the position vector is updated to favour the better chromosome (i.e. winner).

Let the best individuals are ‘a‘ and ‘b‘ then compete both individuals, if both individuals fitness value is same then we assign ‘ a‘ is winner and update the probability vector along the way. Clearly the best individual wins all the competition. The cGA terminates when all the probabilities converge to zero or one.

## VI. CONCLUSION

In this paper we present compact genetic algorithm, an algorithm that mimics the order one behavior of simple genetic algorithm with a given population size and selection rate, but that reduce its memory requirement. In this paper we
explained of the compact algorithm. I am showing the research work in this paper. I will show my final paper which is represent implementation part for solving optimization problem and it will show the minimum cost and time.

## VII. ACKNOWLEDGEMENT

I would like to thank my project guide for guidance and thank H.O.D sir for using Computer lab and thank Director sir for using resources and thank colleague and friends for supporting and thank the anonymous referees for their helpful comments and suggestion that have improved the quality of this manuscript.

## REFERENCES

[1] Charles C. Peck, Atam P. Dhawan, “Genetic algorithms as global random search methods: An alternative perspective”, Evolutionary Computation, Volume 3 Issue 1, MIT Press, March 1995.
[2] D. E. Goldberg, “Genetic Algorithms in Search, Optimization and Machine Learning”, Addison Wesley Publishing Company, Ind. USA, 1989.
[3] http://mathworld.wolfram.com/GlobalOptimization.html.
[4] http://www.maplesoft.com/products/toolboxes/globaloptimization
[5] L.Painton, J.Campbell, “Genetic algorithms in optimization of system reliability” Reliability, IEEE Transactions on Volume: 44 , Issue: 2, Digital Object Identifier: 10.1109/24.387368, Publication Year: 1995 , Page(s): 172 – 178
[6] L. T. Leng, Guided genetic algorithm, Doctoral Dissertation. University of Essex, 1999.
[7] M. Syrjakow and H. Szczerbicka, “Combination of direct global and local optimization methods”, in IEEE Conference on Evolutionary Computation. Perth, Western Australia: IEEE, 1995, pp. 326-333.
[8] Melanie Mitche, “An introduction to genetic algorithm”, MIT press, 1998.
[9] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, S. Dizdarevic, “Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators”, Artificial Intelligence Review , Volume 13 Issue 2, Kluwer Academic Publishers, April 1999.
[10] R. Fletcher, “Practical Methods of Optimization”, John Willey & Sons, 1986.
[11] T. Weise, “Global Optimization Algorithms - Theory and Application”. Available online at www.itweise. de/projects/book.pdf



ISSN: 2278 – 7798 

[12] Törn and A. Zilinskas, Global optimization, in Lecture Notes in Computer Science, vol. 350: Springer-Verlag, 1989.

**Neelu Sahu** received the B.E degree in computer science & engineering from the Institute of Technology, Guru Ghasidas Vishwavidyalaya, Bilaspur, India**,** in 2010. And M.E from the Shri Shankaracharya Group of Institutions, Bhilai, India.

**Prateek kumar Singh** received the B.E degree in information Technology from R.I.T.E.E college, Raipur, India, in 2012. And pursuing M.Tech from the LNCT, Jabalpur, India.



ISSN: 2278 – 7798 