International Journal of Theoretical and Applied Sciences **4**(1): 48-51(2012) **ISSN No. (Print) : 0975-1718**
ISSN No. (Online) : 2249-3247

Ant Colony Optimization Using Genetic Algorithms

Shweta Singh*, G.C. Dubey** and Rajesh Shrivastava***

Department of Mathematics,
**Radharaman Institute of Technology and Science, Bhopal, (M.P.)*
***Government M.G.M. College, Itarsi, (M.P.)*
****Government Benazir College, Bhopal, (M.P.)*

(Received 11 April, 2012, Accepted 12 May, 2012)

ABSTRACT : A Genetic algorithms is a search technique used to find true or approximate solutions to optimization and search problems. Hybrid algorithms is proposed to solve combinatorial optimization problem by using "Ant colony and Genetic programming algorithms". The genetic programming paradigm permits the evolution of computer programs which can perform intermediate calculations, which can perform computation on variable of many different types and reaches to the optimal final solution, by accumulating the most effective sub-solutions.

Keywords : Genetic Algorithms (GA), Ant Colony Optimization (ACO), System Theory, Genetic Programming (GP).

I. INTRODUCTION

Over the last few years, scientist, engineers and economists have extensively used genetic algorithms (GA), to solve optimization problems involving single objective functions. During the last few years several researchers have intended GA. to solve multiobjective problems. The recent resurgence of interest in AP with GA has been spurred by the work on Genetic programming (GP). GP paradigm provides a way to do program induction by searching the space of possible computer programs for an individual computer program that is highly fit in solving or approximately solving the problem at hand [1,3]. The genetic programming paradigm permit the evolution of computer programs which can performance of intermediate calculations, which can perform iterations and recursions to archieve the dosired result, which can define and subsequently use computed values and sub-programs and whose size, shape and complexity is not specified in advance.

GA use relatively low-level primitives, which are defined separately rather than combined a priori into high-level primitives, since such mechanism generate hierarchical structures that would facilitate the creation of new high-level primitives from built-in-low-level primitives [4-8].

II. ANT COLONY OPTIMIZATION

The basic idea of Ant colony optimization (ACO) [5] is to model the problem to solve as the search for a minimum cost path in a graph, and to use artificial ants to search for good paths. The behavior of artificial ants is inspired from real ants: they lay pheromone on components of the graph and they choose their path with respect to probabilities that depend on pheromone trails that have been previously laid by the colony. These pheromone trails progressively decrease by evaporation. Intuitively, this indirect stigmergetic communication means aims at giving information about the quality of path components in order to attracts ants, in the following iterations towards the corresponding areas of the search space.

The first ant algorithm to be applied to a discrete optimization problem has been proposed by Dorigo in [5]. The problem chosen for the first experiments was the Traveling salesman problem and, since than, this problem has been widely used to investigate the solving capabilities of ants. The ACO metaheuristic is a generalization of these first ant based algorithms, and has been successfully applied to different hard combinatorial optimization problem such as quadratic assignment problems [13], vehicle routing problems [6, 9], constraint satisfaction problems [11] or maximum clique problem [12].

III. GENETIC PROGRAMMING

Unfortunately, since every real life problem are dynamic problem, this their behaviors are much complex, GP suffers from serious weaknesses. Thus aim of this work is to enhance the ability of ACO by using GP technique therefore some specific advantage of genetic programming are that no analytical knowledge is needed and still could accurate results.

In applying genetic programming to solve a problem there are five major pre paratory steps. These five steps involve determining.

1. The set of terminals.
2. The set of primitive functions.
3. The fineness measure.
4. The parameters for controlling the run.
5. The method for designating a result and the criterion for terminating a run.

IV. BACKGROUND ON GENETIC ALGORITHM AND THE RELATED WORK

John Holland's pioneering 1975 Adaptation in Natural and Artificial systems described how the evolutionary process in Nature can be applied to artificial systems using in genetic algorithm operating on fixed length character strings.

Holland demonstrated that a population of fixed length character strings can be genetically bred using the Darawinian operation of fitness proportionate reproduction and the genetic operation of recombination. The recombination operation combines parts of two chromosome like fixed length character strings each selected on the basic of their fitness, to produce new offspring strings.

The ACO has been applied to a broad range of hard combinational problems problems are define in terms of components and states, which are sequence of components adding new concept to state.

V. DESCRIPTION GENETIC ALGORITHM

The genetic algorithm (GA) transforms a population (set) of individual objects, each with an associated fitness value, into a new generation of the population using the Darwinian principle of reproduction and survival of the fittest and analogs of naturally occurring genetic operations such as crossover sexual recombination and mutation to form a new population. The new population & then used in the next iteration of the algorithm. Finally the new solutions are used to replace the poorer of the original solution and the process is repeated. Commonly the algorithms terminates when either a maximum number of generations has been produced or a satisfactory fitness level has been reached for the population.

VI. PROPOSED SOLUTION OF ACOGA

In this section we have a combinational optimization problem defined over a set P defined as.

Set $P = P_1, P_2, P_3 \ldots\ldots\ldots P_n$ of basic components decreases trail values over time in order to avoid unlimited accumulation of trails over some component. Daemon actions can be used to implement centralized actions which cannot be performed by single ants.

At each step, each ants computer a set of feasible expansion to its current state, and moves to one of these in probability. The probability distribution is defined as follows, for ant, k the probability of moving from state t to state n to depends on the combination of two values [9].

(*a*) The attractiveness of the move, as computed by some heuristic indication the priori desirability of that move.

(*b*) The trail level of the move, indicating how proficient it has been in the past to make that particular move.

As we know that Daemon action are centralized operation, such as comparing solution values among ants in orders to find the best solution, or running a local search procedure.

Algorithm-1. Ant colony Optimization (ACO)

while termination condition not met do

Schedule Activities

Construct Ants Solutions

Update pheromone

Deamon Actions

end Schedule Activities

end while

A subset Q of components represents a solution the problems; $FC\ 2^p$ is the subset of feasible solution thus a solution Q is feasible if and only if $Q \in F$. A cost function μ is defined over the solution domain $\mu : 2^p \to S$ the objective being to find a minimum cost feasible solution Q^*.

To find $Q^* : Q^* \in F$ and $\mu(Q^*) \leq \mu(Q) \forall Q \in F$. They move by applying a stochastic local decision policy based on two parameters called trades and attractiveness.

Pheromone Trails : The quantity of Pheromone laying on a component represent the past experience of the colony with respect to choosing this component when there is only one objective function, this past experience is defined with respect to this objective. However when there are several objectives, one may consider two different strategies. A first strategy is to consider a single pheromone structure In this case, the quantity of pheromone laid by ants is defined with respect to an aggregation of the different objectives. A second strategy is to consider several pheromone structure. In this case, one usually associates a different ant colony of ants with different objective, each colony having its own pheromone structure further more, an ACO algorithm includes two more mechanisms : trail evaporation and daemon actions. Trail evaporation.

VII. ACOG ALGORITHM

An ACOG is differ from that algorithm given in reference [10] is use genetic programming to enhance performance. It consists of two main

Sections 1. Initialization **2.** Main loop where G_p is used in the second sections. The main loop runs for a user defined number of iterations. These are described in table below :

Algorithm-2

1. Initialization :

(*a*) Set initial parameter that are system : input, output, variable, states, input and output trajectory.

(*b*) Set initial pheromone trail value.

(*c*) Each ant is individually placed on intial state with empty memory

2. While termination conditions not meet do :

(*a*) Construct Ant solution: Each ant constructs a path by successively applying the transition function the probability of moving from state to state depend on: as the attractiveness of the move and the trail level of the move.

(*b*) Apply local search.

(*c*) Best tour check" If there is an improvement, update it.

3. Update Trails :

(*a*) Evaporate a fixed proportion of the pheromone on each raod.

(*b*) For each ant perform the “ant-cycle” pheromone update.

(*c*) Reinforce the best tour with a set number of “elitist ants” performing the “ant-cycle”.

4. Create a now population by applying the following operation, based on pheromone trails and are applied to individual selected from the population with a probability based on fitness :

(*a*) Darwinian reproduction

(*b*) Structure -preserving crossover.

(c) Structure-preserving Mutation.

5. End While

VIII. THEOUTPUT PERFORMANCE OF THE GENETIC PROCESS

In this article, a genetic algorithm based on Ant colony optimization is parameterized by the number of ant colonies and the numbers of pheromone trails. Genetic generation process involves probabilistic steps, because convergence to a globally sub-optimal result, problems become inherent features of genetic generation process. To minimize the effect of these problems must be made. Best -of-run individual from all such multiple independent runs can then be designated as the result of the group of uns. The computational effort required to get the solution would depends primarily on four factors.

S = Population size

n = Number of generation that are run ($n \leq N$; N is the maximum number of generation).

The amount of processing required for fitness cases, and the amount of processing required for test phase also we assume that,

t = to measure the fitness of an individual is its run time.

C = computational effeor

If success occurs on the same generation of every run, the computational effort C world be computed as :

$$C = S.n.\beta.P \qquad \text{... (1)}$$

Since the value of p is too small with respect to other factors, wer shall neglect it, then the computational effort C would be computed as follows.

$$C = S.n_{avr}.\beta \qquad \text{... (2)}$$

where n_{avr} is the average number of executed generations.

Since GPG is a probabilistic algorithm, not all runs are successful at yielding a solution to the problems by Generation G. Thus, the computational effort is computed in this way, first determining the number of independent runs R needed to yield a success with a certain probability second, multiply R by the amount of processing required for each run that is. The number of independent runs R required to satisfy the success predicate by generation i with a probability z which depends on both z and $p(S, i)$ where z is the probability of satisfying the success predicate by generation i at least once in R runs defined by

$$z = 1 - [1 - P(S, i)]R \qquad \text{... (3)}$$

$P(S, i)$ = cumulative probability of success for all generations b/w o and i

$Y(S, i)$ = Instantaneous probability

S = Population size S on a specified generation i

This experimental measurement of $Y(S, i)$ usually requires a substantial number of runs. After taking logarithms for equation (4) we find.

$$R = \left[\frac{\log(1-Z)}{\log[-p(s,i)]} \right] \qquad \text{... (4)}$$

The computational effort C, is the minimal value of the total number individual that must be processed to yield a solution for the problem with z probability.

$$C = S.(n+1).\beta.R \qquad \text{... (5)}$$

where n is the first generation at which minimum number of individual evaluation is produced, it is called generation. Hence the value of C is not necessarily the minimum computational effort possible for the problem.

IX. CONCLUSION

The method proposed the Ant colony algorithm may produce redundant states in the graph, it better to minimize such graphs to enhance the behavior of the inducted system when an ant complete solution or during the construction phase the ant evaluates the solution and modifies the trail value on the components used in its solution. Ant deposit a certain amount of pheromone on the components; that is, either on the vertices or on the edges that they traveres. The genetic programming paradigm permits the evolution of computer programs which can perform alternative computations conditioned on the outcome of intermediate calculations, which can perform computations on variables of many different types which can perform iterations and recursion to achieve the desired result.

REFERENCES

[1] Abraham A. *et al.* Evolutionary Compution: from Genetic Algorithms to Genetic Programming, Studies in Computational Intelligence (SCI) 13, 1-20, www.springerlink.com c_Spring-Verlag Berlin Heidelberg 2006.

[2] Bullnheimer B. Hartl R.F. Strauss C., An Improved Ant system Algorithm fro the Vehicule Routing Problem, *Annals of Operations Research*, vol. **89** (1999) p. 319-328.

[3] Grosan C. and Abraham A.: Hybrid Evolutionary Algorithms Methodologies Architectures, and Reviews, Studies in computational Intelligence (SCI) 75, 1-17, www.springerlink. com c_Springer-Verlag Berlin Heidelberg (2007).

[4] Doerner M. Optimization, Learning, Natural Algorithms (in Italion), Ph.d thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, (1992).

[5] Dorigo M., Stuzle T. Ant Colony Optimization, MIT Press (2004).

[6] Hector Montes A. and Jeremy L. Wyatt," Cartesian Genetic Programming for Image Processing Tasks."

[7] John R. Koza, Margaret Jacks Hall," Survey of genetic algorithms and genetic pro9gramming http:/www-cs-faculty.stanford.edu./koza.

[8] Reccardo Poli , William B. Langdon, Nicholas F. Mcphee, John Koza R., "Genetic Programming: An Introductory Tutorial and a Survey of Techniques and pplications", Technical Report CES-475 ISSN:1744-8050 October (2007). essex. ac.uk/ does /research' publication/ .../2007/ces475 pdf Aooebdex.

[9] Nada M.A. AL-salami, Saad Ghaleb Yaseen, "Ant Colony Optimization." *IJCSNS International Journal of Computer Science and Network Security*. Vol. **8** No. 6 pp. 351-357 June, (2008).

[10] Nada M.A. AL-Salami," System Evolving using Ant Colony Optimization Algorithm", *Journal of Computer Science* **5**(5): 380-387, (2009), ISSN 1579-3636.

[11] C. Solnon, Ants can solve Constraint Satisfaction Problem. *IEEE Transactions on Evolutionary Computation* vol. **2**, p. 221-248(1994).

[12] Solnon C., Fenet S., *A study of ACO capabilities for solving the Maximum Clique Problem, ournal of Heuristics* vol. **12**, no. 3, p. 155-180(2006).

[13] Stiitzle T., Hoos H., MAX-MIN *Ant System. ournal of Future Generation Computer Systems*, vol **16**, p. 889-914(2000).