International Journal of Soft Computing 8 (3): 171-174, 2013
ISSN: 1816-9503

Comparison Between Ant Colony and Genetic Algorithm Using Traveling Salesman Problem

[1]Zaid Ameen Abduljabbar, [2]Mustafa S. Khalefa and [2]Marzanah A. Jabar
[1]Department of Computer Science, Education College, Basra University, Basra, Iraq
[2]Department of Information System, Faculty of Computer Science and Information Technology, UPM, Serdang, Selangor, Malalysia

Abstract: The Travelling Salesman Problem (TSP) is a complex problem in combinatorial optimization. The aim of this study is compare the effect of using two distributed algorithm which are ant colony as a Swarm intelligence algorithm and genetic algorithm. In ant colony algorithm each individual ant constructs a part of the solution using an artificial pheromone which reflects its experience accumulated while solving the problem and heuristic information dependent on the problem. The results of comparison show that ant colony is high efficient than genetic algorithm and it requires less computational cost and generally only a few lines of code.

Key words: Ant colony, genetic algorithm, combinatorial optimization, traveling salesman problem, distributed algorithm

INTRODUCTION

Optimization problems are high importance both for the industrial world as well as for the scientific world. Examples of practical optimization problems include train scheduling, time tabling, shape optimization, telecommunication network design or problems from computational biology. The research community has simplified many of these problems in order to obtain scientific test cases such as the well-known Traveling Salesman Problem (TSP). The TSP Models the situation of a travelling salesman who is required to pass through a number of cities. The goal of the travelling salesman is to traverse these cities (visiting each city exactly once) so that the total travelling distance is minimal (Lawler *et al*., 1985). The TSP problem belongs to an important class of optimization problems known as Combinatorial Optimization (CO) (Lebocey *et al*., 2006).

Due to the practical importance of CO problems many algorithms to tackle them have been developed. These algorithms can be classified as either complete or approximate algorithms. Complete algorithms are guaranteed to find for every finite size instance of a CO problem an optimal solution in bounded time. Yet for CO problems that are NP-hard In complexity theory, NP denotes the set of all (decision) problems solvable by a non-deterministic polynomial time algorithm. These problems are considered hard in the sense they are not solvable in deterministic polynomial time, no polynomial time algorithm exists, assuming that P-NP. Therefore, complete methods might need exponential computation time in the worst-case (Nemhauser and Wolsey, 1988; Garey and Johnson, 1979). This often leads to computation times too high for practical purposes. Thus, the development of approximate methods in which researchers sacrifice the guarantee of finding optimal solutions for the sake of getting good solutions in a significantly reduced amount of time has received more and more attention in the last 30 years (Nemhauser and Wolsey, 1988; Garey and Johnson, 1979).

Ant colony optimization and Genetic Algorithm optimization are the most recent techniques for approximate optimization (Dorigo and Stutzle, 2004; Blum, 2005). The aim of this research is to compare between ant colony and genetic algorithm and show which one is require less combinatorial optimization (time) to find the optimal solution.

ANT COLONY OVERVIEW

Ant System is the original ant colony optimization algorithm proposed by Dorigo *et al*. (1991). Ant Colony Optimization (ACO) is a recent metaheuristic technique that is inspired by the pheromone trail laying and following behavior of some ant species (Dorigo *et al*., 1991). In ACO algorithms, artificial ants are stochastic solution construction procedures that generate solutions using artificial pheromones and heuristic information the ants' solutions are then used to modify the artificial pheromone trails.

Corresponding Author: Zaid Ameen Abduljabbar, Department of Computer Science, Education College, Basra University, Basra, Iraq

This mechanism shifts the stochastic solution construction procedure towards the construction of solutions similar to the better ones seen previously in the algorithm. The definition of the ACO metaheuristic includes also the possibility of using local search: Once ants complete their solution construction phase, local search algorithms can be used to refine their solutions before using them for the pheromone update. Various experimental researches have shown that the combination of solution construction by ants and local search procedures is a promising approach (Dorig and Stutzle, 2004; Blum, 2005).

GENETIC ALGORITHM OVERVIEW

A genetic algorithm is an adaptive search technique based on the principles and mechanisms of natural selection and survival of the fittest. Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics (Haupt and Haupt, 2004). Genetic algorithms are a proven and efficient search technique for solving problems classed as NP-complete (De Jong and Spears, 1989; Singh, 2012; Patalia and Kulkarni, 2012). Genetic algorithms execute iteratively until a stopping criterion is reached either after N generations, reaching convergence or until a satisfactory solution is found.

A genetic algorithm begins from a set of 1 candidate solutions each candidate an encoding for solving the problem. Candidate solutions are typically represented as binary strings and initially created randomly. The fitness/objective function is responsible for decoding a candidate solution into a form applicable to the problem. A collection of candidate solutions form a population. The evolution process of genetic algorithms creates a new population from the current population through the use of genetic operators. After every generation each candidate solution in the population is evaluated using the fitness/objective function. Candidate solutions that perform better at solving the problem receive a higher rated fitness value. Genetic operators ensure fitter solutions survive and continue to produce offspring. The less fit solutions are eventually deleted from the population. This process is termed survival of the fittest (Wilson *et al*., 2007).

BASIC ALGORITHM OF ANT COLONY

Step 1 (Initiation): The amount of the pheromone on each side is initiated into a tiny constant value; allocate m ants randomly to n cities.

Step 2: In ACS, the so-called pseudorandom proportional rule is used: the probability for an ant to move from city i to city j depends on a random variable q uniformly distributed over (0, 1) and a predefined parameter q_0:

$$j = \begin{cases} \arg\max_{u \in \text{allowed}_k^{(i)}} \left\{[\tau_{iu}]^{\alpha}[\eta_{iu}]^{\beta}\right\} & \text{if } q<q_0 \\ j & \text{Otherwise} \end{cases} \quad (1)$$

J is a random variable. This strategy obviously increases the variety of any searching thus avoiding any premature falling into the local optimal solution and getting bogged down.

Step 3: The local pheromone update is performed by all the ants after each construction step. Each ant applies it only to the chosen city:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\tau_0 \quad (2)$$

Where:

$0<\rho\leq 1$ = A decay parameter

τ_0 = 1/n.Lnn is the initial values of the pheromone trails, where n is the number of cities in the TSP and Lnn is the cost produced by the nearest neighbor heuristic

Step 4: Computing of the optimal path. After m ants have travelled through all the cities, compute the length of the optimal.

Step 5: Global updating of pheromone. After all the ants have travelled through all the cities, update only the amount of the pheromone on the optimal path with Eq. 4:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) - \rho\Delta\tau_{ij}(t) \quad (3)$$

$$\Delta\tau_{ij}(t) = \begin{cases} \frac{1}{L_{gb}} & \text{if } (i,j) \in \text{global best tour} \\ 0 & \text{Otherwise} \end{cases} \quad (4)$$

Where:

ρ = The constant

L_{gb} = The length of global best tour

Step 6: If the designated search number is not attained then repeat the above steps (Hlaing and Khine, 2011; Hingrajiya *et al*., 2012).

BASIC GENETIC ALGORITHM

Genetic algorithm is started with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population. This is motivated by a hope that the new population will be better than the old one. Solutions which are selected to form new solutions (offspring) are selected according to their fitness; the more suitable they are the more chances they have to reproduce. This is repeated until some condition (for example, number of populations or improvement of the best solution) is satisfied. It is well known that problem solving can be often expressed as looking for the extreme of a function.

This is exactly the case with the following problem: some function is given and GA tries to find the minimum of the function (Sivanandam and Deepa, 2008; Gupta and Khurana, 2012). The basic genetic algorithm is as follows:

- [Start] generate random population of n chromosomes (suitable solutions for the problem)
- [Fitness] evaluate the fitness f(x) of each chromosome x in the population
- [New population] create a new population by repeating the following steps until the new population is complete
  - [Selection] select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
  - [Crossover] with a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents
  - [Mutation] with a mutation probability mutate new offspring at each locus (position in chromosome)
  - [Accepting] place new offspring in a new population
- [Replace] use new generated population for a further run of algorithm
- [Test] if the end condition is satisfied, stop and return the best solution in current population
- [Loop] go to step 2

In traveling salesman problem, salesman travels n cities and returns to the starting city with the minimal cost he is not allowed to cross the city more than once. In this problem we are taking the assumption that all the n cities are inter connected. The cost indicates the distance between two cities. To solve this problem we make use of genetic algorithm because the cities are randomly. Initial population for this problem is randomly selected cities. Fitness function is nothing but the minimum cost. Initially the fitness function is set to the maximum value and for each travel; the cost is calculated and compared with the fitness function. The new fitness value is assigned to the minimum cost. Initial population is randomly chosen and taken as the parent. For the next generation, the cyclic crossover is applied over the parent (Sivanandam and Deepa, 2008; Gupta and Khurana, 2012).

Table 1: Comparison results between ant colony and genetic algorithm including four randomly generated 25 city problems

	Ant colony		Genetic algorithm	
Problem 25 city	Best length (optimal solution)	Iterations	Best length (optimal solution)	Iterations
1	5249	652	5608	1627
2	7008	961	7520	1938
3	5819	723	6103	1452
4	6874	837	7359	1876

Table 2: Comparison results between ant colony and genetic algorithm including four randomly generated 50 city problems

	Ant colony		Genetic algorithm	
Problem 50 city	Best length (optimal solution)	Iterations	Best length (optimal solution)	Iterations
1	10,749	2639	10,842	7718
2	8,618	2950	8,924	9630
3	13,826	2708	13,979	8225
4	11,010	2986	11,362	6291

RESULT OF EXPERIMENT ONE

This experiment comprises four randomly generated 25 city problems. The connections and distances between cities are set randomly for each 25 city problem. The comparison results are shown in Table 1. Ant colony was run using 15 ants. The best length is the optimal solution. Ant colony almost offers the best performance.

RESULT OF EXPERIMENT TWO

This experiment comprises four randomly generated 50 city problems. The connections and distances between cities are set randomly for each 50 city problem. The comparison results are shown in Table 2. Ant colony was run using 37 ants. The best length is the optimal solution. Ant colony almost offers the best performance.

RESULT OF EXPERIMENT THREE

This experiment comprises four randomly generated 75 city problems. The connections and distances between cities are set randomly for each 75 city problem. The comparison results are shown in Table 3. Ant colony was run using 53 ants. The best length is the optimal solution. Ant colony almost offers the best performance.

RESULT OF EXPERIMENT FOUR

This experiment comprises four randomly generated 100 city problems. The connections and distances between cities are set randomly for each 100 city problem. The comparison results are shown in Table 4. Ant colony was run using 72 ants. The best length is the optimal solution. Ant colony almost offers the best performance.

Table 3: Comparison Results between ant colony and genetic algorithm including four randomly generated 75 city problems

	Ant colony		Genetic algorithm	
Problem 75 city	Best length (optimal solution)	Iterations	Best length (optimal solution)	Iterations
1	15,028	3974	15,198	11,207
2	17,492	4471	17,809	13,119
3	14,891	4298	15,251	13,055
z4	15,726	4672	15,904	12,846

Table 4: Comparison results between ant colony and genetic algorithm including four randomly generated 100 city problems

	Ant colony		Genetic algorithm	
Problem 100 city	Best length (optimal solution)	Iterations	Best length (optimal solution)	Iterations
1	20,751	6802	20,836	16,253
2	21,020	7725	21,284	17,849
3	21,948	7937	22,103	18,006
4	22,194	8367	22,402	16,809

CONCLUSION

This comparison is the first direction and depends on the basic algorithm for the mentioned optimization methods. The ealier results show that ant colony is the best to find the optimal solution with less number of iterations (less combinatorial optimization). At the same time the ant colony requires only few lines of code. The best number of ants are 15 for 25 cities, 37 for 50 cities, 53 for 75 cities and 72 for 100 cities and it's the best number over the selected range in these experiments (10-100) ants. The best value of p is 0.8.

Futures directions are require studies towards find the relation between number of ants and cities, selection method, mutation method and mutation probability.

REFERENCES

Blum, C., 2005. Ant colony optimization: Introduction and recent trends. Physics Life Rev. J., 2: 353-373.

De Jong, K.A. and W.M. Spears, 1989. Using genetic algorithm to solve NP-complete problems. Proceedings of the 3rd International Conference on Genetic Algorithms, June 4-7, 1989, Morgan Kaufmann Publishers, San Mateo, CA, pp: 124-132.

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

Dorigo, M., V. Maniezzo and A. Colorni, 1991. The ant system: An autocatalytic optimizing process. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milano, Italy.

Garey, M.R. and D.S. Johnson, 1979. Computers and Intractability A Guide to the Theory of NP Completeness. W.H. Freeman and Compan, UK.

Gupta, A. and S. Khurana, 2012. Study of traveling salesman problem using genetic algorithm. Int. J. Manage. IT Eng., 2: 575-588.

Haupt, R.L. and S.E. Haupt, 2004. Practical Genetic Algorithms. 2nd Edn., John Wiley and Sons, New York, ISBN: 9780471455653, Pages: 272.

Hingrajiya, K.H., R.K. Gupta and G.S. Chandel, 2012. An ant colony optimization algorithm for solving travelling salesman problem. Int. J. Scient. Res. Public., 2: 1-6.

Hlaing, Z.C.S.S. and M.A. Khine, 2011. An ant colony optimization algorithm for solving traveling salesman problem. Proc. Int. Conf. Inform. Communi. Manage., 16: 54-59.

Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan and D.B. Shmoys, 1985. The Travelling Salesman Problem. John Wiley & Sons, New York, USA.

Lebocey, P., J. Fortune, A. Puret, N. Monmarche, P. Gaucher, M. Slimane and D. Lastu, 2006. On the popularization of artificial insects: An interactive exhibition for a wide audience to explain and demonstrate computer science and robotic problem solving taking inspiration of insects. Proceedings of the 5th International Conference on Ant Colony Optimization and Swarm Intelligence, (ANTS'06), Springer-Verlag, Berlin, Heidelberg, pp: 476-483.

Nemhauser, G.L. and L.A. Wolsey, 1988. Integer and Combinatorial Optimization. John Wiley & Sons, New York, USA.

Patalia, T.P. and G.R. Kulkarni, 2012. Comparative analysis of threshold acceptance algorithm, simulated annealing algorithm and genetic algorithm for function optimization. Global J. Res. Eng. Numerical, 12: 23-27.

Singh, R., 2012. Genetic algorithm for parallel process scheduling. Int. J. Comput. Appli. Inform. Technol., 1: 72-76.

Sivanandam, S.N. and S.N. Deepa, 2008. Introduction to Genetic Algorithms. Springer, USA., ISBN: 354073-189X, Pages: 442.

Wilson, W., P. Birkin and U. Aickelin, 2007. The motif tracking algorithm. Int. J. Automat. Comput., 4: 100-106.