An Improved Modular Hybrid Ant Colony Approach for Solving Traveling Salesman Problem

Sudip Kumar Sahana
Assistant Prof, Dept of CSE,
Birla Institute of Technology,
Ranchi, Jharkhand, India.
Mob: +91-9470131644
sudipsahana@gmail.com

Dr.(Mrs).Aruna Jain
Reader, Dept of IT,
Birla Institute of Technology,
Ranchi, Jharkhand, India.
Mob: +91-9939186200
jaruna16@rediffmail.com

ABSTRACT

Our primary aim is to design a framework to solve the well known traveling salesman problem(TSP) using combined approach of Ant Colony Optimization (ACO) and Genetic Algorithm (GA). Several solutions exists for the above problem using ACO or GA and even using a hybrid approach of ACO and GA. Our framework gives the optimal solution for the above problem by using the modular hybrid approach of ACO and GA along with heuristic approaches.We have incorporated GA, RemoveSharp and LocalOpt heuristic approaches in ACO module, hence each iteration calls the GA and heuristics within ACO module which results in a higher amount of pheromone deposited in the optimal path for global pheromone update. As a result the convergence is quicker and solution is optimal.

Keywords

Traveling Salesman Problem(TSP),Ant Colony Optimization (ACO), Genetic Algorithm (G.A), Heuristics, Optimization, Pheromone.

1. INTRODUCTION

In the approach discussed in this paper we distribute the search activities over "ants," that is, agents with simple basic capabilities which, to some extent, mimic the behavior of real ants.

Ant colony algorithm[2] determines optimal solution by simulating the process of ants searching for food. The ants collective behavior reflects an information positive feedback phenomenon. The increased amount of pheromone attributed as the positive feedback. This optimization technique does not rely on mathematical description of the specific issues, but has strong global optimization feature[3], high performance[4] and flexibility. Three main aspects to determine ACS are:

1.1 ACS (Ant Colony System) State Transition Rule

Ants prefer to move from one place to another(i.e one node to other node) which are connected by short edges with a high amount of pheromone[2]. It can be done by using following rule.

An ant positioned on node r chooses the city s as shown in Fig1 to move by applying the rule given by Eq. (1)

$$PI_k(r,s)= \begin{cases} \dfrac{(r,s).[\eta(r,s)]^{\beta}}{\sum (r,u)[\eta(r,u)]^{\beta}}, & \text{if } s \epsilon J_k(r),\ u \epsilon J_k(r) \\ 0 & \text{otherwise} \end{cases} \quad -(1)$$

where τ is the pheromone, $\eta = 1/d$ is the inverse of the distance d (r,s), $J_k(r)$ is the set of cities that remain to be visited by ant k positioned on city r (to make the solution feasible), and ß is a parameter which determines the relative importance of pheromone versus distance (ß >0).

In Eq. (1) we multiply the pheromone on edge (r,s) by the corresponding heuristic value

η (r,s). In this way we favor the choice of edges which are shorter and which have a greater amount of pheromone.

1.2 ACS Local Updating Rule

While building a solution (i.e., a tour) of the TSP, ants visit edges and change their pheromone level by applying the local updating rule[5] of Eq. (2):

$$(r,s) \leftarrow (1-\rho).(r,s)+\rho.\Delta(r,s) \quad -(2)$$

where $0< \rho <1$ is a parameter.

We have experimented with several values for the term Δ (r,s).A good choice was inspired by Q-learning[3] , an algorithm developed to solve reinforcement learning problems which allows an agent to learn such an optimal policy by the recursive application of a rule. $\Delta(r,s)=\gamma.\max(s,z)$ where $z \epsilon J_k(s)$ and $0<\gamma<=1$.Alternate choices may be $\Delta(r,s)=t_0$ or $\Delta(r,s)=0$.

1.3 ACS Global Updating Rule

Once all ants have built their tours,pheromone is updated on all edges by using the following rule:

$$(r,s) \leftarrow (1-\alpha)*(r,s)+\alpha*\Delta(r,s). \quad -(3)$$

where $0<\alpha<1$ is pheromone decay [4][6]parameter and we assume $\alpha=0.2$ to get a better effect of probability on the globally shortest path.

Where, $(r,s) = \begin{cases} (Lgb)^{-1} & \text{if } (r,s) \in \text{global best tour} \\ 0 & \text{otherwise} \end{cases}$

and Lgb is length of globally best tour.

DOI: 10.5176_2010-2283_1.2.49

1.4 Genetic Algorithm

Genetic algorithm [1] is a relatively new optimization technique which can be applied to various problems, including those that are NP-hard. The technique does not ensure an optimal solution, however it usually gives good approximations in a reasonable amount of time. This, therefore, is a choice to try on the traveling salesman problem [5], one of the most famous NP-hard problems.

Genetic algorithms are based on natural evolution and use a "survival of the fittest" technique, where the best solutions survive and are varied until we get a good result.

2. HYBRID GENETIC ALGORITHM

In Hybrid Genetic Algorithm [8] some heuristic functions are considered with the crossover function of GA. We have designed a modified algorithm[13] by removing the Initial Heuristics (IH) part as it is not well fitted for the large number of cities. If the number of cities increases the performance of IH degrades. And also we have removed the multiple instances of local search. Only single instance we have used over here for the sake of simplicity as it also performs well. The modified Hybrid Genetic Algorithm [13] is designed to use the improvement of offspring produced by crossover. The offspring is obtained by crossover [6] between two parents selected randomly. The tour improvement heuristics: RemoveSharp and LocalOpt are used to bring the offspring to a local minimum [7]. If cost of the tour of the offspring thus obtained is less than the cost of the tour of any one of the parents then the parent with higher cost is removed from the population and the offspring is added to the population. If the cost of the tour of the offspring is greater than that of both of its parent then it is discarded. The modified Hybrid Genetic Algorithm works as below:

Step 1:

- Select two parents randomly.
- Apply Crossover between parents and generate an offspring
- Apply RemoveSharp algorithm to offspring
- Apply LocalOpt algorithm to offspring
- If TourCost(offspring) < TourCost(any one of the parents) then replace the weaker parent by the offspring

Step 2: Shuffle any one randomly selected tour from population

Step 3: Repeat steps 1 and 2 until end of specified number of iterations

2.1 Crossover Algorithm

The crossover operator that is used here is a slight variant of the general crossover operator. The crossover operator uses an "edge map" to construct an offspring which inherits as much information as possible from the parent structures. This edge map stores information about all the connections that lead into and out of a city. Since the distance is same between any two cities, each city will have at least two and at most four edge associations (two from each parent).Crossover algorithm works as follows:

- Step 1:♦Choose the initial city from one of the two parent tours. (It can be chosen randomly or according to criteria outlined in step 4). This is the "current city".
- Step 2:♦Remove all occurrences of the "current city " from the left-hand side of the edge map.
- Step 3:♦If the "current city" has entries in its edgelist go to step 4; otherwise, go to step 5.
- Step 4:♦Determine which city in the edgelist of the "current city", has shortest edge with the "current city". The city with the shortest edge is included in the tour. This city becomes the "current city". Ties are broken randomly. Go to step 2.
- Step 5:♦If there are no remaining unvisited cities, then STOP. Otherwise, randomly choose an unvisited city and go to step 2.

2.2 The RemoveSharp Algorithm

The RemoveSharp algorithm removes sharp increase in the tour cost due to a city, which is badly positioned. The algorithm works as below:

- Step 1: ♦A list (NEARLIST) containing the nearest m cities to a selected city is created.
- Step 2: ♦RemoveSharp removes the selected city from the tour and forms a tour with N-1 cities.
- Step 3: ♦Now the selected city is reinserted in the tour either before or after any one of the cities in NEARLIST and the cost of the new tour length is calculated for each case.
- Step 4: ♦The sequence, which produces the least cost, is selected.
- Step 5: ♦The above steps are repeated for each city in the tour.

2.3 The Local Opt Algorithm

The LocalOpt algorithm will select q consecutive cities (Sp+0 , Sp+1 , ... , S p+q-1) from the tour and it arranges cities Sp+1 , Sp+2 , ... , Sp+q-2 in such a way that the distance is minimum between the cities S p+0 and S p+q-1 by searching all possible arrangements. The value of p varies from 0 to n-q, where n is the number of cities.

3. MODULAR HYBRID ANT COLONY APPROACH

The modular hybrid ant colony [13]algorithm by effectively using Ant Colony Optimization[4], Genetic Algorithm and some heuristics ,combined to optimize the problem and enhancing the throughput is as bellow:

- Initialization: iteration i =0
- Step1: Apply ACS to the problem for iteration i.
  - Calculate initial pheromone on each path.
  - Make local updation on each path.
  - Make global updation on the best path so far.
- Step2: Apply modified hybrid GA.
  - Select two good parents(paths) from output of Step1.
- Step3: Apply Crossover between parents and generate an offspring.
- Step4: Apply RemoveSharp algorithm to offspring.
- Step5: Apply LocalOpt algorithm to offspring.
  - If TourCost(offspring) < TourCost (any one of the parents) then replace the weaker parent by the offspring.
  - Make global updation on the best path so far.
- Step4: Iteration i = i+1

- Step5: Go to Step1 until all the ants converges to a single path (i.e shortest path).

The above algorithm can be modified to achieve an improved performance. Global updation can be done after Crossover, RemoveSharp and LocaOpt each. Thrice global updation will cause a higher pheromone level on the shortest path and hence early convergence. Thus we have formulated a framework for solving the problems of similar types as shown below:

- Initialization: iteration i =0
- Step1: Apply ACS to the problem for iteration i.
  - Assume initial pheromone on each path suitably on the basis of the application.
  - Make local updation on each path.
  - Make global updation on the best path so far.
- Select two good parents(paths) from output of Step1.
- Apply modified hybrid GA(Step2 to Step4).
- Step2: Apply Crossover between parents and generate an offspring.
  - If TourCost(offspring) < TourCost (any one of the parents) then replace the weaker parent by the offspring.
  - Make global updation on the best path so far.
- Step3: Apply RemoveSharp algorithm to offspring.
  - If TourCost(offspring) < TourCost (any one of the parents) then replace the weaker parent by the offspring.
  - Make global updation on the best path so far.
- Step4: Apply LocalOpt algorithm to offspring.
  - If TourCost(offspring) < TourCost (any one of the parents) then replace the weaker parent by the offspring.
  - Make global updation on the best path so far.
- Step5: Iteration i = i+1
- Step6: Go to Step1 until all the ants converges to a single path (i.e shortest path).

4. EXPERIMENTAL DETAILS

Let us consider a TSP problem[9] for six cities in a completely connected graph and the distances between the cities are given as in the figure 1.We have considered r as the source and destination node (city) and initialized with 12 ants.

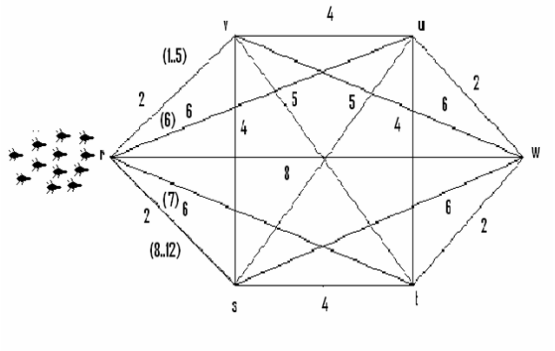

Fig 1

According to the algorithm all ants starts from r and travel all the cities and reaches again at r in different paths causes local updation of pheromone on that paths and a global updation of pheromone on the shortest path out of them. After selecting two promising tours from ACS and applying crossover of GA and other two heuristic functions RemoveSharp& LocalOpt on them, determines the shortest path and results in a global updation each time again. The algorithm proceeds till all ants converge in a shortest path. The implementation was done in Turbo C language on Windows platform in a Pentium 4 machine.

4.1 Complexity Analysis Improved Modular Hybrid ACS verses of Modular Hybrid ACS.

(i). Complexity for step1 (ACS complexity):

Assuming local update takes p unit time and global update takes q unit time.

So Step1 of algorithm 3 having complexity:

$=n(n+1)/2*p$ (for local update) $+n*q$ (for global update) $\approx O(n^2)$.

(ii). Complexity for step2 (Cross over Complexity):

Let each comparison takes l unit of time and adding the city in the edgelist takes m unit of time. So in worst case time complexity $=n*(n*(l+m)) \approx O(n^2)$.

(ii). Complexity for step3 (RemoveSharp Complexity)

Let removing a city from the tour and reinsertion (may be before a node or after anode) takes same unit of time x, m is the size of the NEARLIST and y is the time taken for each comparison. So the complexity is=n*(x+2m*x[for reinsertion before or after a particular node] +2m*y[for comparison]) ≈ O(n). For deadly worst case if m=(n-1),the complexity is $\approx O(n^2)$.

(iii). Complexity for step4 (LocalOpt Complexity)

Time complexity of LocalOpt depends on the value of q, the number of consecutive cities chosen in sequence. So for (q-2)! combinations, in each case (q-1) additions are required to evaluate the cost of sequence and one comparison is to check whether the sequence is minimum or not. Thus n consecutive sequence of q cities having time complexity = n*((q-2)! * (q-1) additions + (q-2)! comparisons) ≈ O(n).

So the overall time complexity of the algorithm is= No of iteration $i*[O(n^2)$ [for ACS]+ $O(n^2)$[for crossover]+ O(n)[for RemoveSharp]+ O(n)[for LocalOpt] $\approx O(n^2)]$.

In case of Improved Modular hybrid ACS the value of i (no of iteration) is dramatically reduced (discussed in section 4.2) than Modular Hybrid ACS below and hence results in improved complexity.

4.2 Experimental Results

The performance graph for the Hybrid GA, Modified Hybrid GA and Modular Hybrid ACS are as shown in the Fig 2..We have used Log Log plot graph to clearly indicate the small performance difference in the Hybrid GA and Modified Hybrid GA. Modular Hybrid ACS have a greater performance relative to others.

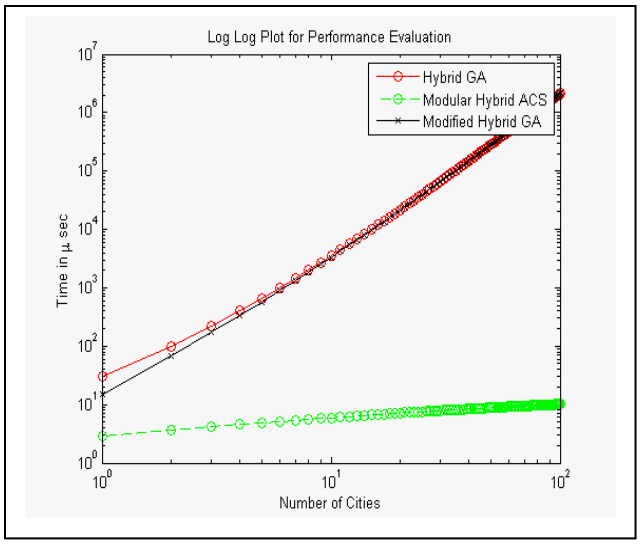

Fig 2

Using Improved Modular Hybrid ACS algorithm on problem of Fig 1 for the traveling salesman requires only one iteration to solve the problem whereas Modular Hybrid ACS requires only two and simple ACS requires three iterations respectively. Table 1 shows the details of the path distances followed by each ant in simple ACS, Modular Hybrid ACS and Improved Modular Hybrid ACS algorithm in each iteration and their convergence. The performance improves in case of large number of cities in comparison with simple ACS and Modular Hybrid ACS.

Table 1

Iteratin Number	No. of Ants converges Using ACS with path length.	No.of Ants convergesUsing Modular hybrid ACS with path-length.	No. of Ants convergesUsing improvedModular hybrid ACS with path length.
1st	T1,2=16, T6=20, T10=18,T3=18, T7=18 , T11,12 =20, T4,5=20, T8,9=16.	T1to4=16,T5=20T6=18,T7to10 =16,T11=20, T12=18.	T1to6=16, T7to12=16. (Converged to shortest path)
2nd	T1to4=16, T5,6=20, T7to10=16, T11,12=20.	T1to6=16, T7to12=16. (Converged to shortest path)	
3rd	T1to6=16,T7to12 =16.(Converged to shortest path)		

The results are shown iteration wise. In the first iteration Improved Modular Hybrid ACS having better performance than Modular Hybrid ACS and simple ACS all ants chooses the shortest path as shown in Fig 3.

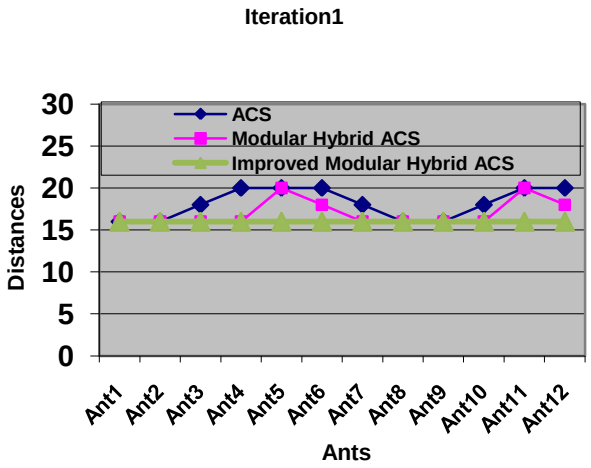

Fig 3

In the second iteration of Modular Hybrid ACS all ants converges to the shortest path whereas in simple ACS only 77.77% ants chooses the shortest path as shown in Fig 4

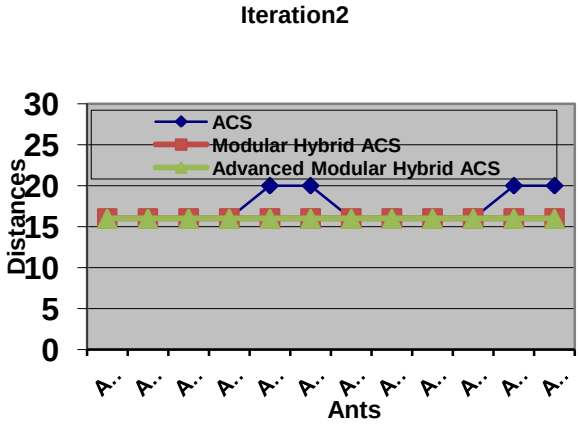

Fig 4

In the third iteration of simple ACS all ants converges to the shortest path as shown in Fig 5.

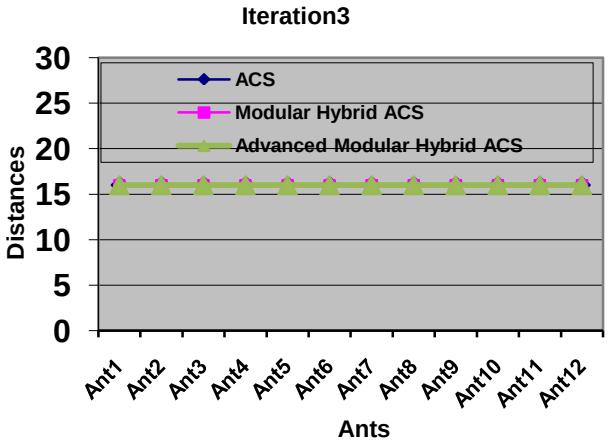

Fig 5

5. CONCLUSION AND FUTURE SCOPE

This paper introduces a new search methodology based on a distributed autocatalytic process and its application to the solution of a classical optimization problem. The general idea underlying the Ant System paradigm is that of a population of agents each guided by an autocatalytic process directed by a greedy force.

The heuristics which are used over here results in near optimal solutions in most of the cases and improvement in a few cases. These heuristics are simple, straightforward and easy to implement as compared to other algorithms. Incorporating these heuristics inside the ACS gives the earlier convergences, hence better performance. The main contributions of this paper are as under.

We employ positive feedback as a search and optimization tool.

1. For initialization the output of ACS is taken which results in a better performance as compared to random initialization.

2. A good starting solutions with local search employs best strategy. The convergence rate is very fast when heuristics are used.

3. Size of NEARLIST can be varied depending on the distribution of the cities.

4. Global pheromone update after applying heuristics leads to the earlier convergence.

We have shown that the ACS is a very powerful and efficient algorithm to provide starting solutions and RemoveSharp & LocalOpt are very good local optimizers. And a proper sequence of combination makes the framework efficient. And thus Improved Modular Hybrid ACS have a greater performance than Simple ACS as well as Modular Hybrid ACS.

This approach can be extended to different problems like vehicle routing problems[10], network routing problems[11], scheduling problems[12], etc.

6. ACKNOWLEDGMENTS

Our thanks to B.I.T (Mesra), Ranchi, India from where we got the support for the research work and special thanks to JoC as the medium to reach this article to the intellectual persons.

REFERENCES

[1]. H. Bersini, C. Oury, and M. Dorigo, "Hybridization of genetic algorithms," *Tech. Rep. No. IRIDIA/95-22*, 1995, Université Libre de Bruxelles, Belgium.

[2]. M. Dorigo and L.M. Gambardella, "A study of some properties of Ant-Q," *Proceedings of PPSN IV–Fourth International Conference on Parallel Problem Solving From Nature*, H.–M.Voigt, W. Ebeling, I. Rechenberg and H.–S. Schwefel (Eds.), Springer-Verlag, Berlin, 1996, pp.656–665.

[3]. Sarayut Nonsiri , Siriporn Supratid,"Modifying Ant Colony Optimization", 2008 IEEE Conference on Soft Computing in Industrial Applications, June 25-27, 2008, Muroran, JAPAN.

[4]. M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization," IEEE Comput. Intell. Mag., vol. 1, no. 4, pp. 28–39, Nov. 2006.

[5].E. L. Lawler, Jan Karel Lenstra, A.H.G. Rinnooy Kan, D.B.Shmoys. The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization . Wiley Press, USA,2007.

[6] Bin Li, Liping Chen, Zhengdong Huang and Yifang Zhong. Productconfiguration optimization using a multiobjective genetic algorithm Volume30, Numbers 1-2 / August, 2006, pp. 20-29.

[7].Jinhui Yang, Xiaohu Shi, Maurizio Marchese, Yanchun Liang, An ant colony optimization method for generalized TSP problem, Progress in Natural Science 18 (11) (2008) 1417-1422.

[8]. G.Jayalakshmi, S.Sathiamoorty andR.Rajaram," A hybrid genetic algorithm–Anew approach to solveTravelling Salesman Problem", *Proceedings of IJCES,2001*.

[9] Lijie Li, Shangyou Ju, Ying Zhang," Improved Ant Colony Optimization for the Traveling Salesman Problem", 2008 International Conference on Intelligent Computation Technology and Automation.

[10]Yu Bin , Yang Zhong-Zhen , Yao Baozhen ," An improved ant colony optimization for vehicle routing problem", European Journal of Operational Research 196 (2009) 171–176.

[11] Sharvani.G.S, N.K. Cauvery, T.M.Rangaswamy," Different types of Swarm Intelligence algorithm for Routing", 2009 International Conference on Advances in Recent Technologies in Communication and Computing, doi:10.1109/ARTCom.2009.157.

[12] Kumar S, Rao CSP. Application of ant colony, genetic algorithm and data mining-based techniques for scheduling. Robot Comput Integr Manuf (2009), doi:10.1016/j.rcim.2009.04.015.

[13] Sudip Kumar Sahana, Dr (Mrs)Aruna Jain," A Modular Hybrid ant Colony approach for travelling Salesman Approach",2010 Annual International Conference on Infocomm Technologies in Competitive Strategies(ICT 2010), October 25-26,2010,MandarinOrchand,Singapore.

Sudip Kumar Sahana received the B.E degree in Computer Technology from Nagpur University, India, in 2001, and the M.Tech. degree in Computer Science in 2006 from the B.I.T (Mesra), Ranchi, India, where he is currently working as Asst. Prof. in the Department of Computer Science and Engineering in addition to continuing Ph.D. degree. His research and teaching interests include soft computing, grid computing, network traffic management and artificial intelligence.

Aruna Jain is working as Reader in the Department of Information Technology, B.I.T (Mesra), Ranchi, India. She is M.Sc (Physics) from Nagpur University, India, in 1983. and M.Tech.(Computer Science) in 1999 from B.I.T (Mesra). She also received her Ph.D.(Engineering) in 2008. Her research and teaching interests include Computer networks, Network security, Soft Computing and Web Engineering. She is author and reviewer of number of research papers in the field of Computer Science.