Survey of solving TSP for by Ant System, Tabu Search and Genetic algorithm

***Abstract*—Ant System(AS) ,Tabu Search (TS) and Genetic Algorithm (GS) are the general algorithm to solve Traveling Salmeman Problem(TSP). In this paper just compare between AS , TS and GS for find iteration and time to solve the problem.**

I. INTRODUCTION

The Traveling Saleman Problem (TSP) is a general problem for testing any algorithm, however , no one can confirm that it's have some algorithm to solve TSP by short time and shortest-path. Ant System(AS) , Tabu Search (TS) and Genetic Algorithm (GA) are the most useful algorithm for solve this problem, then, we should use them to solve it because if we can confirm that what the best algorithm for solving TSP.

2. Review on recent research in algorithm for solving TSP

2.1 Ant System

Ant System (AS)[1] was firstly proposed by three Italy scholars in 1991, Dorigo, Colorni and Maniezzo, and has received increasing attention from researchers, having been used to solve many difficult problems in optimization of discrete systems. It is a bionic optimization algorithm which simulated intelligence behavior of ant colony in insect kingdom. It has some merits such as strong robustness, distributed computation and the use of a constructive greedy heuristic , as a novelty optimization algorithm, not liked Genetic Algorithm and Simulated Annealing Algorithm, ACA has not formed systematic analysis approach and the solid mathematic foundation, many theory questions wait for further studying, for example, long the search time, slow convergent speed and low searching efficiency and cannot expand the hunting zone etc. In view of these flaws, in recent years the domestic and foreign scholars proposed the massive corrective methods to the ant colony algorithm, for example, Max-Min Ant System (MMAS)[2] was put forwarded by T. Stutzle and H.H. Hoos, and the Best-Worst Ant System(BWAS)[3] proposed by literature. These improvement algorithms have played a certain promotion to enhance the performance of ant system. In this survey, surveyors are classifying AS in to three groups that used to solve TSP

Classification of Ant System which used to solve TSP

There are many researches that use varies way of AS in solving TSP, some are improved from the main Ant Colony Optimization (ACO)[4], and some are use a hybrid algorithm between ACO and other genetic algorithms. In this survey, represent three groups of AS which base on behavior of researching methods ; Improved AS, Max-Min Ant System(MMAS), Ant Colony System(ACS).

A. Ant System (AS)

As mentioned above, AS is the first ACO algorithm presented, and it is the original version of various ant algorithms as well. An iteration is defined as the interval in (t, t+1) during which each of the m ants constructs a solution. In other words, every ant traversals all of the n cities at each iteration. AS together with other ant algorithms employs the tabu table to record the already visited cities of an iteration in order to obtain feasible answers. In the construction of a solution, ants choose the next city to be visited via certain search strategy which is a stochastic mechanism. When ant k is in city i, the probability of going to city j is given by:

$$p_{ij}^{k}=\begin{cases}\dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l\notin tabu_k}\tau_{il}^{\alpha}\eta_{il}^{\beta}} & if\ \ j\notin tabu_k,\\ 0 & otherwise,\end{cases} \qquad (1)$$

where $tabu_k$ is the tabu table of ant k which is comprised of the cities that have been visited in the current iteration. The parameters α and β control the relative importance of the pheromone τ_{ij} versus the visibility η_{ij}. The visibility represents the heuristic information, which is given by:

$$\eta_{ij}=\frac{1}{d_{ij}}, \qquad (2)$$

where d_{ij} denotes the distance between city i and city j. This transition rule is also shared by other algorithms. After each iteration, the pheromone τij linked to the path connecting city i and j, is updated according the following formula:

$$\tau_{ij}(t+1)=(1-\rho)\cdot\tau_{ij}(t)+\sum_{k=1}^{m}\Delta\tau_{ij}^{k}, \qquad (3)$$

where ρ is the evaporation rate, m is the number of ants, and $\Delta\tau_{ij}^{k}$ is the quantity of pheromone laid on path(i,j) by ant k:

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k} & if\ ant\ k\ used\ path(i,j)\ in\ its\ tour,\\ 0 & otherwise,\end{cases} \quad (4)$$

where Q is a constant and L_k is the tour length of ant k.

2.2 Moderate Ant System (MAS)

MAS[5] is a improve version based on AS, differs from AS only in search strategy (i.e., transition rule). The pheromone concentration determines the choices of ants in real world, while in ACO algorithms combination of pheromone and visibility determines the choices of ants. Hence, $infomation_{ij}$ is used to represent the integrated factors on the path from city i to city j, and $infomation_{ij}$ is given by:

$$infomation_{ij}=\tau_{ij}^{\alpha}\eta_{ij}^{\beta}.$$

Drawing on the search strategy of Monomorium ant species, the new transition rule should make the ants tend to select paths with moderate *information* instead of too much or too little *information*. Suppose that ant k is at city i at step t of an iteration. When selecting the next city as part of the solution, ant k has n-t adjacent cities available. According to Figure 3, from city i's point of view, the *information* values on the n-t adjacent paths can be regarded as a stochastic variable follow a normal distribution $N(\mu,\sigma 2)$ since n in TSP is usually a big enough number. The parameters of $N(\mu,\sigma 2)$ are estimated as follow:

$$\mu=\frac{1}{n-t}\cdot\sum_{j\notin tabu_k} infomation_{ij},$$

$$\sigma^2=\frac{1}{n-t}\cdot\sum_{j\notin tabu_k}(infomation_{ij}-\mu)^2.$$

In addition, $attraction_{ij}$ is defined as follows:

$$attraction_{ij}=\begin{cases}\dfrac{1}{\sqrt{2\pi}\sigma}\cdot e^{-\frac{(info_{ij}-\mu)^2}{2\sigma^2}} & if\ j\notin tabu_k,\\ 0 & otherwise,\end{cases}$$

which denotes the attraction of path from city i to city j for the ants in MAS. If let k be an ant located at city i, $q0\in[0,1]$ (default is 0.5 in MAS) be a parameter, and q a random value in [0, 1], then the next city j is selected according to the following formula:

If $q \le q0$:

$$p_{ij}^{k}=\begin{cases}1 & if\ j=\arg\max_{j\notin tabu_k} arrtration_{ij},\\ 0 & otherwise,\end{cases}$$

else ($q > q0$) and (1) is used.

An Adaptive Dynamic Ant System Based on Acceleration for TSP (ADAS)

[6] It is well known the pheromone in cities decide ants' routing selections. So the parameters' values that affect the pheromone are the key. In AS, there are many parameters such as α, β, ρ, Q and m. α affects the global best solutions to get and β does the convergence speed. ρ, Q and m affects both of the global best and convergence speed but are not discussed in this paper. In ACS, they are const and isolated in algorithm processing. The parameters' static and isolatism are the key factors that the global best solution is hard to arrive an simulation time are very large in ACS. So in the following part, we narrate a method to make the parameters interrelated in order to get the best balance between the global best and convergence speed.

Transition probability

Each ant k can probabilistically decide the next city to move to, according to formula :

$$P_{ij}^{k}(t)=\begin{cases}\dfrac{\tau_{ij}^{\alpha(t)}(t)\bullet\eta_{ij}^{\beta(t)}(t)}{\sum_{z\in allowed_k}\tau_{iz}^{\alpha(t)}(t)\bullet\eta_{iz}^{\beta(t)}(t)} & if\quad j\in allowed_k\\ 0 & otherwise\end{cases}$$

Where $\alpha(t)$ is the heuristic factor and $\beta(t)$ is the expected heuristic factor of cycle t, and they are defined as the following formula. The other parameters are defined as above.

$$\begin{cases}\begin{cases}\alpha(t+n)=\alpha(t)-1/2A_{OD}(t)Z^2\\ \beta(t+n)=\beta(t)+1/2A_{OD}(t)Z^2\end{cases} & \text{if } E_o(t\text{-}1)>\Delta E_2\\ \begin{cases}\alpha(t+n)=\alpha(t)-1/2A_{OD}(t)Z^2\\ \beta(t+n)=\beta(t)+1/2A_{OD}(t)Z^2\end{cases} & \text{if } E_o(t\text{-}1)<\Delta E_1\\ \begin{cases}\alpha(t+n)=\alpha(t)\\ \beta(t+n)=\beta(t)\end{cases} & \text{else}\end{cases}$$

Where $Z=1/t$ which ensures that the larger t quantity the greater change $\alpha(t)$ and $\beta(t)$ have *to get better solution*,

$$A_{OD}(t)=\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}(t)\ \text{ and }\ E_O(t)=\frac{1}{n}\sum_{i=1}^{n}E_i(t).$$

Formula (8) shows that the ability of the global best should be enhanced and the convergence speed should be deduced when $E_O(t)<\Delta E_1$, and the aim is opposite when $E_O(t)>\Delta E_2$.

Global pheromone update

The pheromone trails update is modified as the following formula.

$$\tau_{ij}(t+n)=\rho_i(t+n)\tau_{ij}(t)+\Delta\tau_{ij} \quad (10)$$

Where:

①
$$\rho_i(t+n)=\begin{cases}A_{OD}(t)\rho_0 & if\quad E_i(t)>\Delta E_2\\ (1-A_{OD})(t)\rho_0 & if\quad E_i(t)<\Delta E_1\\ \rho_0 & if\quad \Delta E_1<E_i(t)<\Delta E_2\end{cases}$$

Where $\rho_0(0<\rho_0<1)$ is a user-defined parameter, and the other variables are defined as above mentioned.

$$② \ \Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

Where:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q_{ij}(t+n)/L_k, & \text{if the } k-th \text{ ant walks along} \\ & \text{the routing(i, j) in its tour} \\ & \text{during time interval(t, t + n)} \\ 0, & otherwise \end{cases}$$

$Q_{ij}(t+n)$ will be changed on the $Lev_{ij}(t)$ *of routing (i,j),*and given by the following formula :

$$Q_{ij}(t+n) = \begin{cases} Q_{ij}(t)(1-A_{OD}(t)\bullet Z) & if \quad lev_{ij}(t) \le \frac{1}{n}\sum_{j=1}^{n} lev_{i,j}(t) \\ Q_{ij}(t)(1+A_{OD}(t)\bullet Z) & if \quad lev_{ij}(t) > \frac{1}{n}\sum_{j=1}^{n} lev_{i,j}(t) \\ Q_{ij}(t) & \text{else} \end{cases}$$

Where: $Z=1/d_{ij}$, n is the city quantity.

The complexity of the ADAS is $0(NC\bullet n^2\bullet m)$ if we stop the algorithm after NC cycles. In fact step 1 is $0(n^2+ m)$, step 2 is 0(m), step 3 is $0(n^2\bullet m)$, step 4 is O(n2•m), step5 is $0(n^2)$, step 6 is 0(n•m). Since we have experimentally found a linear relation between the number of towns and the best number of ants, the complexity of the algorithm is $0(NC\bullet n^3)$. So the complexity of the ADAS is as same as the AS.

2.3 Solving the traveling salesman problem using cooperative genetic ant systems (CGAS)

[7]Unlike those hybrid intelligent ACO algorithms proposed previously, in which GA or PSO played only a partial role in fine tuning the parameters for accelerating ACO convergence, we here propose a new algorithm that is designed to execute both AS and GA concurrently and cooperatively. This algorithm maximizes the advantages offered by both AS and GA independently, which provides a better chance in reaching the global optimal solutions to TSPs. A generic expression of CGAS is given in following Algorithm :

```
Set parameters; initialize pheromone trails
Construct FirstAntSolutions (FAS) by AS
Initialize GA using FAS
while termination condition not met do
  Construct NewAntSolutions (NAS) by AS
  Get NewAntGeneration (NAG) by GA
  Select NewBestSolution (NBS) from NAS and NAG
  Update Pheromones
endwhile
Terminate with NBS as the solution
```

Although this generic algorithm looks simple, it requires a good understanding of some key concepts adopted in this algorithm which differ CGAS from others. Firstly, the strategy of selecting the next city for an ant to visit is based on natural ordering and selection. In each city, we create a sorted list of a certain number of cities that are the closest to it. In a natural selection process, a closer city has a higher probability to be chosen for the next move. To make the use of this list more efficient, this sorted list should only contain a certain number of closest cities. This constant (C_0) depends on the number (n) of cities for a particular TSP, and not exclusive, a rough guide would be C_0 = (5% - 15%) * n.

For ant k in city i to select the next city j to visit, ant k will firstly consult the sorted list c(i) of city i to select the closest city from it. If this city is in set S_k, it will be the next city to visit. If c(i) has no intersection with S_k, then the traditional mechanism of AS is used to select a city outside the list. This selection process is summarized as:

$$j = \begin{cases} \min c(i) & if \ j \in S_k, \arg\max \tau_{ij}\eta_{ij}^{\beta} otherwise. \end{cases}$$

Since this natural ordering and selection strategy is adopted for choosing the next city to visit during iteration, local pheromone update is no longer required.

The solution (sequence of cities visited) of each ant's first iteration constitutes a component of the initial population for GA. Afterwards AS and GA are executed concurrently in the while loop. In the end of each iteration, the two best solutions with the shortest length of tour from both AS (A_{best}) and GA (G_{best}) are compared to determine the shortest path for the round. Whichever is shorter, it is used to replace the other. This keeps the best solution of each round evolving during the entire process.

B. Max-Min Ant System (MMAS)

MMAS[2] differs from the AS in two main aspects: only the best ant is allowed to update the pheromone trails, and the value of pheromone on the paths is bound. The pheromone update function is implemented as follows:

$$\tau_{ij}(t+1) = \left[(1-\rho)\cdot\tau_{ij}(t) + \Delta\tau_{ij}^{best}\right]_{\tau_{\min}}^{\tau_{\max}},$$

where τmax and τmin are the upper bound and lower bound of the amount of pheromone on the paths, respectively; the operator $[x]_b^a$ is defined as follows:

$$[x]_b^a = \begin{cases} a & if \ x > a, \\ b & if \ x < b, \\ x & otherwise; \end{cases}$$

and $\Delta\tau_{ij}^{best}$ is:

$$\Delta\tau_{ij}^{best} = \begin{cases} \frac{1}{L_{best}} & if \ the \ global \ best \ solution \ contains \ path(i, j), \\ 0 & otherwise. \end{cases}$$

L_{best} is the length of the tour obtained by the best ant. It can be the best solution found in the current iteration—iterationbest, L_{ib} or the best solution found since the beginning of the algorithm—global-best, L_{gb} , or a combination of both (subject to the algorithm designer's decision).

With respect to the upper and lower bounds on the pheromone values, τ_{max} and τ_{min} tend

to be set empirically and are determined on a case-by-case basis. Nevertheless, there are still some guide lines that have been proposed for defining τ_{max} and τ_{min} on the basis of analytical considerations.

2.4 A Novel Max-Min Ant System Algorithm (NMMAS)

In MMAS [2], the upper bound τ_{max} is set to an estimate of the asymptotically maximum value, and the lower bound τ_{min} is set to $\varepsilon\tau_{max}$ where $0 < \varepsilon < 1$. For TSP, They are set by

$$\tau_{\max} = \frac{1}{1-\rho}\frac{1}{f(s^{gb})}$$

$$\tau_{\min} = \frac{\tau_{\max}(1-\sqrt[n]{p_{best}})}{(avg-1)\sqrt[n]{p_{best}}}$$

Where $avg = n/2$, p_{best} is significantly higher than 0.5.

From formula above, it can be found that the upper and lower bounds of pheromone trail are relevant to the value of objective function. They have to reset each time when a new global-best solution is found. It is one of inconvenience for practical problem. Another is using only one solution for the pheromone update. By this choice, solution elements which frequently occur in the best found solutions get a large reinforcement, that is, it is useful the ants exploit the solution space. But, it limits the capabilities of exploration as the search concentrate too fast. It may lead to trap in poor quality solutions. So we introduce a novel Max-Min ant system, called NMMAS[8], to overcome the shortage of MMAS.

Pheromone Trail Limits

In experiment, we find out that the pheromone trail value can be independence of the value of objective function. But, to achieve better performance it has something to do with the heuristic value. If the gap between the pheromone trail value and the heuristic value is too large, the role of heuristic value in random proportional rule will be lose. The heuristic value in TSP is defined as the inverse of distance. So in NMMAS, one random sampling is needed to obtain the length of tour, and the interval of pheromone is determined.

From formula above, the ratio of the upper bound and the lower bound is over 1000. So the lower bound of pheromone trail is set to $m_1\tau_{max}$, where m_1 is a real constant number, $0 < m_1 < 1$. Then, if upper bound which has nothing to do with the value of objective function, then the limit of pheromone trail value is also too. In practice, the length of tour is corresponded to one interval. At the beginning of algorithm, random sampling is used to get the length. Then, we can choice the interval though the range of length. For example, if the length $L <= 2000$, then $\tau_{max} = 0.1$; if $2000 < L <= 20000$, $\tau_{max} = 0.01$; …

Pheromone Trail Update

An effective ACO has to achieve an appropriate balance the exploitation and the exploration. To get a better solution, algorithm needs to explore the solution space as much as possible at the beginning phase. The more information we gather at the start of the algorithm, the better solution we obtain at the end of the algorithm. But, using only one solution to update pheromone is not conducive to explore. Although the quality of solution is less than the iteration-best solution's in the iteration, it may contain the useful information which can help us find optimal solution. So pheromone is updated by two phases in NMMAS. Let *N*max be a maximum number of algorithm iterations, $0 < m2 < 1$ be a real number, and *NC* be an iteration number. In first phase: *NC* < *m*2*N*max, the first *l* solutions in one iteration are used to update pheromone trail, called update by order. Then, in phase two: NC < *m*2*N*max, only the iteration-best solution is used to update pheromone. Using first *l* solutions for the pheromone update is the most important means of exploration in NMMAS. The pheromone trail updating by order is done according to the following formula:

$$\tau_{ij}(t+1) = [(1-\rho)\tau_{ij}(t) + \sum_{\omega=1}^{l}\Delta\tau_{ij}^{\omega}(t)]\begin{matrix}\tau_{\max}\\ \tau_{\min}\end{matrix}$$

Where $\sum_{\omega=1}^{l}\Delta\tau_{ij}^{\omega}(t)$ is the first *l* ant updating the pheromone according to the rank which is sorted by the tour length.

$\Delta\tau_{ij}^{\omega}(t)$ is defined as follows:

$$\Delta\tau_{ij}^{\omega}(t) = \begin{cases}\dfrac{\rho\tau_{\max}}{\omega} & \text{if ant } \omega \text{ travel on edge}(i,j)\\ 0 & \text{otherwise}\end{cases}$$

Where $\Delta\tau_{ij}^{\omega}(t)$ is the quantity of updating which caused by ant □ on edge (i, j) . □ is the order of solution *S*_ in this iteration. It is shown that the increasing volume of pheromone is also independent the value of objective function. This is different from the mechanism of MMAS. In order to maintain consistency of algorithm, in phase $\Delta\tau_{ij}^{\text{best}}(t)$ is defined as follows:

$$\Delta\tau_{ij}^{\text{best}}(t) = \begin{cases}\rho\tau_{\max} & \text{if edge belongs to } S^{\text{best}}\\ 0 & \text{otherwise}\end{cases}$$

Pheromone Trail Initialization

In NMMAS, we initialize the pheromone trail to the half of τ_{max}, that is $\_0 = 0.5\tau_{max}$, which differs from the choice chosen in MMAS. With the

experimental investigation in TSP, choice of the half of τmax is better than the τmax's. The experimental results presented in next section confirm the conjecture that the larger exploration of the search space due to setting τ0 = 0.5τmax improves MMAS' performance.

C. Ant Colony System (ACS)

The most significant contribution of ACS is the introduction of a local pheromone update in addtion to the pheromone update applied at the end of the construction process, which is more in line with the natural behavior of real ants.

The local pheromone update is executed by all the ants after each construction step of the *n* steps of an iteration. Moreover, each ant applies it only the last path which has just been traversed:

$$\tau_{ij} = (1-\varphi)\cdot\tau_{ij} + \varphi\cdot\tau_0 ,$$

where $\varphi \in (0, 1]$ is the pheromone decay coefficient, and τ_0 is the initial value of the pheromone on the path.

Similar to the MMAS algorithm, the pheromone is also updated at the end of each iteration by the iteration-best or the global-best ant only. However, the update function is slightly different:

$$\tau_{ij}(t+1) = \begin{cases} (1-\rho)\cdot\tau_{ij}(t) + \rho\cdot\Delta\tau_{ij} & \text{if the global best} \\ & \text{contains path}(i, j) \\ \Delta\tau_{ij} & \text{otherse,} \end{cases}$$

where

$$\Delta\tau_{ij} = \frac{1}{L_{best}} .$$

As in MMAS, L_{best} can be either L_{ib} or L_{gb}.

ACS also uses a different transisiton rule, called the pseudorandom proportional rule. If let *k* be an ant located at city *i*, $q0 \in [0, 1]$ be a parameter, and *q* a random value in [0,1], then the next city *j* is selected according to the following formula:

if $q \le q0$:

$$p_{ij}^k = \begin{cases} 1 \;\; if \;\; j = \arg\max_{j \notin tabu_k} \tau_{ij}\cdot\eta_{ij}^{\beta}, \\ 0 \;\; otherwise, \end{cases}$$

else ($q > q0$) and (1) is used.

Additionally, ACS employs the candidate lists to make the selections prefer some nearer cities during the construction process.

2.5 Improved Ant Colony Optimization (HK-ACO)

An improved ant colony optimization(HK-ACO)[9] by using Held-Karp low bound to dynamically control the balance between intensification and diversification.

Probability Selection Mechanism

From TSP equation, α and β are two parameters which determine the relative influence of the pheromone trail and the heuristic information. The role of the parameters α and β is analyzed in the following. Supposed α = 0, the TSP equation turn to be following equation:

$$p_{ij}^k = \frac{\eta_{ij}^{\beta}}{\sum_{l\in N_i^k}\eta_{ij}^{\beta}}$$

From the equation 1, the selection probability increases with the increase of β . Furthermore, here $\eta_{ij} = 1/d_{ij}$ is an a priori available heuristic value. Therefore, If α = 0, the closest cities are more likely to be selected. This corresponds to a classical stochastic greedy algorithm. On the other hand, suppose β = 0, the equation 1 turn to be following equation.

$$p_{ij}^k = \frac{[\tau_{ij}(t)]^{\alpha}}{\sum_{l\in N_i^k}[\tau_{ij}(t)]^{\alpha}}$$

From above equation, if β = 0, only pheromone amplification is at work. This method will lead to the rapid convergence of a stagnation situation with the corresponding generation of tours which, in general, are strongly suboptimal. Hence a trade-off between the influence of the heuristic information and the pheromone trail exists. In order to achieve the tradeoff between heuristic information and the pheromone, this paper introduces a new approach to adjust the parameters α and β according the following equation:

$$\alpha = C - d_{ij}/(L_{bsp} + d_{ij} + L_{rsp})$$

$$\beta = B + d_{ij}/(L_{bsp} + d_{ij} + L_{rsp})$$

In equation above, parameter C and B are constant, and set to be 1, 5 correspondingly that are suggested to be advantageous. L_{bsp} is the length of subtour visited by ant. L_{rsp} is the length of subtour not visited by ant except city i and j. $d_{ij}/(L_{bsp} + d_{ij} + L_{rsp})$ express the proportion of the selected edge to the optimal length. The tour is divided to three parts, which are demonstrated in the figure 1. Thus, the influence between the heuristic information and the pheromone trail can be adjusted according to the equation previous.

Held-Karp lower bound procedure

L_{rsp} is the length of city unvisited by ant and the length is unknown. In the proposed selection mechanism defined in previous equation, the L_{rsp} needn't calculate explicitly. Instead, the Held-Karp lower bound is applied to estimate the proximate value. The Held-Karp lower bound provides a very good problem-specific estimation of optimal tour length for the traveling salesman problem. This measure, which is relatively quick and easy to compute, has enormous practical value when evaluating the quality of near optimal solutions for large problems where the true optima are not known. The Held-Karp procedure used for

the experiments described in this paper is described in detail in second Algorithm. In the Held-Karp procedure, the Volgenant-Jonker formula is used for experiments to determine how many near neighbors of each city should be stored for the traveling salesman problem. The idea is to minimize the number of edges required in the subgraph for the main iteration scheme, whilst still getting a good estimate of the Held-Karp lower bound. Ideally the best value obtained from the iteration sequence on the subgraph should equal the final value from the $O(n^2)$ 1-tree calculation.

2.6 Ant Colony Optimization for the Traveling Salesman Problem Based on Ants with Memory

Being different from the preexistent attempts to Ant System, we are trying to introduce a novel character into the agent. We let the new ants memorize the best-so-far solution. This model was called Ants with Memory (Mant)[10]. Generally speaking, the Mant can be used in any version of ACO algorithms. In Mant algorithm, we choose random part of the best-so-far solution and make it as the new solution proportion. The Mant based on ACO algorithm is given by table:

Initialize parameters, pheromone trails,
While (termination condition not met)**Do**
  While (not first loop)**Do**
    Generate random number i, j(0<i<j<n),
    Put visited tag from city i+1 to j-1
    Make i as start city and j as end city
  Construct solutions as normal ACO
  Apply local search(could use 2-opt/3-opt etc.)
  Update pheromone trails
  End
End

Probabilistic Mant

Aimed at weaken the memory property of Mant, We introduce the probabilistic strategy into Mant. That is, we control the Mant use their memory property by a probability as used by Dorigo in ACS[4][5](it is called pseudorandom proportional rule). We call the new amended model of Mant with probability p1 as MantP1, for better illustration, we also provide Mant with probability p2 as MantP2, here p2=1-p1. the detailed algorithm shows in table:

Initialize parameters, pheromone trails,
While (termination condition not met)**Do**
  While (not first loop)**Do**
    Generate random number q,
    If ($q< q_0$), construct solutions as Mant
    Else, construct solutions as normal ACO
  Apply local search(could use 2-opt/3-opt etc.)
  Update pheromone trails
  End
End

In MantP2 algorithm, just switch ($q<q_0$) into ($q>q_0$) in MantP1 algorithm.

2.7 An improvement of the ant colony optimization algorithm for solving Travelling Salesman Problem

[11]The probability of some paths that are likely to be selected becomes low gradually when the algorithm implements for a certain period. In order to increase the possibility of exploring these paths, the pheromone needs to be re-initialized. At this time, the pheromone of each path could be divided into two types: the pheromone concentration of path is close to the pheromone concentration of current optimal path, and there is great disparity between the two pheromone concentrations, which shows that the ant cannot find a better path than current optimal path. Therefore, there are two types how to re-initialize the pheromone concentration. The pheromone concentration must be re-initialized when the algorithm is implementing for a certain time, since the shortest path is more or less fix onto the only one path that is probably not the best one. The pheromone concentration of a set of paths, less than $(\tau_{min}+\tau_{max})/2$, is set to τ_{min}, this because we can make sure that there is no optimal path in these paths. On the contrary, the one, greater than $(\tau_{min}+\tau_{max})/2$, is set to $(\tau_{min}+\tau_{max})/2$, for the optimal is probably found in these paths. Thus, the speed of process searching the optimal path is accelerated by re-initializing pheromone concentration separately.

The improved algorithm, based on MMAS algorithm, can be described as following steps:

Initialize

At time zero the parameter and the pheromone concentration of each path are initialized (the pheromone concentrations have the same initial value) during which m ants are positioned on n towns. The starting town and the nodes allowed to visit are initialized for each ant.

Iteration

FOR p=1 TO n //traverse all nodes, and come back to the starting nodes.

FOR k=1 TO m // one-step transfer of ant colony

{ assume that the ant is located at node i

IF p<n THEN //not all the towns are traversed.

{the ant moves to next town according to the State Transition formula}

ELSE //traverse all nodes

}

Update the whole pheromone of this iteration

FOR k=1 TO m

{ compute } //compute the ant's travel length in this iteration, and update the whole pheromone

Re-initialization after several iterations

IF (satisfy the condition of initialization) THEN

Reset the pheromone concentration of each path

according to the rule presented above, and go into the next iteration

ELSE

{ add 1 to the number of iteration, initialize the node list

allowed to visit of each node

GOTO step 2 //go into the next iteration

}

2.8 Solving TSP using Tabu search algorithm

1. tabu search(TS)
2. tabu insertion search(TIS)

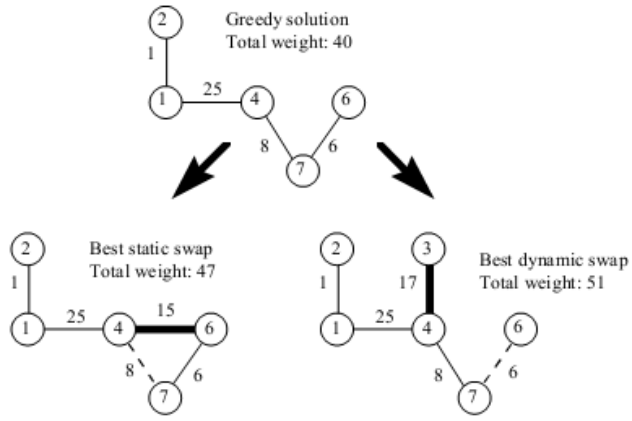
basic Tabu method

Tabu search begins in the same way as ordinary local or neighborhood search, proceeding iteratively from one point (solution) to another until a chosen termination criterion is satisfied.

Each x ∈X has an associated neighborhood N(x) ⊂ X , and each solution x ′ ∈ N(x) is reached from x by an operation called a move.

As an initial point of departure, we may contrast TS with a simple descent method where the goal is to minimize f(x) (or a corresponding ascent method where the goal is to maximize f(x)). Such a method only permits moves to neighbor solutions that improve the current objective function value and ends when no improving solutions can be found. A pseudo-code of a generic descent method is presented in Figure 2.1. The final x obtained by a descent method is called a local optimum, since it is at least as good or better than all solutions in its neighborhood. The evident shortcoming of a descent method is that such a local optimum in most cases will not be a global optimum, i.e., it usually will not minimize f(x) over all x ∈X .

1) Choose $x \in \mathbf{X}$ to start the process.
2) Find $x' \in \mathbf{N}(x)$ such that $f(x') < f(x)$.
3) If no such x' can be found, x is the local optimum and the method stops.
4) Otherwise, designate x' to be the new x and go to 2).



Insertion Method

1.Select two cities v_i, v_k randomly from $v = \{v_i, v\sim, \dots, v_j\}, t = \{v_i, v_k, v_i\}$

2.If T=V, stop; else determine v_j (v_j £ T), and two consecutive vertices v_i,v_k(v_i,v_k E T) such that (d_{ij}+d_{jk}-d_{ik}) is minimal

3.Insert v_j between v_i,v_k, and get a new tour T={ ..., v_i,v_j,v_k ,.__ }, repeat step2)

2.9 Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator

Sequential Constructive crossover(SCX), for a genetic algorithm that generates high quality solutions to the Traveling Salesman Problem (TSP). The sequential constructive crossover operator constructs an offspring from a pair of parents using better edges on the basis of their values that may be present in the parents' structure maintaining the sequence of nodes in the parent chromosomes. The efficiency of the SCX is compared as against some existing crossover operators; namely, edge recombination crossover (ERX) and generalized N-point crossover (GNX) for some benchmark TSPLIB instances.

Genetic Algorithm

Genetic algorithms (GAs) are based essentially on mimicking the survival of the fittest among the species generated by random changes in the gene-structure of the chromosomes in the evolutionary biology. In order to solve any real life problem by GA, two main requirements are to be satisfied:

(a) a string can represent a solution of the solution space, and

(b) an objective function and hence a fitness function which measures the goodness of a solution can be constructed / defined.

A simple GA works by randomly generating an initial population of strings, which is referred as gene pool and then applying (possibly three) operators to create new, and hopefully, better populations as successive generations. The first operator is reproduction where strings are copied to the next generation with some probability based on their objective function value. The second operator is crossover where randomly selected pairs of strings are mated, creating new strings. The third operator, mutation, is the occasional random alteration of the value at a string position. The crossover operator together with reproduction is the most powerful process in the GA search. Mutation diversifies the search space and protects from loss of genetic material that can be caused by reproduction and crossover. So, the probability of applying mutation is set very low, whereas the probability of crossover is set very high.

Sequential constructive crossover operator (SCX)

The search of the solution space is done by creating new chromosomes from old ones. The most important search process is crossover. Firstly, a pair of parents is randomly selected from the mating pool. Secondly, a point, called crossover site, along their common length is randomly selected, and the information after the crossover site of the two parent strings are swapped, thus creating two new children. Of course, this basic crossover method does not support for the TSP.

The sequential constructive crossover (SCX) operator constructs an offspring using better edges on the basis of their values present in the parents' structure. It also uses the better edges, which are present neither in the parents' structure. As the ERX and GNX, the SCX does not depend only on the parents' structure; it sometimes introduces new, but good, edges to the offspring, which are not even present in the present population. Hence, the chances of producing a better offspring are more than those of ERX and GNX. A preliminary version of the operator is reported as local improvement technique.

3. Implementation and performance evaluation

3.1 Ant algorithm

Comparison of various algorithms in optimal solution path of TSP cases.

TSP Case / Algorithm	Oliver30	Att48	Berlin52	Eil51	Eil75	D198
AS	431.67	35061.36	7683.43	434	546	16702.1
MAS				441		17192
ADAS				421	533	15903
CGAS			7542		538	
MMAS	423.74	35061.36	7665.34	427.6		15972.5
NMMAS				427.2		15924.4
ACS	423	33981		426	542	15780.5
HKACO						15780
MAnt						15780.8
MAntP1						15780.7
MAntP2						15781.1
Improvement ACO	423.74	35061.36	7545.48			

TSP Case / Algorithm	KroA100	KroA200	Att532	Rat783	Pcb1173	Pr2392	Pcb3038
AS	21429		29064				
MAS			32043	10302	69318	456459	167507
ADAS	21280		28131				
CGAS	21282	29368					
MMAS	21320.3		32387	10421	68719	458502	168967
NMMAS	21298.2						
ACS			33540	8911.9		387686.9	235131
HKACO			27686	8808	56897		
MAnt				8907.3		388345.6	
MAntP1				8906.2		387634.7	
MAntP2				8910.7		388270.3	
Improvement ACO							

3.2 Tabu Search algorithm

compare result Between TIS and TS

Problem	Problem Size	Best olution published	Method	Probability of Obtain the Optimum	Convergence Speed (Iteration steps)		
					min	max	Average
Gr17	17	2085	TIS	100%	5	3406	1372.8
			TS	6%	5	2740	1080.3
Bays29	29	2020	TIS	100%	29	8764	2196.6
			TS	49%	29	9678	3206.4
ctsp	31	15404	TIS	100%	60	11907	2706
			TS	95%	60	16904	3933.7

3.3 GENETIC ALGORITHMS

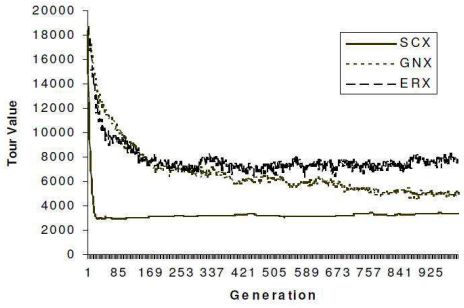
COMPUTATIONAL EXPERIMENTS

For comparing the efficiency of the different crossover operators, genetic algorithms using SCX, ERX and GNX have been encoded in Visual C++ on a Pentium 4 personal computer with speed 3 GHz and 448 MB RAM under MS Windows XP, and for some TSPLIB instances. Initial population is generated randomly. The following common parameters are selected for the algorithms: population size is 200, probabiliy of crossover is 1.0 (i.e., 100%), probability of mutation is 0.01 (i.e., 1%), and maximum of 10,000 generations as the terminating condition. The experiments were performed 10 times for each instance.

Instance	n	Opt. Sol.	ERX			GNX			SCX		
			Best (%)	Avg (%)	Avg Time	Best (%)	Avg (%)	Avg Time	Best (%)	Avg (%)	Avg Time
br17	17	39	0.00	0.00	2.10	0.00	0.00	0.26	0.00	0.00	0.11
ftv33	34	1286	1.94	4.98	70.22	15.47	19.49	7.64	0.00	3.58	2.25
ftv35	36	1473	3.19	5.20	76.39	17.58	18.56	1.48	0.00	0.59	9.69
ftv38	39	1530	4.38	5.45	160.87	6.99	13.18	4.03	0.24	0.46	6.89
p43	43	5620	1.44	1.89	213.99	2.46	2.58	22.33	0.05	0.10	22.98
ftv44	45	1613	5.46	6.39	157.23	14.01	15.71	18.46	0.62	0.93	19.22
ftv47	48	1776	5.97	8.48	200.70	20.10	20.38	42.67	0.51	1.73	25.99
ry48p	48	14422	2.04	2.31	185.59	15.18	18.13	39.33	0.59	0.60	25.73
ft53	53	6905	18.03	19.51	122.75	18.29	23.33	29.35	1.03	1.77	36.73
ftv55	56	1608	13.18	14.41	328.59	22.51	24.71	23.74	0.62	1.45	35.11
ftv64	65	1839	25.24	27.48	326.95	25.23	29.87	91.39	0.49	1.54	76.56
ft70	70	38673	10.40	10.56	561.14	6.53	7.81	90.13	0.70	0.86	74.19
ftv70	71	1950	30.41	34.56	432.31	20.72	23.04	135.97	2.15	2.75	58.69
kro124p	100	36230	30.96	37.15	542.57	25.72	28.58	178.64	4.24	4.93	142.02
ftv170	171	2755	62.45	66.15	526.46	51.00	60.69	483.21	6.13	8.93	259.60

Summary of the results by the crossover operators for asymmetric TSPLIB instances.

This picture gives the result for fifteen asymmetric TSPLIB instances of size from 17 to 171. The solution quality of the algorithms is insensitive to the number of runs. Only one instance, br17 of size 17, could be solved exactly by ERX and GNX, whereas three instances, br17, ftv33 and ftv35, could be solved exactly, at least once in ten runs, by SCX. Between ERX and GNX, on the basis of quality of best solution value and average solution value, for the instances from ftv33 to ftv64, ERX is found to be better; but for four instances ft70, ftv70, kro124p and ftv170, GNX is found to be better. That means, as size of the problem increases GNX is found to be better than ERX. It is to be noted that we have implemented only the original versions of ERX and GNX for the comparative study. On the basis of
quality of the solution, as a whole, for all the instances SCX is found to be the best one. On the basisof time of convergence, GNX is found to be better than ERX, and SCX is the best one.



Performance of different crossover operators on the instance ftv170.

This picture shows performance of different crossover operators for the instance ftv170 (considering only 1000 generations). All crossover operators have some randomized factors, which make them more efficient when trying to copy an allele. The more randomized these operators are, the more possibilities of progress should have. Among them GNX operator has wide range of variations, but it is not the best. Also, ERX operator has some variations, but is the worst. On the other hand, SCX provides us best results. But it has limited range of variations and gets stuck in local minimums quickly.

Instance	n	Opt. Sol.	ERX			GNX			SCX		
			Best (%)	Avg (%)	Avg Time	Best (%)	Avg (%)	Avg Time	Best (%)	Avg (%)	Avg Time
bayg29	29	1610	0.00	0.25	18.11	9.25	10.62	0.65	0.00	0.00	2.19
eil51	51	426	1.41	2.03	157.32	18.78	20.11	20.44	0.00	0.63	4.59
berlin52	52	7542	0.00	3.19	78.57	19.32	22.24	40.11	0.00	0.24	5.10
eil76	76	538	5.20	5.95	286.90	20.07	20.76	141.96	0.00	0.87	128.94
pr76	76	108159	9.08	9.75	230.67	25.13	26.36	142.89	0.11	1.43	131.10
kroA100	100	21282	27.43	32.60	583.55	54.93	68.47	110.55	4.04	4.37	48.75
kroC100	100	20749	40.22	42.25	222.51	54.69	59.02	201.09	1.80	2.77	123.76
eil101	101	629	27.03	27.72	531.80	32.59	32.80	219.37	0.75	1.12	226.42
lin105	105	14379	30.05	33.31	728.44	48.13	50.30	264.80	2.52	2.67	185.90
brg180	180	1950	65.77	74.76	706.00	58.46	59.66	516.69	0.00	0.51	636.67
d198	198	15780	69.04	78.92	870.77	65.71	73.35	304.23	4.09	4.56	542.23

Summary of the results by the crossover operators for symmetric TSPLIB instances.

4. Result and analysis

According to all the experiments above, TS present that can do good implement with some small size of problem and use small amount of iterations but not guarantee of the optimal path solution. While the GA can implement in more flexible way but, it use more time in calculating and implementing also not guarantee the optimal path solution. Otherwise, AS can do any size of the problem and use less time to calculate than genetic algorithm. However, AS can guarantee the optimal path solution but, should give some time for iteration to implement.

5. Conclusion

This survey showed the result of test,Ant System,Tabu Search and Genetic Algorithm,its shown that Ant System can found shortest-path at period of time, however it can't guarantee its iteration and time. Tabu Search can work with a short time but sometime it can't found the best path and if it work with a large problem it just have low performance. Genetic Algorithm have a high flexibility to solve with several heuristic problem but it take more time for equation compute. So we can't summary who is the best algorithm for solve Traveling Saleman Problem because they have difference advantage and disadvantage.

Reference

[1] M. Dorigo, A.C., and V. Maniezzo, *positive feedback as a search stragegy.* 1991: p. 91-106.

[2] T. Stützle, a.H.H.H., *Max-Min ant system.* 2000. **16**(Future Generation Computer Systems): p. 889–914.

[3] Yan, Z., et al. *BEST-WORST Ant System*. in *Advanced Computer Control (ICACC), 2011 3rd International Conference on*. 2011.

[4] Dorigo, M., M. Birattari, and T. Stutzle, *Ant colony optimization.* Computational Intelligence Magazine, IEEE, 2006. **1**(4): p. 28-39.

[5] Ping, G. and L. Zhujin. *Moderate ant system: An improved algorithm for solving TSP*. in *Natural Computation (ICNC), 2011 Seventh International Conference on*. 2011.

[6] Hong-biao, M., W. Jiang, and R. Zi-hui. *An Adaptive Dynamic Ant System Based on Acceleration for TSP*. in *Computational Intelligence and Security, 2009. CIS '09. International Conference on*. 2009.

[7] Dong, G., W.W. Guo, and K. Tickle, *Solving the traveling salesman problem using cooperative genetic ant systems.* Expert Systems with Applications, 2012. **39**(5): p. 5006-5011.

[8] Zhaojun, Z. and F. Zuren. *A novel Max-Min ant system algorithm for traveling salesman problem*. in *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*. 2009.

[9] Lijie, L., J. Shangyou, and Z. Ying. *Improved Ant Colony Optimization for the Traveling Salesman Problem*. in *Intelligent Computation Technology and Automation (ICICTA), 2008 International Conference on*. 2008.

[10] Bifan, L., W. Lipo, and S. Wu. *Ant Colony Optimization for the Traveling Salesman Problem Based on Ants with Memory*. in *Natural Computation, 2008. ICNC '08. Fourth International Conference on*. 2008.

[11] Tiankun, L., et al. *An Improvement of the Ant Colony Optimization Algorithm for Solving Travelling Salesman Problem (TSP)*. in *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*. 2009.

[12] Yonghui Fang, Guangyuan Liu, Yi He, and Yuhui Qiu, "Tabu search algorithm based on insertion method", in *Proceedings of the 2003 International Conference on Neural Networks and Signal Processing, 2003*, 2003, vol. 1, pp. 420- 423 Vol.1.

[13] R. Battiti, G. Tecchiolli, I. Nazionale, and F. Nucleare, "The Reactive Tabu Search", 1993.

[14] C.H. Papadimitriou and K. Steglitz. *"Combinatorial Optimization: Algorithms and Complexity"*. Prentice Hall of India Private Limited, India, 1997.

[15] C.P. Ravikumar. *"Solving Large-scale Travelling Salesperson Problems on Parallel Machines"*.Microprocessors and Microsystems 16(3), pp. 149-158, 1992.