Solving Traveling Salesman Problem by Ant Colony Optimization Algorithm with Association Rule

Gao Shang Zhang Lei Zhuang Fengting Zhang Chunxian
School of electronics and information, Jiangsu University of Science and Technology, Zhenjiang 212003,China
E-mail:gao_shang@hotmail.com

Abstract

The traveling salesman problem (TSP) is among the most important combinatorial problems. Ant colony optimization (ACO) algorithm is a recently developed algorithm which has been successfully applied to several NP-hard problems, such as traveling salesman problem, quadratic assignment problem and job-shop problem. Association rule (AR) is the key in knowledge in data mining for finding the best data sequence. A new algorithm which integrates ACO and AR is proposed to solve TSP problems. Compare with the simulated annealing algorithm, the standard genetic algorithm and the standard ant colony algorithm, the new algorithm is better than ACO.

1. Introduction

Recently Traveling Salesman Problem (TSP) is the importance research core for combinatorial problems in the world. TSP is proven to be NP-hard. Research methods applied to TSP has been extended to some fine heuristics such as simulated annealing algorithm [1], tabu search algorithm [2][3], genetic algorithms[4], ant colony algorithm [5][6] etc. In this paper we present a new method to efficiently solve TSP problems. We used association rules to assist ant colony algorithm for solving TSP problems.

2. The Basic ACO Algorithm

In this section we introduce the basic ACO algorithm. We decided to use the well-known traveling salesman problem as benchmark, in order to make the comparison with other heuristic approaches easier. Given a set of n towns, the TSP can be stated as the problem of finding a minimal length closed tour that visits each town once. We call d_{ij} the length of the path between towns i and j. In the case of Euclidean TSP, d_{ij} is the Euclidean distance between i and j (i.e., $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. An instance of the TSP is given by a graph (N, E), where N is the set of towns and E is the set of edges between towns (a fully connected graph in the Euclidean TSP).

Let $b_i(t)$ ($i = 1,2,\cdots,n$) be the number of ants in town i at time t and let $m = \sum_{i=1}^{n} b_i(t)$ be the total number of ants. Each ant is a simple agent with the following characteristics:

- it chooses the town to go to with a probability that is a function of the town distance and of the amount of trail present on the connecting edge.
- to force the ant to make legal tours, transitions to already visited towns are disallowed until a tour is completed (this is controlled by a *tabu* list).
- when it completes a tour, it lays a substance called trail on each edge (i, j) visited.

Let $\tau_{ij}(t)$ be the intensity of trail on edge (i, j) at time t. Each ant at time t chooses the next town, where it will be at time $t+1$. Therefore, if we call an iteration of the ACO algorithm the m moves carried out by the m ants in the interval $(t, t+1)$, then every n iterations of the algorithm (which we call a cycle) each ant has completed a tour. At this point the trail intensity is updated according to the following formula

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

where ρ. is a coefficient such that (1- ρ) represents the evaporation of trail between time t and $t+n$,

Third International Conference on Natural Computation (ICNC 2007)
0-7695-2875-9/07 $25.00 © 2007 IEEE

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

where $\Delta\tau_{ij}^{k}$ is the quantity per unit of length of trail substance (pheromone in real ants) laid on edge (i, j) by the k-th ant between time t and $t + n$. It is given by

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & \text{if k - th ant uses edge (i, j) in its tour (between time t and t + n)} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

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

The coefficient ρ must be set to a value ρ <1 to avoid unlimited accumulation of trail. In our experiments, we set the intensity of trail at time 0, $\tau_{ij}(0)$, to a small positive constant *c*.

In order to satisfy the constraint that an ant visits all the n different towns, we associate with each ant a data structure called the *tabu list*, that saves the towns already visited up to time t and forbids the ant to visit them again before n iterations (a tour) have been completed. When a tour is completed, the *tabu* list is used to compute the ant's current solution (i.e., the distance of the path followed by the ant). The *tabu* list is then emptied and the ant is free again to choose. We define $tabu_k$ the dynamically growing vector which contains the *tabu* list of the kth ant, $tabu_k$ the set obtained from the elements of $tabu_k$, and $tabu_k(s)$ the s-th element of the list (i.e., the s-th town visited by the k-th ant in the current tour).

We call *visibility* η_{ij} the quantity $1/d_{ij}$. This quantity is not modified during the run of the ACO algorithm, as opposed to the trail which instead changes according to the previous formula (1).

We define the transition probability from town i to town j for the k-th ant as

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(t) \cdot \eta_{ij}^{\beta}}{\sum_{s \in allowed_k} \tau_{is}^{\alpha}(t) \cdot \eta_{is}^{\beta}} & if\ \ j \in allowed_k \\ 0 & otherwise \end{cases} \quad (4)$$

where $allowed_k = \{ N - tabu_k \}$ and where α and β are parameters that control the relative importance of trail versus visibility. Therefore the transition probability is a trade-off between visibility (which says that close towns should be chosen with high probability, thus implementing a greedy constructive heuristic) and trail intensity at time t (that says that if on edge (i, j) there has been a lot of traffic then it is highly desirable, thus implementing the autocatalytic process).

3. Association Rules

As the database about business information grows rapidly, data mining become an important subject in the real world application and has captivated more attention in database management [7]. Many data mining algorithms provide efficient and powerful solutions to real world problems [8][9][10][11][12][13][14]. Business information are collected from sales data in bar-code or credit card transaction information which consist of customers ID, bought items, transaction time etc. Many data mining technologies analyze sequences that help us understanding relationship among data such as knowledge rules, constraints, regularities. Association rules (AR) are used to finding sequences for items in database[10][11][13][14]. AR finds the maximal sequences that have a certain user-specified constraint minimum support.

Due to increasing complexity of business related database, data mining becomes an importance research area. Many data mining technologies analyze sequences which help us understanding relationship among data such as knowledge rules, constraints, regularities. There are good data mining algorithms to find sequential patterns such as Apriori[15], Association Rule[10][13][14], Scale-up[15],etc. There are good algorithms to find sequential patterns. Sequential pattern finds the maximal sequences that have a certain user-specified constraint minimum support.

In this paper we also present a new algorithm that constructs sequential pattern. We introduce the Ant Association Rule that provide new method to find the maximal sequence satisfying the minimum support constraint. The proposed method can be applied on huge and complex database.

Association rules was first introduce in [10]. It analyzes relation among items that bought by customers. Data mining association rules can be reduced into two sub-problems [15].

(1). Search all large item sets that satisfy predetermined minimum support.

(2). Generate the association rules for the database using the large item sets.

Given a database of customer record about sold item sets, the mining sequential pattern is the maximal sequences among all sequences that meet predefined minimum support s% [15]. Many algorithms are efficient in finding sequences among all item sets in the database. Usually using association rules to find the large item sets that satisfying minimum support s%. It is decomposed into the following three steps.

(1) Scanning the item sets to find the large item sets that meet the minimum support s%

(2) Generate the candidate set of item sets according to priority of large item sets.

(3) Iterates 1,2 steps to generate the association rules for the database.

As an example, we are given a database with the transaction data shown in Fig.1.

Customer_Id	Items
1	A C E
2	B D E
3	A B C
4	B E
5	C D E

Figure.1 A database with the transaction data

With minimum support of 2 customers (i.e. 40% minimum support), we can simply scan all the transactions and find the items which satisfy the constraint. We get the large 1-itemsets as in Fig.2. Candidate sets C_2 are generated based on large 1-itemsets, L_1 , which also satisfy minimum support of 2 customers. Then the large 2-itemsets, L_2 are determined based on the support of each C_2 sets shown in Fig.1. The candidate itemsets, C_3 is generated from L_2. Then scans all transactions and finds the large 2-itemsets in Fig.1. We list all possible candidate 3-itemsets, then prune out item sets that do not satisfy minimum support, then generate large 3-itemsets L_3 . Since there is no candidate 4-itemsets to generate from L_3. The rules ends operation for discover large item set.

Association rules is to find the subsets of large item set for subsets. It is listed as follows

(1) Scan the database ant count the support of itemsets with one element.

(2) $L1$={ large one-itemsets}.// Result of searches for large item sets with one attribute.

(3) For(k=2; k<=*no _ of _ attributes*; k++)

a. Generate candidate itemsets C_k based on L_{k-1} y adding one or more attributes.

b. Find itemset C_k as the subset of L_{k-1}.

c. For each itemsets in C_k , if it does not satisfy the constraint (% *s*), prune it from C_k .

d. Check remaining C_k itemsets.

e. Select L_k sets that is the large itemsets from C_k.

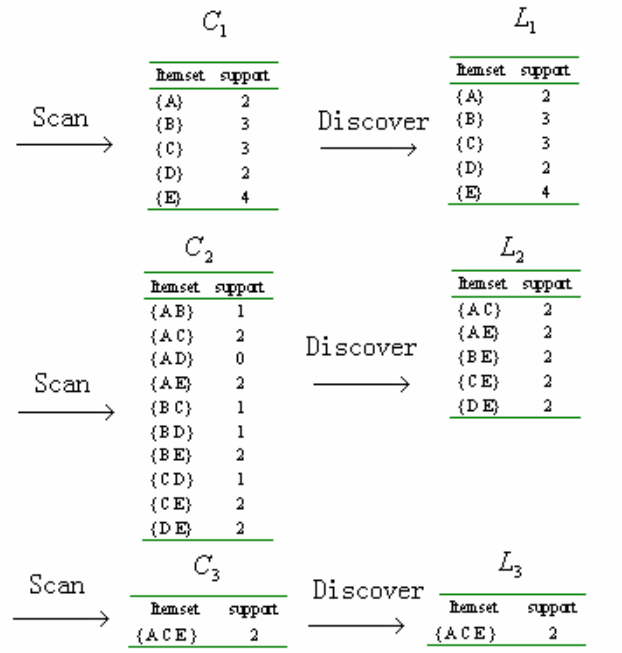

Figure .2 Association rules

4. Association Rules Aided Ant Colony Optimization Algorithm

ACO algorithms is a new way to find solution for optimization problem, such as TSP, ATSP, QAP, JSP etc. In this paper we integrate ACO and Association Rules for solving TSP problems. The procedure of new algorithm are

(1) ACO to solved TSP problems under predefined cycles (NCMAX=200) // to generate some initial solution.

(2) Select proper size of solutions and storage them in database.

(3) Find relation above all cities in database according to association rule.

(4) Sequence the cities based on high support from step 3.

This section compares the results of simulated annealing algorithm, genetic algorithm, ACO algorithm and new algorithm on traveling salesman problem of 30 cities. The parameters of simulated annealing algorithm are set as follows: the initial temperature $T = 100000$, the final temperature $T_0 = 1$, and annealing velocity $\alpha = 0.99$. The

parameters of the genetic algorithm optimization toolbox (GAOT) used to solving TSP are set as follows: the population $N = 30$, the cross probability $P_c = 0.2$, and the mutation probability $P_m = 0.5$. The parameters of the new algorithms are set as follows: $\alpha = 1.5$, $m = 30$, $\beta = 2$, and $\rho = 0.9$. 20 rounds of computer simulation are conducted for each algorithm, and the results are shown in Table 1. The optimal tour of 30 cities by new algorithm is shown in fig. 3. The new algorithm is proved effective.

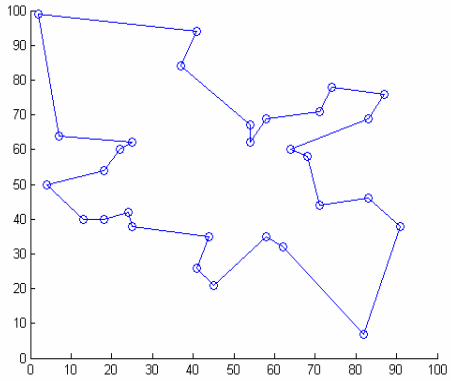

Figure 3. The optimal tour of 30 cities by hybrid algorithm

Table.1 Testing result of algorithms

Algorithms	Average solutions	Best solutions	Worst solutions
Simulated annealing algorithm	438.5223	424.6918	479.8312
Genetic algorithm	483.4572	467.6844	502.5742
Basic ACO algorithm	550.0346	491.9581	599.9331
New algorithm	431.4987	423.7406	447.6865

5. Conclusions

In this paper we proposed a modified ACO algorithm by integrating AR and ACO. Using the test problem we find new algorithm contains better solution quality than ACO. For the future work, we may modify the pheromone according to association rule. It tends to change pheromone dynamically, which may reinforce the ACO capabilities for solving TSP problem.

References

[1]C.S. Jeong, and M.H. Kim, "Fast parallel simulated annealing for traveling salesman problem", Neural Network, Vol. 3 (1990), pp.947-953

[2]F. Glover, "Tabu Search- Part Ⅰ", ORSA J. Computing, Vol. 1, no. 3(1989), pp.190-206

[3]F. Glover, "Tabu Search- Part Ⅱ", ORSA J. Computing, Vol. 2, no. 1(1990), pp.4-32

[4]T.N. Bui, and B.R. Moon, "A new genetic approach for the traveling salesman problem", Proc. Of the 1st IEEE Conference on 1994, Evolutionary Computation, Vol.1(1994), pp.7-12

[5]M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents", Systems, Man and Cybernetics, Part B, Vol. 26 Issue: 1(1996), pp.29 – 41

[6]M. Dorigo, and L.M. Gambardella, "Ant colony system： a cooperative learning approach to the traveling salesman problem", IEEE Trans. Evolut. Comput, 1 (1997), pp.53-66

[7]A. Silberschatz, M. Stonebraker, and J.D. Ullman, Database Research: Achievements and Opportunities into the 21st Century. Report NSF Workshop Future of Database Systems Research (1995)

[8]R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, A. Swami, "An Interval Classifier for Database Mining Application", Proc. 18th Internal Conf. Very Large Data Bases, Vancouver, Canada, (1992), pp.560-573

[9]R. Agrawal, T. Imielinski, A. Swami, "Database Mining: A Performance Perspective", IEEE Trans. Knowledge and Data Engineering, Vol. 5(1993), pp.914-925

[10] R. Agrawal, T. Imielinski, A. Swami, "Mining Association Rules Between Sets of Items in Large Databases", Proc. ACM-SIGMOD Internal Conf. Management of Data, (1993), pp.207-216

[11]R. Agrawal R., and R. Srikant, "Fast Algorithms for Mining Association Rules", Proc. Internal Conf. Very Large Data Bases, Santiago, Chile,(1994), pp.487-499

[12]J. Han,Y. Cai, and N. Cercone, "Data-Driven Discovery of Quantitative Rules in Relational Databases", IEEE Trans Knowledge and Data Engineering, Vol. 5(1993), pp.29-40

[13]J.S. Park, M.S. Chen, and P.S. Yu, "Efficient Hash-Based Algorithm for Mining Association Rules", Proc. ACM-SIGMOD Internal Conf. Management of Data, San Jose, Calif.,(1995), pp.175-186

[14]A. Savasere, E. Omiecinski, S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Database", Proc. Internal Conf. Very Large Data Bases, Zurichm (1995), pp.432-444

[15]R. Agrawal, and R. Srikant, "Mining Sequential Patterns", Proc. Internal Conf. Data Engineering, (1995), pp.3-14