THE ADJACENT VEHICLES QUERY ALGORITHM OF
MICROSCOPIC TRAFFIC SIMULATION

Yuping Zhao, Department of computer science and engineering, University of electronic science and technology, City Chengdu, China, zypuestc@live.com

Shijie Zhou, Department of computer science and engineering, University of electronic science and technology, City Chengdu, China, sjzhou@uestc.edu.cn

Jiaqing Luo, Department of computer science and engineering, University of electronic science and technology, City Chengdu, China, csjluo@hotmail.com

Zhu Song, Department of computer science and engineering, University of electronic science and technology, City Chengdu, China, toni110@163.com

Abstract

Microscopic traffic simulation reflects the characteristics of the entire traffic flow by simulating individual vehicles in the transport network. Therefore, in the microscopic simulation, before a vehicle make behavioural changes, it needs to get the information from surroundings (especially information from its adjacent vehicles) in time. In a large-scale traffic simulation system, the query of adjacent vehicles will be very frequent. This directly affects the efficiency of the whole simulation system. To this end, we propose a position thread AVL (PTAVL) tree data structure to store vehicles on a lane. On this basis, we propose an adjacent vehicles query algorithm based on the PTAVL tree and compare it with the existing query algorithms. The query algorithm based on PTAVL has complexity of $O(1)$ when a vehicle queries its former and follows adjacent vehicles on the same lane. It has complexity of $O(\log n)$ when a vehicle queries its adjacent vehicles on the adjacent lane. Our primary experimental results show that the algorithm can improve the speed of query adjacent vehicles and the simulation speed of the microscopic traffic simulation system.

Keywords: Microscopic traffic simulation, Adjacent vehicles query, Data structure, AVL, Time complexity.
1. INTRODUCTION

With the rapid development of the city, traffic congestion becomes more and more frequent. Traffic congestion has given the people of great inconvenience in work and life (Arnott R et al. 1994). In order to solve the current situation of urban traffic congestion, Intelligent Transport System (ITS) gets more and more attention around the world (James W 2004; Yan Y et al. 2013). As an important part of ITS, traffic simulation is able to simulate the existing transport network (Kramer J G et al. 2010), thereby traffic simulation is not only able to let relevant departments to conduct an assessment of the existing road network and provide the basis for the planning of new roads (Stig O.Simonsson 1992).

According to simulation granularity, traffic simulation can be divided into macroscopic traffic simulation, microscopic traffic simulation and mesoscopic traffic simulation. Microscopic traffic simulation is realized by simulating each vehicle in transport network, so it can reflect the actual traffic conditions veritably. In microscopic traffic simulation, the vehicle needs to obtain data of the surrounding environment before it changes its current state. These data include information of its adjacent vehicles. Accurate and efficient access to this information directly affects the accuracy and efficiency of the entire traffic simulation system. In this case, it will reduce the speed of queries if we do not use efficient query structures and algorithms. Thus it will affect the efficiency of the entire traffic simulation system.

Therefore, we propose a position thread AVL tree (PTAVL tree) structure. In this structure, the vehicle nodes had been stored in a balanced binary tree. At the same time, considering the adjacent relationship of the vehicle's former and follow vehicle in the same lane, we added the thread based on vehicle position on the lane. Through the thread, a vehicle is able to query the former and follow adjacent vehicles on the same lane with the time complexity of $O(1)$. By contrast, the time complexity of search on a balanced binary tree is $O(\log n)$.

On the basis of the structure, we propose an adjacent vehicles inquiry algorithm based on PTAVL. By using of the structure and algorithm, the efficiency of adjacent vehicles query and simulation system is improved significantly.

The main work of this paper includes: proposed a location thread AVL tree structure and an adjacent vehicles query algorithm based on PTAVL tree.

The rest of the paper organized as follows: the section 2 describes the related work, and the section 3 describes the vehicle storage structure and the adjacent query algorithm we proposed, section 4 is the experimental analysis of the proposed structures and algorithms, and section 5 concludes the paper.\footnote{This work is supported by National Science Foundation of China, under Granted NO.61170041, and China Postdoctoral Science Foundation, under Granted NO.2013T60849 and NO.2012M521684.}

2. RELATED WORK

Since the vehicle query the adjacent vehicles frequently in the microscopic traffic simulation system. Therefore, vehicle storage structure and query algorithm directly affects the
efficiency of the simulation system. In research on Parallel Traffic Simulation Model and Key Algorithm, it uses a vector to store all the lanes and all vehicles on the road with a linked list (Ni Anning 2007). In Study and Implement on Parallel Simulation Algorithm of Dynamic Route Solution for Traffic Network, it uses of global sharing data road, driveway and vehicle data queue to store the data of road and vehicles. And it uses the queue data structure to store the entire vehicles of simulation system (Gao Linjie 2006). In Traffic Simulation System based on TRANSIMS (Raney B et al. 2002) and in Vehicle Traffic Simulation (Badler N et al.) they use linked list to store the vehicles on the road. The SUMO (Simulation of Urban Mobility) developed by the German Space Agency use a queue to store vehicles on each lane (Krajzewicz D et al. 2002; Behrisch M et al. 2011). In the systems of using NS models (Nagel K et al. 1992), they are all divided the driveway into lattices (cellular). Each lattice can be only occupied by one vehicle and have empty state or occupied state. We can query the adjacent lattices quickly according to the number of current lattice. Since cellular automaton traffic flow model is a space discrete model, the vehicle runs in each simulation step is discontinuous. The simulation accuracy of using cellular automata simulation system is poor, and therefore cellular structure is not suitable for high-precision traffic Simulation. The vector, queue and linked list storage of vehicles are all based on traversal way to query a vehicle. So in this section, we only focus on the use of vehicles query algorithm based on list structure.

The vehicle inquiry algorithm based on linked list structure uses linked list to store the vehicles on a lane according to their position on the current lane.

Vehicle linked list shown in Figure 1.

![Figure 1. The structure of vehicle linked list](image)

The list store vehicles according to their position in the current lane, so that the order of the vehicle in the vehicle list can truly represent the physical location relationship of the vehicles on the lane. Therefore, the structure can be very convenient for storing vehicles of a lane. Vehicles can join or leave the lane and query the former and follow adjacent vehicles on the same lane with high efficient by using this structure.

When the vehicle has the intention to change lane, it need to query the left-right adjacent vehicle on the adjacent lane, where "left-right adjacent" refers to the former and follow adjacent vehicles on the adjacent lane. As shown in Figure 2, when vehicle 7 has the intention to change to the left lane, it will query the former and follow vehicle which on the left adjacent lane. In this case, the former vehicle of vehicle 7 is vehicle 2 and its follow vehicle is vehicle 3 on the left adjacent lane. For query vehicle 2 or vehicle 3, we had to traverse the vehicle list which in lane A. Therefore, the time complexity of query left or right adjacent vehicle is $O(n)$. As shown in Figure 2, if the vehicle 4 which on lane A want to query it’s adjacent vehicles which on lane B, it had to traverse almost the whole vehicle list. This traverse will reduce the efficiency of traffic simulation seriously. In order to improve the efficiency of query left or right adjacent vehicle we introduced the index structure based on balanced binary tree.
3. DATA STRUCTURES AND ALGORITHMS

3.1 Vehicle storage structure based on balanced binary tree

In the vehicle storage structure based on linked list, according to the characteristics of linked list, we cannot access the vehicles of list randomly. Therefore, the use of storage structure based on linked list is difficult to improve the efficiency of the query. To this end, we introduce a vehicle storage structure based on balanced binary tree, this structure store vehicles according to the position of the vehicle on the lane. The average length of query left-right adjacent vehicle is \( \log n \), by adopting this structure, greatly improving the efficiency of query the left or right adjacent vehicles, especially, when there has large number of vehicles on adjacent lane, query efficiency improvement will become more apparent.

Although the use of vehicle storage structure based on balanced binary tree improve the efficiency of query left-right adjacent vehicles greatly, vehicle query its former or follow vehicles on same lane need to query in the tree, because the time complexity of search in a balanced binary tree is \( O(\log n) \), this is worse than the structure based on linked list. Therefore, we proposed a new vehicle storage data structure, it is based on a position thread AVL tree (PTAVL), this structure combines the advantages of the linked list structure and balanced binary tree structure.

3.2 The vehicle storage structure based on PTAVL

The node of PTAVL contains five domains. They are: Data domain, Lchild domain, Prior domain, Rchild domain, Next domain, as it shown in figure 3.

<table>
<thead>
<tr>
<th>Prior</th>
<th>Lchild</th>
<th>Data</th>
<th>Rchild</th>
<th>Next</th>
</tr>
</thead>
</table>

Figure 3. The node structure of PTAVL tree

In the structure, the Prior domain has a pointer which pointing to the former vehicle on current lane; the Lchild domain has a pointer which pointing to the node’s left child; the Data domain store the information of vehicle; the Rchild domain has a pointer which pointing to the node’s right child; the Next domain has a pointer which pointing to the follow vehicle on current lane of the vehicle which stored in Data domain. Note that, the node which stores the front vehicle of a lane, its Prior domain is null and the node which store the rear vehicle of a lane, its Next domain is also null.
<table>
<thead>
<tr>
<th>Vehicle ID</th>
<th>Drive distance(m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>350</td>
</tr>
<tr>
<td>2</td>
<td>300</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
</tr>
<tr>
<td>4</td>
<td>130</td>
</tr>
<tr>
<td>5</td>
<td>95</td>
</tr>
<tr>
<td>6</td>
<td>88</td>
</tr>
<tr>
<td>7</td>
<td>80</td>
</tr>
<tr>
<td>8</td>
<td>50</td>
</tr>
</tbody>
</table>

*Table 1. Data of drive distance of vehicles on the driving direction of a road*

Assuming there are 8 vehicles on a lane, this 8 vehicles travel distance from the road begins in the forward direction shown in Table 1. According to Table 1 we established a PTAVL tree; it is shown in figure 4. In the figure, the solid line indicates the pointer pointing to left child and the right child, the black dashed arrows indicate the pointer pointing to the successor node, and the red dashed arrows indicate the pointer pointing to the precursor node.

*Figure 4. The PTAVL tree established by the data of table 1*

When a vehicle needs to query the former or the follow adjacent vehicles on the same lane, we can use the Prior domain or the Next domain of the PTAVL. When a vehicle need to query the left-right adjacent vehicles which on the adjacent lane, we can use the binary tree structure of the PTAVL.

3.3 The algorithms of PTAVL

3.3.1 The query of former and follow adjacent vehicles

In PTAVL tree, vehicle can get its former and follow adjacent vehicles through the Prior domain and Next domain of the PTAVL tree which the vehicle was stored in. We can get the former vehicle and the follow vehicle by Prior domain and Next domain directly, so the average length of query former and follow adjacent vehicles is 1, the time complexity is $O(1)$. 
3.3.2 The query of left and right adjacent vehicles

PTAVL tree is also a Binary search tree (BST), it has the character of BST. The algorithm of query left-right adjacent vehicle based on PTAVL is similar with the search algorithm of BST, but there are still some different between them. The general search algorithm of BST only search one node, but the query of left and right adjacent vehicles algorithm need to search two nodes and etc., one as the former and another as the follow.

Assuming the distance of the current vehicle on road is distance; the pointer of vehicle storage structure of adjacent lane is neighbour; the former and follow vehicle which we needed are before and follow. The algorithm as follow:

```
01 SearchNeighborhood( neighbour, distance)
02   If neighbour = NULL
03      before ← NULL; follow ← NULL; return 0
04   Endif
05   If vehicle position on the neighbour lane = distance
06      before ← neighbour; follow ← (neighbour->next)
07   Else if vehicle position on the neighbour lane > distance
08      If neighbour->Lchild =NULL
09         follow ← neighbour
10         before ← (neighbour->prior)
11      Else
12         SearchNeighborhood (neighbour -> Lchild, distance)
13      Endif
14   Else if vehicle position on the neighbour lane < distance
15      If neighbour->Rchild =NULL
16         before ← neighbour; follow ← neighbour->next
17      Else
18         SearchNeighborhood (neighbour -> Rchild, distance)
19      Endif
20   Endif
```

Because a PTAVL tree is a balanced binary tree, in the search process the times of compare with the keywords will not more the depth of the tree, and the largest depth of a balanced binary tree is \((\log_{e}(\sqrt{3(n+1)}) - 2)\) (Horowitz E et al. 1983). Therefore, the time complexity of query left-right adjacent vehicles is \(O(\log n)\) based on PTAVL tree.

4. EXPERIMENTS

4.1 Introduction of the experiment

In this paper, we will compare the algorithm based on PTAVL tree with the algorithm based on AVL tree and linked list. In the experiment, we make a simulation based on the backbone network of Mianyang city, the network as shown in figure 5. To facilitate the experiment, we assume that all roads have equal number of lanes. The variable parameters are: The number of
vehicles in a lane, the lane change rate (lane change rate indicates the rate that a vehicle change lane from its current lane to an adjacent lane in each step).

![Figure 5. The road network used in experiment](image)

In the Experiment, a vehicle just only have 4 operations there are query the former and follow adjacent vehicles, query the left and right adjacent vehicles, leave a lane and move into a lane. We compare the average time cost of vehicle in one step move which use of algorithm based on PTAVL with the use of algorithm based on linked list and the use of algorithm based on AVL according to the 2 parameters.

4.2 Analysis of experimental results

4.2.1 Impact of lane change rate

In this experiment we set the number of vehicles in each lane is 250, number of lanes is 2 and the speed of each vehicle is 10m/s, the impact of lane change rate on different data structures and algorithms as shown in figure 6.

![Figure 6. The impact of lane change rate of vehicles to the time cost in one simulation step of a vehicle](image)

From the figure 6 we can see that with the increase in the rate of lane change, the three vehicle storage structure and algorithm all have increased by amount of time, but the time cost which uses AVL and PTAVL gradually stabilized, on the Contrary, the time cost which uses linked list continued increase. This is because it need query the left-right adjacent vehicles when a vehicle change lane, and the PTAVL or AVL is more efficient than linked list on query left-right adjacent vehicles. In contrast, the time cost which uses PTAVL less
than the other two data structures.

4.2.2 The impact of number of vehicles on each lane

In this experiment we set the speed of each vehicle is 10m/s, the number of lanes is 2 and the lane change rate is 30%, the impact of number of vehicles in each lane as shown in figure 7.

From the figure 7 we can see that With the increase in the number of vehicles the time cost of using the linked list structure and algorithm increase rapidly, and the time cost of use of AVL vehicle structure and algorithm rendering logarithmic growth trend, it is because the use of AVL structures and algorithms, the time complexity of vehicle check former and follow adjacent vehicles in same lane and check the left or right adjacent vehicles on the adjacent lane are both $O(\log n)$.

![Figure 7. The impact of vehicle numbers on a lane to the time cost in one simulation step of a vehicle](image)

From those Experiments we can see that, the efficient of adjacent vehicle query algorithm based on PTAVL is better than which based on linked list and AVL, especially when vehicle have a high lane change rate or there have a large number of vehicles on lane, the improve of efficiency will become more apparent by using PTAVL.

5. CONCLUSION

The adjacent vehicle query algorithms based on PTAVL tree we proposed in this paper is able to improve the efficient of query adjacent vehicles in microscopic traffic simulation system, and apply it to our microscopic traffic simulation system can significantly improve the efficiency of the whole system.
REFERENCES

Yan Y, Xu C. (2013). A Development Analysis of China's Intelligent Transportation System:
Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things
(iThings/CPSCOM), IEEE International Conference on and IEEE Cyber, Physical and
Effects: The 3rd International Conference on Vehicle Navigation & Information System
RTI/IVHS. September 2-4.
University.
Gao Linjie. (2006). Study and Implement on Parallel Simulation Algorithm of Dynamic
Route Solution for Traffic Network: Jilin University.
Raney B, Voellmy A, Cetin N, et al. (2002). Towards a microscopic traffic simulation of all
of Switzerland: Computational Science—ICCS 2002. Springer Berlin Heidelberg,
371-380.
of the 4th Middle East Symposium on Simulation and Modelling, 183-187.
Overview: SIMUL 2011, The Third International Conference on Advances in System
Simulation, 55-60.
de Physique I, 2(12): 2221-2229.
press.