Author:	Xu, Jiajie; Liu, Guanfeng; Zheng, Kai; Liu, Chengfei; Guo, Haoming; Ding, Zhiming
Title:	On Efficient Passenger Assignment for Group Transportation
Editor:	Matthias Renz, Cyrus Shahabi, Xiaofang Zhou and Muhammad Aamir Cheema
Conference name:	20th International Conference, DASFAA 2015
Conference location:	Hanoi, Vietnam
Conference dates:	20-23 April 2015
Series title and volume:	Lecture Notes in Computer Science: Database systems for advanced applications
Place published:	Switzerland
Publisher:	Springer International Publishing
Year:	2015
Pages:	226-243
URL:	http://hdl.handle.net/1959.3/408723
Copyright:	Copyright © 2015 Springer International Publishing Switzerland. The author's accepted manuscript is reproduced here in accordance with the copyright policy of the publisher. The final publication is available at http://link.springer.com/

On Efficient Passenger Assignment for Group Transportation

Jiajie Xu [1] Guanfeng Liu [1] Kai Zheng [1] Chengfei Liu [2] Zhiming Ding [3]

[1] Department of Computer Science & Technology, Soochow University
[2] Faculty of ICT, Swinburne University of Technology, Australia
[3] Institute of Software, Chinese Academy of Sciences

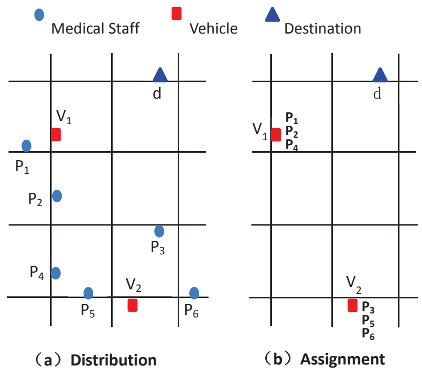
[1] *{xujj, gfliu, kevinz}@suda.edu.cn* [2] *cliu@swin.edu.au* [3] *zhiming@iscas.ac.cn*

Abstract. With the increasing popularity of LBS services, spatial assignment has become an important problem nowadays. Nevertheless most existing works use Euclidean distance as the measurement of spatial proximity. In this paper, we investigate a variant of spatial assignment problem with road networks as the underlying space. Given a set of passengers and a set of vehicles, where each vehicle waits for the arrival of all passengers assigned to it, and then carries them to the same destination, our goal is to find an assignment from passengers to vehicles such that maximum travel time of all passengers is minimized. Such a passenger assignment problem has various applications in real life. However, finding the optimal assignment efficiently is quite challenging due to high computational overhead in fastest path search and combinatorial nature of capacity constrained assignment. In this paper, we first propose two exact solutions to find the optimal results, and then an approximate solution to achieve higher efficiency by trading a little accuracy. Finally, performances of all proposed algorithms are evaluated on real dataset.

1 Introduction

Consider a set of passengers P and a set of vehicles V, all distributed on road networks. Each vehicle waits for a group of passengers (assigned to it) and carries them to a common destination d. Our objective is to find the passenger assignment $A \subseteq P \times V$ that can enable all passengers to arrive destination d at earliest, and vehicles are not allowed to carry more passengers than their capacity limits. Such a problem is called passenger assignment for group transportation, which can find various applications in real life.

An illustrative example is shown in Figure 1 to describe the passenger assignment problem. Assume that several passengers (i.e. $P_1 - P_6$), vehicles (i.e. V_1, V_2) and the final destination d are distributed on road network as Figure 1(a), and each vehicle can carry no more than three staffs. All passengers are required to go to d (by taking vehicles) as soon as possible for some purposes. To save time, it adopts an assemble-based-group-transportation fashion, i.e., passengers go to assemble on vehicle (by private car or taxi) like Figure 1(b), and then vehicles carry them to the final destination. The key issue is how to find the optimal assignment from passengers to vehicles efficiently such as Figure 1(b), such that all passengers can arrive destination and start the activities at their earliest. Likewise, the passenger assignment problem can be widely used in applications like logistic control, resource supply and other group transportation recommendation systems, etc.



Travel Cost Matrix (minutes)

From \ To	V_1	V_2	d
P_1	5	35	-
P_2	8	26	-
P_3	17	16	-
P_4	15	12	-
P_5	21	8	-
P_6	40	10	-
V_1	-	-	22
V_2	-	-	38

Fig. 1. An example of passenger assignment

Motivated by the above example, this paper studies the group transportation oriented passenger assignment problem, which tend to have high computational overhead for two main reasons: first, the large number of combinations of assignment from P to V makes the search space to be extremely large, particularly when the size of passenger or vehicle sets goes up; secondly, numerous fastest path queries (FPQ) need to be processed for deriving the travel cost from passengers to vehicles (i.e. information in travel cost matrix in Figure 1).

Capacity constrained assignment is a classical research problem that has been well studied in literature [5, 4, 6]. Recently lots of efforts have been made to address the capacity constrained spatial assignment (CCSA) problems [16, 15], by using Euclidean distance as the measurement of spatial proximity between two objects. However, existing CCSA solutions cannot be simply applied to our problem for two reasons: (1) The goal of optimization is different. Most CCSA algorithms are mainly designed to minimize the sum of Euclidean distance of all assigned object pairs, while the passenger assignment problem seeks to minimize the travel cost of the most time-consuming passengers, which provide us oppertunities to further speed up the assignment processing; (2) The spatial proximity measurement is different. The network-based spatial proximity used in the passenger assignment problem causes a large number of fastest path searches, which is generally regarded as much more computationally expensive than the evaluation of Euclidean distance.

In this paper, we present a novel strategy to find the optimal assignment based on maximum bipartite matching. It utilizes a set of pruning mechanisms sensitive to the most-time-consuming-passenger, so that the search can be constrained in small bipartite sub-graphs and in less loops. However, though the bounds used in the strategy ensure it has a fairly good performance on assignment step, the overall efficiency is still not satisfactory because of the vast processing cost of the scalable FPQs. Therefore, an approximate solution is further proposed to trade off a little accuracy for higher efficiency. The main contribution of this paper can be summarized as follows:

– We define a problem called passenger assignment for group transportation, which could potentially benefit many applications such as emergency response, supply chain management and traffic planning.
– We propose a novel assignment strategy based on maximum bipartite matching, by which the optimal passenger assignment can be efficiently found.
– We propose an approximate FPQ querying based assignment strategy, which utilizes bounded travel cost computation to reduce search space. It significantly improve the efficiency with little sacrifice on accuracy.
– We implement the proposed algorithms and conduct experiments on real dataset to evaluate the performances of our proposed solutions.

The rest of the paper is organized as follows. Section 2 presents the related work and Section 3 formally defines the passenger assignment problem. Afterwards, we introduce two exact passenger assignment algorithms in Section 4, and an approximate solution called TN-FPS in Section 5. After discussions on experimental results in Section 6, the paper is concluded in Section 7.

2 Related Works

Assignment is a classical problem that has been studied intensively, with many classical matching problems of bipartite graph, such as the maximum perfect matching (MPM) and minimum weight perfect matching (MWPM) problem. So far, there exist a lot of literatures towards bipartite graph matching in the operational research area. A well-known solution is the classical Hungarian algorithm and its variations [10, 12, 1], both having $O(mn + n^2 logn)$ time complexity. Later, increasing attention are paid to assignment with capacity constraints, with many algorithms proposed to address the capacity constrained assignment (CCA) problem, such as successive shortest path algorithm [3][15].

More recently, with the fast development of mobile computing, the problem of assignment and matching on spatial objects becomes popular. Specifically, [17] studied the spatial matching problem, which is reduced to the stable marriage problem for spatial objects, and [16, 15] studied the issue of CCA problems for objects in Euclidean space. However many CCA applications are road network constrained, and they cannot be well supported by [16, 15] because of their criterions and Euclidean distance measure adopted, like the passenger assignment problem of this paper. Therefore, new solutions are needed to support efficient assignment processing for applications in road networks like resource dispatch, evacuation in disaster, intelligent services and supply chain management.

In addition, another key technique our passenger assignment problem relies on is spatial and distance query processing. The basic type of distance query is shortest path query (SPQ). Typical solutions for SPQ include Dijkstra and A* algorithm, which traverses the road network nodes in ascending order of their distance from query position, and runs in $O(nlogn+m)$ time by using Fibonacci heap. [9] discussed SPQ problem on complex terrain space. Recently, many literatures tried to exploit the hierarchical structure of road map in a pre-processing step, and then properly use it to accelerate NDQ, such indexes mainly include the highway hierarchies (HH) [13], contraction hierarchies (CH) [7], and transit-node routing (TNR) [8] algorithms. More recently, the work [14] pre-compute certain shortest path distances called *path oracles* to answer approximate SPQ in $O(log|V|)$ time and $O(\frac{|V|}{\epsilon^2})$ space with is an error bound of ϵ.

Passenger Assignment requires scalable FPQ to find the time cost between objects in P and V. Compared to SPQ, the FPQ is much more challenging because real-time traffic condition needs to be considered, resulting road network with dynamic travel cost on the edges. All the above algorithms rely on heavy pre-processing, and are thus not suitable for dynamic scenarios where the road network topology or edge weights may change frequently. As a result, we have to use some approximate FPQ algorithms, so as to improve the efficiency of query processing. Moreover it is necessary to design robust matching algorithms that can find good solution based on approximate FPQ results.

3 Problem Definition

3.1 Spatial Networks

A spatial network is modeled by a connected and undirected graph $G = (V, E)$, where V is a set of vertices and E is a set of edges. A vertex $v_i \in V$ indicates a road intersection or an end of a road. An edge $e \in E$ is defined as a pair of vertices and represents a segment connecting the two adjacent vertices. For example, edge $e = \{v_i, v_j\}$ represents a road segment that enables travel between vertices v_i and v_j. We use $time(e)$ to denote the time required to pass a road segment e based on real-time traffic condition. Given two locations a and b in spatial network, its fastest path $FP_{a,b}$ is a sequence of edges linking a and b with the minimal total travel cost. We use $TC_{a,b} = \sum_{e \in FP_{a,b}} time(e)$ to represent the travel cost between a and b. Note that the travel cost between any two vertexes may update with the road condition change.

All passengers in P and vehicles in V are embedded in networks and they may be located on edges. If the network distances to the two end vertices of an edge are known, it is straightforward to derive network distance to any point in this edge. Thus, we assume that all data points are on vertices for clear description.

3.2 Passenger Assignment for Group Transportation

Let P be a set of passengers and V a set of vehicles, all distributed on a spatial network G. Each $p \in P$ denotes a passenger with his or her geographical location $p.l$, and each $v \in V$ denotes a vehicle that can carry up to $v.c$ passengers. Each passenger in P moves to an assigned vehicle in V first. Each vehicle starts when its passengers arrive and transport the to a common destination d.

For example in disaster management scenario, medical staffs go to a special vehicle first, and special vehicles then carry them to rescue mission place (where might be dangerous) after all staffs assigned to this vehicle have assembled. To plan the routes for all medical staffs, we need to assign them to special vehicles, and the result is called an assignment. We first define the notions of valid assignment to judge if an assignment is a qualified result.

Definition 1. *(Valid Assignment) An assignment $A \subseteq P \times V$ is said to be a valid assignment if it satisfies:*

(1) capacity constraint, i.e. each vehicle $v \in V$ must appears at most $v.c$ times in assignment A due to its capacity constraint;

(2) assignment A must be full to P, i.e. each passenger $p \in P$ must appear and only appear once in A, i.e. $|A| = |P|$, and $p \neq p'$ must be true for any two pairs $(p, v) \in A$ and $(p', v') \in A$.

For each pair $(p, v) \in A$ in assignment A, passenger p moves toward its assigned vehicle v first. Later, each vehicle v carries $\{p'|(p', v) \in A\}$ to destination when they arrive $v.l$. Obviously, the arrival time of all passengers in P is determined by the one having maximum total travel cost (time for waiting is not included), so we need to define two important notions 'pair cost' and 'critical pair'.

Definition 2. *(Pair Cost) For each pair (p, v) where $p \in P$ and $v \in V$, the cost of this pair $PC_{p,v}$ is measured as the total travel cost from $p.l$ to destination d via $v.l$ on spatial network G such that*

$$PC_{p,v} = TC(p.l, v.l) + TC(v.l, d)$$

Note that, the possible waiting time of a passenger p for other passengers at $v.l$ is not included. For example in Figure 1, the pair cost of (P_2, V_1) is computed as PC_{P_2,V_1} = 8 + 22 = 30.

Definition 3. *(Critical Pair of an Assignment) The critical pair $CP(A)$ of an assignment A is the pair $(p, v) \in A$ that has the maximum pair cost $PC_{p,v}$, and we call the passenger p as critical passenger.*

To ensure that all passengers in P can arrive destination in earliest, e.g. for the rescue missions or goods supply in emergency management scenarios, we further define the cost of a valid assignment based on its critical pair. For example in Figure 1, if assignment is made as Figure 1(b), then the pair (P_3, V_2) is the critical pair because of its greatest pair cost PC_{P_3,V_2} = 54.

Definition 4. *(Cost of an Assignment) The cost $\psi(A) = max\,(\,\{\,PC_{p,v} \,|\, (p, v) \in A\,\}\,)$ of a valid assignment A is quantified to be the pair cost of its critical pair, meaning the maximal travel cost of all passengers in P on the spatial network.*

Problem Formalization. Given a spatial network G, a passenger set P and a vehicle set V as input. Among all valid assignments from set P to set V, we aim to find the one A with the minimal assignment cost $\psi(A)$, which is determined by the pair cost (i.e. total travel cost) of A's critical pair based on G.

4 Exact Algorithms

This section presents two exact methods called IPA and MBMA for computing the optimal passenger assignment.

4.1 Integer Programming based Assignment (IPA)

Integer programming is known to be a widely used processing model for optimization problems. In this section, we introduce an effective integer programming based assignment solution.

As passenger assignment here is a road network constrained problem, we process the FPQs (between passengers/vehicles and vehicles/destination) first to derive all necessary travel cost information by the Dijkstra algorithm. After that, we find the valid assignment A with minimum value of $\psi(A)$ in the assignment step based on the pair cost information derived from last step. Valid assignment means to satisfy the capacity constraint of vehicles, and ensures each passenger to appear in A once and only once. Based on above constraints and optimization goals, we notice the passenger assignment problem can be expressed as the following integer program, where $x = (p, v)$ represents the 0-1 integer vector of the assignment between $p \in P$ and $v \in V$:

$$\begin{array}{ll} \text{minimize} & \tau(A) = max\left(\{\, PC_{p,v} \times x(p,v) \,|\, (p,v) \in A\,\}\right) \\[2ex] \text{subject to} & x(p,v) = 0 \; or \; 1 \qquad \forall p \in P, v \in V \\[2ex] & \sum_{p \in P} x(p,v) \leq v.c \quad \forall v \in V \\[2ex] & \sum_{v \in V} x(p,v) \leq 1 \qquad \forall p \in P \end{array}$$

In general, the purpose of above 0-1 program is to find the passenger assignment A such that: (1) to maximize value of $\tau(x)$ for the returned assignment; (2) all of the subject conditions can be satisfied. Particularly, only those pairs (p, v) having that $x(p, v) = 1$ are included in assignment A, meaning that the assignment from p to v is made. It is obvious that the final result A derived by above program is an assignment being valid and optimal. But the IPA algorithm is inefficient in most cases because of the 'max' operator used in optimization goal (i.e. to get the maximum pair cost), which lead to no bound to be easily found and used to stop the iterations. In contrast, the pruning effect would be greatly improved for cases that to minimize linear expression or using 'sum' operator, because an upper bound (i.e. current minimal value) is utilized to help us break in the middle. Therefore, we further propose another method to find optimal assignment in a more efficient fashion.

4.2 Maximum Bipartite Matching based Assignment (MBMA)

In this section, we introduce a MBMA strategy that can return optimal results efficiently. By constructing a bipartite graph $BG = (P, V, E)$, where the weight of an edge $e(p, v) \in E$ in bipartite graph $w(e) = PC_{p,v}$ is the pair cost between passenger p and vehicle v. We aim to find an valid assignment $A \subseteq P \times V$ on bipartite graph BG with minimum $\psi(A)$. Existing algorithms (e.g. Hungarian algorithm for maximum matching, and Kuhn-Munkres algorithm for minimum weight matching) cannot be applied because of the optimization criterion and capacity constraints we face in this paper. To cover the gap, we design an bipartite matching based algorithm that is more suitable for passenger assignment. The MBMA algorithm works with the following steps:

Step 1. Assignment Initialization Step. It first initializes a valid assignment A based on input bipartite graph using a given maximum bipartite matching strategy tailored for CCA-SN;

Step 2. Assignment Improvement Step. We prune some high weighted bipartite edges, and re-assigns some matches in A to find a better assignment on a bipartite subgraph;

Step 3. Iterative Step. Assignment improvement is processed in loop until it is not possible to perform the assignment improvement step.

4.2.1 Assignment Initialization

Given an input bipartite graph BG, we conduct the assignment operation to derive a valid assignment. Among the existing maximum bipartite matching (MBM) solutions, the Hungarian algorithm is the most famous one, and it finds the maximum matching by finding an augmenting path from each $v \in V$ to V' and adding it to the matching if it exists. We know that each augmenting path can be found in $O(|E| + |V \cup V'|)$ time, and we need to find $|V \cup V'|$ times of the augmenting path. Therefore, we can find maximum matching on $O(mn + n^2 logn)$ time, where $m = |E|$ and $n = |V \cup V'|$. It is a useful technique to our problem in finding a valid assignment from P to V.

Compared to the classical MBM, the assignment for the passenger assignment in this paper must be capacity aware. As stated in Section 3.4, capacity constraint can be addressed by vehicle-to-capacity-unit transformation, but such transformation incurs a much greater bipartite edge set, which in turn leads to more time cost. To handle this problem, we propose an assignment initialization algorithm tailored for our problem as shown in Algorithm 1.

The basic idea of assignment here is similar to Hungarian algorithm, i.e. to explore maximum matching on BG by finding augmenting path from each $p \in P$ to V and adding it to the matching if it exists. Specifically, let A be the matching of BG, a vertex $p \in P$ or $v \in V$ is matched if it is endpoint of edge in A, and it is free otherwise. A path is said to be alternating if its edge alternate between A and $E - A$. An alternating path is augmenting if both endpoints are free, and it has one less edge in A than in $E - A$. The assignment algorithm continuously replace the A edges in augmenting path by the $E - A$ ones to increment size of the matching until it cannot be enlarged.

```
input  : $BG$ - an input bipartite graph
output: $A$ - an assignment result
$A = \phi$;
do
    $p$ = FIND-AUGMENTING-PATH($BG, A$);
    if $p \neq NIL$ then
        $A = A \oplus p$;
    end
while $p = NIL$ ;
return $A$;
```

Algorithm 1: Assignment Initialization Algorithm

The function FIND-AUGMENTING-PATH(BG, M) of Algorithm 1 (Line 3) means to find an augmenting path based on A. This is the key issue of this algorithm as it determines both accuracy and efficiency: (1) from accuracy perspective, it must be capacity aware. Therefore, a counter $v.cn$ ic created for each $v \in V$ to record the times it has been assigned to, and augmenting path via it would be denied if $v.cn > v.c$; (2) from efficiency perspective, we hope the selected augmenting path can be covered by a valid assignment (i.e. feasibility), and also to reduce its assignment cost (i.e. closer to optimal result).

Particularly, we use some heuristics in implementation to speed up the assignment processing. To reduce the cost of assignment, edges with lower weight are encouraged to be chosen. To ensure the augmenting path to be part of valid assignment cover, we tend to avoid using $v \in V$ with low flexibility value $F(v)$ to reserve it for possible future use, and its flexibility is defined as:

$$F(v) = \frac{|v.c - v.cn|}{\sum_{(p,v) \in E - A_{temp}} PR(p, v)}$$

where A_{temp} is a temporal assignment result in processing, and $PR(p, v)$ is the probability of $(p, v) \in A$. In computation, we calculate it as $PR(p, v) = \frac{1}{degree(p)}$, because p has $degree(p)$ candidates for assignment in total. Obviously, $\sum_{(p,v) \in E - A_{temp}} PR(p, v)$ is the total possibility of v to be assigned accordingly. After assignment initialization, a valid assignment can be found if there is any, but the assignment may not be optimal. Therefore, we further seek to improve the assignment result in the next section.

4.2.2 Assignment Improvement

The general idea of assignment improvement is to find a better assignment on a subgraph with edges $E_{sub} \in E$ of bipartite graph BG. The pruning of bipartite edge is thus vital to determine the accuracy and efficiency. Basically, we hope to filter out bipartite graph edges to form such a bipartite sub-graph: firstly, the size of bipartite edge set E_{sub} are supposed to be less for efficient matching purpose; secondly, all the edges (i.e. passenger-vehicle pairs) in optimal assignment must be preserved in the sub-graph.

To achieve above two goals, we conduct *(1) relevance driven edge pruning* and *(2) improvement driven edge pruning* in sequential order, to filter out hopeless bipartite edges and edges unlikely to improve assignment result respectively. Then *(3) improved assignment search* is made on the sub-graph after pruning.

(1) Relevance Driven Edge Pruning. In relevance driven pruning, we try to prune out hopeless bipartite edges, e.g. edges with weights greater than the upper bound of assignment cost, as they are not relevant to query processing anymore. Let function $minW(E)$ take input as an edge set E of bipartite graph BG, and return the minimal weight of the edges in E respectively. To facilitate the derivation of upper bound of assignment cost, we require the edge set E_{sub} of sub-graph to be $weight - bounded$ as defined below.

Definition 1. *(Weight-bounded) An edge set $E_{sub} \subseteq E$ is said to be* weight-bounded *if it satisfies:*

$$minW(E - E_{sub}) \leq w(e) \qquad \forall e = (p, v) \in E_{sub}$$

Therefore, a weight-bounded edge set E_{sub} contains those and only those edges in E that have weight less than or equal to a threshold $minW(E - E_{sub})$. Conversely, all remaining edges in $E - E_{sub}$ have weight (i.e. pair cost) greater or equal to that threshold. Suppose that we are given a weight-bounded edge set E_{sub}, and an valid assignment can be found based on E_{sub}, the following theorem determines the upper bound of optimal assignment, and can be used to help us to filter out bipartite edges that is not relevant to the optimal assignment.

Lemma 1. *If a valid assignment A' is found from weight-bounded edge set $E_{sub} \subseteq E$, then the upper bound of the cost of optimal assignment A is $minW(E - E_{sub})$, and we have $A \subseteq E - E_{sub}$.*

Proof. Consider the edges in E_{sub}. First, their edge weights are less or equal to $minW(E - E_{sub})$. Second, for any bipartite edge $e = (p, v) \in E$ its edge weight is defined as the pair cost $w(e) = PC_{p,v}$. Given $A' \subseteq E_{sub}$, and we have $\psi(A') \leq minW(E - E_{sub})$ accordingly. As A is the optimal assignment, i.e. $\psi(A) \leq \psi(A')$, it must also holds that $\psi(A) \leq minW(E - E_{sub})$. Therefore, optimal assignment A has an upper bound of assignment cost at $minW(E - E_{sub})$. For $A \subseteq E - E_{sub}$, We can proof it by contradiction, i.e. UB would not be the upper bound if there exists an edge $(p, v) \in (E - E_{sub}) \cap A$ as we have $UB < minW(E_{sub}) \leq PC_{p,v}$ in such case.

The Lemma 1 informs us how to conduct bipartite edge pruning based on the upper bound of cost assignment: assume that we can find a valid assignment A at cost $\psi(A)$ from edge set E_i at loop i, then the upper bound of assignment cost becomes $UB = \psi(A)$, and all of the hopeless bipartite edges $\{e \mid e \in BG.E \wedge w(e) > UB\}$ are pruned from the valid edge set E_V (Line 1). Such an upper bound based pruning is definitely meaningful, but not enough yet, because it is unlikely to find an assignment much better than A in the next loop that carries out on E_V. In contrast, we do hope the optimal assignment can be detected in just a few loops for efficiency purpose.

(2) Improvement Driven Edge Pruning. After the relevance driven edge pruning, additional edges (especially those with higher weight) must be pruned as well, so that the assignment result can be improved. Based on E_V after relevance pruning, the problems we face are: (1) the priority of edges for pruning; (2) the ratio of bipartite edges to be preserved. The first problem is relatively easy, as edges with higher weight (i.e. greater pair cost) tend to be removed. We focus on discussing the second problem here.

As all of the bipartite edges in E_V has a potential to be part of optimal assignment, in reality, the ratio of filtering is a trade-off between accuracy and efficiency: the higher the ratio is, the the better assignment result we tend to have (as more high weight edges are pruned), but the less possible to be able to successfully find one (as the less edge candidates we have); on the other hand, if the lower the ratio is, the more likely to find a valid assignment, even though the improvement tend to be not significant. How to find a good balance is an important but challenging problem here.

We notice that the trade-off balance of improvement driven pruning is subject to the lower bound of assignment cost in reality. The low bound can be derived by two lemmas. Let function $\xi(p)$ to denote the minimal pair cost of all possible pairs $\{(p, v)|v \in V\}$ associated to a passenger $p \in P$, we have the following lemmas to find a static low bound of assignment cost.

Lemma 2. *Given a bipartite graph BG, $min(\xi(p \in P))$ is a lower bound of assignment cost.*

Proof. From the view of P, each $p \in P$ must be assigned. If there is an assignment A that $\psi(A) < min(\xi(p \in P))$, then the $p \in P$ leading to $min(\xi(p \in P))$ is not assigned as no associated edge can be used. Therefore A must not be a valid assignment, and we thus have $\psi(A) \geq min(\xi(p \in P))$.

As we can see, Lemma 2 can give us a static lower bound of assignment cost, which can be computed based on the input data. Furthermore, Lemma 3 can help us to update the lower bound along with the assignment processing.

Lemma 3. *If a valid assignment cannot be found on a weight-bounded edge set $LB = E_{sub} \subseteq E$, then $minW(E - E_{sub})$ is a lower bound of assignment cost.*

Proof. For any valid assignment A, we know $A \subsetneq E_{sub}$, so there must be an edge $e = (p, v)$ such that $e \in E - E_{sub}$ and $e \in A$. Given that $\psi(A) \geq w(e) \geq minW(E - E_{sub})$, then we know $LB = minW(E - E_{sub})$ is the lower bound of assignment cost.

The lower bound of assignment cost is an important parameter, which is used to divide edges in E_V into two sets, i.e. one set $E_C = \{e | e \in E_V \wedge w(e) \in [LB, UB]\}$ and another set $E_V - E_C$ (weight bounded to E_C), towards which different criterions are used. For $E_V - E_C$, we preserve all of them because their edge weights are even less than LB (definitely not critical pair). In contrast, edges in E_C are potential critical pair, so improvement driven pruning on E_C is necessary. A straightforward pruning approach is the ϵ (0< ϵ <1)cut pruning method (ϵCP-method), through which we only keep a ratio of ϵ edges in E_C with minimum edge weights. Though this method is practical, its performance relies on parameter ϵ that cannot be set in a rational and automatical way.

To reduce the loops in assignment processing, a more intelligent method is thus highly sought after to set ϵ in rational. Basically, the value of ϵ is subject to two factors: (1) the abundance of choices, measured as the square of ratio between the number of requested edges to that of valid edges in E_V; (2) the ratio of E_C in E_V, where we tend to be more aggressive (smaller ϵ) if their ratio is greater, and to be conservative otherwise (e.g. binary cut ϵ = 0.5). Putting the two factors together, we can normalize $\epsilon = \sqrt{\frac{|P|}{|E_V|}} \times \frac{|E_V| - |E_C|}{2 \cdot |E_V|}$, where $\frac{|E_V| - |E_C|}{2 \cdot |E_V|}$ means to be aggressive when majority of valid edges falls in E_C, and the figure of ϵ is in the range of [0, 1].

(3) Improved Assignment Search. Based on the sub-graph BG' after pruning, we try to find an improved assignment result by bipartite matching. But the improvement driven pruning may lead us unable to find a valid assignment. We say assignment improvement is successful if a valid assignment can be found, and unsuccessful otherwise. In cases it is unsuccessful, we can have a more precise lower bound of assignment cost from Lemma 3, to adjust the bipartite sub-graph to find a valid assignment in the next round. We thus further discuss how to execute iteratively to find the optimal assignment.

4.2.3 Iterative Processing

In this part, we discuss how to iteratively improve the assignment until an optimal assignment can be derived, particularly about the iterative processing procedure, the reuse of intermediate results, and stop condition. Algorithm 2 shows the mechanism of MBMA algorithm that put together assignment initialization and improvement steps. The processing starts from the assignment initialization (Line 2), and move to improvement step if a valid assignment is found. In improvement step, we use the upper and lower bounds to guide assignment evolution. The upper bound and lower bound are initialized and updated based on the Lemmas in previous section. If an improvement is successful, the upper bound is updated (Line 13); Otherwise, we adjust the lower bound and have it to be increased (Line 17). Improvement is made in loop until the lower bound equals to the upper bound, indicating the optimal assignment is found.

```
input : Passenger set $P$, vehicle set $V$, and road network $G$
output: $A$ - an optimal assignment
bipartite graph $BG = construct(O, V, G)$;
$A = assignment(BG)$;
if $A$ is NIL then
    return NIL;
end
else
    $LB = min(\xi(p \in P))$;
    $UB = \psi(A)$;
    $E_{sub} = \{e \mid w(e) \leq UB\}$;
    do
        $A = assgnImpr(BG, UB, LB)$;
        $E_{bcp} \leftarrow$ set of edges pruned by binary cut pruning;
        if $A$ is not NIL then
            $UB = \psi(A)$; $E_{sub} = \{e|w(e) \leq UB\}$;
        end
        else
            $E_{sub} = E_{sub} - E_{bcp}$; $LB = minW(BG.E - E_{sub})$;
        end
    while $LB \geq UB$ ;
    return $A$;
end
```

Algorithm 2: MBMA Algorithm

In addition, we further optimize the assignment processing by two points. In assignment improvement, we only adjust the assignment result in previous loop based on the new bipartite sub-graph (after pruning). It is thus not necessary to do the complete matching on bipartite graph in each loop, as an improved assignment tends to be found in just a few operations. Also, it can be processed in parallel if BG can be expressed by un-connected bipartite sub-graphs.

Complexity Analysis. The computational overhead of MBMA algorithm mainly for spatial query processing and assignment computation. Identical to IPA algorithm, we need $|P| + |V|$ times of fastest path search, hence the time cost of spatial query processing is $O((|P|+|V|) \times (|G.E| \times |G.V| + |G.V|^2 log|G.V|))$; As for assignment processing, assume m and n are the size of edge and vertex set of the used bipartite graph or subgraph, we go through up to $O(m)$ loops for assignment improvement according to Algorithm 4. In each loop, we find the maximum weight matching, with a time complexity $O(mn + n^2 log\, n)$. The MBMA algorithm thus costs $O(m^2 n + mn^2 log\, n)$ time in the worst case.

5 TN-FPS based Algorithm

Above solutions tend to be time consuming because of the high cost on scalable FPQs processing by A* or Dijkstra, so we further use approximate FPQ querying techniques to speed up the execution. Motivated by [8], this paper adopts a Transit Node based Fastest Path Search (TN-FPS) algorithm to find approximate FPQs results based on the *transit nodes* (i.e. important traffic intersections) selected by historical trajectory data. Particularly, the approximations of TN-FPS are only allowed if they have no or few affect on assignment accuracy. Though TN-FPS may also affect assignment precision, the effect is usually trivial because of the *observation* that passengers tend to be assigned to close vehicles, rather than those far-away.

Above observation informs us important guidelines for algorithm design: for close passenger and vehicle pairs, to return their exact fastest path for assignment precision purpose; for long distance pairs, to derive approximate fastest path for efficiency purpose. Given two far-away locations a/b (that close to n/n'), the travel cost can thus be approximated the following equation:

$$apprTC(a, b) = TC_{a,n} + TC_{n,n'} + TC_{n',b}$$

$apprTC(a, b)$ is the travel cost if a passenger goes to b from a via n and n'. It is a lower bound of the accurate travel cost according to Lemma 1, and the assignment cost is thus not under-estimated when approximation occurs.

Lemma 1. *Approximate travel cost $apprTC(a, b)$ is a lower bound of travel cost $TC_{a,b}$ from location a to b.*

The proof is omitted dy to the lack of space. The number of transit nodes (*tn*) is in reality an efficiency and accuracy trade-off: the more transit nodes are, the better accuracy can be achieved in despite of more computational cost. We set *tn* = 20 by empirical, and users can revise it for extra precision or efficiency requirements. Two major problems of TN-FPS algorithm are: (1) the selection of transit nodes; (2) how the queries are processed.

(1) Transit nodes selection. In the selection of transit nodes, we apply the Trajectory Analysis (TA) based method, which seeks to identify important intersections from trajectory data to ensure that: firstly, the transit nodes are evenly distributed over the road network; secondly, transit nodes are meaningful intersections passengers likely to pass. Basically, trajectory data is the motion history of moving objects, and it is modeled as a sequence of time stamped geo-locations $Tr = (p1, p2, ...pn)$, and each point p_i has its location $p_i.loc$ and $p_i.time$, and it can be aligned to the vertexes on the road network by some map-matching algorithms [2, 11]. That means, each trajectory can be converted to a network constrained model $Tr_N = (vi, vj, ...vn)$. Through a large trajectory dataset D, the importance of each vertexes can be seen as the frequency been passed, e.g. a intersection passed by a large number of moving objects is always an important junction, and it can be formalized as

$$Freq(v_i) = \frac{\sum_{Tr \in D} NUM(v_i, Tr_N)}{\sum_{Tr \in D} |Tr|}$$

In our approach, the network space is partitioned into —*tn*— of grids. For each grid g, we formally measure the weight of each intersection v inside grid region g as $weight(v) = \frac{Freq(v)}{D_{EU}(v,g.c)}$, where $D_{EU}(v, g.c)$ is the Euclidean distance between v and $g.c$ (i.e. the center of g. Then we select out the intersection that has the greatest weight value as a transit node. In this way, the selected transit nodes are thus rational in both spatial and importance domains.

(2) Approximate Assignment Processing. The TN-FPS algorithm computes the (possibly approximate) all-pair fastest pathes based on the transit nodes as input, and finds the assignment result by the MBMA strategy.

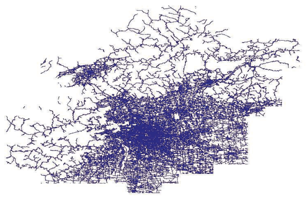
For each passenger p, the Dijkstra based traverse on road network is carried out but only limited to a small spatial space, and its accurate travel cost to all vehicles and transit nodes that it reaches are recorded in a $TC = P \times V$ matrix. To avoid impact from approximations to accuracy of assignment, the network traversal terminates if it meets two conditions: (1) the number of reached vehicles to be larger than a threshold λ. As passengers tend to be assigned to a close vehicle, the accuracy of assignment can be guaranteed if a number of closest vehicles are found; (2) the traverse must reach no less than two or more transit node, so that approximate distances to non-reached vehicles can be calculated. Similarly, the distance cost from vehicles to transit nodes and the destination are also computed. By integrating all of traversal results, we can derive all the needed (accurate and approximate) travel cost information, by which a good assignment can be computed by the MBMA strategy.

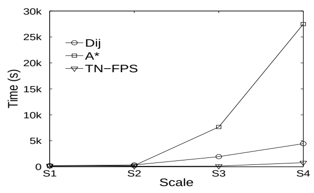
6 Experimental Study

In this section, we conduct extensive experiments on real spatial data sets to demonstrate the performance of the proposed algorithms. The data set used in our experiments are Beijing Road network, which contains 226, 238 directed edges (road segments) and 171,187 vertices (intersections), and Figure 2(a) shows their distribution on the spatial space. The used trajectory data compose over 300,000 moving objects trajectories in Beijing. All algorithms were implemented in JAVA and tested on a HP Compaq 8180 Elite (i5 650) computer with 2-core CPUs at 3.2GHz and 1.12 GHz, 4GB RAM and running Windows XP operating system.

Scale	**No. of Passengers**	**No. of Vehicles**
S1	50	10
S2	100	20
S3	500	100
S4	5000	1000

Table 1. Setting of Test Cases


(a) Road Network



(b) FPQs Runtime

Fig. 2. Road Network, FPQs Runtime

Experiments are based on 100 test cases in four scale settings shown as Table 1. In each test case, we generate the given number of passengers and vehicles based on random distribution over road network. The efficiency and accuracy of different FPQs solutions are compared first, then we overlook the performances of the assignment algorithms, and finally evaluate the final performances by integrating them together.

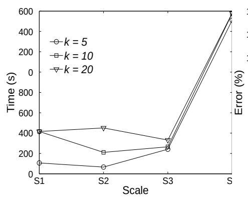
Performances on processing FPQs. Figure 2(b) shows the performances of FPQs processing using the A*, Dijkstra and TN-FPS algorithms. In comparison, the efficiency of TN-FPS significantly outperforms both of the Dijkstra and A* based algorithm according to Figure 2(b). In contrast to A* and Dijkstra algorithms that finds the accurate FPQs results, the result returned by TN-FPS may not be accurate because of the approximation. We thus further evaluate TN-FPS under different setting of tn and k.

Figure 3(a) and Figure 3(b) are the efficiency and accuracy comparison in different tn (i.e. number of transit nodes). From Figure 3(a), we can easily observe that the runtime of FPQs processing is in reverse proportion to the number of tn, which can be explained by the fact a closer transit node tends to exist, so the network traversal of FPQ processing can stop early. Also, Figure 3(b) confirms the assumption that the greater number of transit nodes tends to improve accuracy. In terms of accuracy, we measure the error of the computed results as the ratio such that $Error = \frac{\sum_{p\in P}\sum_{v\in V} apprTC_{p,v}}{\sum_{p\in P}\sum_{v\in V} TC_{p,v}}$.

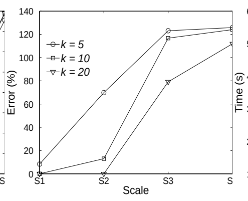
Figure 3(c) and Figure 3(d) show the comparison of algorithm efficiency and accuracy in different k (i.e. number of reached vehicles in search). The efficiency varies only for scale settings S1 and S2 because finding k vehicles for each passenger may traverse a large space of the road network in such cases. In contrast for S3 and S4, the network traverse space becomes no longer sensitive to k given lots of vehicles. Figure 3(d) shows larger k tends to improve the accuracy of FPQs processing because more traverse on road network, but the improvement is only significant for long distance FPQs.

Performances on Assignment. Based on the FPQs results, we further evaluate the performances of the proposed assignment algorithms. Figure 4(a) indicates that the I-PA algorithm tends to have poor efficiency and scalability performances, meaning it can only be used when number of passenger and vehicle are small. This phenomenon is cause by the unsatisfactory pruning effects towards the cost measure of passenger assignment problem. In contrast, the MBMA based algorithm is much more efficient, particularly when the scale of passengers and vehicles are large. Towards MBMA, we further evaluate its performance in different parameter ε, and how their result improve in different loops.

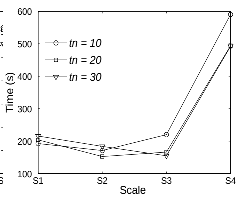
In the MBMA algorithm, ε is an important parameter to determine how we filter out bipartite edges for result improvement. Figure 4(c) and (d) show how ε impacts the



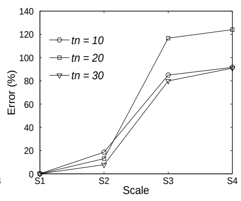
(a) Efficiency via k



(b) Accuracy via k

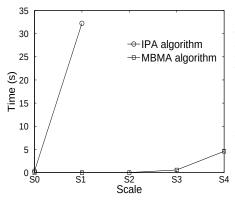


(c) Efficiency via tn

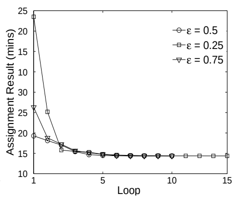


(d) Accuracy via tn

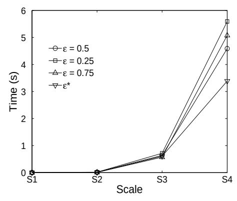
Fig. 3. FPQs Processing Performances via k and tn



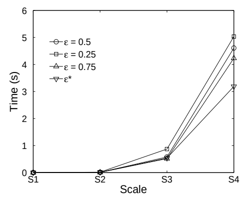
(a) Efficiency via algorithms



(b) Results via Loops



(c) Efficiency via ε (VA=0.8)



(d) Accuracy via ε (VA=0.5)

Fig. 4. Assignment Processing Performances via selected algorithm, loops, ε

efficiency of the algorithm in cases with different capacity abundance, where the vehicle abundance is defined as $VA = \frac{|P|}{\sum_{v \in V} v.c}$, meaning the ratio between requested capacity and all available capacity. In general, greater value of VA incurs more processing cost because the assignment operations are more likely to fail. We can easily observe that the different thresholds tend to have similar efficiency performances, and the adaptive setting of ε leads to the minimum processing cost. Figure 4(b) shows that the evolution of the iterative assignment loops based on different settings of ε (in scale S3).

To sum up, the experimental results implies that the MBMA algorithm can support us to find the optimal assignment efficiently, but the time cost for computing accurate FPQs is usually much greater when classical algorithms like A* or Dijkstra are used. If TN-FPS is applied for approximate FPQs processing instead, the efficiency will be improved greatly and the final passenger assignment result is near-optimal (less than 3% average error rate in all settings).

7 Conclusion and Future Work

In this paper, we define the problem of passenger assignment with road network as the underlying space, and devised two exact assignment algorithms based on integer programming and maximum bipartite matching techniques respectively. To reduce the high computational cost for scalable FPQs, an approximate solution is further introduced to find the near-optimal results in a much more efficient way. Comprehensive experiments are carried out to evaluate the performance of different algorithms.

In the future, we would like to improve the current approach by incorporating speed patterns, so that pre-computed traffic knowledge and the pattern aware spatial indexes can be used to further speed up the assignment.

References

[1] M. L. Balinski and R. E. Gomory. A primal method for the assignment and transportation problems. *Management Sci.*, 10(3):578–593, 1964.

[2] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proceedings of VLDB*, pages 853–864, 2005.

[3] T. Brunsch, K. Cornelissen, B. Manthey, and H. R?glin. Smoothed analysis of the successive shortest path algorithm. In *SODA*, pages 1180–1189, 2013.

[4] G. Dantzig and J. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, 1959.

[5] M. Desrochers, C. Jones, J. K. Lenstra, M. Savelsbergh, and L. Stougie. Towards a model and algorithm management system for vehicle routing and scheduling problems. *Decision Support Systems*, 2(25):109–133, 1999.

[6] R. Duan and H.-H. Su. A scaling algorithm for maximum weight matching in bipartite graphs. In *SODA*, pages 1413–1424, 2012.

[7] R. Geisberger, P. Sanders, D. Schultes, and D. Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *WEA*, pages 319–333, 2008.

[8] D. M. P. S. H. Bast, S. Funke and D. Schultes. In transit to constant time shortest-path queries in road networks. In *ALENEX*, 2007.

[9] M. Kaul, R. C.-W. Wong, B. Yang, and C. S. Jensen. Finding shortest paths on terrains by killing two birds with one stone. *PVLDB*, 7(1):73–84, 2013.

[10] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[11] K. Liu, Y. Li, F. He, J. Xu, and Z. Ding. Effective map-matching on the most simplified road network. In *ACM SIGSPATIAL GIS*, pages 609–612, 2012.

[12] J. Munkres. Algorithms for the assignment and transportation problems. *J. Soc. Indust. Appl. Math.*, 5:32–38, 1957.

[13] P. Sanders and D. Schultes. Engineering highway hierarchies. In *ESA*, pages 804–816, 2006.

[14] J. Sankaranarayanan and H. Samet. Query processing using distance oracles for spatial networks. *IEEE TKDE*, 22(8):1158–1175, 2010.

[15] L. H. U, K. Mouratidis, and N. Mamoulis. Continuous spatial assignment of moving users. *VLDBJ*, 19(2):141–160, 2010.

[16] L. H. U, M. L. Yiu, K. Mouratidis, and N. Mamoulis. Optimal matching between spatial datasets under capacity constraints. *ACM TODS*, 35(2):1–43, 2010.

[17] R. C.-W. Wong, Y. Tao, A. W.-C. Fu, and X. Xiao. On efficient spatial matching. In *VLDB*, pages 579–590, 2007.