An Integrated Taxi Dispatching Strategy Handling both Current and Advance Bookings

Xian WU [a], Der-Horng LEE [b]

[a,b] *Department of Civil and Environmental Engineering, National University of Singapore, Singapore, 117576*
[a] *E-mail: wuxian@nus.edu.sg*
[b] *E-mail: dhl@nus.edu.sg*

Abstract: An integrated taxi dispatching strategy namely the Advance Booking Chain Dispatching strategy (ABC-DS) is proposed in this paper, which aims to handle two types of commonly known taxi bookings: one is the Current Booking (CBK), the customer makes a booking call for a taxi that can reach him/her as early as possible; another is the Advance Booking (ABK), the customer makes a booking call and indicates the pickup time which is normally in half an hour or later. The microscopic traffic simulation is adopted as the modeling and testing approach for the proposed dispatching strategy, by which a sensitivity analysis in terms of different booking demands is conducted. The simulation results show that the proposed ABC-DS can give better operational performance in certain demand levels, which is potential to attract more customers to take the taxi by booking in advance and provide strategic implications for taxi operators.

Keywords: Advance Booking, Agent-based control system, Current Booking, Microscopic traffic simulation, Non-centralized control system

1. INTRODUCTION

One problem existed in the current taxi service market is the imbalance between taxi supply and demand (Santani *et al.*, 2008), which may cause two negative impacts on the demand side and supply side of the taxi service market: one is the longer waiting time of customers who are waiting at the taxi stand or on the street; the other is the longer empty cruising time of taxis. These two negative impacts cause not only the waste of social resources for both customers and taxi drivers but also environmental problems such as the emissions released from taxis when they are searching and waiting for customers on the congested road network.

In response to the aforementioned problem, automatic taxi dispatching approaches have been widely used in many large cities worldwide, in which customers can book taxis directly through phones or mobile devices, and the taxi operator normally employs the taxi dispatching system to deal with the customer bookings (Liao, 2003). Compared with the traditional ways of taking taxis by hailing on the street or waiting at the taxi stand, booking taxis through the dispatching system has more advantages: it provides an alternative way for the customer and the taxi driver to find each other easily. Two types of bookings have been commonly known: one is the Current Booking (CBK): the customer makes a booking call for a taxi that can reach him/her as early as possible; another is the Advance Booking (ABK): the customer makes a booking call and indicates the pickup time which is at least in half an hour later (Lee *et al.*, 2004). To differentiate between the taxi service with bookings to that without bookings, the following two terms are defined in this paper:

- Booking Taxi Service (BTS): provided for the customer who takes the taxi by

booking (either CBK or ABK) through the taxi dispatching system;

- Non-Booking Taxi Service (NBTS): provided for the customer who takes the taxi by either waiting at the taxi stand or hailing on the street.

A number of research studies on the topic of taxi service can be found in the literature (Yang and Wong, 1998; Yang *et al.*, 2002; Lee *et al.*, 2003; Liao, 2003; Lee *et al.*, 2004; Chang and Chu, 2009; Kattan *et al.*, 2010; Seow *et al.*, 2010; Sirisoma *et al.*, 2010; Aquilina, 2011; Yang and Yang, 2011), but only a few of them focused on the dispatching approaches for the BTS (either CBK or ABK). For example, for the CBK, Both Lee *et al*. (2003) and Seow and Lee (2010) explored efficient dispatching strategies for it: the former presented a shortest travel time based dispatching strategy while the latter proposed an agent-based dispatching strategy which enabled taxis to negotiate and cooperate with each other to achieve a group objective; for the ABK, Lee *et al*. (2004) applied the Pickup and Delivery Problem with Time Windows (PDPTW) approach to deal with the ABK, in which a series of ABKs would be automatically chained and then sent to individual taxis in the form of booking packages.

However, there are two limitations existed in the previous research studies on the dispatching approaches. Firstly, those studies focused on dispatching strategies either for CBK or ABK respectively, but yet considered some practical problems triggered by handling them concurrently (e.g., the interference between the two types of booking), which is inadequate to be implemented in the real-world dispatching operations: on one hand, both the CBK and the ABK may become constraints for the dispatching processes of each other, e.g., a taxi with an already confirmed ABK will not decide to accept a new CBK if the delivery of the new CBK may cause the delay of delivering the ABK; on the other hand, dealing only with the CBK or only with the ABK may under-estimate the overall operation performance, since the CBK and the ABK could be scheduled together to achieve better overall taxi service performance.

Secondly, the dispatching strategies proposed in previous research studies yet considered the NBTS, which is also inadequate to be implemented in the real-world dispatching operations: on one hand, same as the CBK, the NBTS may become the constraint for the dispatching process of the ABK, for example, a taxi with an already confirmed ABK will not pick up a customer waiting at a taxi stand or on the street if the delivery of the customer may cause the delay of delivering the ABK; on the other hand, the NBTS will directly affect the demand for BTS, for example, the customer will choose NBTS rather than BTS if the supply of NBTS is sufficient or the supply of BTS is insufficient, which may also indirectly affect the overall taxi service performance.

To handle the aforementioned limitations in previous studies, an improved dispatching strategy namely the Advance Booking Chain Dispatching Strategy (ABC-DS) is proposed in this paper, which has considered ABK, CBK and NBTS in a dynamic manner. This improved strategy is an extension of the works of Lee *et al*. (2004) which has considered ABK in a static manner only; moreover, another dispatching strategy which is similar to the one of the real-world, namely the Separate Dispatching Strategy (Sep-DS) is also developed for the comparison purpose.

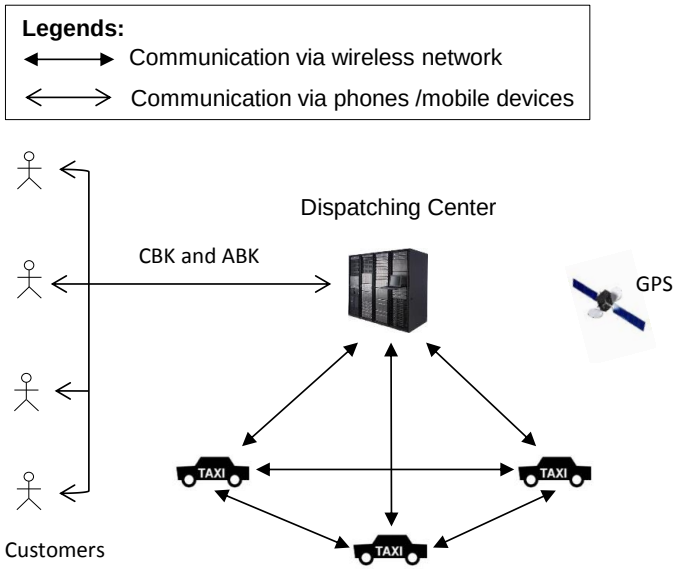
The microscopic traffic simulation is adopted as the modeling and testing approach for the aforementioned two dispatching strategies. A programming plugin using the APIs (Application Programming Interfaces) of the simulator is designed to enable the simulation of dynamic customer behaviours and the dispatching strategies. Various booking demand levels has been tested, and the performance of different dispatching strategies has been evaluated by three performance indicators: 1) the taxi Occupancy Rate (OR): the ratio between the total occupied time and the total operating time of all taxis; 2) the Customer Waiting Time (CWT):

the average waiting time of all customers; and 3) the numbers of completed and unsuccessful bookings.

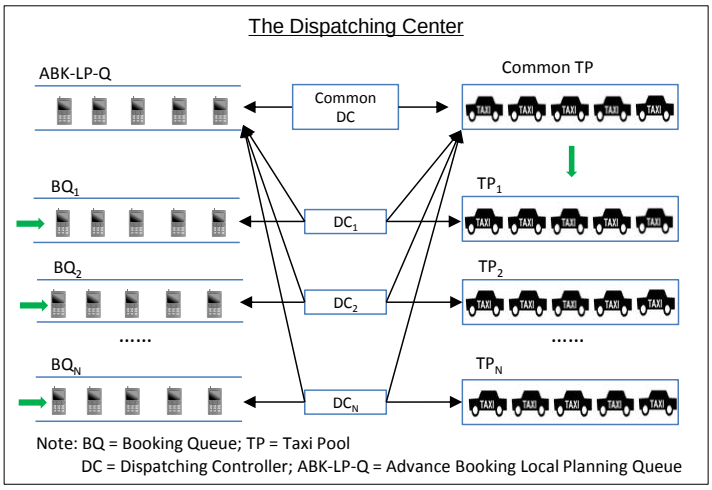
The rest of the paper will be organized as follows: the system architecture will be presented in Section 2; the operations of the proposed dispatching strategy will be introduced in Sections 3 and 4; the simulation experiments and analysis will be shown in Section 5; followed by the conclusion in Section 6.

2. SYSTEM ARCHITECTURE OF THE PROPOSED ABC-DS

2.1. The General System Architecture



(a) The interactions between customers, taxis and the dispatching center



(b) The data structures within the dispatching center

Figure 1. The non-centralized dispatching system architecture

The system architecture of the proposed dispatching strategy is shown in Figure 1, which is a non-centralized (or partially decentralized) dispatching architecture. As shown in Figure 1(a), same as the current dispatching system introduced by Liao (2003), the customer can make the

booking (CBK or ABK) for taxi service though the dispatching center either by phones or mobile devices. Taxis in this architecture are also able to communicate with each other and the dispatching center via wireless networks.

Five *operational states* are defined for the taxi: 1) *Free state*: the taxi is running on the road network, no customer is occupied and no booking request is assigned to the taxi; 2) *On-call state*: the taxi is running on the road network, no customer is occupied but there is one booking request assigned to the taxi, and the taxi is heading for the taxi stand where the booking request comes from; 3) *Boarding state:* the taxi is boarding a customer at the taxi stand, the customer may or may not have placed a booking request; 4) *Boarded state*: the taxi is running on the road with a customer occupied, and the taxi is heading for the destination of the customer; 5) *Alighting state*: the taxi is alighting a customer at the taxi stand.

As shown in Figure 1(b), at the dispatching center, the road network is assumed to be partitioned into N logical areas, which is known by the taxis and the dispatching center. The customer bookings (CBKs and ABKs) and *operational states* of taxis in the same logical area are stored in the Booking Queues (BQs) and Taxi Pools (TPs) of respective logical areas. Each logical area and corresponding BQ and TP are denoted as LA_i, BQ_i and TP_i, where $1 \leq i \leq N$. There is also a common TP which stores the *operational states* of all taxis. The entire dispatching process is controlled by a common Dispatching Controller (DC), and each logical area is also equipped with a designated DC denoted by DC_i, for all $1 \leq i \leq N$. The already assigned but yet-served ABKs will be stored in an Advance Booking Local Planning Queue (ABK-LP-Q), which will also be controlled by the common DC.

2.2. The Taxi Agent

For the taxi side, except the touchscreen, the GPS and the wireless communication devices, the taxi agent (an active software entity) is also installed in the In-vehicle Unit (IU) of the taxi, which is able to: 1) undertake computational tasks and make independent decisions on behalf of the taxi driver; and 2) communicate with the dispatching center and other taxi agents via wireless communication devices.

Two data structures are designed for the taxi agent to facilitate the decision process of dealing with the bookings (CBK or ABK) received from the dispatching center:

- Data structure 1: A pointer pointing to the CBK already confirmed by the taxi; the pointer will be set to *nil* if the taxi has no confirmed CBK;
- Data structure 2: A pointer pointing to the Advance Booking Queue (ABK-Q) which is a collection of ABKs already confirmed by the taxi and assigned to it; the ABKs in the ABK-Q are sorted by the Desired Pickup Time (DPT) indicated by the corresponding customers.

The taxi agent should ensure that the following four general operational rules are satisfied during the entire operating period of the taxi (also shown in Figure 2):

1) The pointer pointing to the CBK and the pointer pointing to the ABK-Q are attached to the taxi throughout the duration of operation;
2) If the taxi has at least one ABK in its ABK-Q, the taxi could confirm with a new CBK only when the delivery of the new CBK will not cause the delay of delivering the existing ABKs in the ABK-Q;
3) If the taxi is currently serving a CBK (i.e., the taxi is in *on-call state*) or occupied with a customer, it could confirm with a new ABK only when the delivery of the currently serviced CBK will not cause the delay of delivering the new ABK and the delivery of the new ABK will not cause the delay of delivering the existing ABKs in the ABK-Q;

4) The taxi will check from time to time to ensure that when the current time is approaching the DPT of any ABK in the ABK-Q, the taxi could decide to assign the ABK (from the ABK-Q) to the CBK pointer and start to head to the pickup location of it.

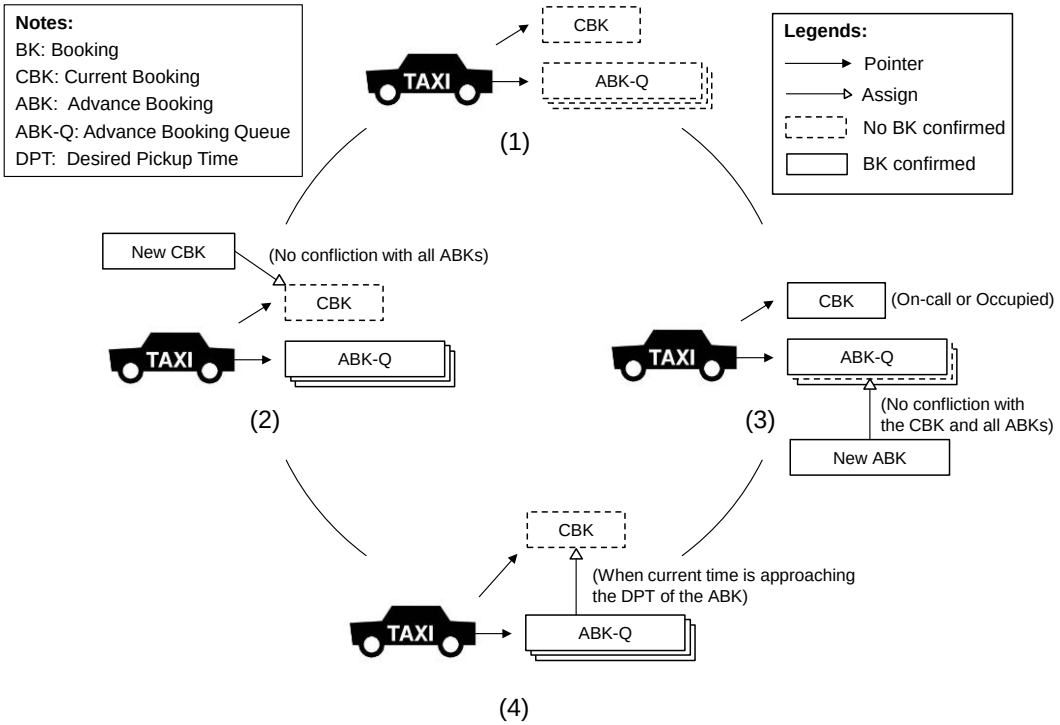


Figure 2. General rules for the decision process of the taxi agent

2.3. The Advance Booking Chain (ABC)

The proposed dispatching strategy ABC-DS enables each taxi agent to setup an Advance Booking Chain (ABC) based on its existing ABK-Q structure:

Definition 1 Advance Booking Chain (ABC): denote the set of all taxi agents as $TA=\{ta_1, \ldots, ta_{N_T}\}$ where $|TA|=N_T$, then the ABC of the taxi agent $ta_i \in TA$ can be denoted as ABC_i, which is a set of sequenced ABKs such that $ABC_i=\{ABK^i_1, \ldots, ABK^i_{N_i}\}$ where $|ABC_i|=N_i$. Denote ABK^i_m as the m^{th} ABK in ABC_i, $p^+_{i,m}$, $p^-_{i,m}$, $t^{DPT}_{i,m}$ and $t^{EDT}_{i,m}$ are the pickup location, delivery location, Desired Pickup Time (DPT) and Estimated Delivery Time (EDT) of ABK^i_m. An ABC should satisfy the following rules:

- All the pickup and delivery locations $p^+_{i,m}$ and $p^-_{i,m}$ (m=1,2,…N_i) should be visited by the taxi of ta_i;
- $p^+_{i,m}$ and $p^-_{i,m}$ (m=1,2,…N_i) should be directly connected in the route traveled by ta_i, and $p^+_{i,m}$ is visited before $p^-_{i,m}$;
- ta_i should arrive at any location $p^+_{i,m}$ (m=1,2,…N_i) within the time interval $[t^{DPT}_{i,m}-\varepsilon, t^{DPT}_{i,m}+\varepsilon]$ where ε is a pre-specified random error of $t^{DPT}_{i,m}$. If the taxi of ta_i arrives to $p^+_{i,m}$ earlier than $t^{DPT}_{i,m}-\varepsilon$, it should wait until $t^{DPT}_{i,m}-\varepsilon$;
- ABC_i may be scheduled whenever a change of the ABKs in ABC_i occurs, which is expected to achieve a better overall taxi service performance.

When new booking requests come in or the ABK-LP-Q is not empty, the dispatching operations of the proposed dispatching strategy should be carried out. Two dispatching operation phases are designed for the proposed strategy: one is the Initial Assignment Phase (IAP), and the other is the Local Planning Phase (LPP). Both of the two phases are introduced in detail in Section 3 and 4 respectively.

3. DISPATCHING OPERATIONS (1) – THE INITIAL ASSIGNMENT PHASE (IAP)

The objective of the IAP is to quickly find a feasible location for a new booking request and also to shorten the response time to the customer who has made the booking. In this section, the core process of the IAP, namely the Cost Computation Process (CCP) will be firstly introduced, which is performed by the taxi agent upon the reception of a broadcasted booking request from the dispatching center for the initial assignment purpose. Then, the general dispatching operations for both the dispatching center and the taxi agent will be introduced.

3.1. The Cost Computation Process (CCP)

The taxi agent will perform the CCP upon the reception of a booking request from the dispatching center. The CCP is a process to decide whether the booking could be assigned to the taxi, and what is the cost of the taxi for taking it. The formulation of the CCP is introduced in this section, which considers both scenarios for the CBK and ABK.

3.1.1. The general description of CCP

Assume at time *t*, once a taxi agent $ta_i \in TA$ receives a new booking request broadcasted from the dispatching center, it will start a new round of CCP immediately. The purpose to perform the CCP is to calculate the cost of ta_i for taking the new booking, and then send the cost value (if any) back to the dispatching center.

3.1.2. Handling the CBK

Denote *CBK** as the CBK that a taxi agent $ta_i \in TA$ just receives from the dispatching center. *CBK** should indicate the pickup location p^+_{CBK*} and the delivery location p^-_{CBK*} of the customer. The cost of ta_i for taking *CBK** is denoted as $c(ta_i, CBK*)$, which is calculated by ta_i for either of the following two cases:

Case 1: $ABC_i = \emptyset$, then:

$$c(ta_i, CBK*) = TT(ta_i, p^+_{CBK*}, t) \quad (1)$$

Where function $TT(ta_i, p^+_{CBK*}, t)$ is the travel time between the current location of ta_i and the current location of the customer who has made *CBK** at time *t*.

Case 2: $ABC_i \neq \emptyset$, then:

ta_i will check if there is any time confliction between *CBK** and $ABK^i_1 \in ABC_i$, where ABK^i_1 is the first (front) ABK in ABC_i:

$$t + TT(ta_i, p^+_{CBK*}, t) + TT(p^+_{CBK*}, p^-_{CBK*}, t) + TT(p^-_{CBK*}, p^+_{i,1}, t) \le t^{DPT}_{i,1} + \varepsilon \quad (2)$$

Where $p^+_{i,1}$ and $t^{DPT}_{i,1}$ are the pickup location and DPT of ABK^i_1.

If Equation (2) is not satisfied, then set $c(ta_i, CBK*) = nil$; otherwise, $c(ta_i, CBK*)$ can be calculated in the following:

$$c(ta_i, CBK*) = TT(ta_i, p^+_{CBK*}, t) + TT(p^-_{CBK*}, p^+_{i,1}, t) - TT(ta_i, p^+_{i,1}, t) \quad (3)$$

After the calculation for either Case 1 or Case 2, ta_i will inform the dispatching center the CCP result if $c(ta_i, CBK*) \neq nil$, which will be included in a message with the following format: {*ID of* ta_i, *ID of CBK**, $c(ta_i, CBK*)$} (or {taxi ID, booking ID, CCP result value}).

3.1.3. Handling the ABK

Denote *ABK** as the ABK that a taxi agent $ta_i \in TA$ just receives from the dispatching center. *ABK** should indicate the pickup location p^+_{ABK*}, the delivery location p^-_{ABK*} and the DPT t^{DPT}_{ABK*} of the customer. The cost of ta_i for taking *ABK** is denoted as $c(ta_i, ABK*)$, which is calculated by ta_i for either of the following four cases:

Case 1: $ABC_i = \emptyset$, then:

Case 1.1: ta_i has already had a CBK which is denoted as CBK_i (either in *on-call state* or *boarded state*), ta_i will check if there is any time confliction between *ABK** and CBK_i:

$$t^{EDT}_{CBK_i} + TT(p^-_{CBK_i}, p^+_{ABK*}, t) \le t^{DPT}_{ABK*} + \varepsilon \quad (4)$$

Where $t^{EDT}_{CBK_i}$ and $p^-_{CBK_i}$ are the EDT and delivery location of CBK_i. Notice that $t^{EDT}_{CBK_i} = t + TT(ta_i, p^+_{CBK_i}, t) + TT(p^+_{CBK_i}, p^-_{CBK_i}, t)$ if ta_i is in *on-call state*; otherwise, $t^{EDT}_{CBK_i} = t + TT(ta_i, p^-_{CBK_i}, t)$ if ta_i is in *boarded state*. If Equation (4) is not satisfied, set $c(ta_i, ABK*) = nil$; otherwise, $c(ta_i, ABK*)$ can be calculated by:

$$c(ta_i, ABK*) = TT(p^-_{CBK_i}, p^+_{ABK*}, t) \quad (5)$$

Case 1.2: ta_i has taken no CBK, the $c(ta_i, ABK*)$ can be calculated by:

$$c(ta_i, ABK*) = TT(p^-_{CBK_i}, p^+_{ABK*}, t) \quad (6)$$

Case 2: $ABC_i \neq \emptyset$ and $t^{DPT}_{ABK*} < t^{DPT}_{i,1}$, then:

Case 2.1: ta_i has already had a CBK which is denoted as CBK_i (either in *on-call state* or *boarded state*), ta_i will check if there is any time confliction among CBK_i, *ABK**, and $ABK^i_1 \in ABC_i$:

$$t^{EDT}_{CBK_i} + TT(p^-_{CBK_i}, p^+_{ABK*}, t) \le t^{DPT}_{ABK*} + \varepsilon \quad (7)$$

$$t^{DPT}_{ABK*} + TT(p^+_{ABK*}, p^-_{ABK*}, t) + TT(p^-_{ABK*}, p^+_{i,1}, t) \le t^{DPT}_{i,1} + \varepsilon \quad (8)$$

Where $t_{CBK_i}^{EDT}$ is calculated by the same way mentioned in Case 1.1. If either Equation (7) or Equation (8) is not satisfied, set $c(ta_i, ABK^*)= nil$; otherwise, the $c(ta_i, ABK^*)$ can be calculated by:

$$c(ta_i, ABK^*) = TT(p_{CBK_i}^-, p_{ABK^*}^+, t) + TT(p_{ABK^*}^-, p_{i,1}^+, t) - TT(p_{CBK_i}^-, p_{i,1}^+, t) \quad (9)$$

Case 2.2: ta_i has taken no CBK, ta_i will check if there is any time confliction between ABK^* and $ABK_1^i \in ABC_i$:

$$t_{ABK^*}^{DPT} + TT(p_{ABK^*}^+, p_{ABK^*}^-, t) + TT(p_{ABK^*}^-, p_{i,1}^+, t) \le t_{i,1}^{DPT} + \varepsilon \quad (10)$$

If Equation (10) is not satisfied, set $c(ta_i, ABK^*)= nil$; otherwise, the $c(ta_i, ABK^*)$ can be calculated by:

$$c(ta_i, ABK^*) = TT(p_{ABK^*}^-, p_{i,1}^+, t) \quad (11)$$

Case 3: $ABC_i \neq \emptyset$ and $t_{ABK^*}^{DPT} > t_{i,N_i}^{DPT}$, then:

ta_i will check if there is any time confliction between ABK^* and $ABK_{N_i}^i \in ABC_i$:

$$c(ta_i, ABK^*) = TT(p_{ABK^*}^-, p_{i,1}^+, t) \quad (12)$$

If Equation (12) is not satisfied, set $c(ta_i, ABK^*)= nil$; otherwise, the $c(ta_i, ABK^*)$ can be calculated in the following:

$$c(ta_i, ABK^*) = TT(p_{i,N_i}^-, p_{ABK^*}^+, t) \quad (13)$$

Case 4: $ABC_i \neq \emptyset$ and $t_{i,1}^{DPT} < t_{ABK^*}^{DPT} < t_{i,N_i}^{DPT}$, then:

ta_i will check if there is a pair of successive ABKs in ABC_i which are denoted as ABK^i_m and ABK^i_{m+1} satisfying that:

$$t_{i,m}^{DPT} + TT(p_{i,m}^+, p_{i,m}^-, t) + TT(p_{i,m}^-, p_{ABK^*}^+, t) \le t_{ABK^*}^{DPT} + \varepsilon \quad (14)$$

$$t_{ABK^*}^{DPT} + TT(p_{ABK^*}^+, p_{ABK^*}^-, t) + TT(p_{ABK^*}^-, p_{i,m+1}^+, t) \le t_{i,m+1}^{DPT} + \varepsilon \quad (15)$$

If either Equation (14) or Equation (15) is not satisfied, set $c(ta_i, ABK^*)= nil$; otherwise, the $c(ta_i, ABK^*)$ can be calculated by:

$$c(ta_i, ABK^*) = TT(p_{i,m}^-, p_{ABK^*}^+, t) + TT(p_{ABK^*}^-, p_{i,m+1}^+, t) - TT(p_{i,m}^-, p_{i,m+1}^+, t) \quad (16)$$

After the calculation for either of the cases mentioned above, ta_i will inform the dispatching center the CCP result if $c(ta_i, ABK^*) \neq nil$, which will be included in a message with the following format: {*ID of* ta_i, *ID of CBK**, $c(ta_i, ABK^*)$} (or {taxi ID, booking ID, CCP result value});

3.2. The General Dispatching Operations

3.2.1. Operations of the dispatching center

A series of dispatching operations are performed by DC_i, for $1 \leq i \leq N$ concurrently, which includes the following operation steps:

Step 1: Distribute a yet-assigned booking in BQ_i to taxis

If $BQ_i = \emptyset$ or $TP_i = \emptyset$, then go to Step 2; otherwise, take out a booking request in the front of BQ_i and attach it to DC_i . If the booking request is a CBK then broadcast it to all taxis in TP_i; otherwise, if the booking request is an ABK then broadcast it to all taxis in the common TP. Notice that taxis with the "one-time-stop" tags will not be considered for this round, and the tags will be then removed permanently. A timer in DC_i will be reset to zero to record the system waiting time of the booking after being broadcasted.

Step 2: System update and await the CCP results of taxis

If there is no booking request attached to DC_i, perform one round of the Update-BQs-TPs process and then go to Step 1; otherwise, if there is a booking request attached to DC_i, perform the Update-BQs-TPs process repeatedly until it receives a message with the CCP result from a taxi, and then insert the message into a Sequence Queue (SQ) denoted as SQ_i, which is increasingly ordered by the value of the CCP result. The message with the CCP result from a taxi should be with the following format: {taxi ID, booking ID, CCP result value}.

Step 3: Continue and repeat Step 2 until the stop criterion is satisfied

The stop criterion is: 1) the system waiting time of the booking is longer than a maximum threshold WT_{max}; or 2) the number of messages collected in the SQ_i is more than a maximum threshold N_{max}, while the system waiting time of the booking is longer than a minimum threshold WT_{min}.

Step 4: Assign the yet-assigned booking to a taxi

If there is no message in SQ_i, the booking will be placed back at the front of the BQ_i, then go to Step 1; otherwise, if there is at least one message in SQ_i, assign the booking to the taxi indicated by the message in the front of the SQ_i (i.e., the taxi with the lowest CCP result value in the SQ_i), and then inform the taxi to let the driver decide whether to accept or reject the booking.

Step 5: Process the feedback from the taxi driver

Upon the reception of the feedback from the taxi driver for the booking assignment described in Step 4, if the taxi driver has accepted the booking assignment, the dispatching center will send a confirmation message to the corresponding customer; otherwise, if the taxi driver has rejected the booking assignment, the dispatching center will compensate the booking by placing it at the front of BQ_i, and attach a "one-time-stop" tag to the corresponding taxi in the TPs for penalty purpose. Then remove the booking request previously attached to DC_i (if this booking request is an ABK, it will also be inserted into the ABK-LP-Q) and go to Step 1.

Note: The Update-BQs-TPs process is defined in the following: 1) Update the *operational state* of taxis in TP_i and the common TP, if notifications from taxis regarding the changing of current *operational state* or the changing of current logical area are received and pending to be processed; 2) Update the yet-assigned bookings in BQ_i, if new bookings are received in respective logical areas and pending to be processed.

3.2.2. Operations of the taxi agent

The taxi agent installed in the IU of each taxi will perform the following operation tasks during the operating period:

- Notify the dispatching center once the current *operational state* of the taxi has changed or the taxi has entered to a new logical area. The notification should be in the following format: {taxi ID, current logical area ID, current *operational state*};
- Start the CCP when it receives a booking request (either CBK or ABK) from the dispatching center. Then inform the dispatching center a message with the CCP result upon the completion of the CCP;
- Notify the dispatch center its taxi driver's decision (accept or reject the booking) when it receives a booking assignment from the dispatching center.

4. DISPATCHING OPERATIONS (2) – THE LOCAL PLANNING PHASE (LPP)

After the IAP for a new ABK, a subsequent LPP will be performed to further improve the booking assignment so far, which is an application of the local search techniques of the Dynamic Pickup and Delivery Vehicle Routing Problem (DPD-VRP) proposed by Fabri and Recht (2006). The motivation for the LPP is in the following:

- Assume that in the IAP, denoted the j^{th} incoming ABK as ABK_j which is assigned to a taxi agent ta_i, and the corresponding cost $c(ta_i, ABK_j)$ is the lowest one that all responding taxi agents can provide;
- The IAP is fast, but it can only provide a feasible but not optimal solution. For example, when another ABK denoted as $ABK_{j'}$ comes, it will be assigned to another taxi agent, say $ta_{i'}$ with the lowest cost $c(ta_{i'}, ABK_{j'})$ that all responding taxi agents can provide; however, if $c(ta_i, ABK_j) + c(ta_{i'}, ABK_{j'}) > c(ta_i, ABK_{j'}) + c(ta_{i'}, ABK_j)$, it will be better to assign ABK_j to $ta_{i'}$ and $ABK_{j'}$ to ta_i respectively;
- Because of the dynamic nature of the ABK, the final optimal solution can only be identified after the last ABK comes; however, at every time t during the taxi operation, there is a current feasible solution that can be improved further.

Thus, a subsequent LPP will be performed to further improve the initial assignment. Several considerations also need to be paid attention in performing the LPP, which are:

- The LPP will be performed only when there is still computational time available. If the next ABK comes quickly after the initial assignment of the current one, there may be no time to further improve it;
- The LPP only deals with the ABKs already assigned but yet to be served;
- A new round of LPP will start after the initial assignment of a new incoming ABK, which will be performed iteratively until either the LPP is finished or another ABK comes.

In the LPP, the dispatching center will broadcast all assigned but yet-served ABKs (one by one) stored in the ABK-LP-Q (as shown in Figure 1(b)) to taxis in the common TP for further improvement of the IAP solution. In this section, the core operation in LPP, namely the *move operation* will be firstly introduced, which will be performed by the taxi agent upon the reception of a broadcasted ABK from the dispatching center for the local planning purpose. Then, the searching strategy for the ABK-LP-Q and general dispatching operations for both the dispatching center and the taxi agent will be introduced in the following.

4.1. The Move Operations

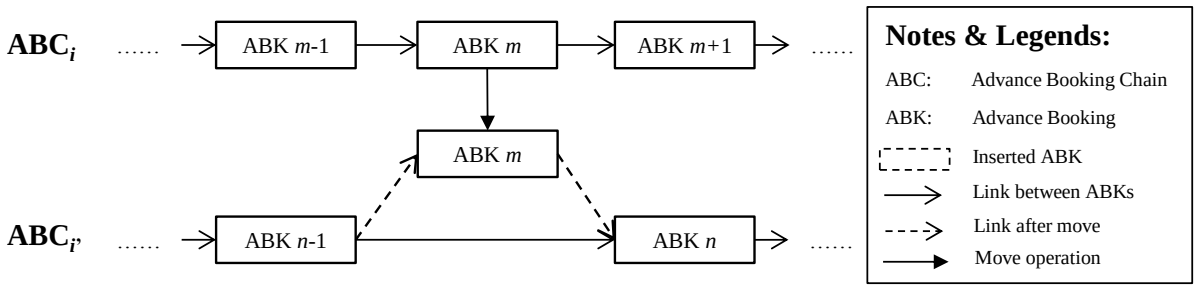
The taxi agent will perform the trail of *move operation* upon the reception of a broadcasted ABK from the dispatching center for the local planning purpose. Denote the broadcasted ABK as ABK^{i}_{m} which is the m^{th} ABK in the ABC_i of the taxi agent $ta_i \in TA$; denote the taxi agent who has received the message as $ta_{i'} \in TA$, so that the ABC of it is $ABC_{i'}$. Two types of *move operations* will be introduced: the first is the *insert move* and the second is the *swap move*.

Definition 2 insert move: the *insert move* is to remove an ABK from the ABC of its current taxi and then insert it into the one of another. Denote the function **Insert(**ABK^{i}_{m}, $ABC_{i'}$**)** performs an *insert move* to try to insert $ABK^{i}_{m} \in ABC_i$ to $ABC_{i'}$ between every possible locations. For example as shown in Figure 3(a), taxi agent ta_i tries to delete ABK^{i}_{m} from ABC_i and then insert it into $ABC_{i'}$ between $ABK^{i'}_{n-1}$ and $ABK^{i'}_{n}$. The cost reduction value $\Delta c\_insertion$ can be calculated by Equation (17).

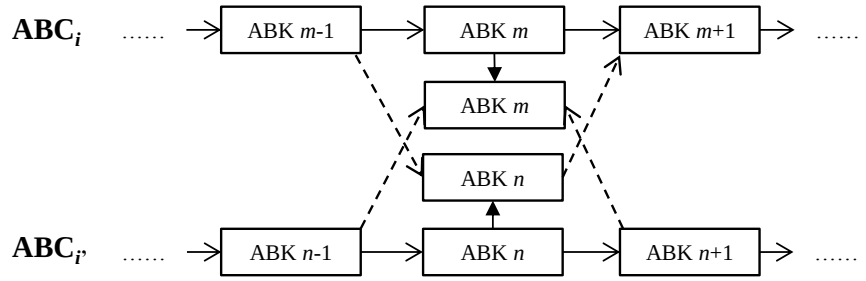
$$\begin{aligned}\Delta c\_insertion = &[TT(p^{-}_{m-1}, p^{+}_{m}, t) + TT(p^{-}_{m}, p^{+}_{m+1}, t) + TT(p^{-}_{n-1}, p^{+}_{n}, t)] - \\ &[TT(p^{-}_{n-1}, p^{+}_{m}, t) + TT(p^{-}_{m}, p^{+}_{n}, t) + TT(p^{-}_{m-1}, p^{+}_{m+1}, t)]\end{aligned} \tag{17}$$

Definition 3 swap move: the *swap move* is to swap two ABKs from two ABCs of taxis. Denote the function **Swap(**ABK^{i}_{m}, $ABC_{i'}$**)** performs an *swap move* to try to swap $ABK^{i}_{m} \in ABC_i$ with each ABK in $ABC_{i'}$. For example as shown in Figure 3(b), taxi agent ta_i tries to swap $ABK^{i}_{m} \in ABC_i$ with $ABK^{i'}_{n} \in ABC_{i'}$. The cost reduction value Δc_swap can be calculated by Equation (18).

$$\begin{aligned}\Delta c\_swap = &[TT(p^{-}_{m-1}, p^{+}_{m}, t) + TT(p^{-}_{m}, p^{+}_{m+1}, t) + TT(p^{-}_{n-1}, p^{+}_{n}, t) + TT(p^{-}_{n}, p^{+}_{n+1}, t)] - \\ &[TT(p^{-}_{m-1}, p^{+}_{n}, t) + TT(p^{-}_{n}, p^{+}_{m+1}, t) + TT(p^{-}_{n-1}, p^{+}_{m}, t) + TT(p^{-}_{m}, p^{+}_{n+1}, t)]\end{aligned} \tag{18}$$



(a) The *insertion move*



(b) The *swap move*

Figure 3. The move operations

The *move operation* (*insertion* or *swap*) will be treated as unsuccessful if $\Delta c\_insertion < 0$ or $\Delta c\_swap < 0$. The constraints described in Section 3.1 should also be satisfied in the iterations of *move operations*; otherwise, the *move operation* will be also treated as unsuccessful.

As indicated in Fabri and Recht (2006), the best computational results can be obtained by the *insert move* in their experiments for the DPD-VRP. Therefore, the LPP in the proposed dispatching strategy will only consider the *insert move*.

Upon the completion of the trial of the *move operation* performed by the taxi agent $ta_{i'} \in TA$, if it is successful, $ta_{i'}$ will send the cost reduction value $\Delta c\_insertion$ back to the dispatching center.

4.2. The Searching Strategy for the ABK-LP-Q

The ABK-LP-Q is employed to store all assigned but yet-served ABKs for the local planning purpose. As introduced in Section 3.2, once an ABK is confirmed to be assigned to a taxi, it will be inserted into the ABK-LP-Q immediately. Moreover, once an ABK becomes unavailable to be moved to other taxis, i.e., the taxi carrying the ABK is running to the pickup location of the ABK, the taxi will notify the dispatching center to remove the ABK from the ABK-LP-Q.

The LPP consists of cycles of operations, in each cycle one ABK in the ABK-LP-Q will be selected to perform the trail of *move operation*, i.e., being inserted to every operating taxi (except the one currently carrying the ABK) for seeking further cost reductions. As discussed in 4.1, only the successful *move operations*, i.e., $\Delta c\_insertion > 0$ for *insert move* will be considered as the candidates for the final *move operations*. Due to the real-time nature of the proposed dispatching strategy, it is expected that ABKs with higher cost reduction values (i.e., $\Delta c\_insertion$) should be trailed as early as possible; however, only after all trails of *move operations* between each ABK in the ABK-LP-Q and each taxi in the common LP then we can only know which ABKs are with higher cost reduction values, which is obviously unrealistic in a real-time system environment. Therefore, by following the approach in Fabri and Recht (2006), the ABKs in ABK-LP-Q will be decreasingly ordered by the *saved cost*, denoted as $c\_saved(ABK^i_m)$ rather than the cost reduction value $\Delta c\_insertion$.

Definition 4 saved cost $c\_saved(ABK^i_m)$: denote ABK^i_m as the m^{th} ABK in the ABC of taxi agent ta_i, then $c\_saved(ABK^i_m)$ is the saved cost when ABK^i_m is removed from its current taxi agent ta_i, which can be calculated in Equation (19):

$$c\_saved(ABK^i_m) = [TT(p^-_{m-1}, p^+_m, t) + TT(p^-_m, p^+_{m+1}, t) - TT(p^-_{m-1}, p^+_{m+1}, t)] \quad (19)$$

Notice that $c\_saved(ABK^i_m)$ is calculated by the taxi agent carrying the ABK, which will be sent back to the dispatching center whenever the taxi agent communicates with the dispatching center for the ABK related tasks.

A Tabu set is also designed in the LPP for the purpose of preventing an ABK from being assigned to a taxi twice (which may cause the block of the local searching) before a certain number successful *move operations* are performed. The Tabu set will be introduced in detail in Section 4.3.

4.3. The General Dispatching Operations

In summary of the ideas of LPP in Sections 4.1 and 4.2, the general dispatching operations for both the dispatching center and the taxi agent of the proposed dispatching strategy is

introduced in detail in this section.

Once a new ABK is successfully inserted to the ABK-LP-Q, a new round of LPP will be triggered. Notice that a data structure namely the ABK-Pointer is defined for facilitating the searching process of the LPP, and the Tabu set which stores the visited ABKs in the LPP is denoted as *TS** .

4.3.1. Operations of the dispatching center

A series of operations are performed by the common DC for the LPP, which includes the following operation steps:

Step 1: Initialization a new LPP

Stop the yet-completed LPP if there is one, update the sequence of the ABK-LP-Q by the *saved cost* of each ABK in the ABK-LP-Q.

Step 2: Distribute a ABK in ABK-LP-Q

If the ABK-Pointer is currently pointing to the end of the ABK-LP-Q, then exit the current LPP. If the ABK currently pointed by the ABK-Pointer is already stored in *TS**, then the ABK-Pointer will be moved to the next position in the ABK-LP-Q, and restart Step 2; otherwise, insert the ABK to *TS** and broadcast the ABK to all taxis in the common TP. A timer will be reset to zero in the common DC to record the system waiting time after the broadcasting of the ABK

Step 3: Wait for the response from taxis

Perform the Update-ABK-LP process and wait until the reception of a message with the cost reduction value (i.e., $\Delta c\_insertion$) from a taxi, and then insert the message into a Sequence Queue (SQ) denoted as *SQ**, which is decreasingly ordered by the cost reduction value. The message should be in the following format: {Taxi ID, ABK ID, cost reduction value}.

Step 4: Continue and repeat Step 3 until the stop criterion is satisfied

The stop criterion is: 1) the system waiting time is longer than a maximum threshold WT_{max}; or 2) the number of messages collected in the *SQ** is more than a maximum threshold N_{max}, while the system waiting time is longer than a minimum threshold WT_{min}.

Step 5: Instruct the final move operation to the related taxis

If there is no message in *SQ**, the ABK-Pointer will be moved to the next position in the ABK-LP-Q, then go to Step 2; otherwise, if there is at least one message in *SQ**, then instruct the two taxis - the 1st taxi is the one assigned with the ABK pointed by the ABK-Pointer and the 2nd taxi is the one indicated by the message in front of *SQ** to perform the final *move operation*, i.e., to move the ABK pointed by the ABK-Pointer from the 1st taxi to the 2nd taxi. The instruction should be in the following format: {ABK ID, 1st Taxi's ID, 2nd Taxi's ID}. Then empty the *SQ**.

Step 6: Process the feedback from the taxi drivers

Upon the reception of the feedback from the taxi driver for the final *move operation* described in Step 5, if the taxi drivers have accepted it, the dispatching center will send the confirmation messages to the corresponding customers and update the sequence of the ABK-LP-Q by the *saved cost* of each ABK in the ABK-LP-Q. Notice that upon the acceptation of the final *move operation* by the taxi drivers, if the total number of successful *move operations* exceeds a max number N_{tabu}, then the Tabu set *TS** will be emptied. Then the ABK-Pointer will be moved to the next position in the ABK-LP-Q, and go to Step 2.

Note: The Update-ABK-LP process is defined as follows: receive the notifications from the taxi agents for the status changing of ABKs (e.g., when a taxi starts to serve an ABK), then remove the corresponding ABKs from the ABK-LP-Q.

4.3.2. Operations of the taxi agent

The taxi agent installed in the IU of each taxi will perform the following operation tasks during the operating period:

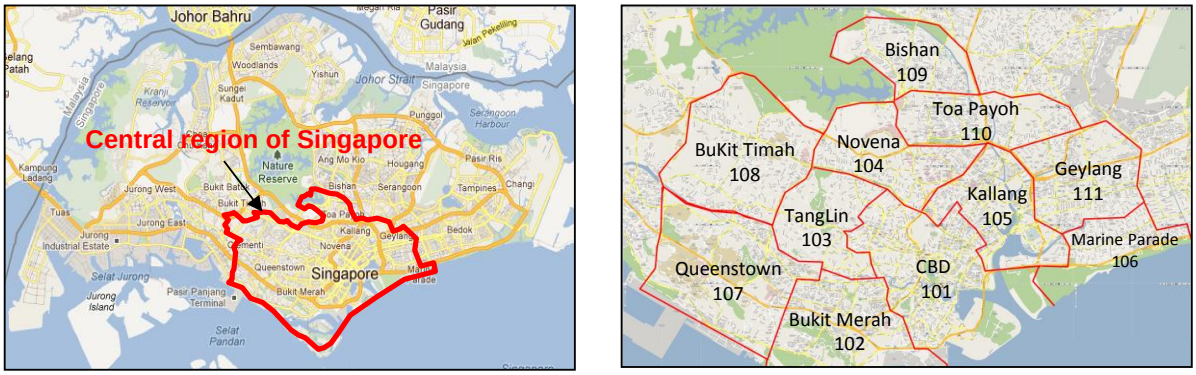
- Notify the dispatch center for the status changing of its ABKs (e.g., when it starts to serve an ABK);
- Calculate the cost reduction value when it receives an ABK (with its *saved cost*) from the dispatching center for local planning. Then inform the dispatching center a message with the cost reduction value upon the completion of the calculation. The details of the calculation process has be introduced in Section 4.1;
- Notify the dispatch center its taxi driver's decision (accept or reject the final *move operation*) when it receives an instruction of the final *move operation* from the dispatching center.

5. SIMULATION EXPERIMENTS

5.1. The Simulation Model

The microscopic traffic simulation is adopted as the modeling and analysis approach in this paper, which includes a microscopic traffic simulation software - PARAMICS (Quadstone, 2010) and a plugin designed by programming with the Application Program Interfaces (APIs). The plugin enables the software to simulate the customer dynamic behaviors, the taxi operations and the the dispatching strategies.

In the simulation model, the central region of Singapore is chosen as the study area which covers an area of around 17km×10.5km. The central region of Singapore and the partition of the study area are both shown in Figure 4. There are totally 11 logical areas, 993 nodes and 2,348 links in the network, as well as 193 zones defined for representing taxi stands.



(a) The study area: central region of Singapore (b) The partition of the study area

Figure 4. The simulation model for the test of the proposed dispatching strategy

5.2. Simulation Settings

There are totally 500 taxis to be simulated by the model introduced in Section 5.1. A series of experiments are conducted for 10 demand levels of the customer which are from 800 arrivals/hour to 8,000 arrivals/hour with the increment of 800 arrivals/hour.

Denote ABK_R as the ratio of customers who will make the ABK, so that the rest (1-ABK_R)% of customers will chose NBTS or make the CBK (notice that in the simulation, customers making CBKs are assumed to be initially waiting at taxi stands for the NBTS until time T_0, and then start to book for taxis). As mentioned in Wang (2004), there are possibilities that customers may prefer the ABK in certain conditions; therefore, in order to test the effect of the ABK to the dispatching performance, we have tested 3 different levels of the ABK_R which are 25%, 50% and 75% respectively.

The proposed dispatching strategy ABC-DS is simulated and compared with another dispatching strategy namely the Separate Dispatching Strategy (Sep-DS). The difference between the ABC-DS and the Sep-DS is that the former allows the formation of the Advance Booking Chain (ABC) while the later does not allow it (in fact, the maximum length of the ABK-Q of each taxi agent is set to 1 in Sep-DS, which is a strategy similar to the one of the real world). The performance of each strategy in each demand scale level is evaluated in terms of the OR and CWT. In addition, the number of completed bookings and the number of unsuccessful bookings (customers waiting at taxi stands more than a time $T_{\max}$) are also recorded in each demand scale level.

Other parameters for the simulation experiments are set in the following: The customer's start-to-book time T_0 is set to 3 minutes, and the maximum time $T_{\max}$ the customer can wait at the taxi stand is set to 10 minutes. The standard deviation of the DPT of all ABKs is set to 30 minutes. The total simulation period is 2 hours plus 30 minutes warm-up time. The max number of successful *move operations* N_{tabu} for emptying the Tabu set *TS** is set to 2. Other parameters of the simulation model are set based on the field observation and survey.

To facilitate the simulation process without loss of generality, the stop criterions for waiting for the response from the taxi agents in both IAP and LPP operations will not be modeled, i.e., the dispatching center will wait until the last taxi agent have responded.

5.3. Simulation Result (1): Test of the CCP in IAP

The test for the CCP is shown in Figure 5, which is grabbed from one of the simulation runs when the demand scale level = 6.

In the test, an ABK denoted as *ABK** and with the pickup location $p^{+}_{ABK*}=135$, the delivery location $p^{-}_{ABK*}=130$, the DPT $t^{DPT}_{ABK*}=5265$ and the EDT $t^{EDT}_{ABK*}=5389$ is broadcasted to all taxis, where the pickup/delivery locations are indicated by the IDs of the taxi stands in the simulation, and the DPT/EDT are indicated by the time passed (in seconds) from the beginning of the simulation. Upon the reception of the information of *ABK**, the taxi agents will calculate the cost for taking the new ABK following the procedures introduced in Section 3.1. In the last, the dispatching center will assign *ABK** to the taxi with the lowest cost which is 354 seconds as shown in Figure 5.

5.4. Simulation Result (2): Test of the LPP

The test for the LPP is shown in Figure 6, which is grabbed from one of the simulation runs when the demand scale level = 6, time t = 3601 seconds. In the test, there are totally 2491 ABKs stored in the ABK-LP-Q for further improvement. Notice that it is artificially assumed

that every ABK in the ABK-LP-Q will be tried with the *move operation* in this test (only), which aims to test the efficiency of the proposed LPP.

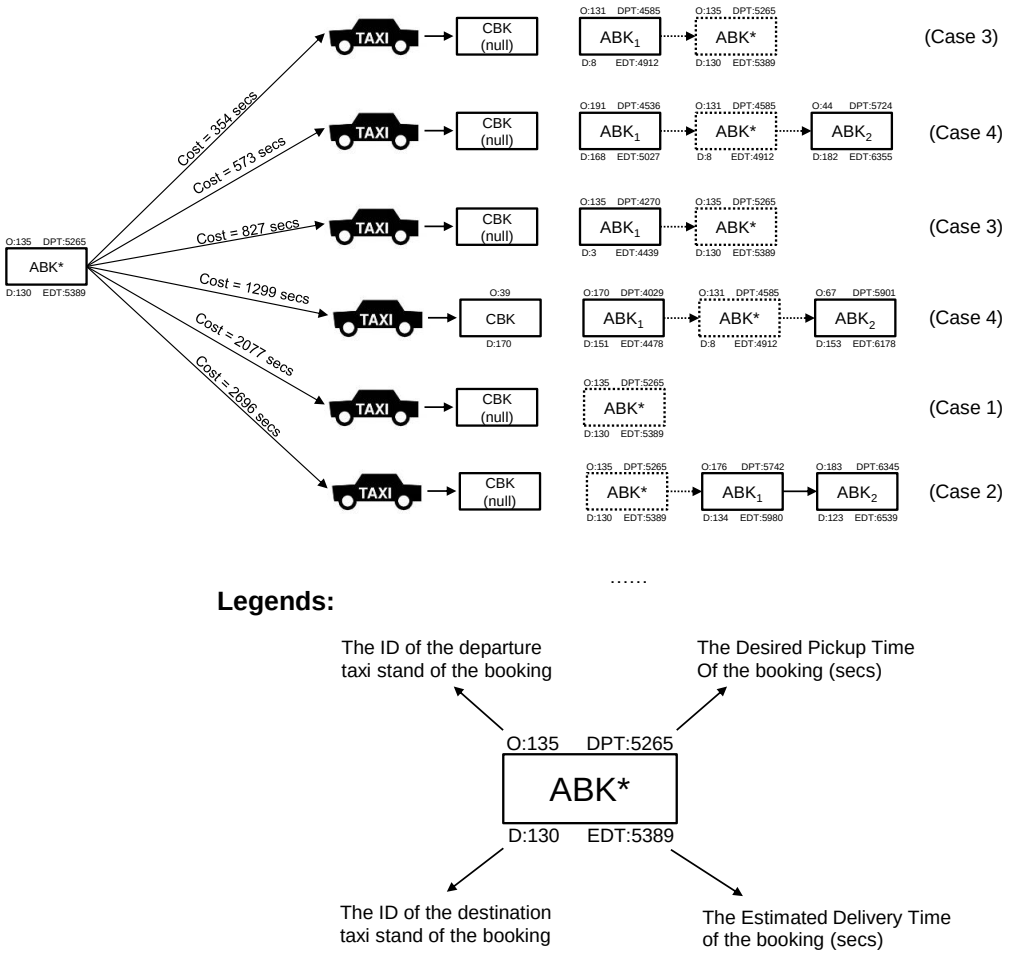


Figure 5. The test for the CCP in the simulation

As shown in Figure 6, all the ABKs with successful *move operations* are drawn based on their sequence numbers processed by the dispatching system.

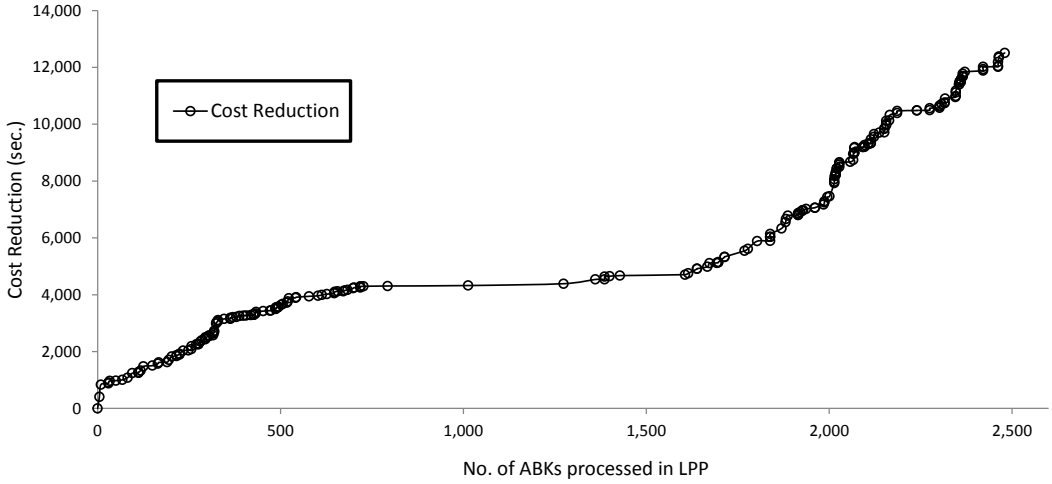


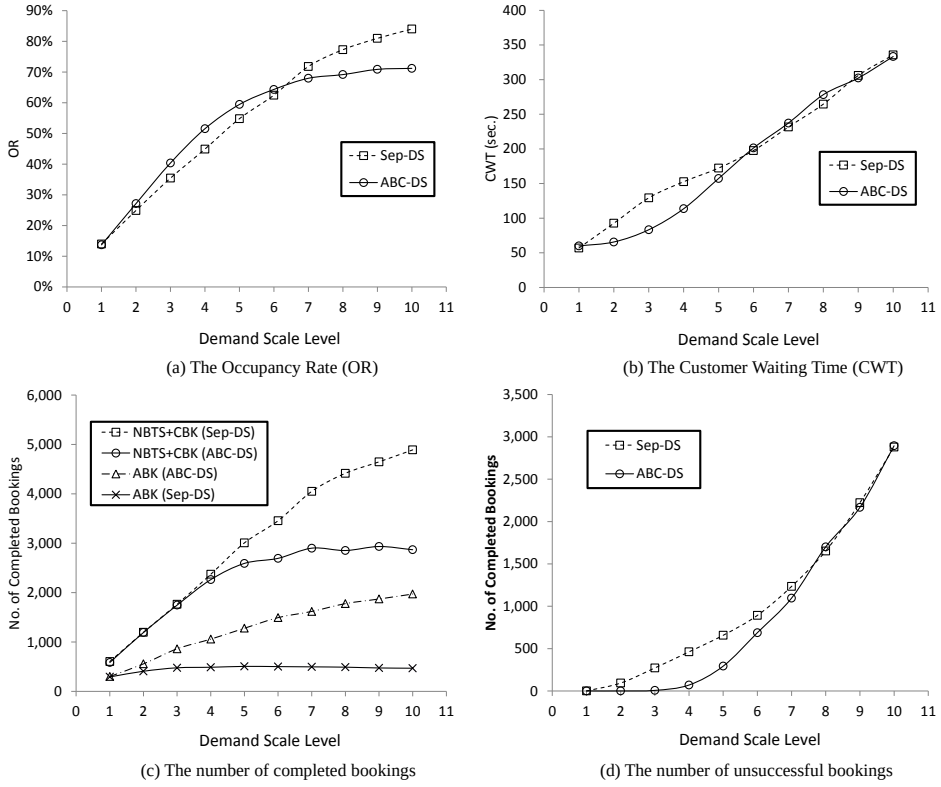
Figure 6. The test for the LPP in the simulation

5.5. Simulation Result (3): Sensitivity Analysis

The simulation results with the analysis for the scenario ABK_R=50% are firstly presented. Then, the simulation results for the scenarios ABK_R=25% and 75% are also presented in the following for the comparison purpose.

5.5.1. Scenario 1: ABK_R=50%

It can be found in Figure 7(a) that the ABC-DS can improve the overall OR up to 6.77% (demand scale level = 4) compared with Sep-DS when the demand scale level < 6. This can be explained as follows: firstly, the ABC-DS chains up ABKs which enables the taxi to avoid unnecessary empty cruising time to a certain extent; secondly, the ABC-DS tries to assign the ABK to the taxi with the highest stimulus (the higher occupancy time with the lower empty cruising time), which may directly lead to the increase of the OR of the entire taxi fleet.



(a) The Occupancy Rate (OR)

(b) The Customer Waiting Time (CWT)

(c) The number of completed bookings

(d) The number of unsuccessful bookings

Figure 7. The simulation results (ABK_R=50%).

However, it is also found in Figure 7(a) that in terms of improving the overall OR, the Sep-DS performs better when the demand scale level is high (> 6). This can be also explained as follows: when the demand scale level is high (> 6), most taxis will be too busy to confirm the incoming CBKs so that the demand for the NBTS will become higher. In such a situation, a taxi under Sep-DS will have higher chance to offer the NBTS to a new customer at the taxi stand where it has just dropped one; however, a taxi under ABC-DS will has little chance to offer the NBTS, since it has to head for the pickup location of the next ABK emptily after it drops one at a taxi stand, which may cause the increase of the taxi empty cruising time, in other words, cause the decrease of the OR of the entire taxi fleet.

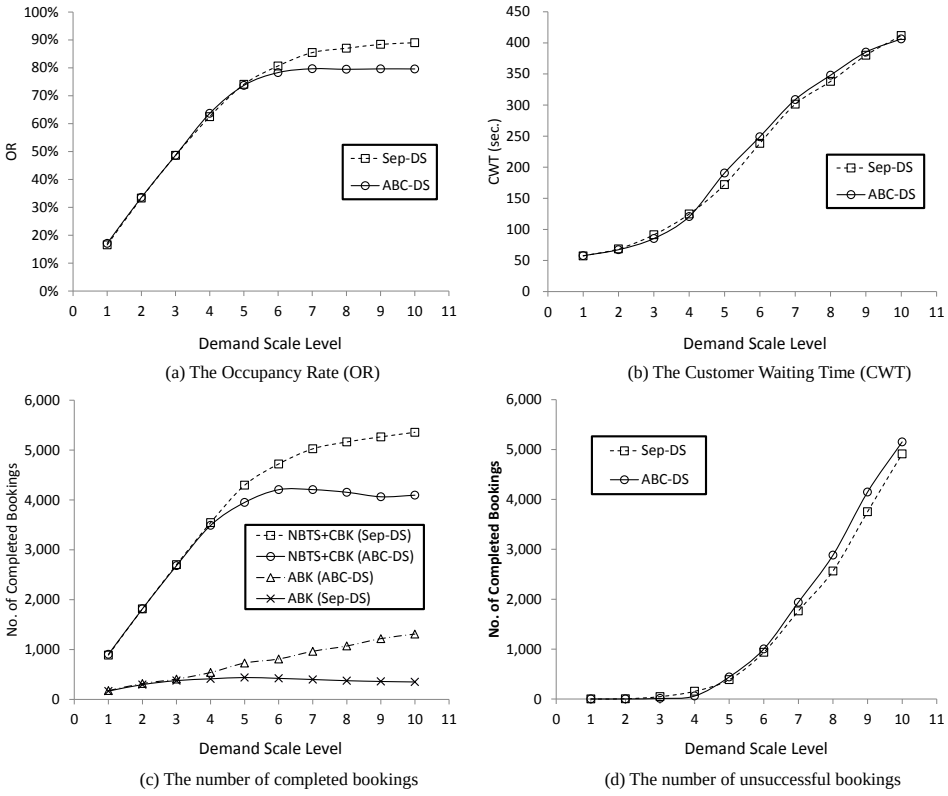
Figure 7(b) shows that the ABC-DS can shorten the CWT as much as 34.65% (demand scale level = 3) when the demand scale level is low (< 6); however, when the demand scale level is high (> 6), the ABC-DS shows no advantage than the Sep-DS in terms of reducing the

CWT.

Figure 7(c) shows that the ABC-DS can complete more ABKs but less NBTS+CBK than Sep-DS. This may be due to the reason that the former strategy allows more than one ABKs to be assigned to one taxi which enables the taxi to serve more ABKs. Figure 7(d) shows that the total number of unsuccessful bookings in the ABC-DS is less when the demand scale level < 7, which indicates that the ABC-DS may be a potential way in attracting more customers to take the taxi by booking in advance.

5.5.2. Scenario 2: ABK_R = 25%

The ratio of ABKs in scenario 2 is much lower than that in Scenario 1, in which the dispatching performance of ABC-DS is no better than that of Sep-DS. For example, in Figure 8(a), the ABC-DS is unable to improve the overall OR compared with Sep-DS when the demand scale level ≤ 5, and even worse when the demand scale level > 5; in Figure 8(b), the ABC-DS shows no advantage than the Sep-DS in terms of reducing the CWT; Figure 8(c) shows the similar pattern of the number of completed bookings as in Figure 7(c); in Figure 8(d), the ABC-DS shows no advantage than the Sep-DS in terms of reducing the number of unsuccessful bookings.



(a) The Occupancy Rate (OR)

(b) The Customer Waiting Time (CWT)

(c) The number of completed bookings

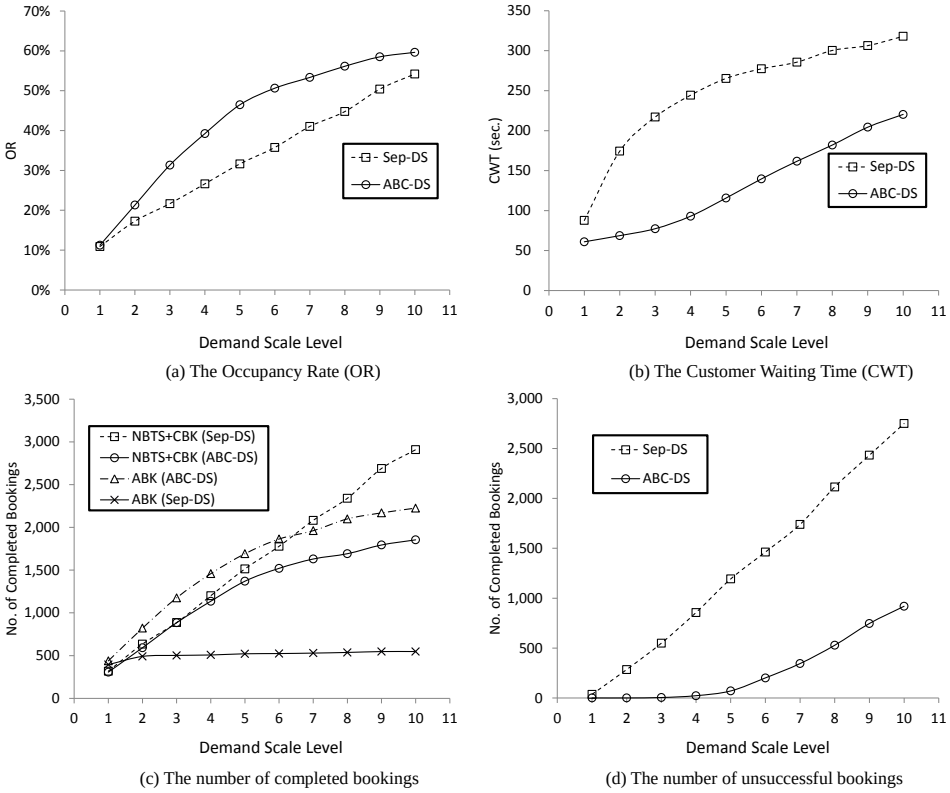
(d) The number of unsuccessful bookings

Figure 8. The simulation results (ABK_R=25%)

5.5.3. Scenario 3: ABK_R=75%

The ratio of ABKs in scenario 3 is higher than that in Scenario 1, in which the dispatching performance of ABC-DS is better than that of Sep-DS. For example, in Figure 9(a), the ABC-DS can improve the overall OR compared with Sep-DS when the demand scale level>1; in Figure 9(b), the ABC-DS is better than the Sep-DS in terms of reducing the CWT; Figure 9(c) shows the similar pattern of the number of completed bookings as in Figure 7(c); in

Figure 9(d), the ABC-DS can reduce the number of unsuccessful bookings compared with Sep-DS when the demand scale level>1.



(a) The Occupancy Rate (OR)

(b) The Customer Waiting Time (CWT)

(c) The number of completed bookings

(d) The number of unsuccessful bookings

Figure 9. The simulation results (ABK_R=75%)

In all, several implications can be obtained from the simulation results, which could become the technical support for taxi operators to make strategic decisions (notice that those taxi operators include large taxi operators with dominant market shares and small taxi operators with lower market shares):

- The ABC-DS is more suitable than the Sep-DS for the scenarios when ABK_R is high (e.g., ABK_R=75%). In these scenarios, all taxi operators are encouraged to employ the ABC-DS rather than the Sep-DS;
- The ABC-DS is no better than the Sep-DS for the scenarios when ABK_R is low (e.g., ABK_R=25%). In these scenarios, all taxi operators are encouraged to employ the Sep-DS rather than the ABC-DS;
- For the scenarios when ABK_R is in the middle range (e.g., ABK_R=50%), the demand scale level is a critical factor for the decision makers to consider: the ABC-DS is recommended to the small operators who have comparatively low booking demands; the Sep-DS is recommended to the large operators who have comparatively high booking demands.

6. CONCLUSIONS

This paper has further improved previous studies on the taxi dispatching approaches, in which an integrated dispatching strategy namely the Advance Booking Chain Dispatching Strategy (ABC-DS) is proposed. This work has extended the works of Lee *et al*. (2004) by further considering the effects of the CBK and the NBTS together with the ABK. Meanwhile, another

dispatching strategy which is similar to the one of the real world, namely the Separate Dispatching Strategy (Sep-DS) is also developed for the comparison purpose. The microscopic traffic simulation is adopted as the modeling and analysis approach for the studied problem. Several implications have ben obtained from the simulation results, which could serve as the technical support for taxi operators to make strategic decisions.

REFERENCES

Aquilina, M. (2011) Quantity De-restriction in the Taxi Market Results from English Case Studies. *Journal of Transport Economics and Policy*, 45, 179-195.

Chang, S.K.J., Chu, C.H. (2009) Taxi Vacancy Rate, Fare, and Subsidy with Maximum Social Willingness-to-Pay Under Log-Linear Demand Function. *Transportation Research Record*, (2111), 90-99.

Fabri, A., Recht, P. (2006) On dynamic pickup and delivery vehicle routing with several time windows and waiting times. *Transportation Research Part B-Methodological*, 40(4), 335-350.

Kattan, L., de Barros, A., Wirasinghe, S.C. (2010) Analysis of work trips made by taxi in Canadian cities. *J. Adv. Transp.*, 44(1), 11-18.

Lee, D.H., Wang, H., Cheu, R.L. (2004) Trip-Chaining for Taxi Advance Bookings:a Strategy to Reduce Cost of Taxi Operations, *the 83th Annual Meeting of the Transportation Research Board*. Transportation Research Board of the National Academies, Washington, D.C.

Lee, D.H., Wang, H., Cheu, R.L., Teo, S.H. (2003) Taxi Dispatch System based on Current Demands and Real-time Traffic Conditions. *Transportation Research Record*, (1882), 193-200.

Liao, Z.Q. (2003) Real-time taxi dispatching using global positioning systems. *Communications of the ACM*, 46(5), 81-83.

Quadstone (2010) **http://www.paramics-online.com**.

Santani, D., Balan, R.K., Woodard, J.C. (2008) Spatio-temporal Efficiency in a Taxi Dispatch System, *the 6th International Conference on Mobile Systems, Applications, and Services*, Breckenridge, Colorado, USA.

Seow, K.T., Dang, N.H., Lee, D.-H. (2010) A Collaborative Multiagent Taxi-Dispatch System. *IEEE Trans. Autom. Sci. Eng.*, 7(3), 607-616.

Seow, K.T., Lee, D.H. (2010) Performance of Multiagent Taxi Dispatch on Extended-Runtime Taxi Availability: A Simulation Study. *IEEE Transactions on Intelligent Transportation Systems*, 11(1), 231-236.

Sirisoma, R., Wong, S.C., Lam, W.H.K., Wang, D., Yang, H., Zhang, P. (2010) Empirical evidence for taxi customer-search model. *Proceedings of the Institution of Civil Engineers-Transport*, 163(4), 203-210.

Wang, H. (2004) Improving Taxi Dispatch Services with Real-Time Traffic and Customer Information. National University of Singapore.

Yang, H., Wong, S.C. (1998) A Network Model of Urban Taxi Services. *Transportation Research Part B-Methodological*, 32(4), 235-246.

Yang, H., Wong, S.C., Wong, K.I. (2002) Demand-Supply Equilibrium of Taxi Services in a Network under Competition and Regulation. *Transportation Research Part B-Methodological*, 36(9), 799-819.

Yang, H., Yang, T. (2011) Equilibrium properties of taxi markets with search frictions. *Transportation Research Part B-Methodological*, 45(4), 696-713.