Design and Modeling of Real-time Shared-Taxi Dispatch Algorithms

Jaeyoung Jung*
Assistant Research Scientist, Ph. D.
Institute of Transportation Studies
University of California, Irvine
4000 Anteater Instruction and Research Bldg (AIRB)
Irvine, CA 92697-3600
Phone: +1–949–824–5989 / FAX: +1–949–824–8385
Email: jaeyounj@uci.edu

R. Jayakrishnan
Professor
Institute of Transportation Studies
Department of Civil and Environmental Engineering
University of California, Irvine
4000 Anteater Instruction and Research Bldg (AIRB)
Irvine, CA 92697-3600
Phone: +1–949–824–2172 / FAX: +1–949–824–8385
Email: rjayakri@uci.edu

Ji Young Park
Associate Research Fellow, Ph. D.
Office for Convergence Technology
The Korea Transport Institute
315, Goyangdaero Ilsanseo-gu
Goyang-si, Gyeonggi-do 411-701, South Korea
Phone: +82-31-910-3163 / FAX: +82-31-910-3227
Email: parkjy@koti.re.kr

*Corresponding author

Submission data: August 1, 2012
Paper length: 5850 + 250*8 (figures and tables) = 7850 words

ABSTRACT
Taxi is certainly the most popular type of on-demand transportation service in urban areas because taxi dispatching systems offer more and better services in terms of shorter wait times and travel convenience. However, a shortage of taxicabs has always been critical in many urban contexts especially during peak hours and taxi has great potential to maximize its efficiency by employing shared-ride concept. There are recent successes in real-time ridesharing projects that are expected to bring substantial benefits on energy consumption and operation efficiency, and thus it is essential to develop advanced vehicle dispatch algorithms to maximize occupancy and minimize travel times in real-time. This paper investigates how taxi services can be improved by proposing shared-taxi algorithms and what type of objective functions and constraints could be employed to prevent excessive passenger detours. Hybrid Simulated Annealing (HSA) is applied to dynamically assign passenger requests efficiently and a series of simulations are conducted with two different taxi operation strategies. The simulation results reveal that allowing ride-sharing for taxicabs increases productivity over the various demand levels and HSA can be considered as a suitable solution to maximize the system efficiency of real-time ride sharing.

Design and Modeling of Real-time Shared-Taxi Dispatch Algorithms

1. INTRODUCTION

Real-time ridesharing is defined as dynamically utilizing the empty seats in passenger cars by assigning passengers on demand, which is quite different from the early version of carpooling projects that were not feasible for real-time response due to the lack of advanced information technologies. As real-time ridesharing projects have been successfully initiated, the potential benefits of ridesharing are expected to be substantial in reducing fuel consumption, carbon emissions, and traffic congestion. For customers, ridesharing can also reduce travel costs for driving and parking. Newer design of Demand Responsive Transit (DRT) with true real-time routing, which can be named RTRT (Real-Time Routed Transit Systems) have emerged in recent years (1, 2, and 3), but those concepts are not fully refined for practical service in real world. Zimride, Avego, and SideCar (4, 5, and 6) are well known services recently initiated for private ridesharing by simply matching drivers and riders in real-time as passengers travel in urban areas. These services utilize vehicles operated by regular car owners and not commercial drivers.

However, it has been known that those private ridesharing services could raise potential concerns about passenger insurance and fare-collection system since the service vehicles are operated by private vehicles and drivers. In addition, rideshare on any given vehicle can be offered only when that private vehicle is moving, and not all the time. To overcome these issues, real-time shared-taxi offers an alternative that is similar. Shared-taxi can be characterized as an on-demand ride-share service operated by an online dispatch center such that the system is capable of taking service requests from individual customers in real-time and establishing service vehicle schedules. While real-time dispatching of such systems is a new concept, shared-ride in taxi, at least in certain forms, is not new. According to a study by Cervero (7), it already flourished in Washington D.C. during World War II due to gas shortage. Taxicab drivers displayed their current destination signs so that riders would hail the cabs to share the ride to the same destinations. For an example of an online real-time response taxi service, Uber (8) in certain U.S. urban areas provides a Smartphone-based on-demand taxi service, though not involving ridesharing yet because taxi-sharing is currently prohibited by law in many cities in the U.S. However, shared-taxi services are being initiated in many countries. In China, the Beijing government recently allowed taxi-sharing due to the shortage of taxicabs during rush hours. That scheme however required all passengers to get in the car at the same location. In Singapore, Taiwan, and Japan, dynamic shared-taxi services are conducted or initiated to link passengers who travel to the same area (9, 10, and 11).

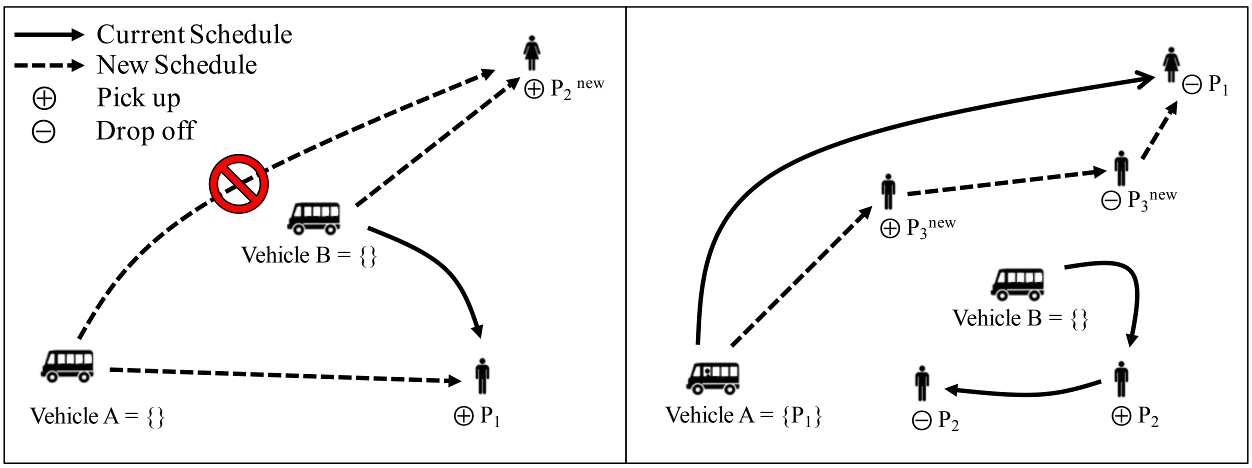
In this paper, an optimization scheme is developed for the real-time vehicle routing in fully flexible shared-taxi systems and a simulation study is conducted to investigate how such a shared-taxi system can improve passenger travel compared to conventional taxi services by utilizing vehicle resources more efficiently. Real-time shared-taxi operation with associated algorithms is studied with realistic scenarios, to evaluate the system performance and the efficiency of solving the vehicle routing problem. The remainder of the paper is organized as follows. In the next section, the real-time shared-taxi is specified as a constrained problem of pickup and delivery for dynamic ride-sharing and three different algorithms are provided. Next, a simulation environment is introduced with two different taxi operation schemes. Finally, the simulation results are discussed with a sensitivity analysis.

2. REAL-TIME SHARED-TAXI DISPATCH PROBLEM

In many countries, taxis are operated by an online dispatch center with the help of communication technologies and geo-location services by utilizing GPS (Global Positioning System) and digital maps. Since providing a quality passenger service requires fast and efficient vehicle dispatch algorithms, it is assumed that online taxi dispatch systems are operated with the help of computer algorithms, advanced communication, and Automatic Vehicle Location (AVL) systems. Cervero (7) differentiates such services from conventional carpooling services in several ways: (1) Vehicles are operated by taxi drivers; (2) Vehicle pickup schedules are assigned dynamically to minimize passenger waiting time and in-vehicle travel time; (3) Vehicle operations are scheduled and controlled by the central dispatch system. The use of shared-ride concept in taxi service allows passengers to satisfy the riding public's preference as well as to save costs.

In real-time taxi dispatch system, when a new request is identified by the system operator, the service request is delivered to the system queue where each customer is labeled with time windows and locations of trip origin and destination. Meanwhile, the dispatch algorithm takes a service request from the queue and finds a best available taxi for the travel request within the time windows. If there's no available taxi to meet the constraints, the dispatch system can reject the request. Once a vehicle is assigned the updated schedule, the vehicle uses the shortest (fastest) path to the pickup and drop-off locations based on real-time traffic information provided by the dispatch system.

In a shared-taxi system, the dispatch algorithm needs to find not only the best available vehicle among candidates, but also an optimal route with the newly updated schedules that avoids the violation of vehicle capacity constraints and the time window constraints of previously assigned passengers as well as the new passenger.



(a) **(b)**

FIGURE 1 Real-time Shared-taxi Dispatch Scenarios: (a) Real-time Routing and (b) Shared-ride.

Seow et al. (12) described a good example of a dynamic taxi-dispatch scenario. Since multiple customers need to be handled with multiple taxicabs in a real-world scenario, it is challenging to group real-time customers to the available taxicabs. Figure 1(a) provides an illustrative example of sequentially dispatching a nearby taxi to service customers when passenger requests P_1 and P_2 arrive at two consecutive times T_1 and T_2, respectively. Based on the commonly adopted First-Come First-Served (FCFS) scheme, a dispatch system assigns a nearest vehicle B to request P_1 first since the P_1 came first. This leaves the vehicle A to be assigned to the remaining request P_2. However, if an algorithm can update P_1 to vehicle A in real-time, it is clear that average waiting times for both P_1 and P_2 can be minimized globally by exchanging or re-assigning the existing schedules, which could also be a potential benefit in case of a sudden vehicle breakdown and unexpected traffic congestion.

A shared-ride example is illustrated in Figure 1(b). When a new request P_3 arrives, the algorithm assigns P_3 to vehicle A by minimizing the travel times of not only P_3, but also all previously assigned passengers, P_1 and P_2. It means that the algorithm should be able to assign P_3 to the best available vehicle in terms of objective function and model constraints. It is clear that properly inserting new passengers into any vehicle's existing schedule is critical because previously assigned passengers can have longer waiting or travel times when shared-ride with new passengers is attempted.

Many studies have investigated how shared-taxi can improve system performance and level of service. Tao (10) proposed a dynamic rideshare matching algorithm for taxi service in Taiwan. The author showed the possibility of dynamic taxi-sharing service via a trial field operation. However, the study focused on ride-matching rather than solving vehicle routing problems. Meng et al. (13) proposed a

Genetic Network Programming algorithm for a multiple-customer strategy for a taxi dispatch system and showed that their Multi-Customer Taxi Dispatch System (MCTDS) can enhance the quality of the taxi service within a grid-type artificial network including 25 intersections. Lee et al. (14) introduced a two-step taxi-pooling dispatch system and provided a sensitivity analysis of the system performance. That study tackled the taxi-pooling problem for a feeder system that transport passengers to a metropolitan rapid transit (MRT) station, which implied that the passenger destinations were limited to one point (a many-to-one problem).

3. MODELING SHARED-TAXI DISPATCH ALGORITHMS

This chapter defines the objective functions and problem constraints. Then, three different algorithms for shared-taxi are introduced and compared: (a) a Nearest Vehicle Dispatch (NVD) algorithm that is most commonly used in real applications; (b) an Insertion heuristic (IS) that handles real-time passenger requests in a fast and simple manner; and (c) a Hybrid Simulated Annealing (HSA) that assign passengers efficiently and dynamically to available vehicles.

Model Constraints and Objectives

Compared to conventional DRT systems that usually have been focusing either pickup or dropoff as passenger time-window constraints, passengers' concerns in shared-taxi will be how long they wait for a service and how long detour they have by allowing their rides with other passengers because taxi trips are characterized as an instantaneous short trip in urban areas. Consequently, three types of constraints are introduced: (1) vehicle capacity; (2) maximum passenger wait time; and (3) maximum detour factor. Differently from a many-to-one problem, a vehicle needs to pick up and drop off passengers continually without service cycle. Thus checking the number of available seats among the vehicles' schedules is essential when inserting a new schedule. It is noted that in practical dynamic vehicle routing, trip requests can be rejected by service providers due to the limited number of vehicles, especially when the passengers have time windows or maximum detour constraints. In this paper, passengers are considered not to wait longer than a certain period. The time window for passenger waiting time (e.g., 15 min) is capable of strictly preventing the indefinite deferment of unassigned passengers. The final constraint is on a maximum detour factor guaranteeing an upper bound on the passengers' in-vehicle traveling time between their origins and destinations. This constraint prevents excessive detours caused by too many passengers being assigned on a vehicle trip. The maximum detour factor thus has an important impact in determining the level of service.

Two types of objectives are considered: (1) Minimizing passenger waiting times and detours caused by ride sharing; (2) Maximizing system profit from accepting passengers selectively based on the current schedule. Since each algorithm could have different numbers of delivered passengers during simulation with the objective function (1), scoring is proposed based on the number of delivered and rejected requests, average waiting time, and average travel time. When scores are found in this way, a lower score (cost) indicates better performance.

$$Cost = T_p \cdot P^r + \sum_{i \in I} TT(P_i^c) + \sum_{i \in I} WT(P_i^c) \tag{1}$$

T_p: Penalty value for a dropped request, 7200 sec
P^r: A set of rejected requests during the simulation
P^c: A set of completed requests during the simulation
$TT(P_i^c)$: Travel time of passenger $P_i^c \in P^c$
$WT(P_i^c)$: Waiting time of passenger $P_i^c \in P^c$

The system profit is proposed based on the profit found from vehicle operating cost (which in turn is based on vehicle distance traveled) and service revenue (which is based on the number of delivered passengers). In common taxi fare collection schemes, the fare starts at a basic flat fare with additional charges applying according to distance traveled and time waited. As the study context is an urban area in South Korea, the relevant fare structure in this study is as per the following three components found in general for taxi fare in South Korea.

- Basic fee: This basic fare covers the first two kilometers.
- Per mile (or kilometer) charge: An additional charge is applied every 144 meters.
- Waiting charge: If the taxi speed drops below 15 km/hour, an additional charge is added every 35 seconds.

In this study, we assume that the service revenue consists of two parts: fixed revenue and distance based revenue. The operating cost can be obtained using the vehicle distance traveled, as follows. In this case, a higher value indicates better performance.

$$Profit = \alpha \sum_{i \in I} P_i^c + \beta \sum_{i \in I} D^P(P_i^c) - \gamma \sum_{j \in J} \mathrm{D}^v(V_j) \quad (2)$$

α: Fixed revenue (basic fare, 2000)
β: Weight of distance based revenue, 1.0
γ: Weight of vehicle operating cost, 0.4
$D^P(P_i^c)$: Passenger door-to-door distance excluding the basic fare distance
$\mathrm{D}^v(V_j)$: Vehicle distance traveled (km)

Nearest Vehicle Dispatch

Nearest Vehicle Dispatch (NVD) is the most widely employed strategy in current on-line taxi dispatch systems with single customer group. NVD has the following two steps. In step 1, when a new passenger request arrives, the algorithm seeks a geographically nearest available vehicle from the passenger's origin location so as to provide quick and efficient response times. Checking feasible time windows is carried out at step 1. Once a nearest vehicle is selected, an optimal schedule is found at step 2 by assuming that the vehicle's pickup and delivery schedule can be independently optimized (similar to the driver or an in-vehicle computer doing that) based on the current location and the existing schedule. Since this greedy algorithm only considers reaching the passenger with the shortest distance possible, it doesn't need a complicated dispatch algorithm. However, passengers could necessarily detour because the algorithm doesn't consider existing schedules, the time spent by passengers on board, or the trip origins and destinations of those passengers.

Insertion Heuristic

While NVD searches only for a nearest feasible vehicle to assign a new passenger, Insertion heuristic (IS) compares all feasible vehicles to find a best available vehicle for its objective. Although this study focuses on a many-to-many vehicle routing problem, the proposed Insertion heuristic is based on a First-Come First-Served (FCFS) policy in which a new request is considered individually and independently from other new requests. The proposed IS has four steps as follows if the objective is to minimize waiting times and detours: (1) Each passenger trip is identified by its origin and destination; (2) Collect available vehicles to insert a new trip request in the corresponding service area; (3) Select a vehicle by minimizing service waiting time and travel time of the new passengers as well as the existing passengers; (4) Update the vehicle schedule with a new request. The detailed procedure is following:

1. A new passenger request, z_i ($i \in I$) comes in, and pickup and delivery locations and the number of passengers in the group are identified.
2. Once the system searches for a vehicle j among all available vehicles J, ($j \in J$), it confirms whether the constraints meet or not with new pickup and dropoff events, $e^{i,pick}$ and $e^{i,drop}$, associated with z_i. If they are acceptable, the incremental cost IC_j ($e^{i,pick}$, $e^{i,drop}$) based on its

current schedule K, ($k \in K$) is calculated to determine the optimal vehicle. In the same procedure, it searches for the best insertion positions for the new events among the current schedules E_j by calculating the expected waiting and travel time of the new passenger as well as previously assigned passengers. The best vehicle is updated with the total cost C_j found and corresponding insertion positions for $e^{i,pick}$ and $e^{i,drop}$.

3. If there are no more available vehicles to consider, the dispatch algorithm assigns the passenger to the vehicle with the minimum incremental cost, IC_j. Otherwise, the passenger request is rejected due to the constraints.
4. Once the best vehicle is determined, the pickup and delivery schedules ($e^{i,pick}$ and $e^{i,drop}$) are inserted into the optimal position among the existing schedules of the vehicle.
5. Dispatch the vehicle following the schedule to serve the new passenger.

$$C_j(E_j) = \sum_{k\in K} \left[WT(e_k^{pick}) + TT(e_k^{drop})\right] \quad (3)$$

$$C_j(E_j, e_j^i) = \min \sum_{k\in K+2} \left[WT\left(e_k^{pick}\right) + WT\left(e_m^{i,pick}\right) + TT\left(e_k^{drop}\right) + TT\left(e_n^{i,drop}\right)\right] \quad (4)$$

$$IC_j(E_j, e_j^i) = C_j(E_j, e_j^i) - C_j(E_j) \quad (5)$$

$WT(e_k^{pick})$: Waiting time of the previously assigned passenger with e_k^{pick}
$WT(e_m^{i,pick})$: Waiting time of the new passenger z_i with the m-th pickup order ($0 < m$)
$TT(e_k^{drop})$: Travel time of the previously assigned passenger with e_k
$TT(e_n^{i,drop})$: Travel time of the new passenger z_i with the n-th dropoff order ($m < n$)

The C_j in (3) can be calculated based on the vehicle's current schedule E_j. Waiting time for passengers can be obtained with pickup events whereas travel time can be calculated with delivery events. IC_j in (5) includes two terms respectively: (a) The total cost calculated for the updated schedule including $e^{i,pick}$ and $e^{i,drop}$; (b) The total cost calculated for the current schedule of vehicle j. Note that since adding a new schedule causes extra costs for a vehicle, the incremental cost should be always positive. The formulation (4) returns not only the total cost, but also the optimal insertion positions for the new events. The cost function can be replaced by the system profit in a similar manner, if needed.

The proposed insertion heuristic is fairly easy and straight-forward to implement, and shows computational efficiency, but it has limitations on dynamic pickup and delivery operations. The primary limitation is that it has no dynamic schemes capable of re-optimizing vehicle schedules by shifting or exchanging previously assigned passenger pickup and delivery schedules. Thus it should be expected that it would normally achieve a sub-optimal solution. The problem of finding an optimal solution is however not easy as the well-known combinatorial issues ensue. This leads us to developing an optimization scheme that while still is heuristic, can reach near optimal solutions, as described next.

Hybrid Simulated Annealing

Simulated Annealing (SA), first suggested by Metropolis et al. (15) and defined by Kirkpatrick et al. (16) is a generic probabilistic meta-heuristic, which is capable of finding an approximately accurate solution for the global optimum of a complex system with a large search space. The name of Simulated Annealing (SA) involves a technique such as heating and cooling of a material in annealing in metallurgy. It is widely known that the heat treatment in metallurgy enables the property of material to change in its hardness and strength. SA is a stochastic relaxation method based on an iterative procedure starting at an initial "higher temperature" with the system in a known configuration, the word "temperature" being used to give an intuitive connection to metallurgy. The iterative procedure of SA improves the cost function until the current temperature cools down. At higher temperatures, the atoms are likely to become unstable from the initial position, which means that the algorithm is allowed to have flexibility in searching the feasible space, while at lower temperatures it has more chances to find improvement with local search than the initial state (16). In comparison with NVD and IS, SA provides a systematic re-optimization scheme to assign new requests as well as update existing vehicle schedules in real-time.

Figure 2(a) shows the general procedure for SA. The algorithm starts from an initial state S_0, which represents an initial solution x at a predefined higher temperature T_0. The procedure to search the global minima consists of comparing energy levels for two consecutive random states, S_t and S_{t+1}. The energy level E indicates the objective function value for a given state vector x. In the shared-taxi problem, S_t is a state vector that contains all vehicles' schedules including passengers not assigned to vehicles due to tight time windows. The objective function consists of two parts, the cost associated with vehicle schedule and the penalty cost for rejected passengers in (6). At each iteration, the current state S_t is replaced by a randomly generated candidate S_{t+1} with a probability that depends both on the difference between the corresponding objective values and on a control parameter, which is gradually decreased over the cooling process. If the new state has a lower energy level than the previous state, $E(S_t) < E(S_{t+1})$, the algorithm proceeds with the current state. Otherwise, the move is determined to be accepted with a probability based on a Boltzmann's function allowing the search to escape a local minimum, $P(E_t,T)=e^{\Delta E/kT}$ where the temperature affects. The following shows the energy levels for both minimization and maximization of shared-taxi problems. Note that a higher energy level would be desirable to maximize system profits as opposed to minimizing passenger travel costs.

Energy level (E_t) for minimizing passenger travel costs:

$$E_t = E^{Cost}(S_t) = \sum_{j\in J} C_v(v_j) + \sum_{i\in I} C_r(z_i^r) \quad (6)$$
$$C_v(v_j) = \left[\sum_{i\in I} TT(z_i) + \sum_{i\in I} WT(z_i)\right] \cdot P(z_i) \quad (7)$$
$$C_r(z_i^r) = T_p \cdot P(z_i^r) \quad (8)$$

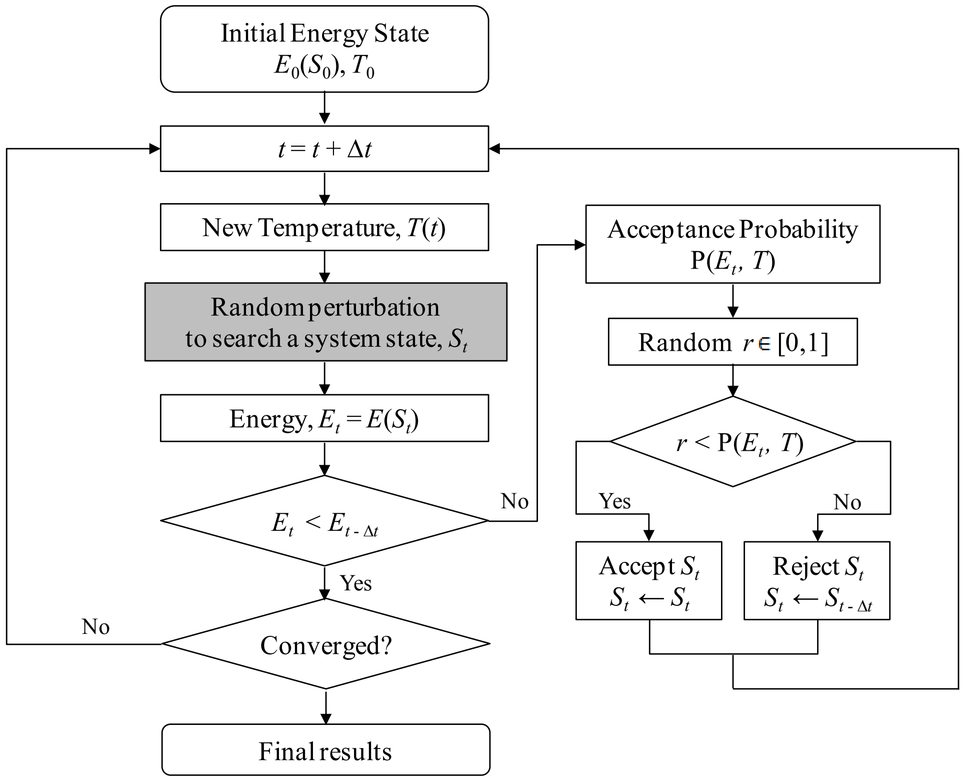
$C_v(v_j)$: Passenger cost associated with vehicle v_j
$C_r(z_i^r)$: Penalty value associated with not assigned passenger group z_i^r
$P(z_i^r)$: Number of passengers in passenger group z_i^r
T_p: Penalty value for a dropped request, 7200 sec

Energy level (E_t) for maximizing system profits:

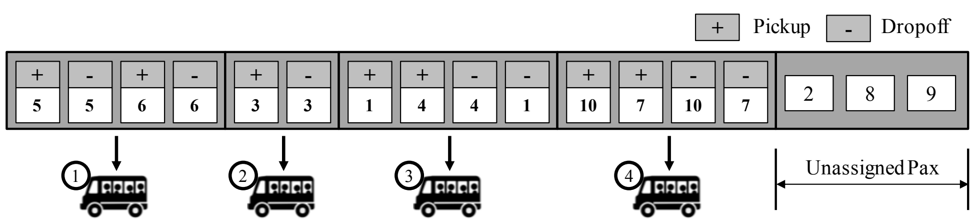
$$E_t = E^{Profit}(S_t) = \sum_{j\in J} G_v(v_j) \quad (9)$$
$$G_v(v_j) = \alpha \sum_{i\in I} z_i + \beta \sum_{i\in I} D^P(z_i) - \gamma \mathrm{D}^v(v_j) \quad (10)$$

$G_v(v_j)$: Profit associated with vehicle v_j
α, β, γ : Parameters for fixed revenue and weights for base revenue and operating cost in (2)
$D^P(z_i)$: Door-to-door distance excluding the basic fare distance of passenger group z_i in vehicle v_j
$\mathrm{D}^v(v_j)$: Expected vehicle distance traveled (km) associated with the current schedule of vehicle v_j

A standard SA procedure is adopted to generate a random neighbor including swap and move. Basically a swap is used for the main operator by exchanging vehicle schedules, ΔS, which can be a critical part of the random perturbation to generate a neighborhood solution. In this case, the energy $E(S_t)$ indicates the cost associated with vehicle routing. The swap procedure is started from randomly selecting two vehicles, v_i and v_j, and then swap two randomly selected customers z_i and z_j from each vehicle. Unassigned requests are characterized by a salvage slot with higher penalty values at the end of the state vector in Figure 2(b). Unassigned ones will be rejected if the system doesn't allow them in the system queue.



(a)



(b)

FIGURE 2 (a) Simulated Annealing Procedure and (b) System State Vector.

When inserting a new customer to the vehicle's schedule, the existing vehicle schedules need to be updated. A heuristic insertion algorithm is adopted to keep the individual vehicle's schedule optimized, called Hybrid Simulated Annealing (HSA). One reason is that the validity of a newly generated neighbor should be checked while generating a neighborhood solution because the swap and move operations can generate infeasible solutions with the aforementioned random search. Unexpected infeasible solutions would however cause a significant impact on system efficiency and solution accuracy. For this reason, many types of SA applications combine a general SA procedure with another heuristic technique that enables the search moves to be within the feasible space. Searching a random candidate, the grayed part in Figure 2(a), can be replaced by the IS algorithm proposed in the previous section due to the characteristics of the shared-taxi problem with constraints such as vehicle capacity and time windows.

SA starts with parameters, iteration I_{iter}, initial temperature T_0, final temperature T_c, cooling rate R_c, and Boltzmann constant K. I_{iter} denotes the number of iterations at a particular temperature. Setting the number of iteration is critical for a multiple vehicle routing problem because a higher number of iteration provides a higher opportunity to move around the search space expanded as the number of vehicles increases. An alternative is to change the number of iterations dynamically, depending on a temperature. For example, a large number of iterations are required to thoroughly explore the local optimum. On the other hand, the number of iterations is not necessarily large at higher temperatures. Setting the temperature plays a critical role in acceptance probability, which means that the value of initial temperature depends on the scale of the cost of a problem-specific objective function. The initial temperature should not be high enough that the algorithm simply conducts a random search, causing excessive computation time. A preliminary search can be used to estimate the initial temperature by calculating an average objective increase, $\delta\Delta f$ in formula (11), where p_0 is a desired average increase of acceptance probability. According to Crama and Schyns (17), usually $p \in$[0.8, 0.9], and R_c and K are normally set with $r \in$ [0.80, 0.99] and $k \in$ [0.1, 1.0], respectively.

$$T_0 = -\delta\Delta f / \ln{(p_0)} \tag{11}$$

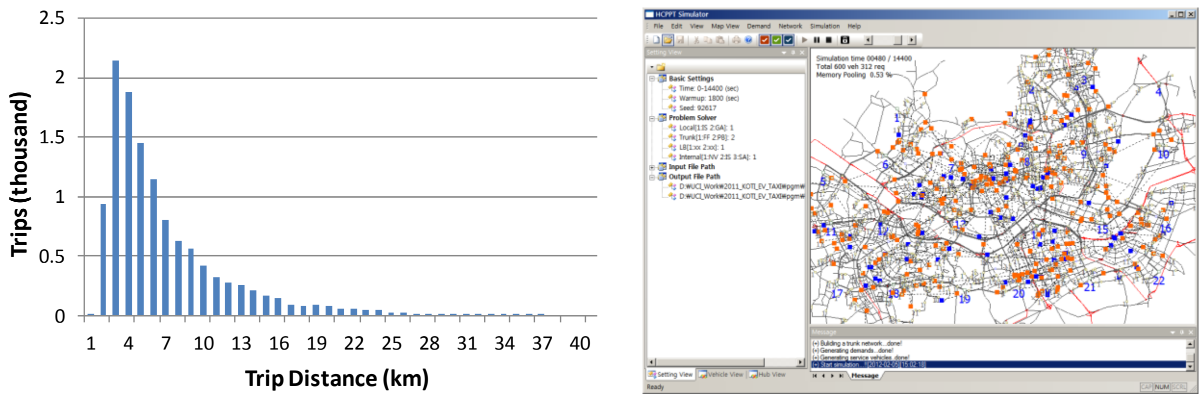
Note that the selection of parameters including the number of iterations and temperature might not only affect the quality of the solution, but also has significant influence on the algorithm's run-time. It is known that a good initial solution improves the quality of solution as well as the convergence time. To generate a good feasible solution for SA in real-time scheduling, other meta-heuristic techniques can be used with combination of parallel computing techniques (18).

4. SIMULATION IMPLEMENTATION

Simulation Design

A shared-taxi simulator is written in Microsoft Visual C++, which is capable of implementing various types of algorithms and visualizing all simulation elements (e.g., tracking vehicles and passengers) as in Figure 3(b). The simulator imports digital maps designed for map display, geo-coding, and includes faster vehicle routing with realistic roadway attributes such as road categories, turning prohibition, one-way, posted speed, numbers of lanes, link lengths, and link shapes. For a simulation in urban area, Seoul area is abstracted from the national transportation network, and it contains 8,382 links and 6,321 nodes. The link shapes used here are as per the transportation network in the KOTI (Korea Transport Institute) regional transportation planning model.

For taxi demand generation, the demand data used are as in the KOTI regional transportation planning model. As of 2011, the trip demand consists of auto, bus, subway, rail, taxi, and other types of demands, which covers Seoul with a total of 560 zones. Under the usual assumption of spatial uniformity of demand around a zone centroid, point-to-point dynamic taxi demands are randomly generated in accordance with destination probabilities from the taxi demand table of each centroid. The real-time service requests arrive according to a Poisson process in a temporal manner. Figure 3(a) shows the trip length distribution of 12,000 trip requests generated based on the taxi demand table with the minimum trip length, 1.5 km for the taxi service. It shows that majority of trip demands are within 10 km. The average trip length is 6.3 km and the expected door-to-door travel time 13.3 min under the assumption that vehicles can travel at 60-90% of the posted speeds on the network.



(a) (b)

FIGURE 3 (a) Trip distribution of requests and (b) Shared-taxi Simulator with Seoul Network.

Simulation Scenarios

According to the Seoul Taxi Association, a total of 72,000 taxi licenses are registered including owner-driver taxis, and a total fleet of 40,000 vehicles are actually operated as of 2011. A total of 600 service vehicles are considered in the simulation, which is equivalent to 1.5% of the total number of vehicles operated in Seoul. The initial vehicle positions are randomly generated over the simulation area. We consider 4-seater taxicabs for shared rides. As for demand levels, four different demand levels (9,000, 12,000, 15,000, and 18,000 requests equivalent to 3.7, 5.0, 6.3, and 7.5 requests/km^2-hr) are considered based on the request pool generated in Table 1. The total simulation time is set to 4 hours including 30 min as a warm-up period. An average of 1-min boarding and alighting times are assumed for each passenger, with a normal distribution $N(1.0, 1.0)$. Three different algorithms, namely, NVD, IS, and HSA are tested. For HSA, 60 sec is considered as the period for successive re-optimization. We set the maximum 6,000 iterations at each annealing temperature and the lowest temperature 0.2.

Two types of customer policies are considered. First, Single Customer Operation (SCO) is the traditional taxi service without ride sharing. Each passenger trip request is matched to the best available empty vehicle according to the objective function. Multiple Customer Operation (MCO) allows taxicabs to pick up multiple customers whose origins and destinations could be different, but minimizing their waiting time and travel time or maximizing profits as long as the vehicle has vacancy.

TABLE 1 Simulation Scenarios

Shared-taxi simulation	
Shared Taxi settings	
Service area (km^2)	605
Simulation time (hours)	4
Warm up (hours)	0.5
Number of service vehicles	600
Vehicle capacity (passengers/vehicle)	1 and 4
Maximum waiting time	15 min
Maximum detour factor	2.0
Operation types[1]	SCO, MCO
Demand and Algorithm settings	
Demand levels (thousand requests/4-hour)	9, 12, 15, and 18
Routing algorithms[2]	NVD, IS, and HSA

[1]: SCO: Single Customer Operation, MCO: Multiple Customer Operation

[2]: NVD: Nearest Vehicle Dispatch, IS: Insertion heuristic, HSA: Hybrid Simulated Annealing

5. SIMULATION RESULTS

Two performance measures are introduced in order to compare the system efficiency and performance, Level-of-Service (LOS) index (ϕ) and Ride-time index (ρ), which are discussed by Black (19) is adopted to compare the system efficiency in shared-ride transportation systems. Note that these indices may be a little misleading, as lower values indicate better. The door-to-door ride time in (12, 13) is the travel time when no other passengers are picked up or dropped enroute (i.e., equivalent to the time for driving a personal auto) given the same network conditions.

$$\phi = \frac{\textit{Avg. passenger waiting time at pickup location}}{\textit{Avg. door-to-door ride time}} \tag{12}$$

$$\rho = \frac{\textit{Avg. vehicle ride time}}{\textit{Avg. door-to-door ride time}} \tag{13}$$

Minimizing Wait and Travel Times

Figure 4 shows performance measures of 4-seater taxi with passenger travel cost minimization. Regarding passenger delivery, HSA apparently performs better than other two algorithms. However, when the passenger demand stays lower levels (9,000 and 12,000 requests), there's no significant difference between IS and HSA in terms of passenger delivery and reject in Figure 4(a) and 4(b). As expected, NVD performs worst due to the lack of optimality whereas both IS and HSA consider passengers' waiting and traveling time. NVD assigns greedily passengers to the nearest vehicles to minimize passenger waiting time without any consideration of passengers' origins and destinations, so it is clearly seen that the LOS index of NVD shows lowest among other algorithms although the ride-time index of NVD shows highest even with lower demand levels in Figure 4(c) and 4(d).

In table 2, average waiting times remain from six to twelve minutes, which are far below the maximum waiting time constraint, 900 sec. That is explained by another bound constraint with maximum detour factor, 2.0. It is also important to note that ride-time index can be slightly over 2.0 because passengers' boarding and alighting times are assumed randomly during the simulation. For instance, excessive longer boarding or alighting times of one passenger will affect the travel time of other passengers on board in shared-ride operation. At higher levels of passenger requests, the ride-time indices of IS and HSA are almost 2.0. The average vehicle distance traveled decreases as the demand increases in IS and HSA, which is very consistent with increased vehicle loads.

Figure 4(e) and 4(f) show the objective function cost comparisons of the simulation results. The normalized costs by the total number of requests are also given. Apparently, HSA outperforms than both NVD and IS. Although there's no difference between IS and HSA at the lowest demand level, IS shows higher costs similar to NVD as the demand level increases. Since IS simply finds the best vehicle to insert a new passenger at a time - not considering the all passengers at the same time - the deterioration of system performance is unavoidable compared with HSA that periodically optimizes the entire vehicle schedules, which can't be achieved by a simple heuristic solution. It is also noted that the penalty value (7200 sec) for rejected requests in HSA can impact significantly on both the number of delivered passengers and the quality of service simultaneously.

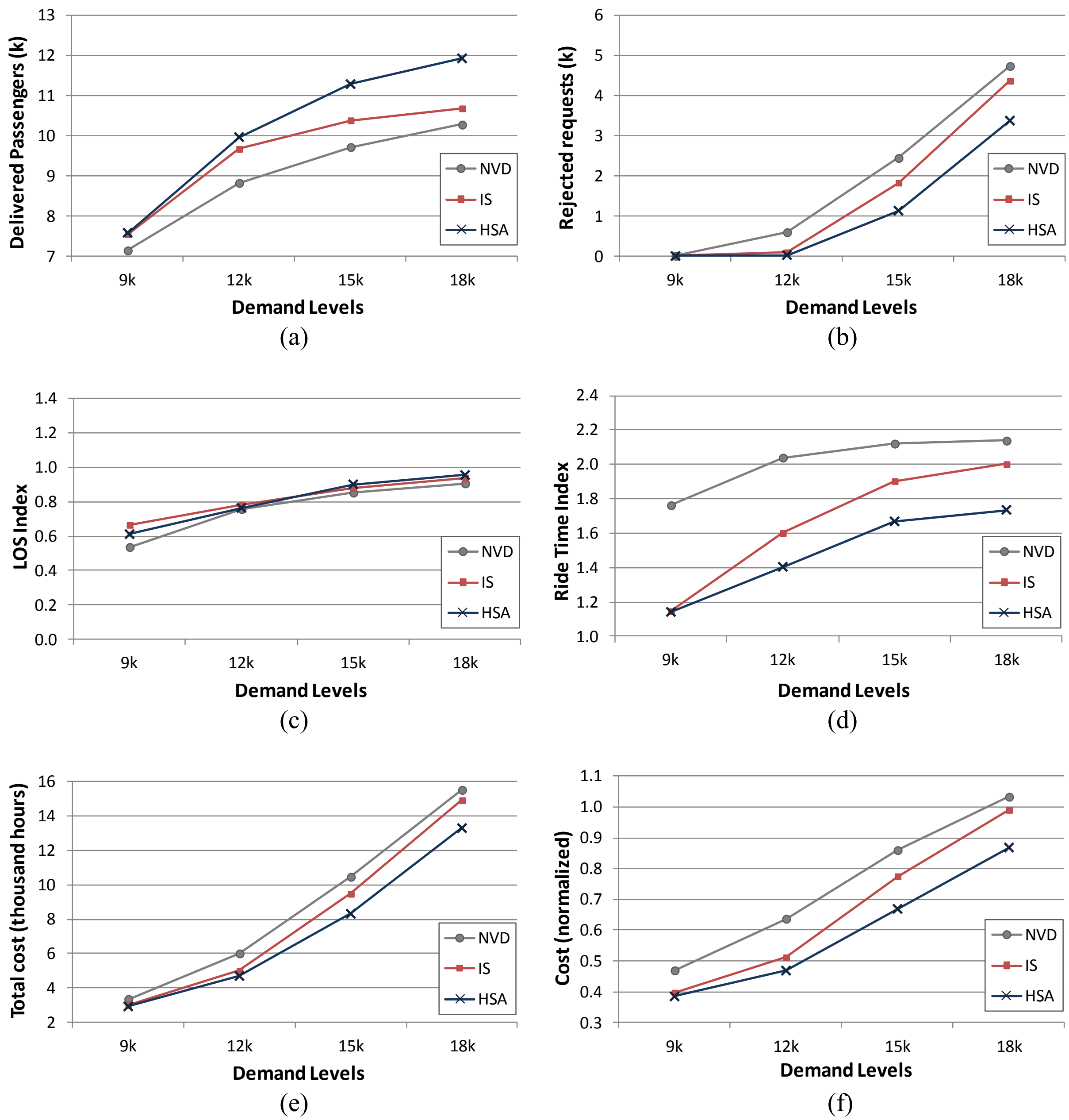


FIGURE 4 Performance measures (Minimizing Cost with MCP): (a) Number of Delivered Passengers, (b) Number of Rejected Requests, (c) LOS Index, (d) Ride time Index, (e) Total Cost, and (f) Normalized Cost.

Maximizing Profit

When applying the profit maximization for algorithms, HSA algorithm tends to accepts passengers' requests selectively to maximize its objective. Figure 5 shows the system performance. Different from the previous results, the numbers of delivered and rejected passengers in Figure 5(a) and 5(b) are very similar over three algorithms. In Figure 5(e), both IS and HSA shows the similar profits at the lowest demand compared to the number of service vehicles while the profit of NVD is far below of IS and HSA. As passenger demand increases, the profit with HSA keeps increasing gradually while the profit with IS stays constant after the demand level, 15,000. This can be explained on the basis of the optimization concept of HSA. The HSA is capable of re-optimizing vehicle schedules by shifting or exchange previously assigned passengers' schedules. The same patterns are observed in the cost minimization in the previous section, but applying the penalty (7200 sec) in HSA for real-time passenger delivery system where rejecting passengers is necessary, might cause additional impacts on system performance unless the penalty values are carefully evaluated, as mentioned earlier, while NVD and IS don't reject any of requests as long as the requests meet the constraints. When comparing passenger door-to-door distances, it is shown that HSA algorithm clearly tends to accept passengers who have longer travel distance in the profit maximization (average door-to-door trip length of delivered passengers, 6.4 km/request) than in the cost minimization (5.9 km). This is because vehicles would have a higher chance to fully utilize its available seats with longer passenger trips rather than with shorter trips.

In table 2, the average vehicle distance travel with IS and HSA (60 - 85 km) with the profit maximization strategy are significantly lower than the values (83 - 95 km) obtained with the cost minimization in the same table. It is reasonable that the dispatch algorithm tries to maximize system profit (equivalent to minimize operating costs), directly linked to vehicle distance traveled. It is noted that the cost minimization strategy not only minimizes the passenger waiting time and travel time, but also tries to maximize the passenger delivery as long as the incremental cost is less than the predefined penalty, while the focus of the profit maximization rejects requests is to achieve a higher system profit without any consideration of passenger delivery.

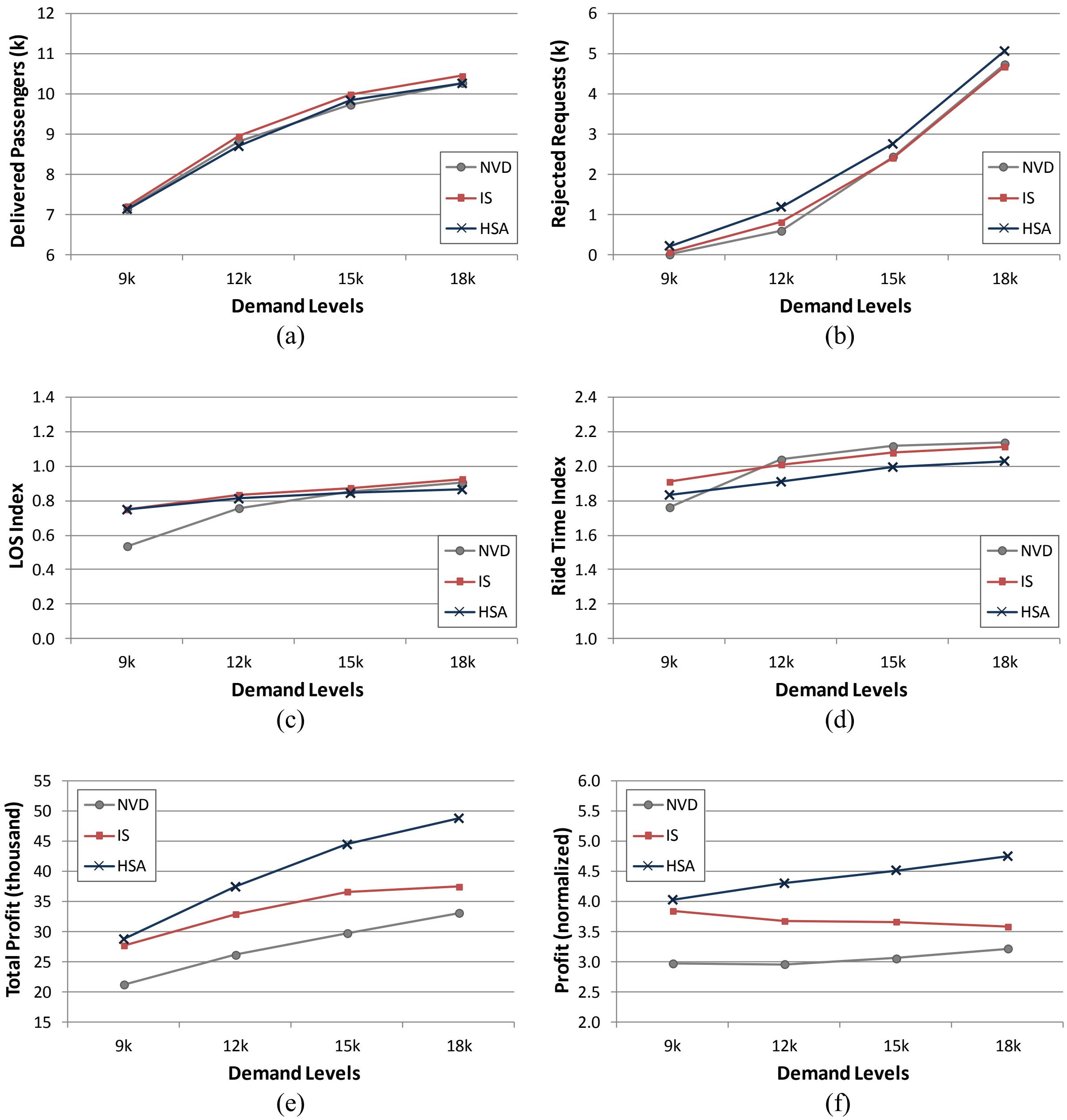


FIGURE 5 Performance measures (Maximizing profit with MCO): (a) Number of Delivered Passengers, (b) Number of Rejected Requests, (c) LOS Index, (d) Ride time Index, (e) Total Profit, and (f) Normalized Profit.

TABLE 2 Detailed performance measures with MCO

		Minimize Cost		Maximize Profit	
Vehicle Routing Schemes	NVD	IS	HSA	IS	HSA
9,000-request					
Total delivered passengers (requests)	7,146	7,551	7,586	7,211	7,140
Average wait time home (min)	6.55	8.74	8.04	9.44	9.69
Average passenger travel time (min)	21.49	15.04	15.01	24.03	23.75
Level-of-Service index	0.54	0.67	0.61	0.75	0.75
Ride-time index	1.76	1.15	1.14	1.91	1.83
Rejected passengers (requests)	12	6	4	66	225
Average vehicle load (passengers/veh)	1.61	0.94	0.93	1.57	1.47
Average vehicle dist. traveled (km)	78.09	92.81	88.31	60.92	62.41
12,000-request					
Total delivered passengers (requests)	8,830	9,679	9,972	8,952	8,704
Average wait time home (min)	8.85	9.79	9.83	10.36	10.84
Average passenger travel time (min)	23.82	20.02	18.06	24.93	25.50
Level-of-Service index	0.76	0.78	0.76	0.84	0.81
Ride-time index	2.04	1.60	1.40	2.01	1.91
Rejected passengers (requests)	601	100	27	814	1,195
Average vehicle load (passengers/veh)	2.44	1.82	1.54	2.15	1.94
Average vehicle dist. traveled (km)	84.27	88.07	87.65	76.41	74.74
15,000-request					
Total delivered passengers (requests)	9,724	10,392	11,292	9,996	9,842
Average wait time home (min)	9.86	10.61	11.27	10.74	11.47
Average passenger travel time (min)	24.50	22.92	20.85	25.50	27.12
Level-of-Service index	0.85	0.88	0.90	0.88	0.84
Ride-time index	2.12	1.90	1.67	2.08	2.00
Rejected passengers (requests)	2,451	1,837	1,136	2,418	2,762
Average vehicle load (passengers/veh)	2.78	2.53	2.14	2.56	2.35
Average vehicle dist. traveled (km)	85.95	85.99	83.88	83.05	82.19
18,000-request					
Total delivered passengers (requests)	10,284	10,689	11,934	10,453	10,270
Average wait time home (min)	10.50	11.06	11.66	11.14	11.96
Average passenger travel time (min)	24.74	23.57	21.19	25.43	28.08
Level-of-Service index	0.91	0.94	0.96	0.93	0.87
Ride-time index	2.14	2.00	1.74	2.12	2.03
Rejected passengers (requests)	4,745	4,377	3,382	4,686	5,075
Average vehicle load (passengers/veh)	2.92	2.77	2.35	2.80	2.57
Average vehicle dist. traveled (km)	85.85	85.19	83.12	84.58	84.09

Benefits of Shared-taxi

Another simulation is performed to investigate the benefit of shared-taxi with the comparison between SCO and MCO in Figure 6. SCO allows only one customer group in a taxi at any time during the service period. There could be two different strategies for the scheduling of single customer vehicles. The first strategy is that the new customers can be assigned only to idling vehicles with no current schedules while the second is that new customers can be assigned to any vehicles satisfying the associate time constraints.

For example, in the second strategy a new schedule could be updated to a vehicle even before the vehicle has finished the current schedule of the passenger on board. From preliminary simulation runs, it was found that the first strategy delivers 25% less number of passengers, but it shows smaller waiting times at home compared to the second strategy. In this simulation, the second strategy is employed due to its similarity in the assumption used in the shared-taxi operation. The cost minimization scheme is applied because the study focuses on the number of delivered passengers and their travel times.

SCO and MCO are compared using HSA in Figure 6. Significantly notable increases can be seen in the number of delivered passengers with MCO in Figure 6(a). At the highest demand level, MCO shows about 4,000 more delivered passengers than SCO. The number of rejected passengers in MCO stays at the lower level at the lower demand levels, and then starts increasing, indicating that the shared-ride system rarely drops passengers given the fleet size and passenger demands. It is very interesting that LOS index with MCO is lower than with SCO in Figure 6(c). It can be explained that MCO has a higher probability to pick up passengers. Consequently the passengers might wait for a shorter time than the case of SCO in which a vehicle can go to pick up a passenger only after dropping off the passenger on board. Ride-time index with MCO keeps increasing as the trip requests increase, but stay below 2.0 due to the proposed constraint in Figure 6(d). It is reasonable that the ride-time indices with SCO stay at the same level across all algorithms and scenarios, at a value of 1.0, because the vehicles are not allowed to be in shared-ride operation.

As for a combined index from summing both ride-time and LOS indices, MCO shows a lower value than SCO at the lowest demand level. This denotes that the proposed shared-ride system could not only serve more passengers, but also provides better quality of service (QoS) by allowing ride-sharing at certain demand levels. Most importantly, MCO has more than tripled its average vehicle load than SCO indicating higher vehicle utilization compared to SOC. However, the number of delivered passengers does not increase as much. It implies that the average vehicle load should be carefully considered as a performance indicator because it could include side effects related to passenger trip lengths. It should be noted however that the passengers' convenience and comfort is not the focus in this study. Those are aspects which significantly affect the demand, but are beyond the scope of this study that focuses on the operational efficiency. Regarding vehicle travel distance in Figure 6(f), MCP shows less average travel distance (85.7 km) than SCO (97.1 km), which can also expect less vehicle operating costs than SCO.

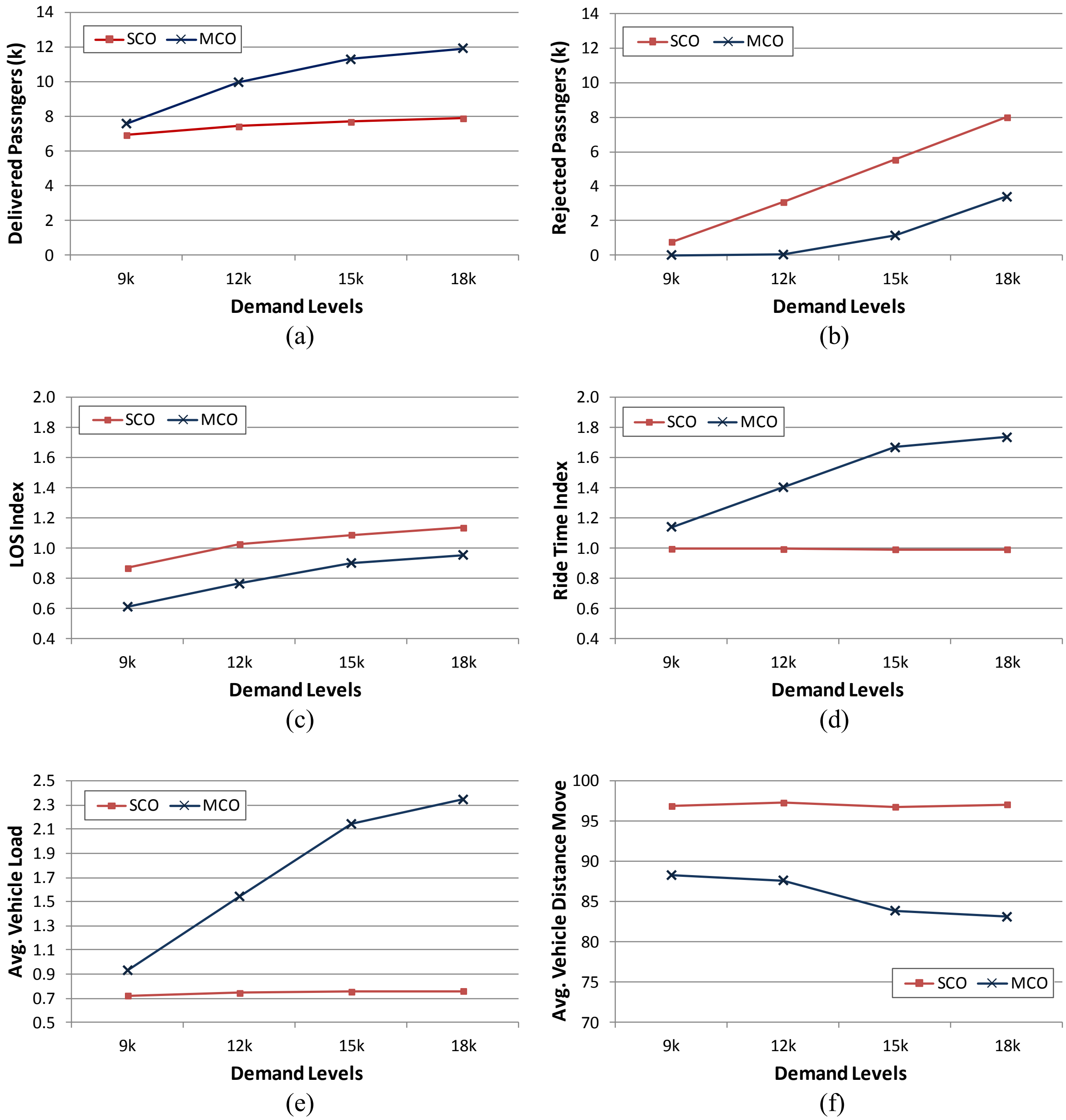


FIGURE 6 Performance Comparison between SCO and MCO: (a) Number of Delivered Passengers, (b) Number of Rejected Requests, (c) LOS Index, (d) Ride time Index, (e) Average Vehicle Load (passengers/vehicle), and (f) Average Vehicle Distance Moved (km) during the simulation.

6. CONCLUSION

Taxi is a convenient and fast and flexible transportation method in urban areas. A dynamic shared-taxi system is proposed with three types of taxi dispatch algorithms. It is assumed that all vehicles can optimize their schedule at the individual vehicle level with given pickup and delivery schedules. It is shown that HSA systematically optimizes the entire schedules of the vehicles in real-time, which is not possible with eight NVD or IS in this study. From the simulation study, it is seen that the proposed shared-taxi system has potential in improving the system performance compared to the conventional form of taxi service. However, increasing travel time of passengers (vehicle ride time) is inevitable, and the demand impacts need to be investigated within the given detour constraint.

In this study, we assumed that all passengers are willing to share their rides with other passengers and the simulation results show great potentials that ride-sharing can reduce the taxi fare by improving the system productivity. However, in real practice, not all passengers want to share their ride. Considering that many passengers are still not allowing ride-sharing for their comfort or convenience, it should be addressed that the proposed shared-taxi model can deal with this issue depending on the passengers' different willingness (e.g., individual preference for maximum waiting time and detour) towards ride-sharing in real-time. Moreover, taxis would have greater capability for it than other transportation means because individual vehicles can dynamically switch their operation between single and multiple customer schemes. The proposed shared-taxi concept will be applied to increase the efficiency of EV (Electric Vehicle) taxi operation, which usually suffers from limitations by short driving range and battery charging. Also, the author want to emphasize that the share-ride system proposed in this study can be easily converted to a real-time shared-van (shuttle) service.

7. ACKNOWLEDGEMENT

This project is funded by the Korea Transport Institute (KOTI), as part of "Electric Vehicle 2011", initiated by the National Research Council for Economics, Humanities and Social Science, South Korea.

REFERENCES

1. Dial, R. B. (1995). Autonomous Dial-a-Ride Transit Introductory Overview. Transportation Research, Vol. 3C, No. 5, pp. 261-275.
2. Cortés, C. E. and Jayakrishnan, R. Design and Operational Concepts of High-Coverage Point-to-Point Transit System. In Transportation Research Record: Journal of the Transportation Research Board, No. 1783, National Research Council, Washington, D.C., 2002, pp. 178-187.
3. Jung, J. and Jayakrishnan, R. High Coverage Point-to-Point Transit: Study of Path-Based Vehicle Routing Through Multiple Hubs. In Transportation Research Record: Journal of the Transportation Research Board, No. 2218, Transportation Research Board of the National Academies, Washington, D.C., 2011, pp.78-87.
4. Zimride. http://www.zimride.com. Accessed July 17, 2012.
5. Avego. http://www.avego.com. Accessed July 17, 2012.
6. SideCar. http://www.side.cr. Accessed July 17, 2012.
7. Cervero, R. Paratransit in America: Redefining Mass Transportation. Praeger Publishers, Westport, Conn., 1997.
8. Uber. http://www.uber.com. Accessed July 17, 2012.
9. Split-it. http://www/split-it.sg. Accessed July 31, 2012
10. Tao, C. C. Dynamic Taxi-sharing Service Using Intelligent Transportation System Technologies. Wireless Communications, Networking and Mobile Computing, 2007, International Conference on. 21-25 Sep, 2007, pp. 3209-3212.
11. Tsukada, N. and Takada, K. Possibilities of the Large-Taxi Dial-a-ride Transit System Utilizing GPS-AVM, Journal of Eastern Asia Society for Transportation Studies, EASTS, Bangkok, Thailand, 2005, pp. 1-10.
12. Seow, K. T., Dang, N. H., and Lee, D. H. A Collaborative Multiagent Taxi-Dispatch System. IEEE Transactions on Automation Science and Engineering, Vol. 7, Issue 3, 2010, pp. 607-616.
13. Meng, Q. B., Mabu, S., Yu, L., and Hirasawa, K. A Novel Taxi Dispatch System Integrating a Multi-Customer Strategy and Genetic Network Programming. Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol. 14, No. 5, 2010, pp. 442-452.
14. Lee, K. T., Lin, D. J., and Wu, P. J. Planning and Design of a Taxipooling Dispatch System. In Transportation Research Record: Journal of the Transportation Research Board, No. 1930, Transportation Research Board of the National Academies, Washington D.C., 2005, pp. 86-95.
15. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. Equations of state calculations by fast computing machines. Journal of Chemical Physics, Vol. 21, No. 6, 1953, pp. 1087-1092.
16. Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by Simulated Annealing. Science, Vol. 220, No. 4598, 1983, pp. 671-680.
17. Crama, Y., and Schyns, M. Simulated annealing for complex portfolio selection problem. European Journal of Operational Research. Vol. 150, Issue 3, 2003, pp. 546-571.
18. Janaki Ram, D., Sreenivas, T. H., and Ganapathy Subramaniam, K. Parallel Simulated Annealing Algorithms. Journal of Parallel and Distributed Computing, Vol. 37, Issue 2, 1996, pp. 207-212.
19. Black, A. Urban Mass Transportation Planning. McGraw-Hill Series in Transportation. McGraw-Hill, New York, 1995.