ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/281445468

# Dynamic Shared-Taxi Dispatch Algorithm with Hybrid Simulated Annealing

**Article** *in* Computer-Aided Civil and Infrastructure Engineering · June 2015

DOI: 10.1111/mice.12157

CITATIONS
137

READS
3,251

**3 authors**, including:

Jae Young Jung
Nike Inc.
**17** PUBLICATIONS **500** CITATIONS
SEE PROFILE

R. Jayakrishnan
University of California, Irvine
**126** PUBLICATIONS **3,639** CITATIONS
SEE PROFILE

**Some of the authors of this publication are also working on these related projects:**




Persistent Traffic Cookies View project

All content following this page was uploaded by Jae Young Jung on 15 March 2018.

The user has requested enhancement of the downloaded file.

# Dynamic Shared-Taxi Dispatch Algorithm with Hybrid Simulated Annealing

Jaeyoung Jung[1*], R. Jayakrishnan[2], and Ji Young Park[3]

[1]*Research & Advanced Engineering, Ford Motor Company*
[2]*Institute of Transportation Studies, Department of Civil and Environmental Engineering, University of California, Irvine, California, United States*
[3]*Department of National Transport Strategy Planning, The Korea Transport Institute, Gyeonggi-do, Korea*

## 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 passenger 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 the shared-ride concept. There are recent successes in dynamic ride-sharing projects that are expected to bring substantial benefits arising from energy consumption and operation efficiency and thus, it is essential to develop advanced shared-taxi-dispatch algorithms and investigate the collective benefits of dynamic ride-sharing by maximizing occupancy and minimizing 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. 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 dynamic ride-sharing.

KEYWORDS: Dynamic Vehicle Routing Problem, Shared-Taxi Simulation, Hybrid Simulated Annealing, Dynamic Shared-ride system

## 1. INTRODUCTION

Dynamic ride-sharing is defined as dynamically utilizing empty seats in passenger cars by assigning passengers on demand, which is quite different from the early version of car-pooling projects that were not feasible for real-time response due to lack of advanced information technologies. As dynamic ride-sharing projects have been successfully initiated, the potential benefits of ride-sharing are expected to be substantial in reducing fuel consumption, carbon emissions, and traffic congestion. For customers, ride-sharing can also reduce travel costs associated with driving and parking. A newer design of Demand Responsive Transit (DRT) with true real-time routing, which can be named RTRT (Real-Time Routed Transit Systems), has emerged in recent years: Dial (1995), Cortés and Jayakrishnan (2002), Hadas and Ceder (2008), Quadrifoglio and Li (2009), and Jung and Jayakrishnan (2011), but those concepts are not fully refined for practical service in the real world. Lyft[1], Carma[2], Sidecar[3], and UberX[4] are well known

* Corresponding Author. Tel.:+1 949-824-5989, E-mail: jyoungjung@gmail.com
[1] http://www.lyft.com, visited on Jan. 22, 2014
[2] http://car.ma, visited on Jan. 22, 2014
[3] http://www.side.cr, visited on Jan. 22, 2014

services that have recently emerged for private ride-sharing by simply matching drivers and riders in real-time as passengers travel in urban areas. These services utilize private vehicles operated by regular car-owners and not commercial drivers.

However, it has been known that these private ride-sharing services could raise potential concerns about passenger insurance and fare-collection systems 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, dynamic 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 in real-time and establishing taxi 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 (1997), the system 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[5] in certain U.S. urban areas provides a Smartphone-based on-demand taxi service, which is often called e-hailing system, though not involving ride-sharing 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 other 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: Split-it[6], Tao (2007), and Tsukada and Takada (2005).

In this paper, an optimization scheme is developed for 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. Dynamic shared-taxi operation with associated algorithms is studied in realistic scenarios to evaluate system performance and efficiency of solving the vehicle routing problem. The remainder of the paper is organized as follows. In the next section, the dynamic shared-taxi problem is specified as a constrained problem of pick-up 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. DYNAMIC SHARED-TAXI-DISPATCH PROBLEM

In comparison to the conventional vehicle routing problems (VRP) such as VRPPD (Vehicle Routing Problem with Pick-up and Delivery) and VRPTW (Vehicle Routing Problem with Time Windows), the dynamic shared-taxi-dispatch problem can be characterized in three aspects. First, the proposed problem can be defined as a dynamic vehicle routing problem (DVRP) by Psaraftis (1995). As opposed to the static problem with predefined demands, the problem can be classified as open-ended type, in which the vehicle schedule may not be known and the solution only covers the short duration of the future due to the unknown (Psaraftis, 1988). Thus, the schedules need to be updated with real-time conditions such as new passenger requests, traffic information, real-time vehicle location, or unexpected vehicle breakdown. Secondly, the vehicle dispatch algorithm needs to solve the problem quickly and efficiently to provide quality passenger service. The dispatcher in a large-scale dynamic system needs to make a decision in a very short time whether a new request can be serviced or not so a fast response time is more critical. Lastly, the proposed problem focuses on a different objective function. Most urban taxi services are designed to minimize response time with strong dynamic demand systems while the conventional dial-a-

[4] http://www.uber.com, visited on Jan. 22, 2014
[5] http://www.uber.com, visited on Jan. 22, 2014
[6] http://www.split-it.sg, Accessed June 10, 2013

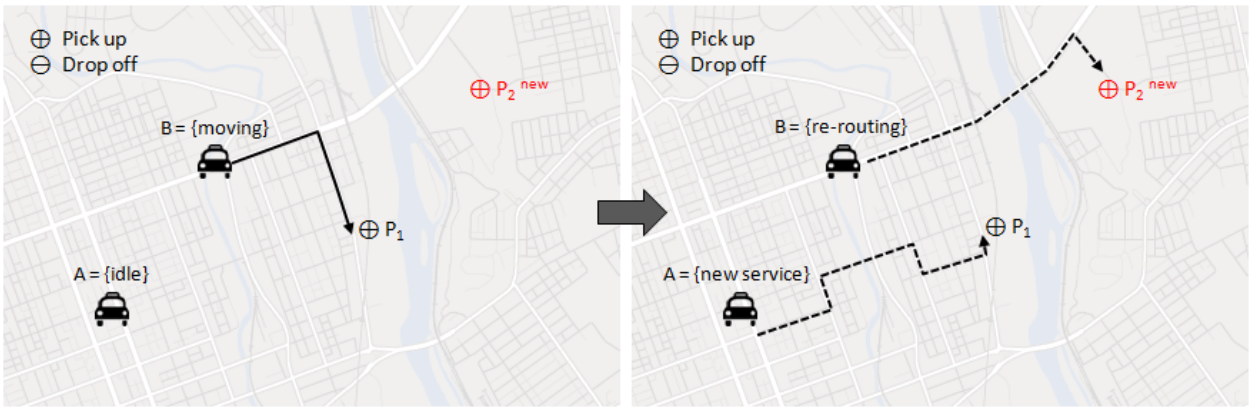
ride systems try to minimize the vehicle operating cost by seeking fewer servicing vehicles given the number of passenger reservations.

Those static or dynamic ride-sharing systems can be a similar DVRP application to real-time shared-taxi problems (Cervero, 1997). Effective usage of empty car seats by ride-sharing can be the same objective, but can be differentiated in some aspects: (1) Vehicles are operated by taxi drivers, which means that vehicle schedules are open-ended; (2) Vehicle operations are scheduled and controlled by a central dispatch system; (3) A system-wide optimal solution is required because use of the shared-ride concept in taxi service allows passengers to satisfy the riding public's preference as well as to save costs. Unlike the shared-taxi system, the ride-sharing system with private drivers is aimed to optimize the cost-savings of individual ride-share participants (Agatz *et al*., 2012).

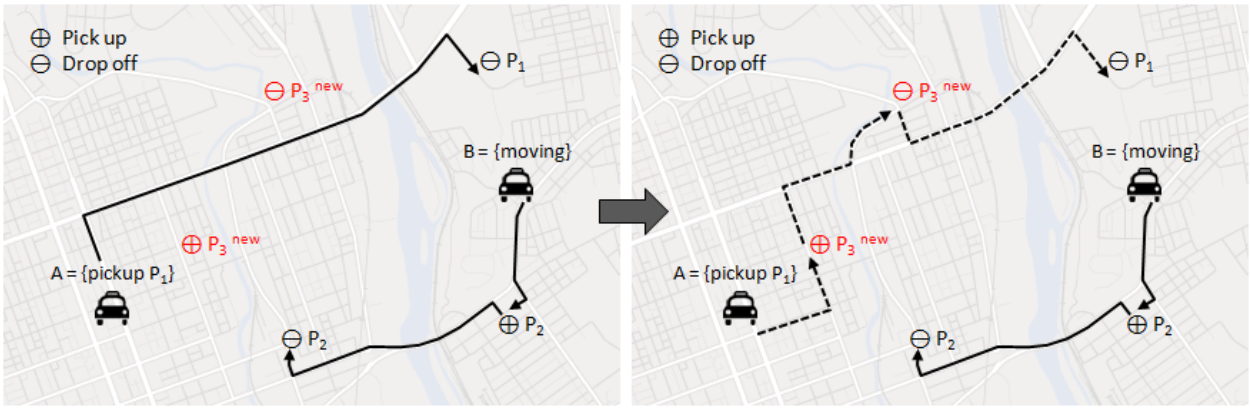
In dynamic taxi-dispatch system, when a new request is identified by the system operator, the service request is delivered to the system queue where each customer is labeled with time windows and locations of trip origin and destination. Meanwhile, the dispatch algorithm takes a service request from the queue and finds the best available taxi for the travel request within the time windows. If there is no available taxi to meet the constraints, the dispatch system can reject the request. The dispatch algorithm needs to find not only the best available vehicle among candidates, but also an optimal route with newly updated schedules that avoids violation of vehicle capacity constraints and time window constraints of previously assigned passengers as well as new passengers.

Seow *et al*. (2010) described a good example of a dynamic taxi-dispatch scenario by grouping 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. 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 vehicle $A$ to be assigned to the new request $P_2$ when $P_2$ shows up. However, if an algorithm can reschedule $P_1$ to vehicle $A$ and $P_2$ to vehicle $B$ 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 or unexpected traffic congestion.

Figure 1(b) illustrates a shared-ride example. 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 to all previously assigned passengers, $P_1$ and $P_2$. As a result, the passenger $P_1$ on board in vehicle $A$ will have a longer travel time by stopping by two points, but the total travel time of all passengers will be minimized within the maximum travel time constraint. It means that the algorithm should be able to assign $P_3$ to the best available vehicle in terms of an objective function and model constraints.




**(a) Re-routing**




**(b) Shared-ride**

**FIGURE 1 Dynamic Shared-taxi dispatch: (a) Re-routing and (b) Shared-ride.**

Despite its potential, there have been only several studies that have investigated how shared-taxi can improve system performance and level of service. Tao (2007) proposed a concept for dynamic taxi-rideshare. The study focused on ride-matching of passengers' origin and destination zones rather than solving vehicle routing problems. The author provided a good overview of the taxi-sharing concept with a trial field operation. Meng *et al*. (2010) 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. The main focus of the dispatch algorithm was to minimize passenger travel time and mitigate associated detours. Lee *et al*. (2005) introduced a two-step taxi-pooling dispatch system and provided a sensitivity analysis of the system performance. The study tackled the taxi-pooling problem for a feeder system that can 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). The proposed model aimed to minimize taxi operating cost as well as passenger traveling time. In dynamic ride-sharing, Herbawi and Weber (2012) proposed a Genetic and Insertion heuristic algorithm for dynamic ride-matching problem with time windows (RMPTW). The concept of dynamic RMPTW is similar to dynamic shared-taxi problems. The study proposed a similar objective function, which is to minimize the total trip distance of vehicles as well as to maximize the number of ride-sharing matched within the maximum distance and time allowed. Genetic algorithm was employed for the base solution then insertion heuristic responses for the newly received offers or requests. Notice that constraints of RMPTW can be different because those rides are offered by private drivers. A similar study, defined as dynamic

Taxi Sharing with Time Windows Problem (TSTWP), was conducted by Santos and Xavier (2013). A greedy randomized adaptive search procedure was proposed, which has two phases in order to minimize the cost paid by served passengers: (1) compute a greedy randomized initial solution and (2) perform a local search to improve the initial solution. A numerical experiment was performed based on the Manhattan's distance. Before discussing the models, it is noted that this study not only focuses on a large-scale optimization algorithm, but also investigates the feasibility of shard-taxi operation and their potential benefits. Due to model complexities and non-linear system performance of proposed concepts, our study requires realistic modeling and simulation, which can deliver a comprehensive evaluation of vehicle operation and system performance over the target area. We utilize the state-of-the-art simulation framework recently developed by Jung and Jayakrishnan (2014).

# 3. MODELING SHARED-TAXI DISPATCH ALGORITHMS

The proposed model in this study shows a more generalized model for shared-taxi service with a realistic simulation environment. This section starts with defining the objective functions and problem constraints. Next, 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 quick and simple manner; and (c) a Hybrid Simulated Annealing (HSA) that assign passengers efficiently and dynamically to available vehicles.

## 3.1. Model Constraints and Objectives

Compared to conventional DRT systems that usually focused on either pick-up or drop-off as passenger time-window constraints, passengers' concerns in shared-taxi will be how long they wait for a service and how long of a 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 at origin; and (3) maximum detour factor. Differing from a many-to-one problem, a vehicle picks up and drops 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 due to the limited number of vehicles, especially when the passengers have time windows. 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 total travel time of passengers; (2) maximizing system profit from selectively accepting passengers based on the current schedule. In (1), the cost evaluation has three terms, in-vehicle time (ride-time) of served passengers, waiting time of served passengers, and penalized value for rejected passenger requests, respectively. The in-vehicle time includes passengers' detour time incurred by ride-sharing as well. Note that it is possible that different proposed algorithms could have different numbers of delivered passengers during simulation. When scores are found in this way, a lower score (cost) indicates better performance.

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

$I$ = Set of all passenger requests during the simulation

$P_i^c$ = Completed request during the simulation, $i \in I$

$P^r$ = A set of rejected requests during the simulation
$T_p$ = Penalty value for a rejected request, 7200 sec
$RT(P_i^c)$ = Ride time of passenger group in the completed request, $P_i^c$
$WT(P_i^c)$ = Waiting time of passenger group in the completed request, $P_i^c$

The system profit is proposed based on the profit found from vehicle operating cost (based on vehicle distance traveled) and service revenue (based on the number of delivered passengers). In common taxi fare collection schemes, the fare starts with a basic flat fare with additional charges 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 generally found 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 simply consists of two parts (distance based profit): fixed revenue (2 km) and distance based revenue (1 meter). The operating cost can be obtained using the vehicle distance traveled as follows. In this case, a higher value indicates better performance. Notice that the second term takes into account the passenger's door-to-door distance without detour because we assume the system may not charge for the detour distance incurred by ride-sharing. The weight factor for vehicle operating cost, 0.4, was assumed based on the gas price (Liquid Petroleum Gas), average taxi mileage in city traffic, and other operating costs.

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

$J$ = Set of all vehicles
$\alpha$ = Fixed revenue (basic fare), 2000
$\beta$ = Weight of distance based revenue, 1.0
$\gamma$ = Weight of vehicle operating cost, 0.4
$D^P(P_i^c)$ = Passenger door-to-door distance excluding the basic fare distance
$D^v(V_j)$ = Vehicle distance traveled (km) of vehicle $V_j$, $j \in J$

## 3.2. Nearest Vehicle Dispatch

Currently, nearest Vehicle Dispatch (NVD) is the most widely employed strategy for on-line taxi-dispatch systems. In general, NVD has the following two steps. In step 1, when a new passenger request arrives, the algorithm seeks to find the nearest geographically available vehicle from the passenger's origin location so as to provide a quick and efficient response time by checking feasibility such as vehicle capacity, waiting time, and detour factor. Once a nearest vehicle is selected, an optimal schedule is found by assuming that the vehicle's pick-up 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 does not need a complicated dispatch algorithm. As expected, the passenger waiting time is minimized. However, passengers may detour longer when sharing their rides because the algorithm does not consider both previously assigned schedules, the time spent by passengers on board, or the trip destinations of those passengers.

## 3.3. Insertion Heuristic

Whereas NVD searches only for a nearest geographically feasible vehicle to assign a new passenger, Insertion heuristic (IS) considers all feasible vehicles to find a best available vehicle. 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 passenger requests. The proposed IS for shared-taxi algorithm is based on the heuristic method for the standard static vehicle routing problem generalized by Campbell and Savelsbergh (2004). Each vehicle $j$ is denoted by $V_j$ ($j \in J$). When a new service request arrives, the request is identified as $z_i$ ($i \in I$). At any moment, a vehicle $V_j$ has its current schedule $K_j$ containing $k$ pick-up and delivery events, defined as $(\mathrm{e}_1^{1,+}, \mathrm{e}_2^{1,-}, \ldots, \mathrm{e}_{k-1}^{i,+}, \mathrm{e}_k^{i,-})$, in which $\mathrm{e}_{k-1}^{i,+}$ and $\mathrm{e}_k^{i,-}$ represent the ($k$-1)-th and ($k$)-th events for pick-up (+) and drop-off (−) for passenger $z_i$. The objective is to minimize the incremental cost, waiting times and detours as follows:

$$\arg\min_j IC(z_i, V_j) = C_j(K_j, z_i) - C_j(K_j) \tag{3}$$

$$C_j(K_j) = \sum_{k \in K}[WT(\mathrm{e}_k) + RT(\mathrm{e}_k)] \tag{4}$$

$$C_j(K_j, z_i) = \arg\min_{m,n} \sum_{k \in K}\left[WT(\mathrm{e}_k) + RT(\mathrm{e}_k) + WT\left(\mathrm{e}_m^{i,+}, \mathrm{e}_n^{i,-}\right) + RT\left(\mathrm{e}_m^{i,+}, \mathrm{e}_n^{i,-}\right)\right] \tag{5}$$

where

$C_j(K_j)$ = Total cost of vehicle $V_j$'s with schedule $K_j$
$C_j(K_j, z_i)$ = Total cost of vehicle $V_j$ when adding a new request $z_i$
$WT(\mathrm{e}_k)$ = Waiting time (cost) associated with $k$-th event in $K_j$
$RT(\mathrm{e}_k)$ = Ride time (cost) associated with $k$-th event in $K_j$

$IC(z_i, V_j)$ in (3) represents an incremental cost of vehicle $V_j$, including two terms respectively: (a) The total cost of the updated schedule of vehicle $V_j$ including the new request; (b) The total cost $C_j$ based on the current schedule as shown in (4). In (5), the total cost can be updated by new pick-up and drop-off events. Note it also returns 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 computational procedure can be represented in Algorithm 1. First, *FeasibleArea*($j$,$z_i$) filters available vehicles based on the passenger's geographical location and the vehicle occupancy. For example, the majority of vehicles might not be able to reach at $z_i$'s pick-up location if the waiting time window is tight. Those vehicles can be immediately removed out of the candidate list. *GetOptPosition*($j$, $z_i$) corresponds to Eq. (3) by seeking the best pick-up and drop-off insertion positions within vehicle $V_j$'s existing schedule. In this function, a candidate tour is constructed and the feasibility is checked using a function *FeasibleTour*($\mathrm{e}_m^{i,+}$, $\mathrm{e}_n^{i,-}$, $K$) in which vehicle capacity and time window constraints are met.

Algorithm 1: Insertion Heuristic – Cost Minimization

```
J = set of available vehicles
Initialization: j* = ∅, ic* = +∞, and [e*^{i,+}, e*^{i,-}] = [∅, ∅]
for j ∈ J do
    if FeasibleArea(j, z_i) then
```

```
        [e_m^{i,+}, e_n^{i,-}], ic = GetOptPosition(j, z_i)
        if [e_m^{i,+}, e_n^{i,-}] != [∅, ∅] and ic < ic* then
            [e_*^{i,+}, e_*^{i,-}] = [e_m^{i,+}, e_n^{i,-}]
            j* = j
        end if
    end if
end for
Insert(j*, e_*^{i,+}, e_*^{i,-})
Update(j*)
```

Function: *GetOptPosition*$(j, z_i)$

```
z_i = a new passenger request and K = set of events for vehicle j
Initialization: ic* = +∞, [m*, n*] = [∅, ∅]
c = Objective(K)
for m=(1, k)∈K do
    for n=(2, k+1)∈K do
        if FeasibleTour(e_m^{i,+}, e_n^{i,-}, K) then
            ic = Objective(e_m^{i,+}, e_n^{i,-}, K) - Objective(K)
            if ic < ic* then
                ic* = ic
                [m*, n*] = [e_m^{i,+}, e_n^{i,-}]
            end if
        end if
    end for
end for
Return [m*, n*], ic*
```

The proposed IS is fairly easy and straight-forward to implement and shows reasonable computational efficiency with the time complexity of $O(n^3)$ but has limitations on large-scale dynamic pick-up and delivery operations. Since the insertion heuristic based on an FCFS priority scheme does not consider all new requests at the same time, it may potentially lead to a sub-optimal solution. In other words, there is no re-optimization scheme such as exchanging a passenger assigned to one vehicle's schedule to another vehicle's schedule later. The problem of finding an optimal solution is however not easy due to well-known combinatorial issues. This leads us to developing an optimization scheme that while still is heuristic, can reach near optimal solutions, as described in the next section.

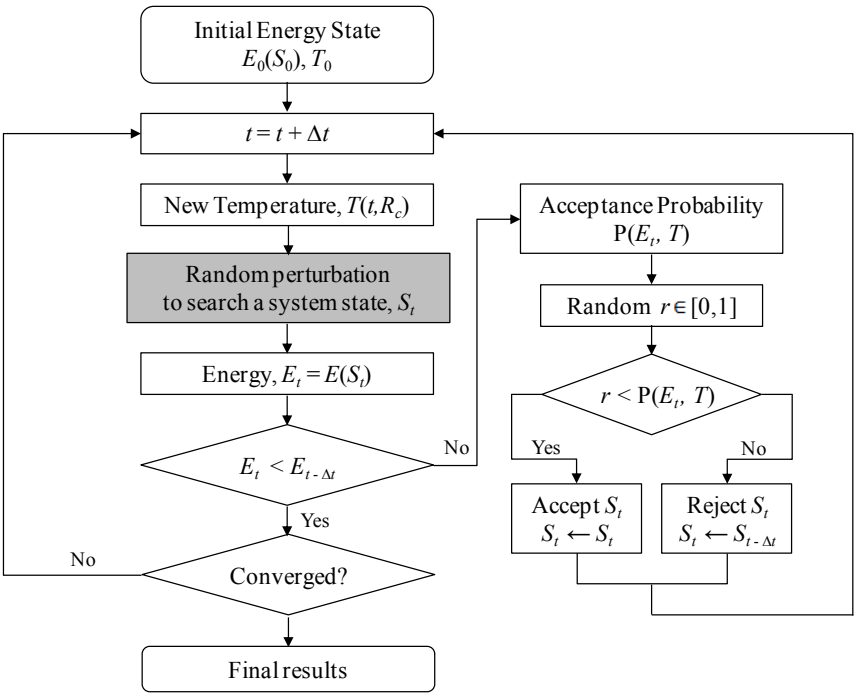
## 3.4. Hybrid Simulated Annealing

### 3.4.1 Simulated Annealing for Shared-Taxi Dispatch Algorithm

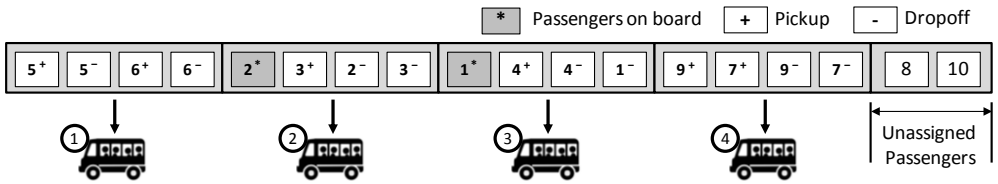
Simulated Annealing (SA), first suggested by Metropolis *et al*. (1953) and defined by Kirkpatrick *et al*. (1983) is a generic probabilistic meta-heuristic, which is capable of finding an approximately accurate solution for the global optimum of a complex system with a large search space. It has been applied successfully to a wide variety of complicated combinatorial optimization problems in many applications such as vehicle routing problems (Van Breedam 1995, Chiang and Russell 1996, Czech and Czarnas 2002), optimal location problems (Liu *et al*. 1994, Ng *et al*. 2010), network design problems (Fan and Machemehl 2006, Zeferino *et al*. 2009). SA is a stochastic relaxation method based on an iterative procedure starting at an initial "higher temperature" with the system in a known configuration in which the word "temperature" is used to give an intuitive connection to metallurgy. The iterative procedure of

SA improves the cost function until the current temperature cools down. More detailed information can be found in Kirkpatrick *et al*. (1983).

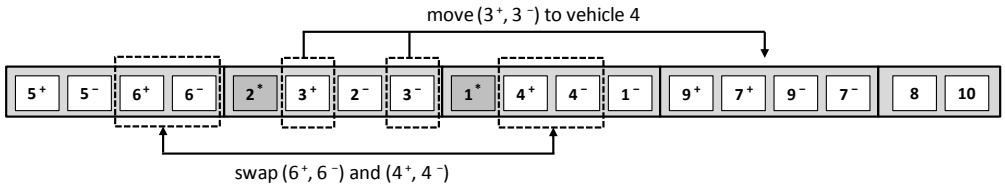
In comparison with NVD and IS that are simply inserting a new request into its existing schedule, a successive re-optimization method is used for SA by assuming that $n$ service requests arrive during the previous time period (e.g., $H \leq 60$ sec) in the scheduling horizon. In other words, SA provides a systematic re-optimization scheme to assign new requests as well as updates existing schedules in real-time. At each end of the time horizon, SA starts with parameters, initial solution $S_0$, iteration $I_{iter}$, initial higher temperature $T_0$, final lower temperature $T_c$, cooling rate $R_c$, and Boltzmann constant $K$. $I_{iter}$ denotes the number of iterations at a particular temperature.




**(a) Simulated Annealing Procedure**




**(b) System State Vector $S_t$**




**(c) Random Perturbation**

**FIGURE 2 (a) SA Procedure, (b) System State Vector $S_t$, and (c) Random Perturbation**

Figure 2(a) shows the general procedure for SA. At a predefined higher temperature, $T_0$, the algorithm starts from an initial solution $S_0$ and its objective value $E(S_0)$, which is called system state vector and energy level respectively. The procedure to search the global minima consists of comparing energy levels for two consecutive random states, $S_t$ and $S_{t+1}$. In this case, the state vector $S_t$ represents vehicle schedules and the energy $E(S_t)$ indicates the cost associated with vehicle routing in the shared-taxi

problem. In Figure 2(b), $S_t$ consists of many cells representing service events for all vehicles in the system as described. For example, vehicle $V_1$ has four cells with passenger ID={5, 6} with four pick-up and drop-off events. It is noted that each vehicle $V_j$ could have passengers on board at any decision time. In addition to vehicle schedules, the state vector considers unassigned passenger requests characterized by a salvage slot with higher penalty values at the end of state vector. Unassigned passenger requests will be dropped due to vehicle capacity constraints and time constraints. At each iteration, the current state $S_t$ is replaced by a randomly generated candidate, $S_{t+1}$, which is called random perturbation or random walk. 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 new state, $S_t$. 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, $P(E_t, T) = \mathrm{e}^{\Delta E/kT}$ where the temperature is affected. In this manner, SA tends to allow a random search at higher temperature, but it only allows a local improvement at the lower temperature.

A standard SA procedure is adopted to generate a random neighbor including move and swap. Basically a move is used for the main operator by moving a passenger schedule from a random vehicle to another, which can be a critical part of the random perturbation to generate a neighborhood solution in Figure 3(c). The swap procedure is started from randomly selecting two vehicles, $V_{r1}$ and $V_{r2}$, and then swapping two or more random schedules from each vehicle.

The following shows the two energy levels, (6) and (9), for both cost minimization and profit maximization of shared-taxi problems in accordance to the objectives mentioned in (1) and (2). Two passenger sets are considered. $I_j$ denotes the requests assigned to $V_j$ and $I_r$ represents the rejected passenger requests. The energy level for the cost minimization in (6) consists of two parts, the cost associated with vehicle schedule in (7) and the penalty cost for rejected passengers in (8). Note that the penalty value (7200 sec) should be more than the longest passenger door-to-door trip (around one hour) at least. In (9), the energy level can be simply replaced with a maximization problem in (9), which is a sum of the profit associated with individual vehicles as shown in (10). 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:

$$E_t = E^{Cost}(S_t) = \sum_{j \in J} C_v(V_j) + \sum_{i \in I_r} C_r(z_i) \quad (6)$$

$$C_v(V_j) = \sum_{i \in I_j} [TT(z_i) + WT(z_i)] P(z_i) \quad (7)$$

$$C_r(z_i) = T_p P(z_i) \quad (8)$$

$C_v(V_j)$ = Passenger cost associated with vehicle $V_j$
$C_r(z_i)$ = Penalty value associated with unassigned passenger request $z_i$
$P(z_i)$ = Number of passengers in passenger request $z_i$
$T_p$ = Penalty value for passengers in the salvage slot, 7200 sec

Energy level ($E_t$) for maximizing system profits:

$$E_t = E^{Profit}(S_t) = \sum_{j \in J} G_v(V_j) \quad (9)$$

$$G_v(V_j) = \alpha \sum_{i \in I_j} z_i + \beta \sum_{i \in I_j} D^P(z_i) - \gamma D^v(V_j) \quad (10)$$

$G_v(V_j)$ = Profit associated with vehicle $V_j$
$\alpha, \beta, \gamma$ = 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 request $z_i$ in vehicle $V_j$
$D^v(V_j)$ = Expected vehicle distance traveled (km) associated with the current schedule of vehicle $V_j$

Setting the number of iteration at each temperature can be critical because a higher number of iteration provides a higher opportunity to move around the search space which expands 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. Therefore, setting the temperature plays a critical role in acceptance probability and the value of the initial temperature depends on the scale of the cost of a problem-specific objective function. The initial value should not be high enough that the algorithm simply conducts a random search causing excessive computation time. Kirkpatrick et al. (1983) suggested $T_0 = \Delta f_{max}$ where $\Delta f_{max}$ is the maximum cost (e.g., energy level) difference between any two neighboring solutions. Alternatively, the initial temperature can be calculated by estimating an average objective improvement, $\delta\Delta f$ in formula (11), where $p_0$ is a desired average increase of acceptance probability and $\delta$ is a scale parameter. In this study, a separate procedure generating a set of random trials is performed to measure the difference between the best and average solution values, $\Delta f$, at each re-optimization. According to Crama and Schyns (2003), usually $p \in$[0.8, 0.9], $R_c \in$ [0.80, 0.99] and $K \in$ [0.1, 1.0] are set.

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

Note that the selection of parameters including the number of iterations and temperature might not only affect the quality of the solution, but also 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 dynamic scheduling, other meta-heuristic techniques can be used in combination with parallel computing techniques by Ram and Sreenivas (1996).

### 3.4.2 Hybrid Simulated Annealing

Since the proposed dynamic shared-taxi problem involves a constrained search space with vehicle capacity, maximum waiting time window, and maximum detour time factor, the validity of a newly generated neighbor should be checked while generating a neighborhood solution by random walk. If the move and swap operations generate infeasible solutions, those unexpected infeasible solutions would cause a significant impact on system efficiency and solution accuracy. A naïve approach can keep generating numerous solutions until a feasible solution is found, but it can cause unnecessary computational burdens. In this study, a heuristic technique called Hybrid Simulated Annealing (HSA) is adopted to keep the solutions feasible and the individual vehicle's schedule optimized. Searching a random candidate, the grayed part in Figure 2(a), can be replaced by the IS algorithm introduced in the previous section.

When inserting a passenger request to the vehicle's schedule of swap or move procedures, the existing vehicle schedules can be updated without any constraint violation by utilizing *GetOptPosition*$(j, z_i)$ in Algorithm 1. Algorithm 2 shows the proposed random perturbation procedure. First, HSA selects a random passenger schedule $z_r$ from a random vehicle $j^{r1}$ from the current system state vector $S_t$. A function, *MoveEvents*$(j^{r1}, z_r)$, is called to move $z_r$ from $j^{r1}$ to an arbitrary vehicle $j^*$ in the same state vector. If $j^*$ is feasible to accept $z_r$, the pick-up and drop-off events associated with $z_r$ are inserted into the optimal position among $j^*$'s schedule by calling *GetOptPosition*. Then, the previous events in $j^{r1}$ are

deleted. If *MoveEvents* function returns false, which means there are no available vehicles for $z_r$, it goes into the salvage slot. It is noted that a random passenger $z_r$ can also be selected from the salvage slot to a random vehicle with a certain probability. When selecting $z_r$ from a vehicle, each passenger event in a vehicle has a different probability of being swapped, $p(z_r)$. For example, the passenger ID=9 of the vehicle $V_4$ in Figure 2(b) has a lower degree of probability to be selected than passenger ID=7, $p(z_9) < p(z_7)$, because moving an imminent pick-up event from one vehicle to another might cause operational inefficiency in practice especially when the vehicle is already headed to the pick-up event.

The advantage of the proposed HSA is the random walk's capability of keeping individual vehicles' schedules both feasible and optimal while the vehicle selection procedure remains random. 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. For example, combining a local optimization technique with a meta-heuristic approach has shown the computational efficiency and a better solution quality in vehicle routing for the real-time feeder system by Jung *et al*. (2011).

Algorithm 2: Random Perturbation for Hybrid Simulated Annealing

```
J = set of available vehicles in system state vector S_t
L = Number of Random Perturbation
for i=(1,L) do
    j^r1 = GetRandomVehicle()
    z_r = GetRandomPassenger(j^r1)
    if MoveEvents(j^r1, z_r) = FALSE then
        InsertSalvage(z_r)
    end if
end for
UpdateStateVector(S_{t+1})
```

Function: *MoveEvents*($j$, $z_r$)

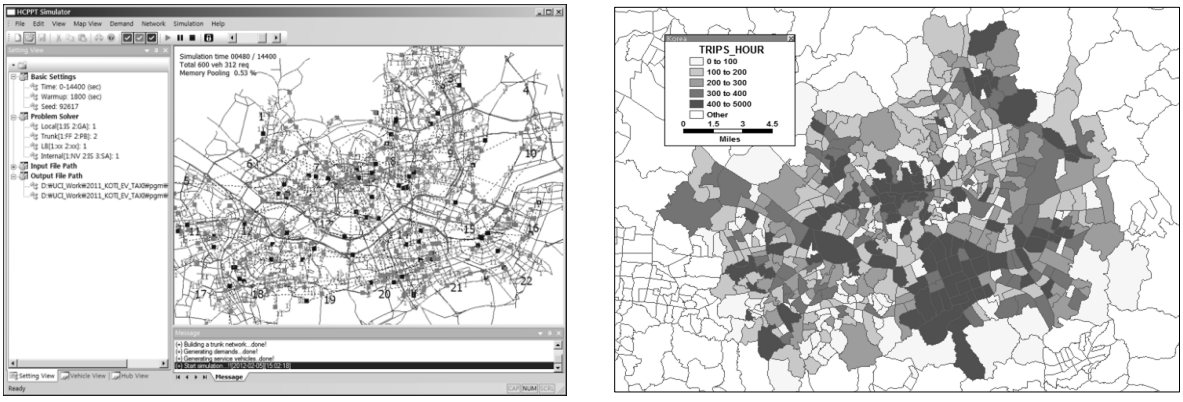
```
J = set of available vehicles
Initialization: j* = ∅ and [e*^{r,+}, e*^{r,-}] = [∅,∅]
while j* = ∅ do
    j^r2 = GetRandomVehicle(J, j^r2≠j)
    if j^r2 = ∅ then
        return FALSE
    end if
    if FeasibleArea(j^r2, z_r) then
        [e_m^{i,+}, e_n^{i,-}] = GetOptPosition(j^r2, z_i)
        if [e_m^{i,+}, e_n^{i,-}] != [∅,∅] then
            [e*^{i,+}, e*^{i,-}] = [e_m^{i,+}, e_n^{i,-}]
            j* = j^r2
        end if
    end if
    Remove(j^r2 ∈ J)
end while
Remove(j, z_r)
Insert(j*, e*^{r,+}, e*^{r,-})
Update(j*)
return TRUE
```

# 4. SIMULATION IMPLEMENTATION

## 4.1. Simulation Design and Stochastic Taxi Demand

The simulation framework for large-scale flexible transit system recently proposed by Jung and Jayakrishnan (2014) is used for the simulation study. The platform is written in Microsoft Visual C++ and it is capable of implementing various algorithms and visualizing all simulation elements as shown in Figure 3(a). The simulator imports digital maps designed for map display, geo-coding, and includes a vehicle route planning algorithm with network attributes such as road categories, turning prohibition, one-way, posted speed, number of lanes, link lengths, and link shapes. For a simulation in urban area, the Seoul area in Korea is abstracted from the national transportation network which contains 8,382 links and 6,321 nodes provided by the Korea Transport Institute (KOTI).

For taxi demand, the stochastic taxi demand in the KOTI regional transportation planning model is used. As of 2011, the trip demand consists of auto, bus, subway, rail, taxi, and other types of demands which covers the Seoul area with a total of 560 zones. Figure 3(b) show the stochastic taxi demand used in this study, which is represented in both trip origin and destination. In the KOTI regional transportation model, the taxi demand was modeled based on the 2010 Korea National Household Travel Survey and taxi operation survey and GPS data. The original dataset has 560 zones based on 424 administrative districts. The collected Taxi GPS (e.g., pick-up and drop-off locations) is matched to corresponding zones and combined with the survey according to KOTI taxi demand model. The taxi demand table that includes origin-destination demand rates is directly imported into the simulation. Under the usual assumption of spatial uniformity of demand around a zone centroid, point-to-point dynamic taxi demands are randomly generated in accordance to the destination probabilities from the taxi demand table of each centroid. Random locations are generated from zonal polygons and those locations are projected to the nearest link. Detailed procedures are described in the study (Jung and Jayakrishnan, 2014). The real-time service requests arrive according to a Poisson process in a temporal manner. A total of 18,000 service requests are generated with the minimum trip length as 1.5 km for the taxi service. The majority of trip demands are within 10 km and the average trip length is 6.3 km. The expected door-to-door travel time is 13.3 min under the assumption that vehicles can travel at 60-90% of the posted speeds, depending on road categories in the network.




**(a)** **(b)**

**FIGURE 3 (a) Shared-taxi Simulator with Seoul network and (b) Stochastic Taxi Demand.**

## 4.2. Simulation Scenarios

According to the Seoul Taxi Association, a total of 72,000 taxi licenses are registered including owner-driver taxis and as of 2011, a total fleet of 40,000 vehicles are actually operating. A total of 600 service vehicles are considered in the simulation, which is equivalent to 1.5% of the total number of vehicles operating in Seoul. The initial vehicle locations 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 service 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 four 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 random uniform distribution $U(0.5, 1.5)$. Three different algorithms, namely, NVD, IS, and HSA are performed. For HSA, 60 sec is considered to be the period for successive re-optimization. We set up the maximum 6,000 iterations depending on each annealing temperature, the highest temperature obtained by generating initial random solutions, the lowest temperature ($T_0$= 0.2), Boltzmann constant ($K$= 1.0), and cooling rate ($R_c$= 0.85).

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. For MCO, the maximum waiting time window is set to 15 min, which is a sufficient value to promote ride-sharing in urban areas, but not too long compared to the average door-to-door distance, which is equivalent to 13.3 min from our simulation.

**TABLE 1 Simulation Scenarios**

| Shared-taxi simulation | |
|---|---|
| **Shared Taxi settings** | |
| Service area (km$^2$, Seoul in South Korea) | 605 |
| Simulation time and Warm up (hours) | 4, 0.5 |
| Number of service vehicles | 600 |
| Vehicle capacity (seats/vehicle) | 4 |
| Maximum waiting time (min) | 15 (900 sec) |
| Maximum detour factor | 2.0 |
| Operation types[1] | SCO, MCO |
| **Demand and algorithm settings** | |
| Stochastic demand levels (thousand requests/4-hour) | 9, 12, 15, and 18 |
| Vehicle 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 in shared-ride transportation systems, Level-of-Service (LOS) index ($\phi$) and Ride-time index ($\rho$), which are discussed by Black (1995). Note that these indices may be a little misleading as lower values indicate better. The door-to-door ride time in (12) and (13) is the travel time when no other passengers are picked up or dropped en-route (i.e., equivalent to the time for driving a personal auto) given the same network conditions.

$$\phi = \frac{Avg.\ passenger\ waiting\ time\ at\ pick\text{-}up\ location}{Avg.\ door\text{-}to\text{-}door\ ride\ time} \quad (12)$$

$$\rho = \frac{Avg.\ vehicle\ ride\ time}{Avg.\ door\text{-}to\text{-}door\ ride\ time} \quad (13)$$

## 5.1. Minimizing Passenger Travel Time

Figure 4 shows the performance measures of a 4-seater taxi with passenger travel cost minimization. Regarding passenger delivery, HSA apparently performs better than the other two algorithms in Figure 4(a). It shows more passenger delivery performance (16% and 10%) than NVD and IS respectively. However, when the passenger demand stays at lower demand levels (9,000 and 12,000 requests), there is no significant difference between IS and HSA in terms of passenger delivery. Figure 4(b) reports the number of rejected passenger requests. Notice that IS and HSA are able to serve almost all requests at the lower demand levels.

Figure 4(c) and 4(d) shows LOS and ride-time indices. NVD shows the best performance in the LOS index because it assigns the passengers to the nearest vehicles minimizing the waiting time without any consideration of passengers' destinations, so it is clearly seen that this greedy concept results in the lack of optimality in ride-time index as shown in Figure 4(d), in which passenger detour increases significantly with NVD. According to simulation results, average waiting times remain from six to twelve minutes in minimizing cost, which are far below the maximum waiting time constraint of 15 min. It is explained that the solutions are mostly constrained by the maximum detour factor, 2.0. As the demand level increases, ride-time indices increase rapidly. At higher levels of passenger requests, the ride-time indices of NVD and IS are almost 2.0 or more in Figure 4(d). It is noted that ride-time index can be slightly over the maximum detour factor due to the passengers' boarding and alighting times 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.

The average passenger travel time in Figure 4(e) is defined as the sum of both waiting time and in-vehicle ride time. Most importantly, Figure 4(f) shows the objective function cost comparisons of the simulation results. Apparently, HSA outperforms both NVD and IS. 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 to HSA that periodically optimizes the entire vehicle schedules, which can not be achieved by a simple heuristic solution. It is also noted that the penalty value (7200 sec) for rejected requests are included in the calculation of total cost. In conclusion, HSA shows outstanding performance in both passenger delivery and shorter passenger travel time.

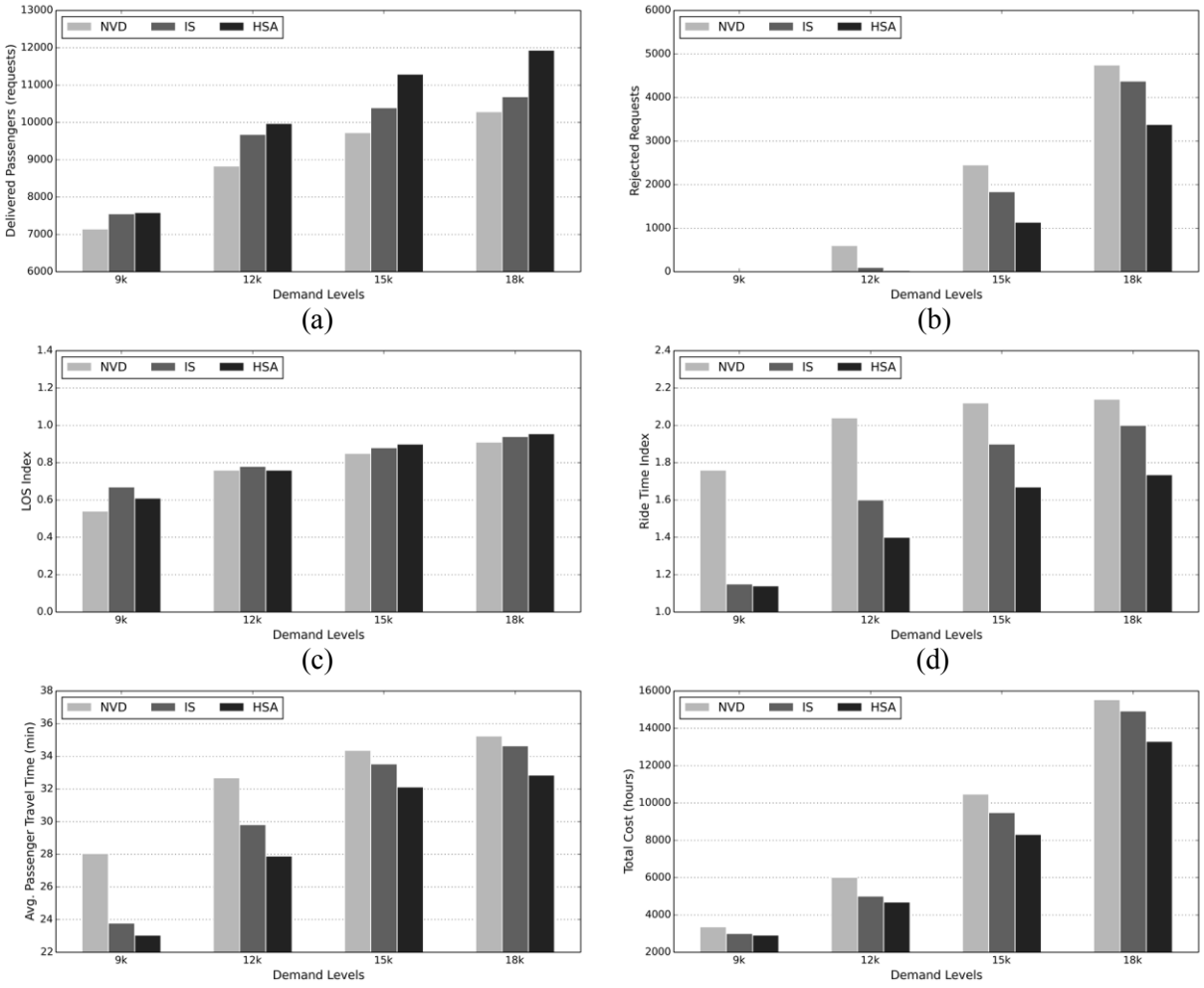



**FIGURE 4 Performance measures (Minimizing Cost with MCP): (a) Number of Delivered Passengers, (b) Number of Rejected Requests, (c) LOS Index, (d) Ride-time Index, (e) Average Passenger Travel Time, and (f) Total Cost.**

## 5.2. Maximizing Operating Profit

First, it is noted that when applying the profit maximization, HSA algorithm tends to accept passengers' requests selectively to maximize its profit. No penalties are applied for rejected request. Figure 5 shows the system performance measures for profit maximization. As opposed to the previous results, the number of delivered and rejected passenger requests in Figure 5(a) and 5(b) are very similar over three algorithms. Interestingly, HSA shows even lower passenger delivery than other algorithms and higher number of request rejection. This indicates that HSA does not guarantee higher numbers of customers, but it focuses on the passengers that help the system profit increase. Unlike the previous section, there is no significant difference in ride-time index in Figure 5(d). In Figure 5(f), HSA reports the best profits for all demand levels compared to both NVD and IS results. It achieves around 47% and 30% more profit compared to NVD and IS. As the passenger demand increases, the profit with HSA keeps increasing significantly. Although LOS and ride-time indices of three algorithms are not significantly different, HSA shows the highest passenger travel times in Figure 5(e) because it actually accepts passengers who travel longer

distance. Notice that the LOS and ride-time indices are normalized statistics based on passengers' door-to-door travel.

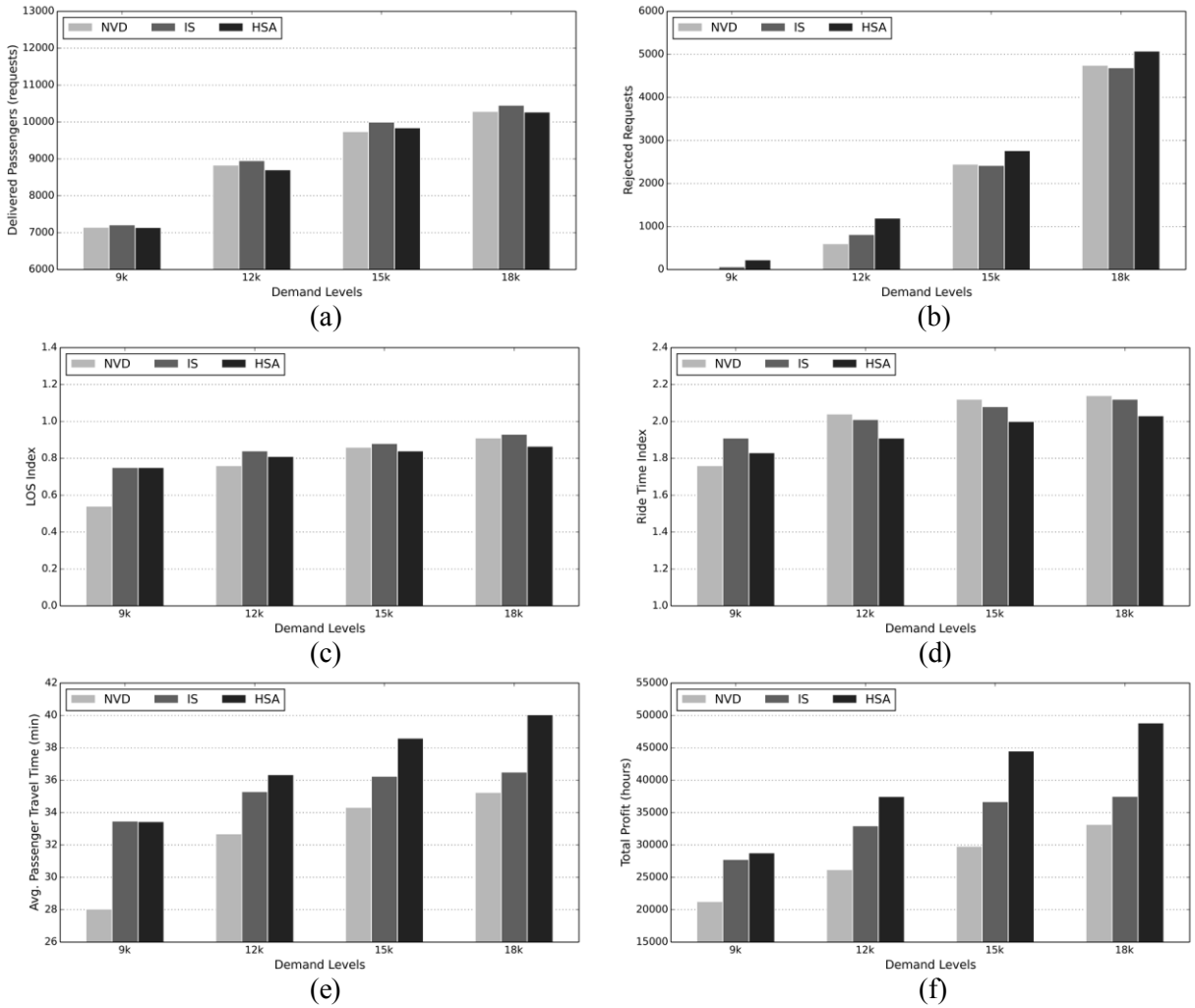



**FIGURE 5 Performance measures (Maximizing Profit with MCO): (a) Number of Delivered Passengers, (b) Number of Rejected Requests, (c) LOS Index, (d) Ride-time Index, (e) Average Passenger Travel Time, and (f) Total Profit.**

Figure 6 provides a side by side comparison between two objective functions. First, comparing the distributions of accepted passenger requests at 18,000-request, has, in maximizing profit, in Figure 6(b) clearly tends to accept passengers who travel longer distances. It also shows a substantial decrease in the number of accepted requests (less than 5 km) and an increase in longer distance trips (more than 5 km) compared to Figure 6(a). Notice that NVD and IS might efficiently assign passengers to vehicle schedules, but they try to accept all passenger requests in FCFS manner as long as all constraints are honored. A similar pattern can be seen in the comparison between Figure 6(c) and 6(d). Even though the passenger door-to-door distance increases with HSA, the average passenger travel time per unit distance (km) is lower than those of other algorithms as shown in Figure 6(e) and 6(f), meaning that HSA achieves efficient passenger traveling performance on both objective functions.

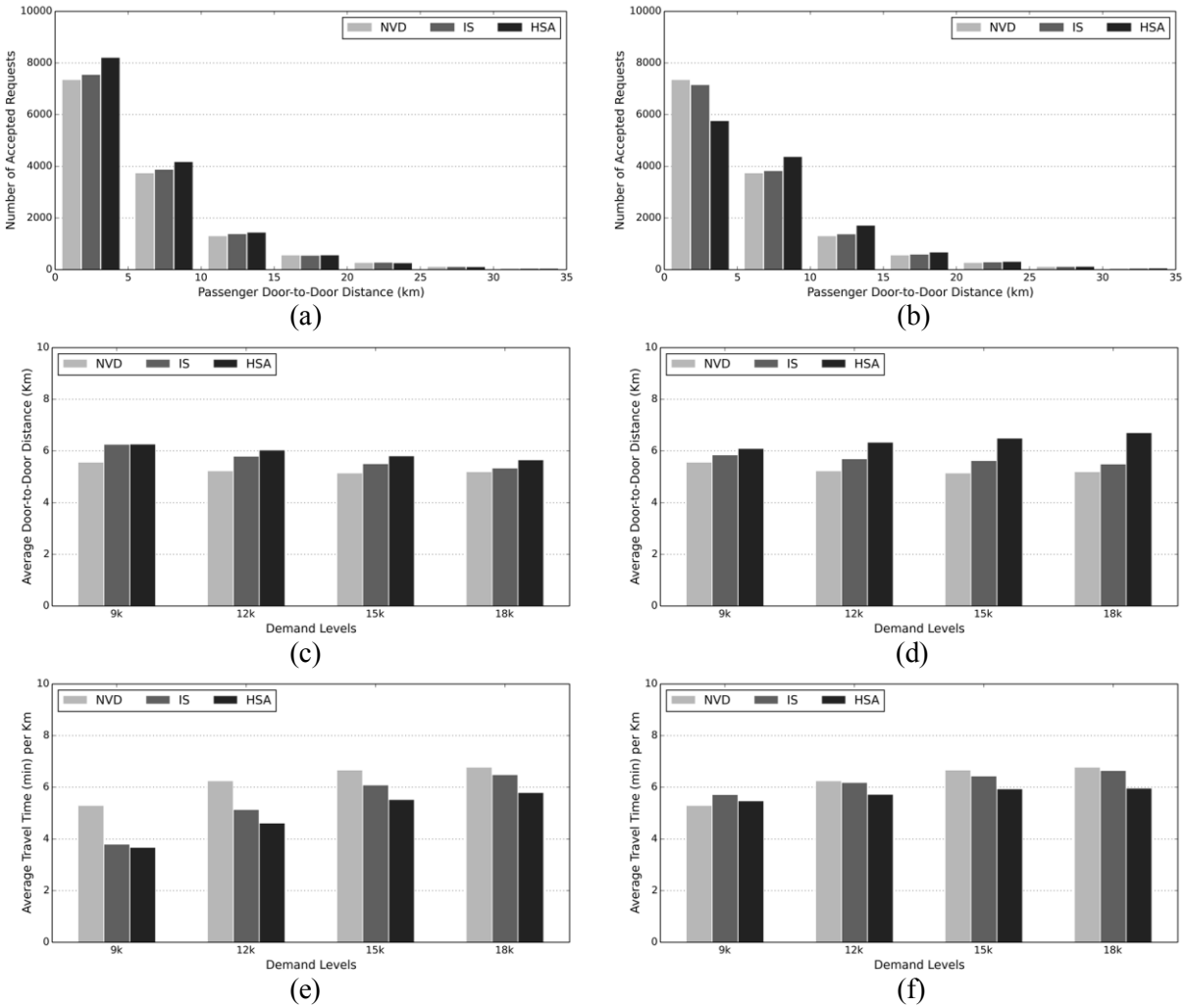
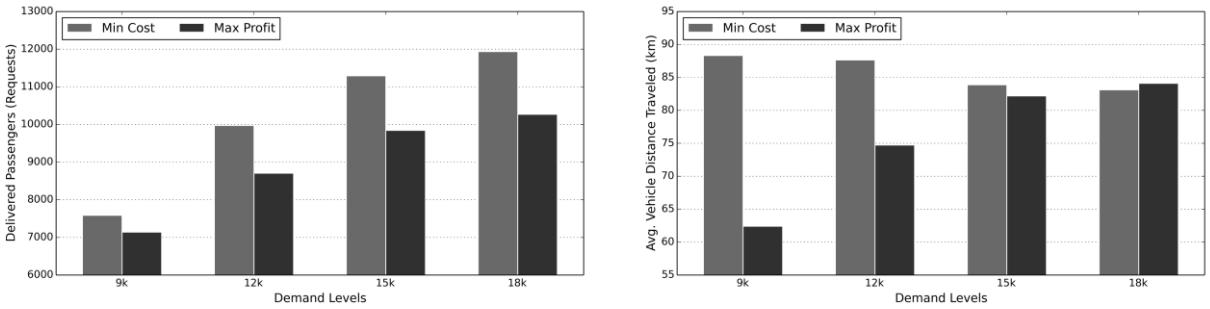



**FIGURE 6 Distribution of Accepted Requests by Door-to-Door Distance (18,000-requests): (a) Minimizing Cost and (b) Maximizing Profit, Average Door-to-Door distance: (c) Minimizing Cost and (d) Maximizing Profit, and Average Passenger Travel Time per Km: (e) Minimizing Cost and (f) Maximizing Profit.**

Figure 7 compares the two objective models, indicating different impacts on the passenger delivery and fleet distance traveled. The cost minimization with HSA in Figure 7(a) tends to deliver as many passengers as possible due to its penalty whereas the profit maximization takes passenger requests that can contribute more system profits. It is noted that the cost minimization strategy not only minimizes the passenger waiting time and ride 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 lacks any consideration of passenger delivery.

In Figure 7(b), the average vehicle distance traveled (62 - 84 km) with the profit maximization strategy stays lower than the values (83 - 88 km) obtained with the passenger travel time minimization at the lower demand levels. The result is consistent with the objective function of the profit maximization that tries to minimize the vehicle operating distance.




(a) (b)

**FIGURE 7 Performance measures with HSA (Minimizing cost and Maximizing profit): (a) Number of Delivered Passengers, (b) Average Vehicle Distance Traveled.**

## 5.3. Benefits of Shared-taxi

Another simulation is performed to investigate the benefit of shared-taxi with the comparison between SCO and MCO in Figure 8. SCO allows only one customer group in a taxi at any time while MCO allows multiple passenger groups with four seats. There could be two different strategies for the scheduling of SCO. 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 associated time constraints. For example, in the second strategy, a new pick-up schedule can be assigned to a vehicle even before the vehicle has finished the current schedule of the passenger on board. From our preliminary simulation runs, it was found that the first strategy delivers 25% fewer passengers, but it shows shorter waiting times at home compared to the second strategy. In this section, 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.

Figure 8(a) and 8(b) reports the side by side comparison with the number of passenger requests for in-system, rejected, and delivered during the simulation. The number of completed requests with SCO remains at the same level while notable increases can be seen in those with MCO. At the highest demand level, MCO shows about 4,000 more completed requests than SCO. The number of rejected requests with MCO remains at the lowest level at the lower demand levels (9,000 and 12,000) 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 even lower than with SCO in Figure 8(c). It can be explained that MCO has a higher probability of picking up new passengers while vehicles are performing. Consequently, the passengers might wait for a shorter time than the case of SCO in which a vehicle can go pick up a passenger only after dropping off the passenger on board. Ride-time index with MCO in Figure 8(d) keeps increasing as the trip requests increase, but it stays below 2.0 due to the maximum detour constraint. It is reasonable that the ride-time index with SCO stays 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.

When combining both ride-time and LOS indices, MCO shows an overall lower value than SCO at the lowest demand level. This reveals that the proposed shared-ride system could not only serve more passengers, but also provide 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 detour 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 7(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.

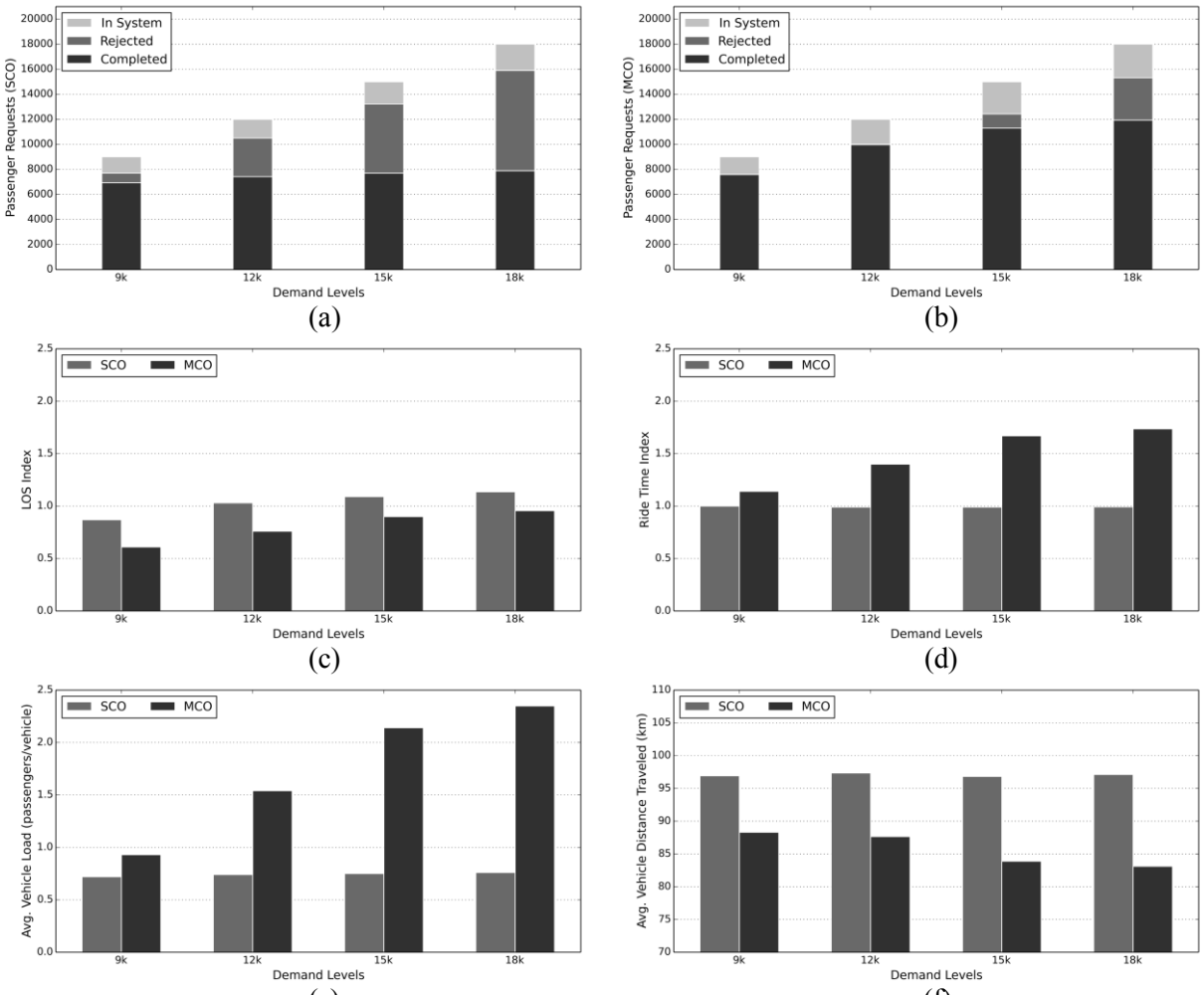



**FIGURE 8 Performance Comparison between SCO and MCO: (a) Passenger Delivery with SCO, (b) Passenger Delivery with MCO, (c) LOS Index, (d) Ride-time Index, (e) Average Vehicle Load, and (f) Average Vehicle Distance Traveled.**

## 5.4. Computational Performance

The proposed HSA can generate outstanding solutions given the level of demands, but it is inevitable for HSA to have extra computational burdens compared to NVD and IS since HSA uses an iterative combinatorial re-optimization method. To measure computational time, we use CPU-second as an embedded form in C++ codes with Intel® Core i5 2.4 GHz with 2G RAM. The computational performance measured in this section involves all of the processing times from preparing model variables to update vehicles' schedules with new real-time requests. The computational times for NVD and IS to insert one request are almost negligible with the simulated problem size. Considering that the vehicle load factor with the given problem is usually less than 3.0, it is conceivable that a vehicle usually has around six events and finding an optimal insertion position within a vehicle, $O(n^2)$, is computationally stable even though the original problem can be classified as NP-hard. Table 2 reports the average computational time

with different demand levels simulated in the previous section. The average request arrivals mean the average number of new requests arrived at each re-optimization procedure. Note that the computational times achieved in this study can vary depending on the parameters used in HSA.

**TABLE 2 Computational Performance with HSA**

| | Demand Levels | | | |
|---|---|---|---|---|
| | 9000-request | 12,000-request | 15,000-request | 18,000-request |
| Average computational time (sec) | 9.7 | 11.6 | 12.6 | 13.2 |
| Average request arrivals (req/60 sec) | 37.7 | 50.2 | 63.1 | 75.8 |

The result shows longer calculation times as HSA involves more numbers of requests. The maximum average computational time is reported with 13.2 sec at the highest demand level. Compared to NVD and IS, HSA takes longer computational time, but it is acceptable because it not only handles multiple requests at a time, but also re-optimizes the existing schedules. It is also noted that the proposed HSA can perform considerably faster when being extended with parallel programming techniques (Malek *et al*., 1989; Ram and Sreenivas, 1996; Sarma and Adeli, 2001; Czech and Czarnas, 2002). Therefore, the computational time and scalability for dynamic shared-taxi problem will not be a critical issue in this study. One can suggest that the dispatcher can switch between IS and HSA as the demand level changes to save overall computational sources.

# 6. CONCLUSION

Taxi is a convenient and flexible transportation method, but the shortage of taxicabs during peak hours has been critical in many urban areas. In this study, a dynamic shared-taxi problem is designed to maximize its utility. Three types of taxi-dispatch algorithms – NVD (Nearest Vehicle Dispatch), IS (Insertion Heuristic), and HSA (Hybrid Simulated Annealing) – are introduced by assuming that all vehicles can optimize their schedule at the individual vehicle level with given pick-up and delivery schedules. The proposed hybrid scheme combining Simulated Annealing and Insertion Heuristic systematically optimizes the entire schedules of the vehicles in real-time, which is not possible with either NVD or IS. 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.

First, it is noted that the taxi demand in this study is imported from the KOTI regional transportation planning model, in which the survey and GPS data from taxi are based on the conventional street taxi-hailing condition whereas the simulation study assumes an e-hailing system by on-line, in which case actual demand patterns can be different. Secondly, we assumed that passengers are willing to share their rides with others and the simulation results show great potentials that ride-sharing can reduce the taxi fare by improving the system productivity. However, in practice, not all passengers want to share their ride unless incentives are offered depending on waiting and detour times. 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' differing level of willingness (e.g., individual preference for maximum waiting time and detour) towards ride-sharing dynamically. Finally, 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 in fast varying demand conditions. The proposed shared-taxi concept can be applied to increase the efficiency of EV (Electric Vehicle) taxi operation, which usually suffers from limitations by short driving range and battery charging as discussed by Jung et al. (2014). Also, the authors want to emphasize that the share-ride system proposed in this study can be easily converted to a dynamic shared-van (shuttle) service.

# REFERENCES

Agatz, N., Erera, A., Savelsbergh, M., & Wang, X. (2012), Optimization for dynamic ride-sharing: A review. European Journal of Operational Research 223, pp. 295–303. doi:10.1016/j.ejor.2012.05.028

Black, A. (1995), Urban Mass Transportation Planning, McGraw-Hill Series in Transportation, McGraw-Hill, New York.

Campbell, A.M. & Savelsbergh, M. (2004), Efficient Insertion Heuristics for Vehicle Routing and Scheduling Problems, Transportation Science, Vol. 38, No. 3, pp. 369–378.

Cervero, R. (1997), Paratransit in America: Redefining Mass Transportation. Praeger Publishers, Westport, Conn.

Chiang, W.-C. & Russell, R.A. (1996), Simulated annealing metaheuristics for the vehicle routing problem with time windows, Ann. of Oper. Res. 63, 3–27.

Cortés, C.E. & Jayakrishnan, R. (2002), Design and Operational Concepts of High-Coverage Point-to-Point Transit System, In Transportation Research Record: Journal of the Transportation Research Board, No. 1783, pp. 178-187.

Crama, Y. & Schyns, M. (2003), Simulated annealing for complex portfolio selection problem, European Journal of Operational Research, Vol. 150, Issue 3, pp. 546-571.

Czech, Z.J. & Czarnas, P. (2002), Parallel Simulated Annealing for the Vehicle Routing Problem with Time Windows, In: 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, Canary Islands–Spain, pp. 376–383.

Dial, R. B. (1995), Autonomous Dial-a-Ride Transit Introductory Overview, Transportation Research, Vol. 3C, No. 5, pp. 261-275.

Fan, W. & Machemehl, R. (2006), Using a Simulated Annealing Algorithm to Solve the Transit Route Network Design Problem, Journal of Transportation Engineering 132, pp. 122–132.

Hadas, Y. & Ceder, A. (2008), Multiagent approach for public transit system based on flexible routes, In Transportation Research Record: Journal of the Transportation Research Board, No. 2063, pp. 89-96.

Herbawi, W.M., Weber, M. (2012), A Genetic and Insertion Heuristic Algorithm for Solving the Dynamic Ridematching Problem with Time Windows, in: Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference, GECCO ’12. ACM, New York, NY, USA, pp. 385–392. doi:10.1145/2330163.2330219

Jung, J., Chow, J., Jayakrishnan, R., and Park, J.Y. (2104), Stochastic dynamic itinerary interception refueling location with queue delay for electric taxi charging stations, In Transportation Research Part C: Emerging Technologies, vol. 40, pp. 123-142

Jung, J. & Jayakrishnan, R. (2011), 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, pp.78-87.

Jung, J. & Jayakrishnan, R. (2014), Simulation Framework for Modeling Large-Scale Flexible Transit Systems, Accepted in Transportation Research Record: Journal of the Transportation Research Board (#14-4756).

Jung, J., Jayakrishnan, R., & Nam, D. (2011), High Coverage Point-to-Point Transit: Local Vehicle Routing Problem with Genetic Algorithms, 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 1285-1290.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983), Optimization by Simulated Annealing, Science, Vol. 220, No. 4598, pp. 671-680.

Lee, K. T., Lin, D. J., & Wu, P. J. (2005), Planning and Design of a Taxipooling Dispatch System, In Transportation Research Record: Journal of the Transportation Research Board, No. 1930, pp. 86-95.

Liu, C.-M., Kao, R.-L., & Wang, A.-H. (1994), Solving Location-Allocation Problems with Rectilinear Distances by Simulated Annealing, The Journal of the Operational Research Society 45, pp. 1304–1315.

Malek, M., Guruswamy, M., Pandya, M., & Owens, H., (1989), Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem, Ann. of Oper. Res 21, pp. 59–84.

Meng, Q. B., Mabu, S., Yu, L., & Hirasawa, K., (2010), A Novel Taxi DispatchTaxi-dispatch System Integrating a Multi-Customer Strategy and Genetic Network Programming, Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol. 14, No. 5, pp. 442-452.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953), Equations of state calculations by fast computing machines, Journal of Chemical Physics, Vol. 21, No. 6, pp. 1087-1092.

Ng, M.W., Park J., & Wallder, S.T. (2010), A Hybrid Bilevel Model for the Optimal Shelter Assignment in Emergency Evacuations, Computer-Aided Civil and Infrastructure Engineering 25(8), pp. 547-556.

Psaraftis, H.N. (1988), Dynamic Vehicle Routing Problems, Vehicle Routing: Methods and Studies. B. L. Golden and A. A. Assad (Editors). Elsevier Science Publishers B. V.

Psaraftis, H.N. (1995), Dynamic vehicle routing: Status and prospects, Ann. of Oper. Res. 61, pp. 143–164.

Quadrifoglio, L. & Li, X., (2009), A methodology to derive the critical demand density for designing and operating feeder transit services, Transportation Research Part B: Methodological 43, pp. 922–935.

Ram, J.D., Sreenivas, T.H., & Ganapathy Subramaniam, K., (1996), Parallel Simulated Annealing Algorithms, Journal of Parallel and Distributed Computing, Vol. 37, Issue 2, pp. 207-212.

Santos, D.O. & Xavier, E.C. (2013), Dynamic Taxi and RidesharingRide-sharing: A Framework and Heuristics for the Optimization Problem, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI'13. AAAI Press, Beijing, China, pp. 2885–2891.

Sarma, K.C. and Adeli, H. (2001), "Bi-Level Parallel Genetic Algorithms for Optimization of Large Steel Structures", *Computer-Aided Civil and Infrastructure Engineering*, 16 (5), pp. 295-304.

Seow, K. T., Dang, N. H., & Lee, D. H. (2010), A Collaborative Multiagent Taxi-Dispatch System, IEEE Transactions on Automation Science and Engineering, Vol. 7, Issue 3, pp. 607-616.

Tao, C. C. (2007), Dynamic Taxi-sharing Service Using Intelligent Transportation System Technologies, Wireless Communications, Networking and Mobile Computing, International Conference on. 21-25 Sep, pp. 3209-3212.

Tsukada, N. & Takada, K., (2005), Possibilities of the Large-Taxi Dial-a-ride Transit System Utilizing GPS-AVM, Journal of Eastern Asia Society for Transportation Studies, EASTS, Bangkok, Thailand, 22 pp. 1-10.

Van Breedam, A. (1995), Improvement heuristics for the Vehicle Routing Problem based on simulated annealing, European Journal of Operational Research 86, pp. 480–490.

Zeferino, J.A., Antunes, A.P., & Cunha, M.C., (2009), An Efficient Simulated Annealing Algorithm for Regional Wastewater System Planning, Computer-Aided Civil and Infrastructure Engineering 24, pp. 359–370.

View publication stats