Contents lists available at SciVerse ScienceDirect

Mathematical and Computer Modelling

journal homepage: www.elsevier.com/locate/mcm

Charge scheduling of electric vehicles in highways

Shun-Neng Yang [a,b,*], Wei-Sheng Cheng [b], Yu-Ching Hsu [a], Chai-Hien Gan [a], Yi-Bing Lin [b]

[a] *Information and Communications Research Laboratories, ITRI, Chutung, Hsinchu, 31040, Taiwan, ROC*

[b] *Department of Computer Science, National Chiao Tung University, Hsinchu, 30010, Taiwan, ROC*

ARTICLE INFO

Article history:
Received 21 July 2011
Received in revised form 16 November 2011
Accepted 20 November 2011

Keywords:
Charge scheduling
Charging station selection
Electric vehicle

ABSTRACT

Today, charging stations (CSs) for electric vehicles (EVs) are much less popular than gas stations. Therefore, searching and selecting CSs is an important issue for the drivers of EVs. This paper investigates the EV charging problem. We propose two types of CS-selection algorithms. The first type only utilizes local information of an EV. The second type utilizes the global information obtained through interactions between the EVs and a Global CS-selection (GCS) server through the mobile telecommunications network. Our study indicates that by using the global information (specifically the workload status of each CS), the EVs can be effectively charged with short waiting times at the CSs.

1. Introduction

Electric Vehicles (EVs) provide energy-efficient transport that reduces carbon emission [1–3]. Generally, the electricity for EV driving can last from 85 to 160 km [4]. When an EV travels for a long trip, it must be recharged in the electric charging stations (CSs). At every CS, there are a number of fast-charging poles that are pole-like charging equipments [5]. With the present fast-charging technology, the charging time of an EV typically exceeds 30 min [6,7]. If an EV arrives at a CS whose fast-charging poles are all occupied, it must wait until the EVs in front of the queue have completed their charging. Therefore, in EV recharging, it is important that an EV selects the least loaded CSs (that are not very "busy") to reduce the waiting time.

If an EV has an on-board unit (OBU, a GPS-based navigator), then through GPS and the map installed in the OBU, the EV can be guided to the next CSs for charging. Such CS-selection decision is made based on the location of the EV (obtained from GPS) and the CSs (from a pre-installed map), and the remaining electricity (from the EV meter). However, the information of the OBU is "local", which cannot provide guidance about the waiting time of the EV before it can be charged at a CS.

To minimize the waiting times at the CSs, "global" information (such as the current queue lengths of the CSs) must be provided to the OBUs. Such global information can be maintained by a server in the network, and an OBU can access the information through wireless communications [8].

This paper[1] addresses the CS-selection problem for EV charging on a highway; that is, when an EV drives on a highway for a long distance, how the CSs are selected for recharging with "short" waiting times. We consider the CS-selection algorithms based on local information and global information, and compare their performances in terms of waiting times at the CSs.

2. The CS-selection algorithms

This section takes the Taiwan National Expressway 1 (TNE1) as an example to describe the EV charging problem. Then we describe the CS-selection algorithms based on local or global information.

* Corresponding author at: Information and Communications Research Laboratories, ITRI, Chutung, Hsinchu, 31040, Taiwan, ROC. Tel.: +886 3 591 2286; fax: +886 3 582 0263.

E-mail addresses: takeshi@itri.org.tw, snyang@cs.nctu.edu.tw (S.-N. Yang), wscheng@cs.nctu.edu.tw (W.-S. Cheng), YuChing@itri.org.tw (Y.-C. Hsu), chgan@itri.org.tw (C.-H. Gan), liny@cs.nctu.edu.tw (Y.-B. Lin).

[1] This paper is an extension of the conference version [ICPADS 2011].

0895-7177/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mcm.2011.11.054

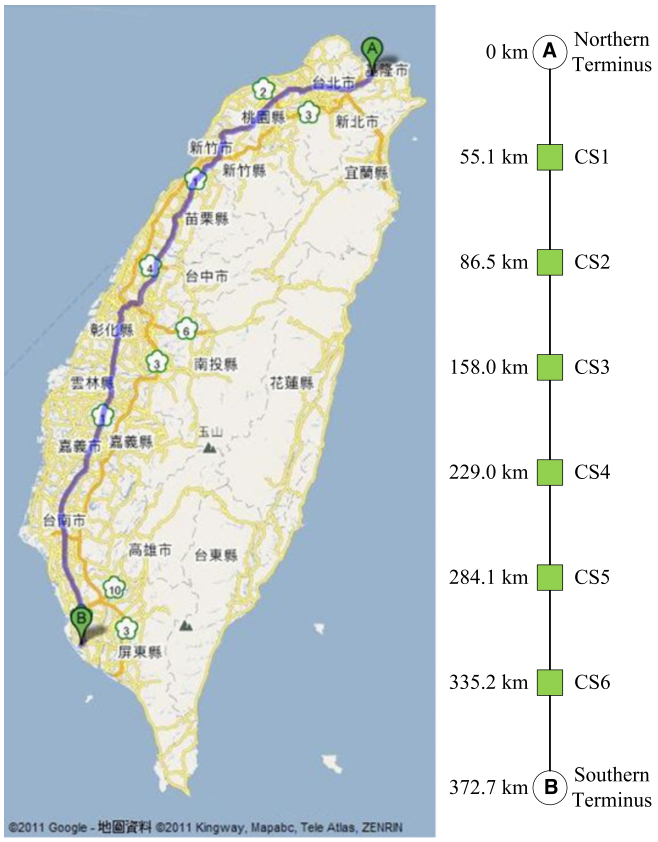


Fig. 1. The route of the Taiwan National Expressway 1 and the intermediate CSs.

Fig. 1 illustrates the route of TNE1. The length of TNE1 is 372.7 km with six intermediate service areas [9]. We assume that the CSs are located at the service areas (one CS per service area). The distance between two CSs ranges from 30 to 70 km. When an EV travels for a long trip (e.g., more than 100 km) through TNE1, it must be recharged in the CSs.

When an EV travels in TNE1, its OBU can select the next CS for charging based on the local information (i.e., the current position of the EV, the remaining electricity, and the distances to the CSs) [10]. From the local information, the OBU compiles a list of reachable CSs, and then select the next CS for charging. Basically, there are three local CS-selection algorithms:

- Local Algorithm 1 (LA1): Shortest-first (that selects the nearest CS).
- Local Algorithm 2 (LA2): Random (that randomly selects a CS).
- Local Algorithm 3 (LA3): Longest-first (that selects the farthest CS).

The problem of CS-selection based on the local information is that, we cannot estimate the workload at a CS (and therefore the waiting time in that CS), and the EV may not select the CS with the shortest waiting time. To resolve this issue, we can utilize the global information (queue lengths of the CSs) through wireless communications (such as WCDMA [11]) between the OBUs and a Global CS-selection (GCS) server in the network, and then use this information to select the CS for charging.

Fig. 2 illustrates two snapshots of a GCS scenario. When an EV arrives at a CS, it reports to the GCS server ((1) in Fig. 2(a)) the charging time required (translated from electricity) by the OBU. Consider the EVs queued at CSs; e.g., CS1 ((2) in Fig. 2(a)) and CS2 ((3) in Fig. 2(a)). The GCS server maintains one queue table for every CS. Each row in the table includes the identifier (ID) of the EV and the required charging time of the EV. In Fig. 2(a), EV1, EV2 and EV3 are queued in CS1, and therefore there are three records in CS1's queue. Similarly, EV4 and EV5 are queued in CS2, and there are two records in CS2's queue. Fig. 2(a) also illustrates that EV6 is driving from CS1 to CS2. A few minutes later, EV1 leaves CS1 and EV6 arrives at CS2 as illustrated in Fig. 2(b). At this moment, EV1's record at CS1 is deleted and a record is created for EV6 in CS2's queue.

With GCS, two CS-selection alternatives are proposed. In Global Algorithm 1 (GA1), when the EV finishes the current charging at station CS*, the next station CS for charging is determined immediately after the EV leaves CS*. The message flow is described as follows (see Fig. 3):

Step 1. The EV sends the CS-selection request to the GCS server.

Step 2. The GCS server computes the predicted waiting time w_p of the EV based on the queue length of each CS. Let T_r be the summation of required charging times in the queue of a CS and N_F be the number of fast-charging poles at a CS,

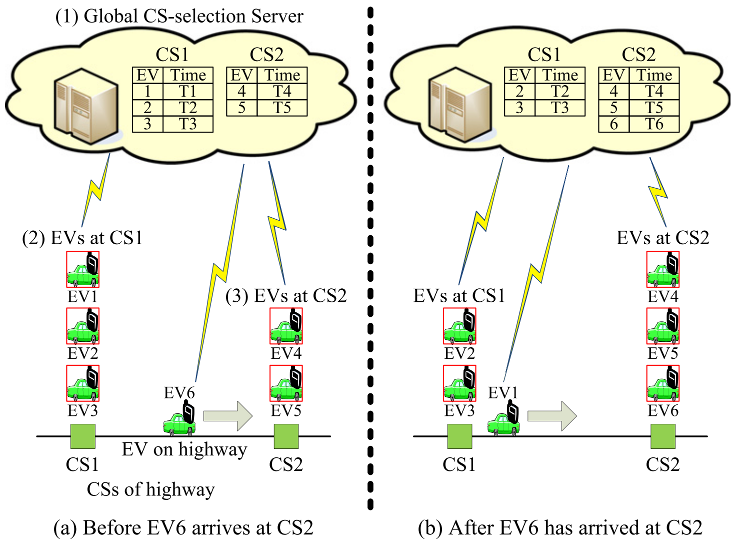


Fig. 2. Two snapshots of a Global CS-selection scenario.

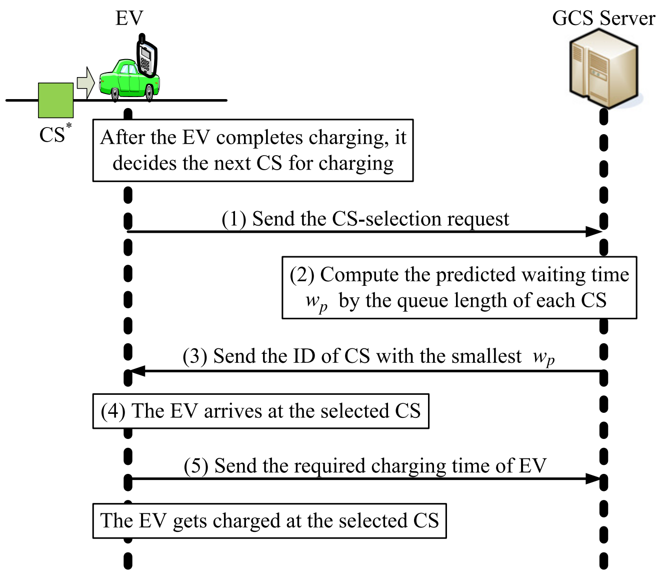

EV
GCS Server
CS*
After the EV completes charging, it decides the next CS for charging
(1) Send the CS-selection request
(2) Compute the predicted waiting time w_p by the queue length of each CS
(3) Send the ID of CS with the smallest w_p
(4) The EV arrives at the selected CS
(5) Send the required charging time of EV
The EV gets charged at the selected CS

Fig. 3. The message flow of Global Algorithm 1 (GA1).

then w_p of the EV at the CS is computed as:

$$w_p = \frac{T_r}{N_F}. \tag{1}$$

In Fig. 2(b), we estimate $T_r = T4 + T5 + T6$ in the queue of CS2.

Step 3. Based on (1), the GCS server selects the CS with the smallest w_p and sends the CS ID to the EV.

Step 4. The EV arrives at the selected CS.

Step 5. The EV sends the required charging time to the GCS server so that the server can compute (1) for the next EV who will query the GCS server for selecting the charging station.

Note that in *Step* 2, the actual T_r value when the EV arrives at CS2 may be different from the estimated value, and (1) may incur inaccuracy.

In Global Algorithm 2 (GA2), the EV and the GCS server exchange messages when the EV is about to arrive at the next charging station CS. The message flow is described as follows (see Fig. 4):

Steps 1–3. Same as GA1.

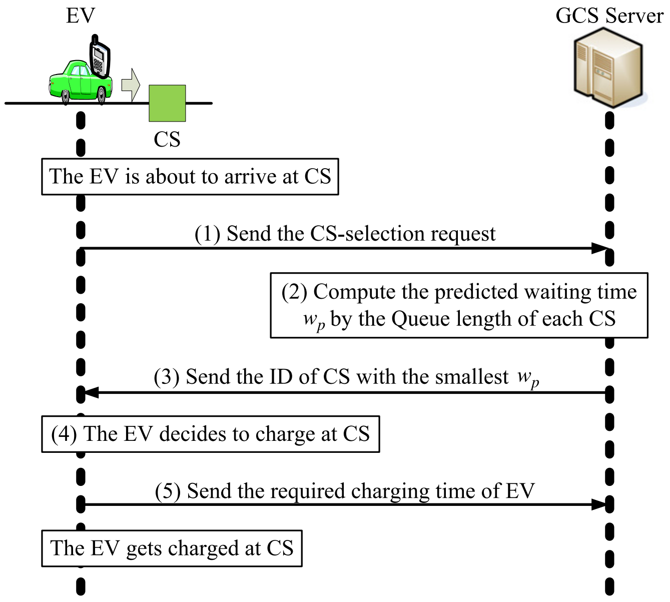

EV
GCS Server
CS
The EV is about to arrive at CS
(1) Send the CS-selection request
(2) Compute the predicted waiting time w_p by the Queue length of each CS
(3) Send the ID of CS with the smallest w_p
(4) The EV decides to charge at CS
(5) Send the required charging time of EV
The EV gets charged at CS

Fig. 4. The message flow of Global Algorithm 2 (GA2).

Step 4. If the selected charging station is CS, then the EV enters CS for charging (*Step* 5 is executed). Otherwise, the EV goes to the next charging station and repeats GA2. For the description purpose, we assume that the EV decides to charge at CS.

Step 5. The EV sends the required charging time to the GCS server as GA1. Then the EV is charged at CS.

3. The simulation model

This section describes the simulation model. For the description purpose, we consider the distance between EVs and CSs instead of the relative locations of a two-dimensional plane. Therefore we use the linear expressway in our simulation model. We assume that each EV moves from the Northern Terminus to the Southern Terminus. In the simulation, the system parameters include the length of highway L, the total number N of EVs, the distance M that a fully-charged EV can drive, the number N_c of CSs on the highway, the number N_F of fast-charging poles at a CS, and the time T_c that an EV spends for charging from 0% to 100% of power.

Several objects are defined. An *EV* object represents an EV with the attributes including the current location l (the distance from the Northern Terminus), the velocity v, the percentage of power left p, the time τ when the EV arrives at the next CS, the total waiting time w (at the CSs), the next CS s for charging, and the number n_s of the CSs visited by the EV.

A *CS* object represents a charging station with the attributes including the location l and the number n_f of the occupied fast-charging poles. It also includes an array of N_F FCP objects (i.e., N_F fast-charging poles) and the queue q_w of waiting EVs.

A *FCP* object represents a fast-charging pole of a CS with two attributes. The first attribute is an *EV* object representing the EV being charged by the fast-charging pole. The t_o attribute represents the charging-over time for the EV.

In this simulation, an event e includes the timestamp t_s, the event type *type*, and the associated *EV* object. Three event types are defined: *Arriving-HW* (an EV enters the highway), *Arriving-CS* (an EV arrives at a CS), and *Charge-Complete* (an EV finishes charging).

The inter-arrival time t_a between two consecutive *Arriving-HW* events e_1 and e_2 is a random number drawn from a random number generator G_A, where $e_2.t_s = e_1.t_s + t_a$. All events are inserted into the event list *e_list*, and are deleted and then processed from *e_list* in the non-decreasing timestamp order. In the simulation, a clock t is maintained to indicate the simulation progress, which is the timestamp of the event being processed.

In the simulation, w_t is the total waiting time of all EVs, and the total waiting time of an EV is $W = w_t/N$. The variable s_t is the total number of visited CSs of all EVs, and the total number of visited CSs of an EV is $S_c = s_t/N$. The average waiting time at a visited CS for an EV is computed as $W_{\text{avg}} = W/S_c$.

In this paper, we use two simulation flowcharts to illustrate how the local and global algorithms work. Specifically, we consider LA2 and GA2. For other algorithms, the flowcharts are similar and the details are omitted.

Fig. 5 illustrates the simulation flowchart for algorithm LA2. Initially, Step 1 sets the highway length L to 372.7 km, the number N of EVs to be simulated is set to 1,000,000, and the distance M is set to 160 km. The number N_c, the time T_c, and the charging pole number N_F are set to 6, 30, and 10 respectively. All temporary variables are initialized to 0. In the simulation, six CS objects $cs(j)$, $1 \leq j \leq 6$, represent the charging stations. The locations $cs(j).l$ are set according to the locations in

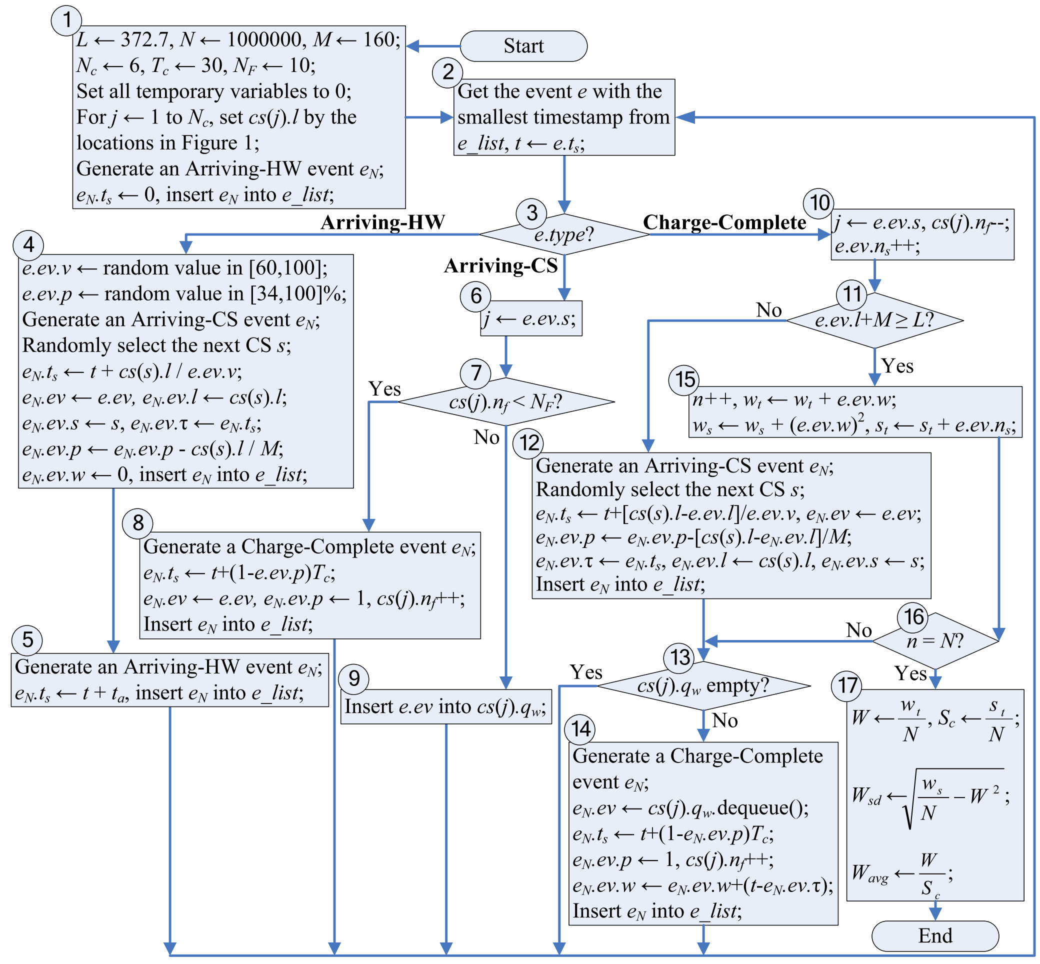

Start
1
$L \leftarrow 372.7$, $N \leftarrow 1000000$, $M \leftarrow 160$; $N_c \leftarrow 6$, $T_c \leftarrow 30$, $N_F \leftarrow 10$; Set all temporary variables to 0; For $j \leftarrow 1$ to N_c, set $cs(j).l$ by the locations in Figure 1; Generate an Arriving-HW event e_N; $e_N.t_s \leftarrow 0$, insert e_N into e_list;
2
Get the event e with the smallest timestamp from e_list, $t \leftarrow e.t_s$;
3
$e.type$?
Arriving-HW
Charge-Complete
Arriving-CS
4
$e.ev.v \leftarrow$ random value in [60,100]; $e.ev.p \leftarrow$ random value in [34,100]%; Generate an Arriving-CS event e_N; Randomly select the next CS s; $e_N.t_s \leftarrow t + cs(s).l / e.ev.v$; $e_N.ev \leftarrow e.ev$, $e_N.ev.l \leftarrow cs(s).l$; $e_N.ev.s \leftarrow s$, $e_N.ev.\tau \leftarrow e_N.t_s$; $e_N.ev.p \leftarrow e_N.ev.p - cs(s).l / M$; $e_N.ev.w \leftarrow 0$, insert e_N into e_list;
5
Generate an Arriving-HW event e_N; $e_N.t_s \leftarrow t + t_a$, insert e_N into e_list;
6
$j \leftarrow e.ev.s$;
7
$cs(j).n_f < N_F$?
Yes
No
8
Generate a Charge-Complete event e_N; $e_N.t_s \leftarrow t+(1-e.ev.p)T_c$; $e_N.ev \leftarrow e.ev$, $e_N.ev.p \leftarrow 1$, $cs(j).n_f$++; Insert e_N into e_list;
9
Insert $e.ev$ into $cs(j).q_w$;
10
$j \leftarrow e.ev.s$, $cs(j).n_f$--; $e.ev.n_s$++;
11
$e.ev.l+M \geq L$?
No
Yes
15
n++, $w_t \leftarrow w_t + e.ev.w$; $w_s \leftarrow w_s + (e.ev.w)^2$, $s_t \leftarrow s_t + e.ev.n_s$;
12
Generate an Arriving-CS event e_N; Randomly select the next CS s; $e_N.t_s \leftarrow t+[cs(s).l-e.ev.l]/e.ev.v$, $e_N.ev \leftarrow e.ev$; $e_N.ev.p \leftarrow e_N.ev.p-[cs(s).l-e_N.ev.l]/M$; $e_N.ev.\tau \leftarrow e_N.t_s$, $e_N.ev.l \leftarrow cs(s).l$, $e_N.ev.s \leftarrow s$; Insert e_N into e_list;
16
$n = N$?
No
Yes
13
$cs(j).q_w$ empty?
Yes
No
14
Generate a Charge-Complete event e_N; $e_N.ev \leftarrow cs(j).q_w$.dequeue(); $e_N.t_s \leftarrow t+(1-e_N.ev.p)T_c$; $e_N.ev.p \leftarrow 1$, $cs(j).n_f$++; $e_N.ev.w \leftarrow e_N.ev.w+(t-e_N.ev.\tau)$; Insert e_N into e_list;
17
$W \leftarrow \frac{w_t}{N}$, $S_c \leftarrow \frac{s_t}{N}$; $W_{sd} \leftarrow \sqrt{\frac{w_s}{N} - W^2}$; $W_{avg} \leftarrow \frac{W}{S_c}$;
End

Fig. 5. The simulation flowchart for algorithm LA2.

Fig. 1. The first *Arriving-HW* event e_N is generated and $e_N.t_s$ is set to 0. The event e_N is inserted into the event list *e_list*. Step 2 retrieves an event e from *e_list* and the current time t is set to $e.t_s$. Step 3 checks $e.type$. If it is *Arriving-HW*, then Step 4 is executed. If $e.type$ is *Arriving-CS*, then Step 6 is processed. Otherwise, $e.type$ is *Charge-Complete*, and the simulation flow proceeds to Step 10.

Step 4 processes the *Arriving-HW* event e. It sets the velocity $e.ev.v$ to a random value between 60 and 100 km/h, and the EV power $e.ev.p$ is set to another random value between 34% and 100%. The value 34% is the minimal required power to drive to CS1 (i.e., $55.1/160 \fallingdotseq 34\%$). An *Arriving-CS* event e_N is generated and the next CS s is randomly selected (according to LA2). The timestamp $e_N.t_s$ is set to $t + cs(s).l/e.ev.v$, where $cs(s).l/e.ev.v$ is the required time that $e.ev$ drives to $cs(s).l$. Variable $e_N.ev$ is set to $e.ev$. The location $e_N.ev.l$ is updated to $cs(s).l$. The next CS $e_N.ev.s$ for charging is set to s. The time $e_N.ev.\tau$ is set to $e_N.t_s$. The power $e_N.ev.p$ is set to $e_N.ev.p - cs(s).l/M$, where the linear equation $cs(s).l/M$ represents the power consumption for $e_N.ev$ to drive to $cs(s).l$ [12]. The time $e_N.ev.w$ is set to 0. Then e_N is inserted into *e_list*. Step 5 generates the next *Arriving-HW* event e_N and $e_N.t_s$ is set to $t + t_a$. The event is inserted into *e_list*.

Step 6 simulates the *Arriving-CS* event e; that is, the EV arrives at CS j, where $j = e.ev.s$. Step 7 checks if $cs(j).n_f$ is less than N_F. If so, there are free fast-charging poles that can serve the EV. Step 8 generates a *Charge-Complete* event e_N and $e_N.t_s$ is set to $t + (1 - e.ev.p)T_c$, where $(1 - e.ev.p)T_c$ is the required time to charge $e.ev$. Variable $e_N.ev$ is set to $e.ev$, and $e_N.ev.p$ is set to 1. Event e_N is inserted into *e_list* and the number $cs(j).n_f$ is incremented by one. On the other hand, if all charging poles are occupied at Step 7, then Step 9 inserts $e.ev$ in the waiting queue $cs(j).q_w$. The simulation proceeds to Step 2.

For a *Charge-Complete* Event, Step 10 identifies CS j that completes the EV charging; that is, $j = e.ev.s$. The number $cs(j).n_f$ is decremented by one and the number $e.ev.n_s$ is incremented by one. Step 11 checks if $e.ev.l + M$ is $\geq L$. If so, the EV has left highway and Step 15 is executed to collect the output statistics for $e.ev$. Otherwise, Step 12 generates an *Arriving-CS* event e_N and randomly selects the next CS s for charging. Variable $e_N.t_s$ is set to $t + [cs(s).l - e.ev.l]/e.ev.v$, $e_N.ev$ is set to $e.ev$, and $e_N.ev.p$ is set to $e_N.ev.p - [cs(s).l - e_N.ev.l]/M$. Step 13 checks if $cs(j).q_w$ is empty. If so, then Step 2 is executed. Otherwise, the charging station serves the next waiting EV $e_N.ev$ in queue $cs(j).q_w$. Step 14 generates a *Charge-Complete* event e_N for that EV. The time $e_N.ev.w$ is updated to $e_N.ev.w + (t - e_N.ev.\tau)$, where $(t - e_N.ev.\tau)$ is the time $e_N.ev$ spent in queue $cs(j).q_w$.

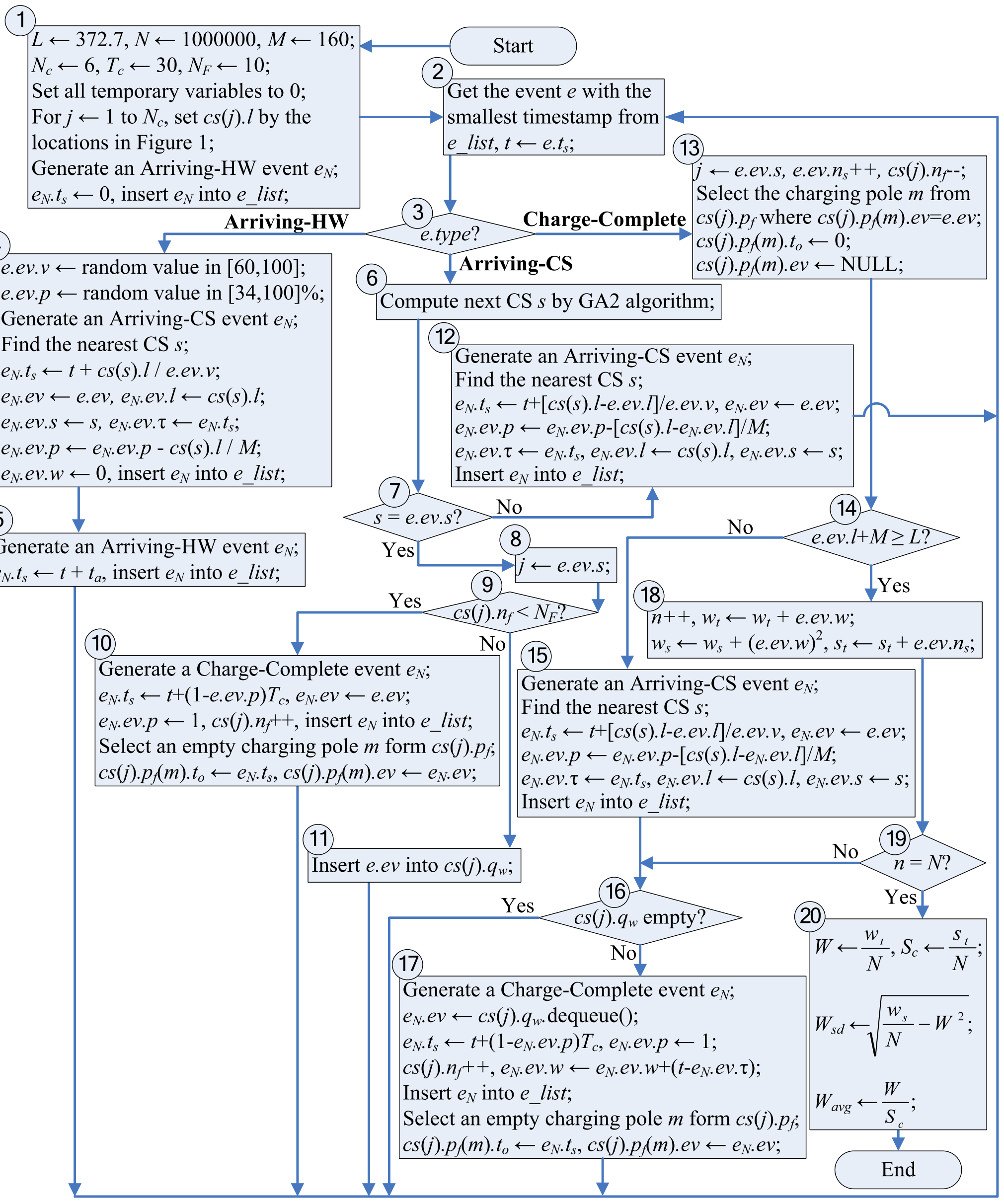

Start
1
L ← 372.7, N ← 1000000, M ← 160;
N_c ← 6, T_c ← 30, N_F ← 10;
Set all temporary variables to 0;
For j ← 1 to N_c, set cs(j).l by the locations in Figure 1;
Generate an Arriving-HW event e_N;
e_N.t_s ← 0, insert e_N into e_list;
2
Get the event e with the smallest timestamp from e_list, t ← e.t_s;
3
e.type?
Arriving-HW
Charge-Complete
Arriving-CS
4
e.ev.v ← random value in [60,100];
e.ev.p ← random value in [34,100]%;
Generate an Arriving-CS event e_N;
Find the nearest CS s;
e_N.t_s ← t + cs(s).l / e.ev.v;
e_N.ev ← e.ev, e_N.ev.l ← cs(s).l;
e_N.ev.s ← s, e_N.ev.τ ← e_N.t_s;
e_N.ev.p ← e_N.ev.p - cs(s).l / M;
e_N.ev.w ← 0, insert e_N into e_list;
5
Generate an Arriving-HW event e_N;
e_N.t_s ← t + t_a, insert e_N into e_list;
6
Compute next CS s by GA2 algorithm;
7
s = e.ev.s?
No
Yes
8
j ← e.ev.s;
9
cs(j).n_f < N_F?
Yes
No
10
Generate a Charge-Complete event e_N;
e_N.t_s ← t+(1-e.ev.p)T_c, e_N.ev ← e.ev;
e_N.ev.p ← 1, cs(j).n_f++, insert e_N into e_list;
Select an empty charging pole m form cs(j).p_f;
cs(j).p_f(m).t_o ← e_N.t_s, cs(j).p_f(m).ev ← e_N.ev;
11
Insert e.ev into cs(j).q_w;
12
Generate an Arriving-CS event e_N;
Find the nearest CS s;
e_N.t_s ← t+[cs(s).l-e.ev.l]/e.ev.v, e_N.ev ← e.ev;
e_N.ev.p ← e_N.ev.p-[cs(s).l-e_N.ev.l]/M;
e_N.ev.τ ← e_N.t_s, e_N.ev.l ← cs(s).l, e_N.ev.s ← s;
Insert e_N into e_list;
13
j ← e.ev.s, e.ev.n_s++, cs(j).n_f--;
Select the charging pole m from cs(j).p_f where cs(j).p_f(m).ev=e.ev;
cs(j).p_f(m).t_o ← 0;
cs(j).p_f(m).ev ← NULL;
14
e.ev.l+M ≥ L?
No
Yes
15
Generate an Arriving-CS event e_N;
Find the nearest CS s;
e_N.t_s ← t+[cs(s).l-e.ev.l]/e.ev.v, e_N.ev ← e.ev;
e_N.ev.p ← e_N.ev.p-[cs(s).l-e_N.ev.l]/M;
e_N.ev.τ ← e_N.t_s, e_N.ev.l ← cs(s).l, e_N.ev.s ← s;
Insert e_N into e_list;
16
cs(j).q_w empty?
Yes
No
17
Generate a Charge-Complete event e_N;
e_N.ev ← cs(j).q_w.dequeue();
e_N.t_s ← t+(1-e_N.ev.p)T_c, e_N.ev.p ← 1;
cs(j).n_f++, e_N.ev.w ← e_N.ev.w+(t-e_N.ev.τ);
Insert e_N into e_list;
Select an empty charging pole m form cs(j).p_f;
cs(j).p_f(m).t_o ← e_N.t_s, cs(j).p_f(m).ev ← e_N.ev;
18
n++, w_t ← w_t + e.ev.w;
w_s ← w_s + (e.ev.w)^2, s_t ← s_t + e.ev.n_s;
19
n = N?
No
Yes
20
W ← w_t/N, S_c ← s_t/N;
W_sd ← √(w_s/N − W^2);
W_avg ← W/S_c;
End

Fig. 6. The simulation flowchart for algorithm GA2.

At Step 11, if the EV leaves the highway, Step 15 increments n by one. The total time w_t is increased to $w_t + e.ev.w$, the sum-of-square of time w_s is increased to $w_s + (e.ev.w)^2$, and the total number s_t is increased to $s_t + e.ev.n_s$. Step 16 checks the number n of EVs that leave the highway. If n equals to N, then Step 17 is executed. Otherwise, the simulation flow goes to Step 13. Step 17 computes the outputs W, W_{sd} (standard deviation of W), S_c, and W_{avg}.

The simulation flowcharts of other proposed local algorithms are similar to that in Fig. 5. The main difference of LA1, LA2, and LA3 is the selection algorithm of the next CS. In LA1, we select the nearest CS instead of randomly selecting one in Step 4 and 12 in Fig. 5. In LA3, we select the farthest CS instead of randomly selecting one in Step 4 and 12 in Fig. 5.

Fig. 6 illustrates the simulation flowchart for algorithm GA2. Many steps in Fig. 6 are similar to those in Fig. 5. We only describe the different parts as follows:

Step 4 always selects the nearest CS s instead of randomly selecting one. The EV will determine if it wants to be charged at Step 6. That is, Step 6 executes GA2 algorithm (see Fig. 4). Step 7 checks if s is equal to $e.ev.s$. If so, goes to Step 8 to simulate that the EV is charged at s. Otherwise, the simulation flow proceeds to Step 12, where the EV will repeat GA2 when it arrives at the next nearest CS. Step 10 generates the *Charge-Complete* event which is similar to Step 8 in Fig. 5. In addition, an empty charging pole m from the array $cs(j).p_f$ is selected to charge the EV. The variable $cs(j).p_f(m).t_o$ is set to $e_N.t_s$ and

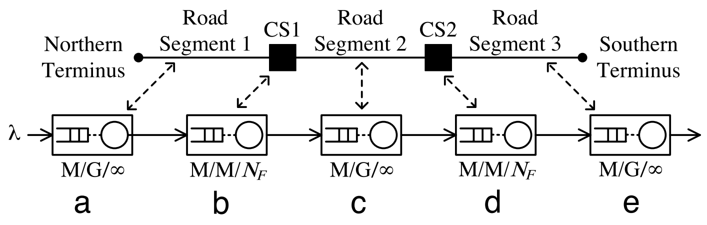


Fig. 7. The queueing model of LA1 with two CSs.

Table 1
The discrepancies of W at CS1 between the analytic and the simulation results ($N_F = 1$).

$\lambda/(N_F\mu)$	0.1 (%)	0.2 (%)	0.3 (%)	0.4 (%)	0.5 (%)	0.6 (%)	0.7 (%)	0.8 (%)	0.9 (%)
Discrepancy of $\lambda = 0.42$	0.02	0.09	0.03	0.17	0.29	0.24	0.26	0.47	0.45
Discrepancy of $\lambda = 0.6$	0.05	0.09	0.1	0.05	0.15	0.22	0.42	0.27	0.26
Discrepancy of $\lambda = 0.78$	0.07	0.07	0.16	0.04	0.1	0.13	0.35	0.35	0.40

$cs(j).p_f(m).ev$ is set to $e_N.ev$. Step 12 generates an *Arriving-CS* event e_N as Step 12 in Fig. 5, except that $e_N.ev.s$ is set to the next nearest CS s.

Step 13 is similar to Step 10 in Fig. 5, which identifies CS $j = e.ev.s$. The number $e.ev.n_s$ is incremented by one and the number $cs(j).n_f$ is decremented by one. In addition, Step 13 finds the charging pole m that is charging the EV; that is, $cs(j).p_f(m).ev = e.ev$. After the EV has left the CSs, the charging pole m is free. Therefore, variables $cs(j).p_f(m).t_o$ and $cs(j).p_f(m).ev$ are set to 0 and NULL respectively. Step 17 is similar to Step 14 in Fig. 5, which generates a *Charge-Complete* event e_N for the next waiting EV $e_N.ev$ in queue $cs(j).q_w$. This EV will occupy charging pole m.

The simulation flowchart of GA1 is similar to that in Fig. 6 except that it skips Steps 6, 7, and 12 (i.e., in GA1 algorithm, Step 3 directly proceeds to Step 8 when *e.type* is *Arriving-CS*).

4. Validation of the simulation model

This section describes an analytic model of highway charging stations to validate the simulation model described in Section 3. For the description purpose, we consider LA1 algorithm and use it to simply explain the validation process. Fig. 7 illustrates how the simulation of LA1 algorithm can be mapped to a queueing network.

For simplicity, we only consider two charging stations, where the highway starts from the Northern Terminus. After an 85 km road segment (Road Segment 1), the EV will arrive at the charging station CS1. Then after another 85 km road segment (Road Segment 2), the EV will arrive at CS2. Then after the last 85 km road segment (Road Segment 3), the EV will leave the highway from the Southern Terminus. According to LA1, the EV is always charged when it arrives at a charging station. The above system can be modeled by a tandem queueing network where a road segment is modeled by an $M/G/\infty$ queue (Fig. 7(a), (c), and (e)), and a charging station is modeled by an $M/M/N_F$ queue (Fig. 7(b) and (d)), where N_F is the number of fast-charging poles.

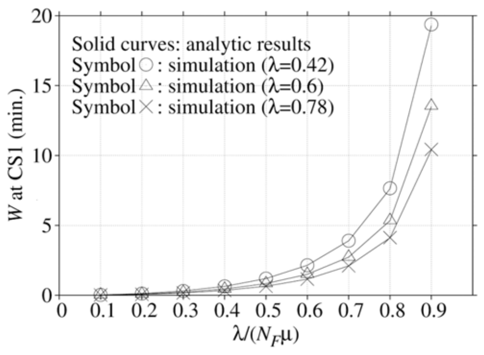
The arrival process of EVs at the Northern Terminus is Poisson with the rate λ. Therefore, Road Segment 1 is an $M/G/\infty$ queue with arrival rate λ and the service times have a uniform distribution with the mean $1/\mu^*$. From the queueing theory [13], the departure process of the Road Segment 1 queue is also Poisson with the rate λ in steady state, which is the arrival process of the CS1 queue. Our example considers $N_F = 1, 3$ for CS1, and the service time is exponentially distributed with the mean $1/\mu$. Thus the CS1 can be modeled as an $M/M/N_F$ queue. From Burke's theorem [14], the departure process of the CS1 queue is also Poisson with the rate λ. The queues that model Road Segments 2 and 3 are the same as that for Road Segment 1, and the CS2 queue is the same as that for CS1.

In our experiments, $\lambda = 0.42, 0.6, 0.78$. The utilization $\lambda/(N_F\mu)$ is set in the range $0.1 \leq \lambda/(N_F\mu) \leq 0.9$, which means that $0.16 \leq 1/\mu \leq 7.8$ (the minimum value of $1/\mu = 0.42/(0.9 \times 3) \fallingdotseq 0.16$ and the maximum value of $1/\mu = 0.78/(0.1 \times 1) = 7.8$). The expected time to travel a road segment is $1/\mu^* = 63.75$ min, which is selected for the following reason: the velocity of each EV is uniformly distributed between 60 and 100 km/h with the mean value 80 km/h and thus the service time is also uniformly distributed with the mean value $85/80 \times 60 = 63.75$ min.

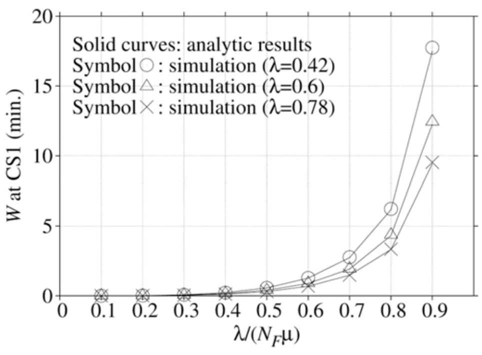
Fig. 8(a) and (b) plot the average waiting times W at CS1 for analytic (the solid curves) and simulation (symbols $\circ$, $\triangle$, and $\times$ for $\lambda = 0.42, 0.6, 0.78$ respectively) results, where $N_F = 1$ and 3, respectively. The discrepancies between the analytic and the simulation results are within 0.5% for $N_F = 1$ (see Table 1) and within 1.5% for $N_F = 3$ (see Table 2).

Fig. 9(a) and (b) plot CS2's W curves for analytic and simulation results ($N_F = 1$ and 3). The discrepancies between the analytic and simulation results are within 0.5% for $N_F = 1$ (see Table 3) and within 1.1% for $N_F = 3$ (see Table 4).

From Tables 1–4, it is clear that the simulation is consistent with the analytic model.



(a) $N_F = 1$.

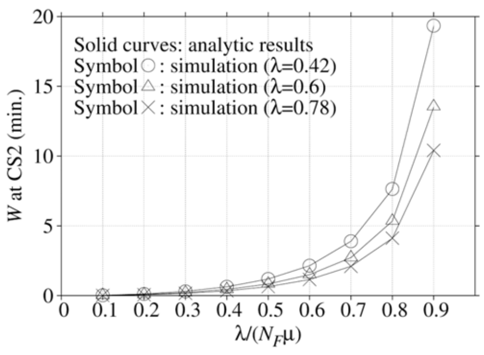


(b) $N_F = 3$.

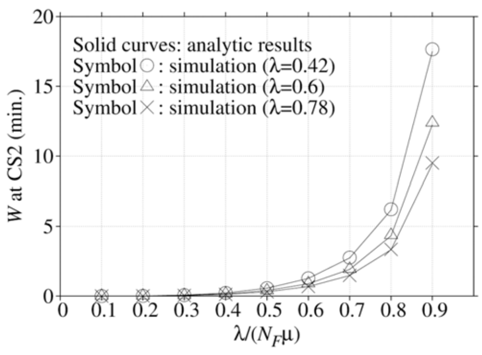
Fig. 8. The W curves at CS1 for the analytic and the simulation results ($N_F = 1, 3$).

Table 2
The discrepancies of W at CS1 between the analytic and the simulation results ($N_F = 3$).

$\lambda/(N_F\mu)$	0.1 (%)	0.2 (%)	0.3 (%)	0.4 (%)	0.5 (%)	0.6 (%)	0.7 (%)	0.8 (%)	0.9 (%)
Discrepancy of $\lambda = 0.42$	0.01	0.13	0.1	0.4	0.63	0.88	0.96	0.99	1.32
Discrepancy of $\lambda = 0.6$	0.31	0.04	0.12	0.22	0.44	0.81	0.7	0.95	1.42
Discrepancy of $\lambda = 0.78$	0.06	0.03	0.26	0.03	0.32	0.61	0.81	0.77	1.07



(a) $N_F = 1$.



(b) $N_F = 3$.

Fig. 9. The W curves at CS2 for the analytic and the simulation results ($N_F = 1, 3$).

Table 3
The discrepancies of W at CS2 between the analytic and the simulation results ($N_F = 1$).

$\lambda/(N_F\mu)$	0.1 (%)	0.2 (%)	0.3 (%)	0.4 (%)	0.5 (%)	0.6 (%)	0.7 (%)	0.8 (%)	0.9 (%)
Discrepancy of $\lambda = 0.42$	0.02	0.07	0.03	0.02	0.11	0.17	0.15	0.33	0.3
Discrepancy of $\lambda = 0.6$	0.02	0.1	0.08	0.04	0.17	0.16	0.09	0.17	0.32
Discrepancy of $\lambda = 0.78$	0.01	0.13	0.02	0.04	0.03	0.08	0.21	0.41	0.22

Table 4
The discrepancies of W at CS2 between the analytic and the simulation results ($N_F = 3$).

$\lambda/(N_F\mu)$	0.1 (%)	0.2 (%)	0.3 (%)	0.4 (%)	0.5 (%)	0.6 (%)	0.7 (%)	0.8 (%)	0.9 (%)
Discrepancy of $\lambda = 0.42$	0.52	0.13	0.2	0.45	0.44	0.63	0.86	0.95	0.82
Discrepancy of $\lambda = 0.6$	0.67	0.01	0.05	0.16	0.36	0.72	0.69	0.98	1.04
Discrepancy of $\lambda = 0.78$	0.05	0.14	0.26	0.04	0.31	0.68	0.69	0.88	0.86

5. Performance evaluation

This section conducts performance evaluation for the proposed algorithms. In the simulation experiments, the arrivals of EVs are a Poisson process with the mean λ. We note that the average vehicle traffic of Sijhih at TNE1 in the southward direction is equivalent to 12 cars per minute [15]. If we assume that 3.5%–6.5% of the vehicles are EVs, then the range of EVs' arrival rate per minute can be reasonably set from 0.42 (=12 × 3.5%) to 0.78 (=12 × 6.5%) in the experiments. The

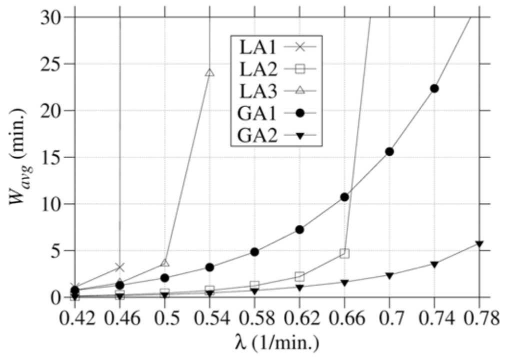


Fig. 10. The W_{avg} performance.

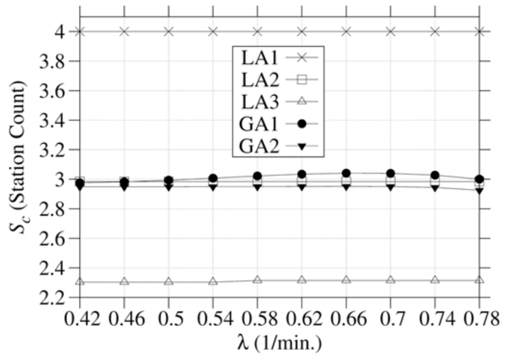


Fig. 11. The S_c performance.

velocity of EV is uniformly distributed between 60 and 100 km/h, which are the lower and the upper speed limits of TNE1. The initial power of EV is uniformly distributed between 34% and 100%.

Fig. 10 compares the average waiting time W_{avg} for the proposed algorithms. The figure indicates the trivial result that W_{avg} increases as λ increases. The W_{avg} values for LA1 and LA3 are larger than those for other algorithms because these two local algorithms select specific CSs (the nearest and the farthest CSs) for charging. These CSs have higher workloads, which result in long queues. W_{avg} for LA1 is larger than that for LA3 because EVs in LA1 visit more CSs than those in LA3 do. LA2 randomly selects the CSs for next charging, which uniformly distributes workloads to the CSs and thus has the best performance among the local algorithms. The performances of global algorithms are better than local algorithms (when λ is higher than 0.66) due to the fact that they utilize the queue length information of the CSs to distribute the workload. GA2 algorithm is better than GA1 because the queue length information of the nearest CS in GA2 is more accurate than that in GA1. For the range of λ we investigated, GA2 has the best W_{avg} performance among all algorithms. The advantage of the global algorithms over the local ones becomes very significant when λ is large.

Fig. 11 illustrates the S_c (the number of CSs visited by an EV) performance. Since every EV in LA1 visits every CS, the S_c value of LA1 is the largest. On the other hand, every EV in LA3 always selects the farthest CS. Thus LA3 has the smallest S_c value. The S_c values of other algorithms (LA2, GA1, and GA2) are between those for LA1 and LA3. The figure also indicates that S_c of GA2 is smaller than GA1 and LA2. Note that for the total waiting time W (i.e., $W = W_{\text{avg}} \times S_c$), we have:

$$W(\text{LA1}) > W(\text{LA3}) > W(\text{GA1}) > W(\text{LA2}) > W(\text{GA2}) \quad \text{for } \lambda \le 0.66 \quad \text{and}$$
$$W(\text{LA1}) > W(\text{LA3}) > W(\text{LA2}) > W(\text{GA1}) > W(\text{GA2}) \quad \text{for } \lambda > 0.66.$$

Fig. 12 compares the standard deviations W_{sd} among these algorithms. The figure indicates that W_{sd} (GA2) has the smallest value among all algorithms, and therefore GA2 is better than other algorithms.

In GA2, an EV always interacts with the GCS server when it is about to arrive at a CS. If the EV decides to charge at a CS, then three messages are exchanged (see *Steps* 1, 3, and 5 in Fig. 4). Otherwise, two messages are exchanged (see *Steps* 1 and 3 in Fig. 4). Our simulation indicates that in GA2, on the average, an EV will exchange 11.74 messages with the GCS server when it travels through the TNE1. On the other hand, in GA1, only 9.04 messages are exchanged between an EV and the GCS server in GA1. Therefore the communication cost of GA2 is 1.3 times of GA1. To conclude, GA2 achieves better waiting time performance at the cost of slightly higher communication cost.

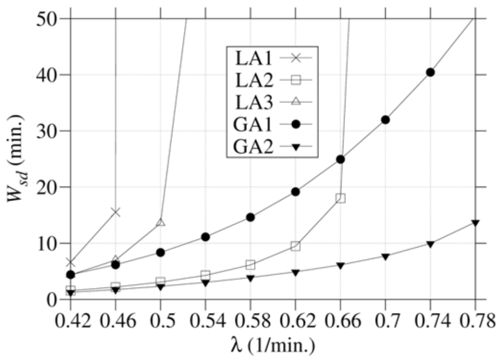


Fig. 12. The W_{sd} performance.

6. Conclusions

This paper proposed two types of CS-selection algorithms for an EV to charge electricity when driving on a highway. The first type only utilizes local information of an EV. The second type utilizes the global information obtained from a Global CS-selection (GCS) server through the mobile telecommunications network. We have conducted simulation experiments to investigate the waiting time performance for these algorithms. Our experiments indicate that the global algorithms have better waiting time performance than the local algorithms. The advantage of the global algorithms over the local ones becomes very significant when the arrivals of EVs are large.

In the future, we will consider intelligent GCS that can more accurately predict the waiting time when an EV actually arrives at a CS, say, 30 min later. Such prediction is not trivial, because other EVs who may arrive at the CS earlier need to be identified by the GCS server, and complicated computation is required.

Acknowledgments

This paper was supported in part by the ITRI/NCTU JRC Research Project under A352BA4111, the ITRI Advanced Research Program under A301AR2400 and A301AR2F00, the National Science Council (NSC) Project under NSC 100-2219-E-009-006, and the Technology Development Program of the Ministry of Economic Affairs (MoEA TDP), Taiwan.

References

[1] US Fuel Economy, Electric vehicles. [Online] Available: http://www.fueleconomy.gov/feg/evtech.shtml, 2011.
[2] N. Schofield, H.T. Yap, C.M. Bingham, A H_2 PEM fuel cell and high energy dense battery hybrid energy source for an urban electric vehicle, in: IEEE International Conference on Electric Machines and Drives, May 15, 2005, pp. 1793–1800.
[3] P.-S. Kim, Cost modeling of battery electric vehicle and hybrid electric vehicle based on major parts cost, in: The Fifth International Conference on Power Electronics and Drive Systems, PEDS 2003, vol. 2, November 17–20, 2003, pp. 1295–1300.
[4] Automotive Research & Testing Center (ARTC), News release of introduction to intelligent electric vehicle: i-EV. [Online] Available: http://www.artc.org.tw/chinese/06_news/down.aspx?file_name=tw_media_news_339796265.doc&file_value=3, October 27, 2010.
[5] D. Aggeler, F. Canales, H.Z.-D.L. Parra, A. Coccia, N. Butcher, O. Apeldoorn, Ultra-fast DC-charge infrastructures for EV-mobility and future smart grids, in: Innovative Smart Grid Technologies Conference Europe, ISGT Europe, 2010 IEEE PES, October 11–13, 2010, pp. 1–8.
[6] J.K. Nor, Art of charging electric vehicle batteries, in: WESCON/'93, Conference Record, September 28–30, 1993, pp. 521–525.
[7] T. Winkler, P. Komarnicki, G. Mueller, G. Heideck, M. Heuer, Z.A. Styczynski, Electric vehicle charging stations in magdeburg, in: IEEE Vehicle Power and Propulsion Conference, VPPC, September 2009, pp. 60–65.
[8] C.-C. Huang-Fu, C.-L. Chen, Y.-B. Lin, Location tracking for WAVE unicast service, in: Vehicular Technology Conference, VTC 2010-Spring, 2010 IEEE 71st, May 16–19, 2010, pp. 1–5.
[9] Service Center of Taiwan National Expressway 1, Guide of Taiwan national expressway 1. [Online] Available: http://www.n1.area.com.tw/, 2003.
[10] K. Imai, T. Ashida, Y. Zhang, S. Minami, EV range extender: Better mileage than plug-in hybrid?, VPPC'08, in: Vehicle Power and Propulsion Conference, 2008, IEEE, 2008, September 3–5, pp. 1–3.
[11] Y.-B. Lin, A.-C. Pang, Wireless and Mobile All-IP Networks, John Wiley & Sons, 2005.
[12] O. Sundström, C. Binding, Optimization methods to plan the charging of electric vehicle fleets, in: Proceedings of the International Conference on Control, Communication and Power Engineering, Chennai, India, July 28–29, 2010.
[13] N.M. Mirasol, The output of an $M/G/\infty$ queuing system is Poisson, Operations Research 11 (2) (1963) 282–284.
[14] C.-H. Ng, B.-H. Soong, Queueing Modelling Fundamentals: With Applications in Communication Networks, second ed., John Wiley & Sons, 2008.
[15] Ministry of Transportation and Communications, Monthly statistics of transportation and communications. [Online] Available: http://www.motc.gov.tw/motchypage/monthly_ebook/10004book.pdf, April 2011.