Electric Vehicle Charging Station Placement: Formulation, Complexity, and Solutions

Article *in* IEEE Transactions on Smart Grid · October 2013

DOI: 10.1109/TSG.2014.2344684 · Source: arXiv

3 authors, including:

Albert Lam
The University of Hong Kong
62 PUBLICATIONS **1,165** CITATIONS

SEE PROFILE

Electric Vehicle Charging Station Placement: Formulation, Complexity, and Solutions

Albert Y.S. Lam, Yiu-Wing Leung, and Xiaowen Chu

***Abstract*—In order to ensure environmental sustainability, many countries will electrify their transportation systems in their future smart city plans. The number of electric vehicles (EVs) running in a city will grow significantly. There are many ways to recharge EVs' batteries and charging stations will be considered as the main source of energy. The locations of charging stations are critical; they should not only be pervasive enough such that an EV anywhere can easily access a charging station within its driving range, but also widely spread so that EVs can cruise around the whole city upon recharging. Based on these new perspectives, we formulate the Electric Vehicle Charging Station Placement Problem (EVCSPP) in this paper. We prove that the problem is NP-hard. We also propose four solution methods to tackle EVCSPP and evaluate their performance on various artificial and practical cases. As verified by the simulation results, the methods have their own characteristics and they are suitable for different situations depending on the requirements for solution quality, algorithmic efficiency, problem size, nature of the algorithm, and existence of system prerequisite.**

***Index Terms*—Charging station, electric vehicle, location, smart city.**

I. Introduction

DUE to the world's shortage of fossil fuels, nations compete to secure enough reserves of natural resources for sustainability. Seeking alternative energy sources becomes crucial to a nation's future development. One of the major fossil fuel consumptions is transportation. Many daily heavily demanded vehicles are powered by gasoline. A major consequence of burning fossil fuels is the release of tremendous amount of harmful gases, which partially constitutes the global warming effect and deteriorates people's health. Electricity is considered as the most universal form of energy, which can be transformed from and to another form effectively. By converting the endurable renewable energy, like solar and wind energies, to electricity, we can manipulate energy in a much cleaner manner. Electrification of transportation, like deployment of electric vehicles (EVs), can not only alleviate our demand on fossil fuels, but also foster a better environment for living. Therefore, EVs will become the major components in the future transportation system.

EVs take the central role in this paper and they have been being studied actively since the boom of the smart grid. Incorporating EVs into an existing self-contained transportation system is challenging. Solely expanding the population of EVs in a city without enough road connections and corresponding charging and parking infrastructure will suppress the practicability of EVs due to their limiting moving ranges. Moreover, existing gas stations are primarily designed for gas refueling; combining charging infrastructure with the conventional gas stations may not be appropriate as the relatively longer charging process will saturate the limited space of the gas stations. We need to carefully plan EV charging facilities to modernize the transportation system. To be precise, we study how EVs will be integrated into the transportation system seamlessly with a focus on charging stations and this will help make our cities "smart". To do this, we study the Electric Vehicle Charging Station Placement Problem (EVCSPP) by finding the best locations to construct charging stations in a city.

Technology advances rapidly while the smart city plan takes a much longer timespan. The plan should cater for the residents and thus we focus on the long-term human aspects rather than the technological ones. An EV should always be able to access a charging station within its capacity anywhere in the city. Charging stations should be built widely enough such that the moving range of an EV can be extended to every corner of the city by having the EV recharged at a charging station available nearby. We study the locations where charging stations should be constructed in a city such that we can minimize the construction cost with coverage extended to the whole city and fulfillment of drivers' convenience. In this paper, we formulate the problem as an optimization problem, based on the charging station accessibility and coverage in the city. We also study its complexity and propose various methods to solve the problem.

Most of the existing work on EVs is related to studying the operational influence of EVs on the grid, i.e., how power is transferred from and to the grid. Besides charging scheduling [2], in a vehicle-to-grid (V2G) system, hundreds of EVs coordinate to act as a power source selling power back to the grid or to support auxiliary services like regulation. A multi-layer market for V2G energy trading was proposed in [3]. The market price was settled via double auction and the mechanism could maximize the EVs' revenues. [4] investigated the joint scheduling of EV and unit commitment and this allowed us to optimize the system's total running cost with the presence of EVs. [5] discussed the incorporation of photovoltaic (PV) equipment into charging stations. It considered that charging facilities equipped with PV panels and the stored solar energy, together with the power requested from the grid, can be used to power EVs. [6] and [7] studied the impact of EV charging to the performance of power distribution networks with the presence of charging stations, which can represent rapid heavy loads. [6] illustrated the effect of fast-charging EVs in terms

A preliminary version of this paper was presented in [1].

The authors are with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong (e-mail: {ayslam, ywleung, chxw}@comp.hkbu.edu.hk).

of power-flow, short-circuit and protection while [7] proposed a new smart load management strategy to coordinate EVs for peak load shaving, power loss minimization, and voltage profile improvement. However, this paper is dedicated to studying the locations for building charging stations, which is an important aspect in the smart city plan.

Both [8] and [9] investigated the location and sizing issues of charging stations; [8] handled the two issues separately while [9] considered a joint optimization for both. With consideration of environmental factors (e.g., load locations, load balance, power quality, etc.) and service radius of charging stations, candidate sites in [8] were selected with a two-step screening method instead of optimization. [9] constructed an optimization problem in which various kinds of costs (including construction, operating, and charging costs) were minimized with traffic flow and charging requirement constraints. [10] determined the charging station locations with real-world public parking information of Seattle as inputs. It formed a mixed-integer program (MIP) by minimizing the total access costs to drivers' destinations from charging stations. In operations research, the study of placing facilities, such as gas stations and fire stations, is generally cast as facility location problems [11], e.g., the Maximal Covering Location Problem [12]. It concerned for the distances or times for travel to individual facilities from locations in a graph. However, in this paper, we focus on the long-term issues of charging station placement for smart city planning, where the short-term factors (e.g., instantaneous loads) will be of less importance. To the best of our knowledge, we are the first to study charging station placement from the new perspectives of the drivers' convenience and EVs' accessibility. Other factors, like traffic conditions, may also be taken into account but they are out of the scope of this work.

Our main contributions include formulating the new problem EVCSPP, analyzing its complexity, and proposing several solutions to the problem. The rest of this paper is organized as follows. We formulate the problem in Section II and discuss its complexity in Section III. Section IV presents four solution methods. In Section V, simulation results are provided for performance evaluation and we also compare the solution methods in terms of characteristics and suitability for different situations. Finally we conclude this paper in Section VI.

II. Problem Formulation

A. System model

We model a city with an undirected graph $G = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ and $\mathcal{E}$ denote the sets of possible sites for constructing charging stations and connections between pairs of sites, respectively. Suppose $|\mathcal{N}| = n$. Let $d : \mathcal{N} \times \mathcal{N} \to \mathbb{R}^+$ be the distance function, where $d(i, j)$ denotes the distance of the shortest path from nodes i to j by traversing the connections.[1] Let f_i be the capacity of node i representing the average capacity of charging service supported if a charging station is constructed at location i. It is related to the size of the site and traffic conditions around. Each node i also has a demand requirement F_i, which refers to its average local charging demand. The more EVs are based at location i, the higher F_i is. F_i can be estimated from the population size and the EV penetration rate. Without loss of generality, we assume that some F_i's are positive while some are of zero value.

We define D to be the average distance able to be traversed by most typical EVs available in the market when being fully charged. A subset of nodes $\mathcal{N}' \subset \mathcal{N}$ is said to be reachable by D if the following conditions hold:

C1) For each $i \in \mathcal{N}'$, there exists a node $j \in \mathcal{N}'$ such that $d(i, j) \leq D$;

C2) For each $i \in \mathcal{N}$, the total capacity, constituted from those nodes $j \in \mathcal{N}'$ such that $d(i, j) \leq \alpha D$ with discount factor $\alpha \in (0, 1]$, is greater than or equal to F_i; and

C3) For any $i, j \in \mathcal{N}'$, suppose h_{ij} be the number of hops of the shortest path from i to j in G. The distance of the path $d(i, j)$ should be smaller than or equal to $h_{ij}D$.

$\mathcal{N}'$ represents the set of locations which have been selected to have charging stations constructed. A city is well planned if $\mathcal{N}'$ is reachable by D. With Condition C1, an EV, which has been fully charged at a location, can recharge at another site within distance D away. C1 guarantees that EVs will not be confined in one single location (or area). Condition C2 says that the local charging demand at a location (e.g., F_i at node i) must be satisfied by the total charging capacities contributed by those charging stations located within distance αD away. α is used to model the tolerance of drivers to move away from its location for charging. Its maximum value is one because an EV can traverse for a distance at most D. The smaller α, the more conservative the model is, i.e., more charging stations should be placed around every possible location. With Condition C3, the charging station network, where each charging station is separated with another of at most distance D, spans the whole city. Note that we use one single D to characterize the accessibility of the whole city for all EV models because the distribution of the charging stations should cater for all possible EVs traveling on the roads. To do this, we should assign D with a more conservative value, e.g., the maximum travel distance of the most basic EV model in the market when being fully charged. To summarize, the conditions all together guarantee that the serving areas of the charging stations cover every corner of the city for all possible EVs.

B. Formulation

Let x_i be the decision (Boolean) variable indicating if node i is chosen for placement and c_i be its construction cost. We minimize the total cost as the objective, i.e., $\sum_{i=1}^{n} c_i x_i$.

For each i, we define $\mathcal{N}_i^{\alpha D} = \{j \in \mathcal{N} | d(i, j) \leq \alpha D\}$, representing the set of nodes (including node i itself) within distance αD from i. We can re-state Condition C2 as $\sum_{j \in \mathcal{N}_i^{\alpha D}} f_j x_j \geq F_i, \forall i \in \mathcal{N}$. As Condition C3 holds for any pair of nodes, C3 implies C1. To re-state C3, we first create a graph $\hat{G} = (\hat{\mathcal{N}}, \hat{\mathcal{E}})$, where $\hat{\mathcal{N}}$ is set to $\mathcal{N}$ and $\hat{\mathcal{E}}$ is equal to $\{(i, j) | i, j \in \mathcal{N}, d(i, j) \leq D, i \neq j\}$ (see the example shown in Fig. 1 of [1]). Consider those nodes i in G with $x_i = 1$ (i.e., $\mathcal{N}'$) and they constitute the corresponding induced subgraph H of $\hat{G}$. Condition C3 is equivalent to having H connected. In

[1] The distance $d(i, j)$ refers to the distance of an actual path connecting locations i and j but not the Euclidean distance.

other words, H has one single connected component. Instead of inspecting the original graph G, we can focus on $\hat{G}$ to formulate the problem. Similar to [13], we adopt a network flow model to address C3. Imagine that there is a source node 0^i attached to node i and it has n units of flow available to be sent along $\hat{G}$ through node i. Let $0 \leq x_0^i \leq n$ be the residue of flow not consumed by the network. Each node j with $x_j = 1$ will consume one unit of flow. For each edge $(j,k) \in \hat{\mathcal{E}}$, we indicate the amount of flow on (j,k) originated from 0^i with variable y_{jk}^i. Hence, we can guarantee those nodes j with $x_j = 1$ being reached from node i on $\hat{G}$ with the following:

$$x_0^i + y_{0i}^i = n, \tag{1}$$

$$0 \leq y_{jk}^i \leq n x_k, \forall (j,k) \in \hat{\mathcal{E}} \cup (0^i, i), \tag{2}$$

$$\sum_{j|(j,k)\in\hat{\mathcal{E}}} y_{jk}^i = x_k + \sum_{l|(k,l)\in\hat{\mathcal{E}}} y_{kl}^i, \forall k \in \hat{\mathcal{N}} \tag{3}$$

$$\sum_{j\in\hat{\mathcal{N}}} x_j = y_{0i}^i, \tag{4}$$

$$0 \leq x_0^i. \tag{5}$$

(1) says that the total amount of flow going out of y_{0i}^i and retained x_0^i in the source 0^i is n. (2) confines that only a sink can receive incoming flow. (3) describes that the total incoming flow to a node is equal to the total outgoing flow plus the amount for a sink. (4) explains that the total flow getting out of the source is equal to the total absorbed by the sinks and (5) restricts non-negative residue remained in the source.

Note that (1)–(5) require node i to be selected for charging station construction. Otherwise, no flow from the Source 0^i is allowed to be delivered to the sinks. To cater for this requirement, we attach a source node to each node in $\hat{\mathcal{N}}$ and the overall mathematical formulation of EVCSPP is modified accordingly as follows:

$$\text{minimize} \quad \sum_{i=1}^{n} c_i x_i \tag{6a}$$

$$\text{subject to} \quad \sum_{j\in\mathcal{N}_i^{\alpha D}} f_j x_j \geq F_i, \forall i \tag{6b}$$

$$x_i = \{0,1\}, \forall i \tag{6c}$$

$$x_0^i + y_{0i}^i = n, \forall i \in \hat{\mathcal{N}} \tag{6d}$$

$$0 \leq y_{jk}^i \leq n x_i x_k, \forall (j,k) \in \hat{\mathcal{E}} \cup (0^i, i), \forall i \in \hat{\mathcal{N}} \tag{6e}$$

$$\sum_{j|(j,k)\in\hat{\mathcal{E}}} y_{jk}^i = x_i x_k + \sum_{l|(k,l)\in\hat{\mathcal{E}}} y_{kl}^i, \forall i,k \in \hat{\mathcal{N}} \tag{6f}$$

$$x_i \sum_{j\in\hat{\mathcal{N}}} x_j = y_{0i}^i, \forall i \in \hat{\mathcal{N}} \tag{6g}$$

$$0 \leq x_0^i, \forall i \in \hat{\mathcal{N}}. \tag{6h}$$

(6a) and (6b) have been discussed before. (6c) confines x_i to be a Boolean variable. (6d)–(6h) correspond to the induced connected subgraph condition (i.e., (1)–(5)) for every node.

Problem (6) is an MIP with Boolean variables x_i's and continuous variables y_{jk}^i's. With the quadratic terms in equality Constraints (6f) and (6g), it is not a mixed-integer linear program (MILP). Hence, this problem is not easy to be solved.

III. Complexity Analysis

The decision version of EVCSPP can be framed as follows:

Let $N(H)$ be the set of nodes associated to the induced subgraph H. Each node i has a capacity $f_i \in \mathbb{Z}^+$ and a demand F_i and is associated with the node set $\mathcal{N}_i^{\alpha D}$. Given an undirected graph $\hat{G} = (\hat{\mathcal{N}}, \hat{\mathcal{E}})$, with node cost $c_i \in \mathbb{Z}^+, \forall i$, a cost bound $C \in \mathbb{Z}^+$, does there exist an induced subgraph H of $\hat{G}$ such that (1) For each $i \in \hat{\mathcal{N}}$, $\sum_{j\in\mathcal{N}_i^{\alpha D}\cap N(H)} f_j \geq F_i$; (2) H is connected; and (3) $\sum_{i\in N(H)} c_i \leq C$?

Theorem 1: The decision version of EVCSPP is NP-complete.

Proof: Similar to [13], we construct a reduction from the vertex cover problem (VCP) to EVCSPP. In graph $\tilde{G} = (\tilde{\mathcal{N}}, \tilde{\mathcal{E}})$, a vertex cover is a subset of nodes $\mathcal{N}' \subset \tilde{\mathcal{N}}$ such that each edge $(i,j) \in \tilde{\mathcal{E}}$ has i, j, or both in $\tilde{\mathcal{N}}$. Without loss of generality, we assume $\tilde{\mathcal{E}} \neq \emptyset$. VCP determines if there exists a vertex cover $\mathcal{N}'$ of $\tilde{G}$ with $|\mathcal{N}'| \leq C$.

We create a graph $\hat{G} = (\hat{\mathcal{N}}, \hat{\mathcal{E}})$, where $\hat{\mathcal{N}} = \tilde{\mathcal{N}} \cup \tilde{\mathcal{E}}$ and $\hat{\mathcal{E}}$ is constructed as follows. For each pair of distinct nodes $i,j \in \tilde{\mathcal{N}}$, we create an edge (i,j) in $\hat{\mathcal{E}}$; for each $e = (i,j) \in \tilde{\mathcal{E}}$, we append (i,e) and (e,j) to $\hat{\mathcal{E}}$. For each $i \in \tilde{\mathcal{N}}$, its cost is set as $c_i = 1$ and zero otherwise. For each $e \in \tilde{\mathcal{E}}$, we set $f_e = 1$ and zero otherwise. We also set $\mathcal{N}_i^{\alpha D} = \tilde{\mathcal{E}}$ and $F_i = |\tilde{\mathcal{E}}|$ for all $i \in \hat{\mathcal{N}}$.

We claim that CVP on $\tilde{G}$ with cost upper bound C if and only if EVCSPP has a solution with cost at most C. Let $\mathcal{N}'$ be a vertex cover of $\tilde{G}$ with $|\mathcal{N}'| \leq C$ and H be the induced subgraph of $\hat{G}$ by nodes $\mathcal{N}' \cup \tilde{\mathcal{E}}$. It is easy to verify that $|\mathcal{N}_i^{\alpha D} \cap N(H)| = |\tilde{\mathcal{E}}|$ and thus $\sum_{j\in\mathcal{N}_i^{\alpha D}\cap N(H)} f_j = |\tilde{\mathcal{E}}| = F_i$. As $\mathcal{N}'$ is a vertex cover, each $e = (i,j) \in \tilde{\mathcal{E}}$ must have at least one of i and j in $\tilde{\mathcal{N}}$ and thus H must contain an edge (e,k) for some $k \in \mathcal{N}'$. Moreover, $\tilde{\mathcal{N}}$ forms a clique in $\hat{G}$. Hence, H must be connected. Since each $e \in \tilde{\mathcal{E}} \subset \hat{\mathcal{N}}$ imposes no cost, H has the same cost as $\mathcal{N}'$ in $\tilde{G}$. Therefore, EVCSPP has a solution with cost at most C.

Consider that an induced subgraph H is a solution of EVCSPP. We set $\mathcal{N}' = N(H) \cap \tilde{\mathcal{N}}$. H contains $\tilde{\mathcal{E}}$: As $f_j = 1$ for $j \in \tilde{\mathcal{E}}$, for any $i \in \hat{\mathcal{N}}$, $F_i = |\tilde{\mathcal{E}}|$ guarantees $\tilde{\mathcal{E}} \subset N(H)$. Since H is connected, each $i \in \mathcal{N}'$ must have an edge with an $e \in \tilde{\mathcal{E}}$ in $\hat{G}$. Moreover, $\mathcal{N}'$ has at most C nodes. Hence, $\mathcal{N}'$ is a vertex cover of $\tilde{G}$ with $|\mathcal{N}'| \leq C$. ∎

With Theorem 1, we have an immediate corollary as follows:

Corollary 1: EVCSPP is NP-hard.

IV. Proposed Solutions

Since the problem is NP-hard, there is no trivial way to solve it. In this section, we propose four solution methods to tackle the problem. They possess their own pros and cons and thus they are suitable for different situations.

A. Method I: Iterative Mixed-Integer Linear Program

(1)–(5) can be used to guarantee that the solution subgraph constituted by all nodes j with $x_j = 1$ is connected as long as $x_i = 1$. If we assume that node i will be one of locations for charging station construction, i.e., $x_i = 1$, problem (6) can be reduced to

$$\text{minimize} \quad \sum_{k=1}^{n} c_k x_k \tag{7a}$$

$$\text{subject to} \quad \sum_{j \in \mathcal{N}_k^{\alpha D}} f_j x_j \geq F_k, \forall k \in \hat{\mathcal{N}}, \tag{7b}$$

$$x_k = \{0, 1\}, \forall k \in \hat{\mathcal{N}} \tag{7c}$$

$$x_0^i + y_{0i}^i = n, \tag{7d}$$

$$0 \leq y_{jk}^i \leq n x_k, \forall (j,k) \in \hat{\mathcal{E}} \cup (0^i, i), \tag{7e}$$

$$\sum_{j|(j,k)\in\hat{\mathcal{E}}} y_{jk}^i = x_k + \sum_{l|(k,l)\in\hat{\mathcal{E}}} y_{kl}^i, \forall k \in \hat{\mathcal{N}} \tag{7f}$$

$$\sum_{j \in \hat{\mathcal{N}}} x_j = y_{0i}^i, \tag{7g}$$

$$0 \leq x_0^i, \tag{7h}$$

$$x_i = 1. \tag{7i}$$

Problem (7) is an MILP and it can be solved with standard MIP solvers applying methods like branch-and-bound. Now the question becomes which node i should be chosen for this purpose.

We write the solution of Problem (7) as $\inf_{x\in\Omega_i} \sum_{j=1}^{n} c_j x_j$, where Ω_i is the solution space constituted by (7b)–(7i). We have the following theorem:

Theorem 2: The solution of Problem (6) can be determined by solving

$$\min_{1\leq i\leq n} (\inf_{x\in\Omega_i} \sum_{j=1}^{n} c_j x_j). \tag{8}$$

Proof: The solution of (6) can induce a connected subgraph of $\hat{G}$ where (6b) is satisfied and the total cost function (6a) is minimized. When we solve (7), its solution induces a connected subgraph of $\hat{G}$ with minimum total cost satisfying (6b) *where node i is included*. To compute (8), we apply (7) to every node i (thus we solve (7) n times) and its solution is the minimum among the n subgraphs. Since (6a) and (6b) exist in every (7), the solution of (6) must be one of the computed subgraphs with some node i included. Hence, addressing (8) can solve (6). ■

With this result, the original mixed-integer *non-linear* program becomes n solvable MILPs. Since we need to go through all nodes iteratively, we call this method Iterative MILP.

Besides the fact that the computational time of solving MILP (7) grows super-linearly with n, this method also suffers from the problem that the number of MILPs (i.e., (7)) needed to be solved also increases with n. Hence, the combined effect of increasing n will make its computation time accelerating extraordinarily fast. Hence, this method is only applicable to small problem instances. If the solver applied to (7) can produce the optimal solution, Method I can guarantee the optimality.

B. Method II: Greedy Approach

Here we present an efficient greedy algorithm, which is applicable to the original formulation (6) and requires much shorter computation time. Before discussing its details, we have the following lemma to facilitate its development.

Lemma 1: Problem (6) (and Problem (7)) is feasible if and only if $x = [x_1, \ldots, x_n] = [1, \ldots, 1]$ is a feasible solution, which gives an upper bound with the objective function values $\sum_{i=1}^{n} c_i$.

Proof: First we consider the only if-direction. As the problem is feasible, there exists a feasible $x' = [x'_1, \ldots, x'_n]$, composed of some 0's and/or 1's, satisfying Constraints (6b)–(6h). If $x'_i = 1$ for all i, then we have the result. Consider that there is at least one j such that $x'_j = 0$. If we produce another x'' by modifying x'_j with value one, besides (6c), x'' will always satisfy Constraint (6b), as we will stay unchanged or increase the sum on the left-hand side of (6b). Moreover, as $0 < \alpha \leq 1$, if $x'_j = 0$ satisfies (6b), there exists at least one node k with $x'_k = x''_k = 1$ within distance D away from node j. In this way, if we have $x''_j = 1$, we will attach node j to the subgraph induced by x' through node k. In other words, the subgraph induced by x'' is still connected, i.e., satisfying (6d)–(6h). We can repeat this process until we change all 0's to 1's and this produces x with upper bound $\sum_{i=1}^{n} c_i$.

The if-direction is trivial. We complete the proof. ■

Corollary 2: If $x = [x_1, \ldots, x_n] = [1, \ldots, 1]$ is not feasible, EVCSPP is infeasible.

Corollary 2 can be used to check the feasibility of a problem instance.

Algorithm 1 Greedy Algorithm

```
1:  Set x_i = 1 for i = 1, ..., n.
2:  repeat
3:      Construct a node set N̄ composed of nodes i with x_i =
        1 where the induced subgraph is still connected when
        x_i is set to zero.
4:      x' ← x
5:      flag ← 0
6:      repeat
7:          Select j with the largest c_j in N̄.
8:          Modify x' by setting x'_j = 0
9:          if x' satisfies (6b) then
10:             x ← x'
11:             flag ← 1
12:         else
13:             x'_j ← 1
14:             Remove j from N̄
15:         end if
16:     until flag = 1 OR N̄ = ∅
17: until N̄ = ∅
```

Assume that we have a feasible problem instance. We construct a greedy algorithm by reducing the total cost as much as possible in each iteration and it results in a sub-optimal solution. Its pseudocode is given in Algorithm 1. In Line 1, we start with the feasible $x = [x_1, \ldots, x_n] = [1, \ldots, 1]$ explained in Lemma 1 and then go through a certain number

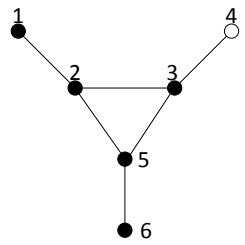

Fig. 1. Node selection in the greedy algorithm.

of iterations (Lines 2–17). In each iteration, we select those nodes in the subgraph induced by x which will not disconnect the subgraph if we remove them from the subgraph and we call this selection $\overline{\mathcal{N}}$ (Line 3). For example, Fig. 1 shows a graph $\hat{G}$ of six nodes, where a dot i and a hole j mean $x_i = 1$ and $x_j = 0$, respectively. In this case, we have $\overline{\mathcal{N}} = \{1, 3, 6\}$. We can see that the resultant x' formed by removing any one node in $\overline{\mathcal{N}}$ will still satisfy Constraints (6d)–(6h). Then we attempt to deselect the one (say node j) with the highest cost c_j in $\overline{\mathcal{N}}$ (Line 7). If the resultant x' satisfies (6b), x' is a feasible point and we proceed to the next iteration (Lines 9–11). Otherwise, we remove j from $\overline{\mathcal{N}}$ (Line 14). Instead of deselecting j (Line 13), we deselect the one with the next highest cost. The iterations terminate when no nodes remain in $\overline{\mathcal{N}}$ (Line 17). The final solution x is the best determined by the greedy algorithm. Note that the resultant solution is usually sub-optimal, especially when the problem size n becomes larger.

C. Method III: Effective Mixed-Integer Linear Program

Recall that in Method I, we need to apply MILP (7) to every node since we are not sure which node i has $x_i = 1$ in the optimal solution and thus it is usually subject to long computation time. However, if we know the node which has unity in the solution, we can save lots of effort by applying (7) to that node only. With Theorem 3, under a general condition, we find that not all nodes i are required to generate the solution as in Method I.

Theorem 3: Suppose that the demand requirements F_i for all i are positive. Then for any node i, at least one node j in i's one-hop neighborhood in $\hat{G}$, i.e., $\mathcal{N}_i^{\alpha D}$, must have $x_j = 1$ in the optimal solution of (6).

Proof: Since all F_i's are positive, $[x_1, \ldots, x_n] = [0, \ldots, 0]$ can never be a solution. Hence, the solution must contain at least one node i with $x_i = 1$. Consider any particular node i. If x_i is set to one in the optimal solution, then the result holds by assigning j to i as $i \in \mathcal{N}_i^{\alpha D}$.

Consider that x_i is set to zero. As F_i is positive, at least one term $f_j x_j$ on the left-hand side of (6b) must be positive. In other words, at least one node (e.g., j) in $\mathcal{N}_i^{\alpha D}$ has $x_j = 1$. Since $\alpha \leq 1$, node j must be a one-hop neighbor of node i. Since all F_i's are positive, the condition applies to every node.

Hence, the result is true for every node in the graph. ■

With this theorem, we compose Method III by choosing any node i and applying Problem (7) with respect to those nodes in $\mathcal{N}_i^{\alpha D}$ only. The number of (7) required to be solved depends on the cardinality of $\mathcal{N}_i^{\alpha D}$. We can minimize the computational time by choosing the node i with the smallest degree in $\hat{G}$. In this way, we can simplify Method I by exploiting the network structure of the graph and the solutions of both Methods I and III are equivalent. Similar to Method I, If the solver applied to (7) can produce the optimal solution, Method III can also guarantee the optimality. Since EVs are movable in a city, it is common to have EVs appearing in every location (node) in a certain timespan, and thus we have positive F_i for all nodes i. Hence, the condition imposed in Theorem 3 is very general.

D. Method IV: Chemical Reaction Optimization

Chemical Reaction Optimization (CRO) is a recently proposed Nature-inspired metaheuristic for optimization [14]. Under certain conditions, it has been proved to be able to converge to the global optimum for combinatorial optimization problems (like EVCSPP) [15] and it has been demonstrated to have very good performance in solving real-world problems, e.g., [16], [17]. CRO is general-purpose and we apply CRO to EVCSPP. In CRO, the manipulated agents are molecules, each of which carries a solution. The molecules explore the solution space of the problem through a random series of elementary reactions taking place in a container. We define four types of elementary reactions, each of which has its own way to modify the solutions carried by the involved molecules. Due to space limitation, we do not illustrate every detail of CRO but explain the necessary modifications based on the framework described in [14]. We basically follow [14] to construct the algorithm. It consists of four elementary reactions, including on-wall ineffective collision, decomposition, inter-molecular ineffective collision, and synthesis. They are implemented as follows:

1) On-wall ineffective collision: It mimics that a molecule hits a wall of the container and then bounces back. This elementary reaction is not vigorous and we only has small modifications to the molecule. Let x and x' be the solutions held by the molecules before and after the change. We apply our greedy approach (Method II)[2] to x to produce x', i.e., $x \xrightarrow{\text{greedy}} x'$.

2) Decomposition: It describes that one molecule hits a wall and breaks into two separate molecules. It involves vigorous changes to the molecules. Let x be the solution held by the reactant molecule and x'_1 and x'_2 be the solutions of the resultant molecules. Here x'_1 and x'_2 are randomly generated in the solution space. A random solution can be produced by modifying Algorithm 1 where the repeat loop (Lines 6–16) iterates for a random number of times (between 1 to n) and we select a random node in $\overline{\mathcal{N}}$ in Line 7.[3] A decomposition can be described as $x \xrightarrow{\text{random}} x'_1 + x'_2$.

3) Inter-molecular ineffective collision: It portrays that two molecules collide with each other and then bounce away. The

[2]Note that we can initiate the greedy algorithm with any x instead of the unity vector $[1, \ldots, 1]$ by skipping Line 1 in Algorithm 1.

[3]Although the generations of x'_1 and x'_2 do not rely on x, the energies stored in the molecules do. Interested readers may refer to [14] for more information.

change is not vigorous. Let x_1 and x_2 be the two reactant molecules and x_1' and x_2' be the resultant molecules. Similar to the on-wall ineffective collision, we apply the greedy algorithm to the respective molecules to modify the solutions, i.e., $x_1 + x_2 \xrightarrow{\text{greedy}} x_1' + x_2'$.

4) Synthesis: Synthesis describes that two molecules collide with each other and then combine into one molecule. The change is vigorous. Similar to decomposition, we produce a new molecule by randomly generating its solution in the solution space. Let x_1 and x_2 be the two reactant molecules and x' be the resultant molecule. We have $x_1 + x_2 \xrightarrow{\text{random}} x'$.

When initializing the algorithm, we assign random solutions in the solution space to the molecules (this can be done by the random solution generation used in decomposition). It is clear that Method II is embedded in this method except that Method II always start with a unity vector x. We can guarantee that Method IV is always superior to Method II in terms of solution quality by having at least one molecule possessing a unity vector as its initial solution. So we can assign the unity vector to some initial molecules (says 10%) in the initialization phase. Since CRO is a probabilistic algorithm, the solutions produced in different run could be different.

V. Performance Study

A. Simulation Results

We perform a series of simulations to evaluate the performance of the four solution methods. All simulations are run on the same computer with Intel Core i7-3770 CPU at 3.40GHz and 16GB of RAM, and conducted in the MATLAB environment. For Methods I and III, the MILP is computed with the CPLEX solver [18] with YALMIP [19]. Recall that Methods I, II, and III are deterministic while Method IV is probabilistic. For illustrative purposes, we repeat Method IV 10 times for each simulation case. After several trial runs, we set the parameter values for Method IV as: "function evaluation limit" = 2000, "initial kinetic energy" = 10, "initial population size" = 40, "initial buffer" = 10, "collision ratio" = 0.5, "synthesis threshold" = 0.5, "decomposition threshold" = 20, and "KE loss rate" = 0.9. We conduct three tests. In the first test, we examine the solutions' performance with changing α. The second aims to study how the computation time grows with the problem size. In the third, we test how the solution methods perform in a real-world scenario.

In the first test, we randomly generate 100 feasible instances. Each instance of G is constructed by randomly placing 50 nodes in an area of 100×100 km^2, where we assign a random value in the range of $(0, 1]$ to the cost c_i, and D, f_i and F_i, for all i, are set to 80 km, 0.5, and 1, respectively. For simplicity, we assume that the nodes are interconnected and the length of the shortest path of each pair of nodes is determined by the Euclidean distance between them. As explained in Section II, we can produce $\hat{G}$ from G. Then we can check the feasibility of each instance with Corollary 2.

We verify the performance of the four methods with respect to the computed solution quality and the computation time. The results are given in Table I. The second column indicates the number of feasible and matched cases among the 100

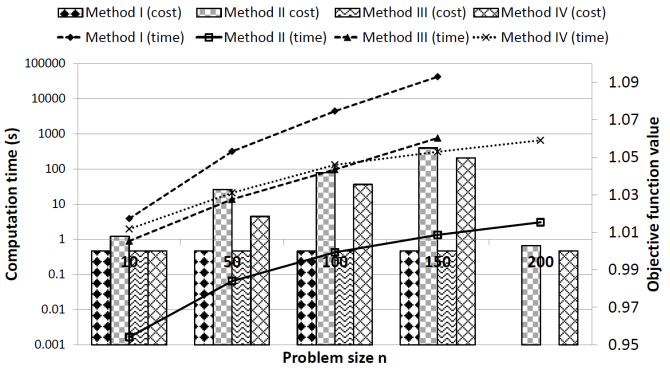

Fig. 2. Computation time changing with problem size.

graphs. All graphs are feasible when α is equal to one. When α decreases, the number of resulted feasible cases will also decrease as Constraint (6b) becomes stronger. The matched cases indicate those of the feasible ones producing the same objective function values by all the four approaches. Regardless the non-statistically significant cases with $\alpha = 0.7$, all the four methods can produce the best solutions for around $\frac{1}{3}$ of the feasible cases. The other columns show the average objective function values and computation times of the four methods among the feasible cases. For Method IV, we also provide the average (among the cases) of the best (among the repeats) and the worst (among the repeats) for reference. Methods I and III always give the best solutions and Method IV always outperforms Method II. In terms of computation time, Method II is the fastest and Method III comes the second. Method IV is the next and Method I takes the longest.

In the second test, we study how the computation time changes with the problem size. The setting is similar but we fix α to one for different values of n. We generate 10 feasible cases for n equal to 10, 50, 100, 150, and 200. Fig. 2 shows the average computation times of the four methods in the logarithmic scale. The corresponding objective function values normalized by computed minimums are also given for reference. All the computation times increase with n. Method I takes the longest computation time which also grows the fastest. Method III needs less time than Method IV when n is small. However, when n is larger than 100, Method III requires more time to compute the solution. In other words, the computation time of Method III grows faster than that of Method IV. Method II needs the shortest time but its computed solution quality is the worst. Although Methods I and III require relatively more time, their solutions are the best. Note that the results of Methods I and III for $n = 200$ is not shown because they are not computable by YALMIP/CPLEX due to the out-of-memory problem. This implies the MILP approaches are not suitable for large problems.[4] As before, Method IV always has better solutions than Method II.

In the third test, we apply the problem to a real-world environment; we determine the locations for building charging

[4]We run the simulations in MATLAB. The out-of-memory problem may happen at another n with a different combination of machine and platform. Here we just demonstrate that Methods I and III are not scalable.

TABLE I
SIMULATION RESULTS FOR $n = 50$ AND $D = 20$ KM

α	Matched/ Feasible cases	Objective function value						Computation time (s)			
		Method I	Method II	Method III	Method IV (mean)	Method IV (best)	Method IV (worst)	Method I	Method II	Method III	Method IV (mean)
1	32/100	9.4875	9.7553	9.4875	9.6119	9.5092	9.7158	478.6861	0.0610	23.6505	20.4516
0.9	20/62	10.8950	11.0986	10.8950	11.0050	10.9235	11.0745	369.5348	0.0618	18.2473	23.3749
0.8	8/23	12.5701	12.8035	12.5701	12.6786	12.5818	12.7829	310.1252	0.0611	15.7985	29.6049
0.7	0/2	13.3116	13.8089	13.3116	13.4800	13.3182	13.7777	331.2596	0.0542	19.9461	34.9514

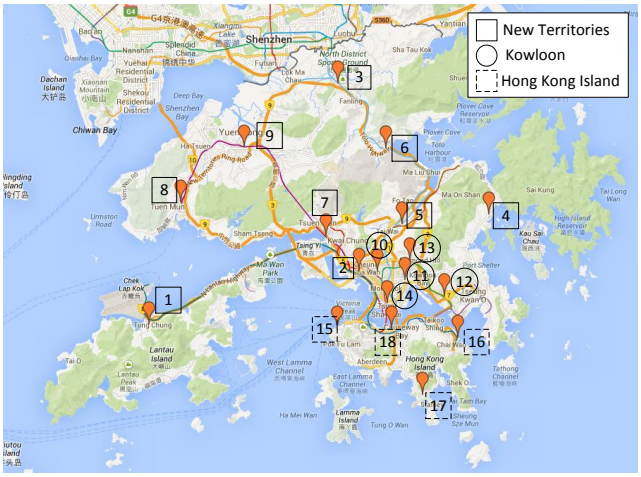

Fig. 3. Charging station distribution (adopted from [20]).

TABLE II
CHARGING STATION DATA (FROM THE 2006 CENSUS)

District	Population (k)	Density (/km^2)	Median monthly per capita (HK$)	f_i
1	137.1	783	5659	1277.14
2	523.3	22421	4833	44.60
3	280.7	2055	5161	486.62
4	406.4	3135	6774	318.98
5	607.5	8842	6232	113.10
6	293.5	2156	5806	463.82
7	288.7	4679	6897	213.72
8	502.0	6057	5172	165.10
9	534.2	3858	4777	259.20
10	365.5	39095	4821	255.79
11	362.5	36178	6897	276.41
12	587.4	52123	4845	191.85
13	423.5	45540	4750	219.59
14	280.5	40136	6034	249.15
15	250.0	20102	9722	99.49
16	587.7	31664	7235	63.16
17	275.2	7083	6563	282.37
18	155.2	15788	10185	126.68

stations in Hong Kong (HK). The HK Government plans to introduce more EVs (e.g., taxi) into the city and the construction of charging stations is one of the crucial steps in the plan. We can see how this can be realized through solving EVCSPP. HK is composed of three zones (New Territories, Kowloon, and HK Island) and each zone is further divided into districts. There are total 18 districts in HK. We select one location in each district for potential charging station construction. The location distribution is given in Fig. 3. The distance between each pair of locations is retrieved from the route connecting them suggested by Google Map [20]. We relate the location parameters to the district data obtained from [21]. We assign the population size to the demand F_i and the median monthly income per capita to the cost c_i. We set the capacity f_i inversely proportional to the density with some proportionality constants. The location parameter values are listed in Table II. We perform simulations with several combinations of D (30, 35, 40, 45, and 50 km) and α (1 and 0.8) and methods' performance is given in Table III where the best objective function values are bold.

Methods I and III always find the best solutions for all cases. Method IV can determine the best solutions in most cases while Method II can still achieve the best solutions in some cases. For computation time, on the average, Method II is the fastest, and then Method IV. Method III comes next and Method I takes the longest. In general, the computation times of Methods I, II, and IV decrease with D since $\hat{G}$ becomes more dense with D and we need less effort to locate the solutions. But Method III does the opposite because nodes tend to have larger degrees with D. Thus the number of MILPs needed to be solved increase together with the minimum degree of $\hat{G}$.

B. Discussion

From the simulations above, we can see that each method has its own characteristics and is suitable for different situations. Here we try to compare them in terms of five different perspectives independently:

1) Solution quality: If the adopted MILP solver can guarantee optimality[5], Methods I and III can obtain the best results. As Method II is embedded in Method IV, Method IV is always superior to Method II but they may not produce the optimal solutions, especially when the solution space is getting larger. They can be ranked as: I=III>IV>II.

2) Computation time: Method II is of the simplest design and takes very limited amount of time to obtain a (usually sub-optimal) solution. Method I needs to apply MILP to all nodes of the problem while Method III requires only a subset. Hence, Method III is always faster than Method I. Method IV is a metaheuristic and we can terminate the algorithm when

[5]Most MIP solvers can generate the optimal solutions when the problem size is small.

TABLE III
SIMULATION RESULTS FOR THE HONG KONG CASE

α	D (km)	Objective function value						Computation time (s)			
		Method I	Method II	Method III	Method IV (mean)	Method IV (best)	Method IV (worst)	Method I	Method II	Method III	Method IV (mean)
1	30	**19 193**	**19 193**	**19 193**	**19 193**	**19 193**	**19 193**	21.76	6.80E-03	4.10	4.05
	35	**15605**	19 193	**15 605**	15 700.8	**15 605**	16 468	21.51	5.88E-03	11.94	3.54
	40	**10 556**	14 348	**10 556**	**10 556**	**10 556**	**10 556**	18.53	5.52E-03	12.90	3.29
	45	**10 556**	14 348	**10 556**	10 558.7	**10 556**	10 583	19.08	5.48E-03	15.38	3.08
	50	**5659**	14 348	**5659**	**5659**	**5659**	**5659**	17.43	5.12E-03	17.64	2.85
0.8	30	**37 237**	**37 237**	**37 237**	**37 237**	**37 237**	**37 237**	20.61	8.24E-03	4.41	6.34
	35	**21 629**	25 236	**21 629**	**21 629**	**21 629**	**21 629**	17.75	6.51E-03	12.07	4.39
	40	**15 605**	19 193	**15 605**	15 796.9	**15 605**	16 661	21.16	5.97E-03	14.73	3.72
	45	**10 556**	14 416	**10 556**	**10 556**	**10 556**	**10 556**	20.49	6.01E-03	16.37	3.37
	50	**10 556**	14 348	**10 556**	**10 556**	**10 556**	**10 556**	19.75	5.03E-03	18.86	3.31

certain stopping criteria are satisfied (in our cases, we limit the computation time by setting a function evaluation limit). We can allow the algorithm to run longer for a better solution. It is a tradeoff between solution quality and speed and it is hard to compare Method IV's efficiency with the others. Hence we just have rank as: II>III>I.

3) Solvable problem size: Solving MLIP is the major building block of Methods I and III and it relies on the adopted solver. As most existing solvers can handle relatively small problems (in our case with MATLAB/YALMIP/CPLEX, the problem is only solvable with $n \leq 150$). However, since the manipulating mechanisms of Methods II and IV are mainly about how to modify and evaluate temporary solutions, they are more resistive to the growing problem size. They can be ranked as: II=IV>I=III.

4) Algorithmic nature: All methods are deterministic except Method IV. In other words, we always come up with the identical result in different runs of the same problem instance. Method IV is probabilistic in nature. For each instance, we repeat the simulations several times to obtain its average performance.

5) Prerequisite: Recall that Method III is valid only when the condition imposed in Theorem 3 holds. Although this condition is very general and held in most practical situations, the other methods do not require it.

We summarize the characteristics of the four methods in Table IV. Note that the above conclusions are drawn independently of each other from our observations on the simulation. When evaluating a particular method, we usually take several aspects into account simultaneously, e.g., correlating solution quality with computation time and problem size. However, we aim to give an extensive assessment and thus we appraise their individual abilities from one aspect to another. In general, each method has its own pros and cons and none is outstanding predominantly. We select the most suitable method according to our need.

As a final remark, Method IV has not been fully optimized. As discussed in [14], its performance is subject to the parameter settings and the duration of the simulation run. We can further improve its performance by determining a better combination of parameter values and allowing it to run for long time. However, we aim to show that the greedy algorithm can be considered as a component in a metaheuristic, whose performance is guaranteed to be better than that of Method II. Seeking the best result for Method IV is out of the scope of this paper. Moreover, Method IV can be interpreted in a broader sense; it is a greedy-algorithm embedded metaheuristic approach where we can replace CRO with any metaheuristic like Genetic Algorithm [22]. Further exploration of Method IV will be left for the future work.

TABLE IV
SOLUTION METHOD CHARACTERISTIC COMPARISON

	Method I	Method II	Method III	Method IV
Solution quality	✓✓✓	✓	✓✓✓	✓✓
Computation time	✓	✓✓✓	✓✓	Depends
Problem size	✓	✓✓✓	✓	✓✓✓
algorithmic nature	Deterministic	Deterministic	Deterministic	Probabilistic
Prerequisite	None	None	Some	None

VI. CONCLUSION

Gasoline is a heavily demanded natural resource and most is consumed on transportation. Its electrification can relieve our dependence on gasoline and tremendously reduce the amount of harmful gases released, which partially constitute global warming and worsen our health. In the 21st century, advancing EV technologies has become one of the keys to boost a nation's economy and maintain (and improve) people's quality of living. For long-term planning, modernizing our cities with EVs is of utmost importance. EVs will be integrated into the transportation system seamlessly and this will help make our cities "smart". To do this, we need to determine the best locations to construct charging stations in the city. In this paper, we focus on human factors rather than technological ones for charging station placement. An EV should always be able to access a charging station within its driving capacity anywhere in the city. Our contributions in this paper include: 1) formulating the problem, 2) identifying its properties, and 3) developing the corresponding solution methods. We formulate the problem as an optimization model, based on the charging

station coverage and the convenience of drivers. We prove the problem NP-hard and propose four solution methods to tackle the problem. Each method has its own characteristics and is suitable for different situations depending on the requirements for solution quality, algorithmic efficiency, problem size, nature of the algorithm, and existence of system prerequisite.

REFERENCES

[1] A. Y. S. Lam, Y.-W. Leung, and X. Chu, "Electric vehicle charging station placement," in *Proc. IEEE Int. Conf. Smart Grid Commun.*, Vancouver, BC, Oct. 2013.

[2] S. Chen and L. Tong, "iEMS for large scale charging of electric vehicles: Architecture and optimal online scheduling," in *Proc. IEEE Int. Conf. Smart Grid Commun.*, Tainan City, Taiwan, Nov. 2012.

[3] A. Y. S. Lam *et al.*, "A multi-layer market for vechicle-to-grid energy trading in the smart grid," in *Proc. 1st IEEE INFOCOM Workshop on Green Netw. and Smart Grids*, Orlando, FL, Mar. 2012.

[4] J. J. Q. Yu, V. O. K. Li, and A. Y. S. Lam, "Optimal V2G scheduling of electric vehicles and unit commitment using chemical reaction optimization," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, Jun. 2013.

[5] F. Guo, E. Inoa, W. Choi, and J. Wang, "Study on global optimization and control strategy development for a PHEV charging facility," *IEEE Trans. Veh. Technol.*, vol. 61, pp. 2431–2441, Jul. 2012.

[6] M. Etezadi-Amoli *et al.*, "Rapid-charge electric-vehicle stations," *IEEE Trans. Power Del.*, vol. 25, pp. 1883–1887, Jul. 2010.

[7] A. S. Masoum *et al.*, "Smart load management of plug-in electric vehicles in distribution and residential networks with charging stations for peak shaving and loss minimisation considering voltage regulation," *IET Generation, Transmission & Distrib.*, vol. 5, pp. 877–888, 2011.

[8] Z. Liu, F. Wen, and G. Ledwich, "Optimal planning of electric-vehicle charging stations in distribution systems," *IEEE Trans. Power Del.*, vol. 28, pp. 102–110, Jan. 2013.

[9] Z. f. Liu, W. Zhang, X. Ji, and L. Ke, "Optimal planning of charging station for electric vehicle based on particle swarm optimization," in *Proc. IEEE Innovative Smart Grid Technol. - Asia*, Tianjin, China, 2012.

[10] T. D. Chen *et al.*, "The electric vehicle charging station location problem: A parking-based assignment method for Seattle," in *Proc. 31th USAEE/IAEE North Amer. Conf.*, Austin, TX, Nov. 2012.

[11] Z. Drezner, *Facility location : applications and theory*, 2nd ed. Berlin: Springer, 2010.

[12] R. Church and C. R. Velle, "The maximal covering location problem," *Papers in Regional Sci.*, vol. 32, pp. 101–118, Jan. 1974.

[13] J. M. Conrad, C. P. Gomes, W.-J. van Hoeve, A. Sabharwal, and J. F. Suter, "Wildlife corridors as a connected subgraph problem," *J. Environmental Economics and Manag.*, vol. 63, pp. 1–18, Jan. 2012.

[14] A. Y. S. Lam and V. O. K. Li, "Chemical-reaction-inspired metaheuristic for optimization," *IEEE Trans. Evol. Comput.*, vol. 14, pp. 381–399, Jun. 2010.

[15] A. Y. S. Lam, V. O. K. Li, and J. Xu, "On the convergence of chemical reaction optimization for combinatorial optimization," *IEEE Trans. Evol. Comput.*, vol. 17, pp. 605–620, Oct. 2013.

[16] J. Xu, A. Y. S. Lam, and V. O. K. Li, "Chemical reaction optimization for task scheduling in grid computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, pp. 1624–1631, Oct. 2011.

[17] A. Y. S. Lam, V. O. K. Li, and J. J. Q. Yu, "Power-controlled cognitive radio spectrum allocation with chemical reaction optimization," *IEEE Trans. Wireless Commun.*, vol. 12, pp. 3180–3190, Jul. 2013.

[18] *IBM ILOG CPLEX V12.1 User's Manual for CPLEX*, 2009.

[19] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *Proc. IEEE Int. Symp. Comput. Aided Control Syst. Design*, Taipei, Taiwan, Sep. 2004.

[20] Hong Kong. (2013, Sep.) Google maps. [Online]. Available: http://maps.google.com

[21] Wikipedia. (2013, Sep.) Districts of hong kong. [Online]. Available: http://en.wikipedia.org/wiki/Districts_of_Hong_Kong

[22] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA: Addison-Wesley, 1989.