Abstract—This paper integrates a set of scheduling entities independently developed for electric vehicle tours, clearly defining interactions and exporting relevant interfaces. The scheduler engine opens two interfaces, one for the map layer and the other for the super scheduler layer. Retrieving the cost matrix from the map layer as well as a set of tour spots to visit from the scheduler layer, our engine invokes an appropriate scheduling entity out of backtracking-based, genetic algorithm, simulated annealing, and 2-opt heuristic schemes. To overcome the inconvenience stemmed from long charging time and short driving range, the scheduling modules define a common fitness function which pursues the reduction of waiting time for battery charging. The proposed design can systematically integrate a new algorithm and makes it possible for them to combine with existing ones through the implemented library functions.

Keywords—Electric vehicle, tour-and-charging scheduler, library engine, waiting time

I. INTRODUCTION

Electric vehicles, or EVs in short, are regarded as one of the key elements in future transport systems. As they can significantly cut down air pollution, many eco-cites, usually having many tourist attractions, are greatly interested in their deployment. EVs can penetrate not just in the form of personal ownership but also through some kinds of sharing systems. In tour cities, EV rent-a-car systems are appearing as a long-term sharing model, in which the rental period lasts for a few days. As contrast to gasoline-powered vehicles, tourists must be more careful in making a tour plan due to the short driving range and long charging time of EVs [1]. Moreover, the charging infrastructure is not yet sufficiently available in most cities, the tour schedule must take into account where to charge EVs.

Without an efficient schedule, tourists may waste their time if battery remaining is not enough to reach the next tour point. For a daily tour, the driving distance is highly likely to exceed the driving range, making it necessary to charge the EV somewhere during a tour. Some tour spots install charging facilities, allowing EVs to be charged while tourists are taking a tour. On the contrary, in a scenic drive course, tour and charging cannot be done simultaneously. A visiting order, or tour schedule, is very important especially in EV tours, but hard for a tourist to decide if the number of tour spots gets larger. Instead, tourists can take advantage of intelligent computer applications for this problem [2]. The integration of computational intelligence is actually what smart transport systems are enthusiastically pursuing.

The tour schedule must reduce the waiting time induced by EV charging, considering the availability of chargers at each tour spot as well as the distance between each destination. The waiting time is the time interval the tourist cannot drive to the next place but wait for their EVs to be charged. As this problem belongs to the NP category, there exist a variety of optimal and suboptimal search techniques. For the given fitness criteria, they enhance the fitness of feasible solutions. As each one has its own advantages, such as accuracy and execution speed, it is desirable to implement them and selectively integrate for a specific requirement. In this regard, our work develops a set of library modules for intelligent tour scheduling, aiming at enhancing the convenience of EV rent-a-car users.

II. LIBRARY DESIGN

A. Design overview

Figure 1 outlines our engine design. It is two interfaces, one for the underlying map layer while the other for high-level schedulers. The essential information for the core engine to make a tour schedule includes the availability of chargers at each tour spot and the inter-destination distance. The map layer, maintaining the road network of a target city, provides such geographical data to the core engine. The core engine part...

Junghoon. Lee is with the Department of Computer Science and Statistics, Jeju National University, Jeju City, Republic of Korea (corresponding author to provide phone: +82-64-754-3594; fax: +82-64-725-2579; e-mail: jhlee@jejunu.ac.kr).

Seulbi. Lee is with the Department of Computer Science and Statistics, Jeju National University, Jeju City, Republic of Korea (e-mail: gwregx@jejunu.ac.kr).

Jihyun. Kang is with the Department of Computer Science and Statistics, Jeju National University, Jeju City, Republic of Korea (e-mail: nativegod12@naver.com).
inter-destination driving cost estimation. Figure 2 plots the road network of Jeju City and each tour spot is marked over the map. For a pair of two location, A* or Dijkstra algorithms calculate the distance between them. Each tour spot is specified by latitude, longitude, closest node, average stay time, and most importantly, charging facility availability. If the distance between each pair of locations is calculated every time a user selects a set of tour spots, the response time can grow behind the tolerable bound. As the number of selectable tour spots is limited, it is reasonable to calculate the distance of every pair of spots in advance through the idle time of the service.

Fig. 2 Map layer display

### C. Core engine library

Given a set of tour spots, the core engine retrieves the cost matrix containing the cost between each pair of spots from the map layer. Then, for a feasible schedule, namely, ordered list of user-selected tour spots, we can estimate the battery consumption and waiting time. Along the sequence, two primitive functions of stay and move are called in turn. Stay increases battery remaining in proportion to the stay time at a spot if it has a charger. Move decreases battery remaining according to the distance between two consecutive spots. If battery remaining is not enough, the waiting time increases by the amount of insufficiency. A schedule having smaller waiting time better fits. Our implementations try to find the schedule having the smallest waiting time. According to our observation, exhaustive search finds an optimal schedule within 1 sec on C language and 2.5 sec on C# implementation, respectively.

We have developed and tested a series of scheduling schemes independently. Our implementation includes exhaustive search [4], genetic algorithm [5], simulated annealing, and 2-opt heuristic. Now, this paper makes each model open an API for inter-module communication. The super scheduler can run one or some of them according to the application requirement. The exhaustive search investigates all feasible solutions, taking the various constraints and pruning branches. The genetic scheduler is built on top of genetic operators such as crossover, selection, and mutation. In addition, simulated annealing techniques do not generally yield an efficient schedule and 2-opt heuristics are not easy to adapt to the waiting time reduction. However, their lightweight execution makes it possible to get good quality initial population for genetic algorithms. Within the engine, we can combine each module for better accuracy.

### III. CONCLUDING REMARKS

This paper has designed an integrated framework consisting of many scheduling entities and a common fitness function, for EV charging and tour. Via the exported library functions, they cannot only combine part of other modules to meet the requirement imposed on scheduling accuracy and response time but also interact with underlying location information and super schedulers. This library allows a variety of schedulers to be implemented for multi-day tour plans and orienteering problems. After evaluation processes, our engine will be re-implemented in .NET platform to seamlessly provide tour-and-charging services for mobile devices [6].

### ACKNOWLEDGMENT

Following are results of a study on the "Leaders INdustry-university Cooperation" Project, supported by the Ministry of Education (MOE).

### REFERENCES


