Transportation Research Part B 70 (2014) 285–302

Contents lists available at ScienceDirect

Transportation Research Part B

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

Online routing and battery reservations for electric vehicles with swappable batteries

Jonathan D. Adler *, Pitu B. Mirchandani [1]

Arizona State University, School of Computing, Informatics, and Decision Systems Engineering, 699 S Mill Ave., Tempe, AZ 85281, United States

ARTICLE INFO

Article history:
Received 20 November 2013
Received in revised form 10 September 2014
Accepted 10 September 2014

Keywords:
Electric vehicles
Approximate dynamic programming
Vehicle routing
Markov chance-decision processes
Linear temporal differencing

ABSTRACT

Electric vehicles are becoming a more popular form of transportation, however their limited range has proven problematic. Battery-exchange stations allow the vehicles to swap batteries during their trip, but if a vehicle arrives at a station without a full battery available it may have to wait an extended period of time to get one. The vehicles can be routed so that they avoid stations without available batteries or to keep batteries available for other vehicles that need them in the future. The batteries can also be reserved during the routing process so that each vehicle is ensured the battery it plans to use is available. This paper provides a method of online routing of electric vehicles and making battery reservations that minimizes the average delay of the all vehicles by occasionally detouring them to the benefit of future ones. The system is modeled as a Markov chance-decision process and the optimal policy is approximated using the approximate dynamic programming technique of temporal differencing with linear models. The solution algorithm provides a quick way for vehicles to be routed using onboard vehicle software connected to a central computer. Computational results for the algorithm are provided using data on the Arizona highway network.

1. Introduction and system operation description

Lowering the use of petroleum products is one of the major goals of the twenty-first century. Gasoline-powered vehicles account for a large amount of the world's petroleum use, and so electric vehicles are being considered as a possible replacement technology. These vehicles have an electric motor rather than a gasoline engine, and a battery to store the energy required to move the vehicle. Governments and automotive companies have recognized the value of these vehicles in helping the environment (Hacker et al., 2009), and are encouraging the ownership of electric vehicles through economic incentives. For many electric vehicles, such as the Nissan LEAF or Chevrolet VOLT, the method of recharging the vehicle battery is to plug the battery into the power grid at places like the home or office (Bakker, 2011). Because the battery requires multiple hours to fully recharge, this method has the implicit assumption that vehicle will be used only for driving short distances. Electric vehicle companies are trying to overcome this limited range requirement with fast charging stations; locations where a vehicle can be charged in less than an hour to near full capacity. Besides being much more costly to operate rapid recharge stations, the electric vehicles still take more time to recharge than a standard gasoline vehicle would take to refuel (Botsford and Szczepanek, 2009). These inherent problems, combined with a lack of refueling infrastructure, are inhibiting a

* Corresponding author. Tel.: +1 401 829 1768.
E-mail addresses: jonadler@asu.edu (J.D. Adler), pitu@asu.edu (P.B. Mirchandani).
[1] Tel.: +1 480 965 2758.

http://dx.doi.org/10.1016/j.trb.2014.09.005
0191-2615/

wide-scale adoption of electric vehicles. The problems are especially apparent in longer trips, or inter-city trips. *Range anxiety*, when the driver is concerned that the vehicle will run out of charge before reaching the destination, is a major hindrance for the market penetration of electric vehicles (Yu et al., 2011).

Another electric vehicle recharging method is to construct *battery-exchange stations*. These stations will remove a battery that is nearly depleted from a vehicle and replace the battery with one that has already been charged (Shemer, 2012). Once the depleted battery is dropped off at the station it is charged until full so that a different vehicle can use it in the future. This method of refueling electric vehicles has the advantage that it is very quick for each vehicle since the driver only has to wait for the battery to be swapped and not for the battery to be charged. An issue with this approach is that all of the vehicles serviced by the battery-exchange station are required to use the identical batteries. As with other new technologies introduced into the market, it is likely that the developers of these batteries will coalesce around a single common standard, as has been the case for other car parts such as tires and wipers. Additionally, the infrastructure of battery-exchange stations is expensive to build, since each station requires battery swapping machines, battery chargers, and many extra batteries on hand. In conjunction to the battery-exchange concept, it is assumed that there exists a viable business model that provides a reasonable profit for companies that establish battery-exchange facilities for the public. Battery-exchange stations have been implemented in the countries of Israel and Denmark by the company Better Place (2013), but unfortunately they have yet to be profitable (Kershner, 2013) and this has forced the company into bankruptcy. The electric vehicle company Tesla Motors Inc., which currently sells plug-in electric vehicles, has recently shown a prototype vehicle with battery-exchange technology (Motavalli, 2013).

Since the construction of battery-exchange stations and their infrastructure is very expensive, they have only been placed in a limited number of locations so far. In addition, the extra batteries that are stored in the station are also expensive, so to keep inventory costs low it is best to stock the stations with as few batteries as required. The number of batteries needed by vehicles visiting the station throughout the day depends on the location of the station, the day of the week, the weather, and many other variables. Further, the manner in which the vehicles arrive affects the number of batteries needed; if the vehicles all tend to arrive around the same time then more batteries are needed since there will not be enough time for them to recharge after being dropped off. Since the exact distribution of cars that will pass through a station is unknown when the stations are constructed, each station has to be designed to handle near the maximum demand that station is expected to receive at any given time. In the event that no full batteries are available at a station, an arriving vehicle will have to wait until a battery is charged before it can leave the station. If a driver of a vehicle is informed in advance that there will be no available batteries at a station they planned to stop at, then that station could be avoided by taking a circuitous route. The decision on how many batteries to place at a station has to balance the company's desire to minimize costs with need for drivers to not have to wait when there are no batteries available. It is possible that by balancing the vehicles across many stations, fewer vehicles would arrive at stations when no batteries are available which would improve the service they receive.

The goal of this paper is to devise an algorithm for real-time routing of electric vehicles with swappable batteries that balances the desire for drivers to have quick trips with the need for the operating company to control the battery swap loads across stations. Further, the algorithm will make reservations for each vehicle at all of the battery-exchange stations on the desired routes. Making reservations will remove the possibility that the batteries the vehicle expect to receive are unavailable due to other vehicles taking them. The objective of the routing and reservation algorithm is to minimize the total expected travel times of not only the vehicle being routed, but of future vehicles as well. Because of this objective, part of the routing and reservation process is to understand how a set of battery reservations could affect future arrivals into the system.

This paper assumes that there is a viable business model for the development and operation of a network of battery-exchange systems. Thus, for our purposes we assume that the stations have already been located and built to hold a set number of batteries that are constantly being charged. In practice, an algorithm like this would require onboard computer units in each vehicle that communicated with a central server that does the routing and reservations. When a fleet of cars is produced to be compatible with a set of battery-exchange stations, the operating company of the stations will likely be involved in the design and production of the vehicles. In the case of the company Better Place, they collaborated with Renault Fluence Z.E. so that their vehicles could be used in the network (Kershner, 2013). Because of this involvement, the operating company can ensure that each vehicle has a compatible unit installed that communicates with the operating company's central server. The operating company could implement a robust routing and reservation software system that provides to each vehicle a route from its origin to its destination, along with where to stop to swap the battery so the vehicle does not run out of charge. At the time the system provides a route, the system would also make reservations at the battery-exchange stations for the batteries needed by the vehicle.

The routing and reservation system would make the route suggestion based on the current battery charge levels at each station along with the pre-existing reservations made by earlier vehicles, which would be stored in the central server. The model in this paper assumes that when the vehicle turns on it will be provided with a single route from the routing and reservation system, and that the driver will take the given route exactly. The steps in this routing and reservation process for each vehicle would be:

1. When the electric vehicle is turned on, the driver would input a destination into the vehicle's computer unit. This, combined with the origin of the trip determined by the GPS location of the vehicle, would be sent to the central server.

2. The central server receives this origin and destination (OD) pair and, using the current battery levels at the stations and the reservations already made, determines which route the vehicle should take and when and where the vehicle should stop to swap its battery.
3. The central server makes reservations for the batteries at each of the stations for the most convenient times the vehicle would require it, subject to availability.
4. The central server sends the selected route and reservation times to the unit onboard the vehicle, and the driver begins to travel the route.

We assume that since the batteries are reserved the vehicles will always find them available precisely when they were reserved for. We do not allow for any randomness in the amount of time it takes to traverse a road, nor that the driver of a vehicle can change its route partway through. Thus the road network itself is fully deterministic, although the arrivals of vehicles into the system are stochastic. Since many different vehicles will be routed, this is an online system that is fully aware of all of the routes taken by the vehicles that have previously arrived in the system. Although future routing and reservation decisions are not known at the time a new route is given, the rate of demand for each origin and destination pair is known and assumed to be fixed over time.

In this research we assume that the goal is to find routes for vehicles that take the least amount of time (as opposed to other metrics such as distance traveled). The amount of time it takes to travel a route is dependent on three components:

1. the time spent with the vehicle driving along roads,
2. the time spent having the battery swapped at each station, and
3. the time spent having the vehicle wait at a station for a battery to become available.

Although the summation of these times can be found directly to compute the travel time, it is also possible that some of these time components are considered worse for drivers than others and a weighted sum is more appropriate. For instance, waiting at a station for a battery to become available could be more frustrating than driving, so for example every minute waiting for a battery could be worth two minutes of driving time. We refer to the weighted sum of the travel time as the *value* of the route.

The simplest routing and reservation software could be designed using a *greedy algorithm* which would find the best route for each particular driver. It may however be in the best interest of the company to suggest that some vehicles take slightly longer routes so that other vehicles have much shorter routes. In this case the company is trying to minimize the average route values across all vehicles, rather than trying to minimize each trip individually. For example consider the road network in Fig. 1 where the edge lengths denote travel times. In this network the electric vehicles have a range of 12 time units, and battery swapping is instantaneous. Here there are two OD pairs for vehicles to take. For vehicles driving from Origin 1 to Destination 1 there is a single route of value 20 that stops at Station 1. For vehicles driving between Origin 2 and Destination 2 there are two routes: one stopping at Station 1 which has a value of 20 and one Stopping at station 2 which has a value of 22. Also assume that Station 1 has a single battery on hand, while station 2 has many batteries available for swaps.

Suppose a vehicle arrives wanting to travel from Origin 2 to Destination 2 and there is an available battery at both Station 1 and Station 2. The shortest route for the vehicle would be to stop at Station 1, however that would require the vehicle to take the only battery at the station. If immediately after routing this vehicle a second vehicle arrived wanting to travel between Origin 1 and Destination 1, it would have no choice but to wait for the battery at Station 1 to become available. Thus, the best global decision may be to have the first vehicle stop at Station 2 taking the slightly longer path, which allows a future vehicle to get a battery at Station 1 without having to wait. This problem of online routing of vehicles to minimize total route value is the focus of this paper.

We will model the system as a *Markov Chance-Decision Process* (MCDP) where the states describe the current reservations at the stations and the actions are routing the vehicles that arrive. An MCDP, which is presented in Hentenryck et al. (2009),

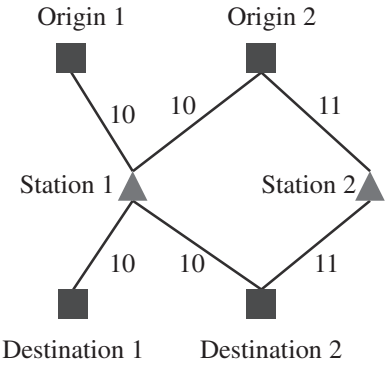

Fig. 1. An example route network with two origin and destination pairs.

is a modification of the standard *Markov Decision Process* (MDP) (Russell and Norvig, 2009). In a MDP, an *agent* moves between *states* by taking *actions* which move the agent to each state with a certain probability. When the agent is in a state a *policy* determines an action for the agent to take. In an MDP the uncertainty occurs after an action is selected and is involved with determining which state the agent will end up in. In an MCDP when the agent is at a state first a random outcome is selected. The outcomes determine which possible states the agent is allowed to transition to, and the agent may transition to any of them with certainty. Thus in an MCDP the uncertainty occurs before action is taken by randomly selecting which possible actions are available to the agent, and there is no uncertainty once the action is chosen.

In the case of routing electric vehicles through a network with reservations, during each interval of time it is unknown whether or not a vehicle will arrive into the system needing to be routed, and if one does arrive what its OD pair will be. When a vehicle does arrive, its OD pair becomes known and the algorithm routes it precisely, and given the routing and reservation decisions made the new state of the system is then exactly known. Then one time step occurs and a new vehicle may arrive; this process is repeated until the end of the day. Note that the time periods are chosen to be small enough so that two or more vehicles do not arrive at the same time, as in a Poisson process, which is a common practice when modeling possible arrivals over time. An area where this practice is often done is in dynamic revenue management for industries such as airlines, where the price of the product varies as people buy the product (e.g. Lee and Hersh, 1993).

The optimal policy for routing may be very complicated and depend on not only if any batteries are available at the different stations, but how exactly many batteries are available and when they are available. Finding the optimal policy for routing vehicles is intractable due to the large number of possible options for routing and reservations over the course of a day, and thus our best hope is to find an approximate solution. We provide a solution using the Approximate Dynamic Programming (ADP) technique of linear temporal differencing (Powell, 2011).

Approximate dynamic programming techniques have been used extensively in transportation literature for many different applications. It has been used successfully for managing fleets of vehicles that have to serve tasks (Powell et al., 2012; Simão et al., 2009; Topaloglu and Powell, 2006). Maxwell et al. used ADP methods to determine where to redeploy ambulances to maximize coverage in a dynamic system (Maxwell et al., 2010). Zhang and Adelman use ADP methods for revenue management of a network of airline flights (Zhang and Adelman, 2009). ADP methods have been used for allocating empty shipping containers in the cargo industry (Lam et al., 2007).

The problem of routing a single electric vehicle through a network where the vehicle has a limited range and must stop at a charging or battery-exchange station was first discussed by Ichimori et al. (1981). The paper solves for the case where the amount of battery charge consumed by the vehicle is proportional to distance it travels. The work was extended to limit the number of stops by Adler et al. in Adler et al. (2014). In the case that the battery consumed along a stretch of road is not dependent on the distance, for example in the case when it also depends on the speed limit of the road, then the problem becomes NP-hard (Laporte and Pascoal, 2011; Smith et al., 2012). The problem of routing an electric vehicle has also been discussed for the situation where the vehicle can recharge on certain roads, like when it is going downhill (Sachenbacher et al., 2011). We note that when the station sizes are not limited by the number of available batteries, then each arriving electric vehicle will be routed using the shortest path methods discussed above since they will not need to compete for available batteries.

The problem of routing multiple electric vehicles to stations as they arrive has been discussed by de Weerdt et al. in De Weerdt et al. (2013) where they route the vehicles based on the forecasted future demand for the day as well as the current vehicles in the system. The algorithm they suggest still only finds the optimal route for each vehicle individually rather than trying to minimize the total route times of all the vehicles. Approximate dynamic programming has been used to find when to charge the batteries at a battery-exchange station (Worley and Klabjan, 2011). Here vehicles arrive at a single battery-exchange station and the batteries are charged at different times to minimize the cost of electricity used.

While in this paper the battery-exchange stations are assumed to have already been placed and have a set number of batteries, the problem of locating the electric vehicle charging stations given fixed demand has been active in operations research. Kuby formulated the flow-refueling location problem which places the stations to try and cover the most OD demand (Kuby, 2005), which has lead to a number of extensions (Kim and Kuby, 2013, 2012; Kuby and Lim, 2007) such as having stations along arcs, having stations with limited capacity, and allowing vehicle detours. He et al. have solved a deterministic routing problem for plug-in hybrid vehicles, vehicles that use both electricity and a gasoline engine, where the problem has drivers recharging their batteries based on the location of the stations at the destinations and the price of charging (He et al., 2013). The problem of placing and sizing the battery-exchange stations while simultaneously finding the routes the vehicles will take between OD pairs has been discussed by Mak et al. (2013). Their analysis assumed that the route between an OD pair is fixed after construction of the stations, while we allow for vehicle routes to change throughout the day as different stations run out of batteries. Nie and Ghamami investigated what size the batteries should and where the stations should be placed be for electric vehicles traveling along a lengthy route with recharging stations (Nie and Ghamami, 2013).

The paper is arranged as follows. The problem is formalized in Section 2 where the problem input and output are defined. In Section 3.1 we show how the problem can be transformed into a Markov chance-decision process. Since the MCDP painfully suffers from the curse of dimensionality, Section 3.2 shows how the problem can be approximated using temporal differencing of linear models. Finally, in Section 4 we illustrate the use of our algorithm to find the approximate solution for the Arizona road network assuming different amounts of demand.

2. Problem description

2.1. Problem input

The road network is represented as an undirected graph $G = (V, E)$ where V is a set of vertices representing intersections and E is a set of undirected edges representing the roads in the network. Placed at certain predefined intersections in the network are n battery-exchange stations. Let $\{b_1, b_2, \ldots, b_n\} = B \subseteq V$ be the intersections that have battery-exchange stations. Each station b_i has, as an input to the problem, x_i batteries for $i = 1, \ldots, n$. It is likely that stations placed in more heavily trafficked locations will have more batteries, so we do not assume that all of the stations have the same number of batteries. We are interested in routing many different vehicles that have different possible origins and destinations. Let $\{(o_1, d_1), (o_2, d_2), \ldots, (o_m, d_m)\} \subseteq V \times V$ be the set of m possible origin and destination pairs, so each pair represents one possible combination of a starting point and an ending point. Let $O = \{o_1, o_2, \ldots, o_m\}$ and $D = \{d_1, d_2, \ldots, d_m\}$. We assume that $o_j \neq d_j$ for any OD pair j, since in that case the vehicles would not travel at all. For an example network see Fig. 2 which contains three OD pairs. Notice that while in Fig. 2 each node is used in at most one OD pair, it is possible that a node can be in multiple OD pairs and it can also be the location of battery-exchange station simultaneously.

The time period in which the battery-exchange stations are open is split into $T + 1$ discrete units $\{0, \ldots, , T\}$. Associate with each edge e a nonnegative integer length $l(e)$ representing the number of time units it takes for a vehicle to traverse the road. Thus in this model the amount of time it takes to traverse a road and the amount of battery charged used are proportional to each other. This is a simplifying assumption since in practice the efficiency of the battery depends on many factors such as the speed limit of the road and current traffic. The assumption was made since it allows for the set of possible routes to be much more quickly found; in the case that the travel time and battery depletion rate are independent functions finding a route through the network is NP-hard.

Let $l'(v_1, v_2)$ for $v_1, v_2 \in V$ be a function that returns the minimum path length in G between two vertices, which can be computed using Dijkstra's algorithm. The battery capacity of the electric vehicles in the network is an integer k representing the number of time units the vehicle can travel between battery swaps. Each vehicle starts at its origin with a full battery. The act of swapping the battery takes g time units. When a battery is dropped off at a station it takes $h(t)$ time units to fill to fully charged, where $h(t)$ is a monotonic increasing function of the amount of battery power used by the vehicle since it last received a fresh battery. In this case, since the battery used is directly proportional to the time the vehicle has spent driving since the last battery-exchange station, the battery used can be measured by the length of the path taken from the previous swapping station. Only once the battery is full may it be swapped into a new vehicle. At the start of each time interval at most one vehicle can arrive in the system. Assume that no vehicle arrives into the system with probability p_0, and a vehicle arrives wanting to travel from o_j to d_j with probability p_j for each OD pair $j = 1, \ldots, m$. Since these probabilities cover all of the possible arrival cases, we have that $\sum_{j=0}^{m} p_j = 1$. Also assume that each OD pair will require stopping to exchange batteries at least once on the way, otherwise they do not need to be considered in the model.

For each OD pair j, let $\mathcal{R}_j$ be the set of routes that the vehicle is allowed to travel between o_j and d_j. Each route $R \in \mathcal{R}_j$ is a sequence $\left(b_1^R, b_2^R, \ldots, b_{|R|}^R\right)$ of battery-exchange stations that the vehicle will visit between the origin and destination in the order the stations will be visited. Thus the length of the shortest path between the stations in the sequence $l'\left(b_i^R, b_{i+1}^R\right)$ for $i = 1, \ldots, |R| - 1$ must each be at most k since the vehicle must be able to travel between the stations without running out of charge. Further, the shortest path between o_j and the first station in the sequence and the shortest path between d_j and the

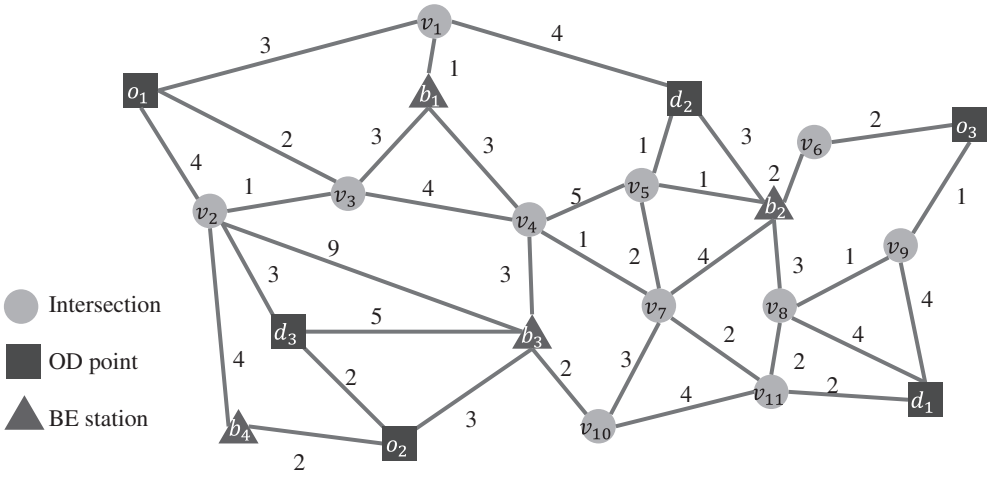

Fig. 2. An example network with OD pairs and battery-exchange stations.

last station in the sequence must both have length at most k. If a vehicle is assigned a route where the vehicle travels between two battery-exchange stops, then there is a shortest path in G for the vehicle to take between those two stations, and the vehicle should always take that path. Thus the travel times between stations can be computed in advance and used for any routes that contain that pair of exchange-stations, and when making routing decisions the travel times between stations can be found by doing a table lookup. For example for the network in Fig. 2 with $k = 10$, for the OD pair (o_1, d_1) one possible set $\mathcal{R}_1$ is $\mathcal{R}_1 = \{(b_1, b_2), (b_3)\}$. In this case the first sequence represents taking the path $o_1 - v_1 - b_1$ with minimal length $l'(o_1, b_1) = 4$, swapping for the first time, followed by path $b_1 - v_1 - d_2 - b_2$ with minimal length $l'(b_1, b_2) = 8$, swapping for the second time, then getting to the destination by $b_2 - v_8 - d_1$ with minimal length $l'(b_2, d_1) = 7$.

The set $\mathcal{R}_j$ for each OD pair j is an input to the model and may be a subset of all feasible routes. The route sets may be restricted to avoid allowing routes that have too high values or too many stops. For example with the network in Fig. 2 another possible value for $\mathcal{R}_1$ is $\{(b_1, b_2), (b_3), (b_4, b_3)\}$, which has the additional sequence where the vehicle stops at station b_4 before visiting b_3. Since the sequence $(b_4, , b_3)$ is more circuitous than (b_3) which visits station b_3 directly, it may be best to exclude it. Let the single element set $\mathcal{R}_0 = \{\emptyset\}$ represent the possible routes to take when no vehicle arrives to be routed: since there are no decisions to make when no vehicles arrive, the only possible choice is to not visit any stations.

While there are many possible methods to generate each $\mathcal{R}_j$ set, one procedure to generate the $\mathcal{R}_j$ sets for each OD pair j is the following. For each pair j, find the shortest feasible route using the algorithm from Adler et al. (2014), and let q_j be the number of stops on the shortest path. The algorithm will also generate a meta-network $G'_j = \left(V'_j, E'_j\right)$ where $V'_j = B \cup \{o_j, d_j\}$ and there is an edge between the vertices if the vehicle can travel between the two points in the road network on a single charge. Then starting at o_j, create $\mathcal{R}_j$ using a decision tree K from the pseudocode in Algorithm 1. The algorithm enumerates the possible paths from the origin, where any path that reaches the destination gets added to $\mathcal{R}_j$. The algorithm only considers paths that have at most y_j stops to avoid arbitrarily long paths, where y_j is an input. Unfortunately, this enumeration could in the worst case take $\mathcal{O}(B^{y_j})$ time, which is exponential in the number of stops allowed, however these sets will be computed in advanced. Further, if due to the size of the network this would be too time consuming other algorithms could be used to generate only a smaller subset of the possible full set of routes.

The battery-exchange stations take reservations throughout the day, and the number of reservations in the system effects the delays the vehicles will receive. For now, we use S_t to represent the reservations that the battery-exchange stations have received at time t, however this will be defined more formally in Section 3.1.2. Let $\rho_1 \geqslant 0$ be the penalty multiplier to swapping time incurred and let $\rho_2 \geqslant 0$ be the penalty multiplier to waiting time incurred. If a vehicle arrives at time t and assigned a route R_t and the system has received S_t reservations, we denote the different times that the vehicle takes along the route as the following: $\psi^{drive}(R_t, S_t)$ is the amount of time units spent driving, $\psi_i^{\text{swap}}(R_t, S_t)$ is the amount of time units swapping batteries at station i, and $\psi_i^{\text{wait}}(R_t, S_t)$ is the amount of time units waiting at station i. We say the route assignment itself has value

$$\psi(R_t, S_t) = \psi^{\text{drive}}(R_t, S_t) + \rho_1 \sum_{i=1}^{n} \psi_i^{\text{swap}}(R_t, S_t) + \rho_2 \sum_{i=1}^{n} \psi_i^{\text{wait}}(R_t, S_t).$$

Note that this value depends not only on the route, but also the current battery reservations at the different stations in the system. Thus ψ is a function of the route being assigned and the system itself.

Having the penalty for the time that the vehicle is waiting at the station or swapping be greater than when the vehicle is travelling may lead to some unusual situations. For instance, it may suggest that drivers take a longer route to the charging station, just so that when the driver arrives there is less of a wait time. In essence the driver would still be waiting, only the wait would be incurred while the vehicle is driving around; the driver still gets the new battery at the same time. This may be warranted, since there is research to suggest that having to have a vehicle stop and wait causes impatience and discomfort in drivers (Naveteur et al., 2013).

Algorithm 1. The pseudocode to generate the set of routes $\mathcal{R}_j$.

On input G'_j, y_j
Initialize $V_K \leftarrow \{(o_j, 0)\}, E_K \leftarrow \emptyset, K \leftarrow (V_K, E_K), \mathcal{R}_j \leftarrow \emptyset$
For $w \leftarrow 0, \ldots, y_j$
  For each $\{(v, w) | (v, w) \in V_k, v \neq d_j\}$
    For each $\{v' | (v, v') \in E'_j\}$
    Set $V_k \leftarrow V_k \cup \{(v', w+1)\}, E_k \leftarrow E_k \cup \left\{((v, w), (v', w+1))\right\}$
    If $v' = d_j$
      Set z as the path between $(v', w+1)$ and $(o_j, 0)$
      Set $\mathcal{R}_j \leftarrow \mathcal{R}_j \cup \{z\}$
    End if
  End for
End for
Return $\mathcal{R}_j$

For each OD pair j and corresponding route set $\mathcal{R}_j$, let L_j be the minimal value of routes in set $\mathcal{R}_j$ assuming each station has an available battery at any time. For example assume that for the network in Fig. 2 we have that $g = 2, \rho_1 = 2$, and $\rho_2 = 4$. Using set $\mathcal{R}_1$ defined above, the value $L_1 = 9 + 2 \cdot 2 + 8 = 21$, since the minimal time under the best conditions would be found using the second sequence (b_3) which has a shortest path of length 9, one swap taking 2 time units (weighted twice as heavily by ρ_1), then a final path of length 8. The value L_j can be thought of as the best case value for a vehicle to travel between OD pair (o_j, d_j). For a vehicle traveling OD pair j assigned at time t with a route R_t value of $\psi(R_t, S_t)$, the *delay penalty* $\psi(R_t, S_t) - L_j$ is the increased value of the route that is assigned compared to the best case value.

2.2. Reservation policy

When a vehicle arrives into the system wanting to traverse OD pair j, it is immediately given a route $R \in \mathcal{R}_j$ to take from o_j to d_j. When the vehicle is assigned the route a battery reservation will be made at each of the battery-exchange stations it will visit. The reservation will state at what time the vehicle will be picking up the battery. A battery can only be reserved for a time when (1) there is a battery available at that time, and (2) the battery to be picked up will not be needed by a vehicle before the battery to be dropped off will be refilled. The second condition ensures that the reservation does not conflict with one already in place. For example, suppose that in the network in Fig. 2 both station b_1 and b_2 have a single battery and again $g = 2$ and $h(t) = 6$ (so the time to charge a battery to full is constant). If a vehicle using pair (o_1, d_1) wants to reserve the battery at station b_1 at $t = 120$, then that battery must be available not only at $t = 120$ but also $t = 121, t = 122, \ldots, t = 127$. Only at time $t = 128$ will the battery of the vehicle be ready to be put into a new one. Having the battery be available between $t = 120$ and $t = 127$ will ensure that no other vehicle can reserve the battery when it is being swapped or when it is charging. The reservation need not be made for the exact time the vehicle arrives, only the earliest available time after a vehicle arrives. That is to say the vehicle may reserve a battery for a later than when it would arrive at the station if taking a battery that is available earlier would violate a previous reservation, however we assume a vehicle will never voluntarily wait if there is a battery available that would not cause a conflict with another reservation. If a vehicle arrives at a station after time T, which for instance could occur if the vehicle turns on to be routed at time $t = T - 1$, assume there will be a battery immediately available at the station for it.

For a more in depth example consider the network in Fig. 2 and a vehicle travelling from o_1 to d_1 wanting to use route (b_1, b_2) and suppose the vehicle is being routed at time $t = 33$. Assume that the two stations each have a single battery, and station b_1 already has a reservation at time 32 and station b_2 has a reservation at time 52. At the time of the assignment the reservations for batteries of the vehicle being routed have to be made at the two stations. Leaving at $t = 33$, the vehicle will arrive at vertex b_1 at time 37. Since the battery at station b_1 is already reserved by another vehicle at time 32, it is therefore not available between times 32 through 39 (due to swapping and charging time of the battery being dropped off by the earlier reservation). Thus a reservation is made at time 40, and with this delay plus the 2 time periods to swap the battery, the vehicle will be done at station b_1 at the end of time period 41. Thus with the 7 time units it takes to travel between b_1 and b_2, the vehicle will arrive at station b_2 at the start of time period 48. The battery at station b_2 is full and available at time 48, but another vehicle already has a reservation at time 52. Thus a reservation cannot be made at time 48 since the battery would then be unavailable to the car arriving at time 52. Since it takes 8 time units to swap and charge, the reservation has to be made at time 60. Therefore the vehicle will leave the station at location b_2 at the start of time period 62 (since it will swap during periods 60 and 61) and will arrive at the destination at time 69. If the waiting penalty and swap penalty are 2 and 4 respectively, the value of this trip is:

$$\begin{aligned}\psi(R_{33}, S_{33}) &= \psi^{\text{drive}}(R_{33}, S_{33}) + \rho_1 \sum_{i=1}^{n} \psi_i^{\text{swap}}(R_{33}, S_{33}) + \rho_2 \sum_{i=1}^{n} \psi_i^{\text{wait}}(R_{33}, S_{33}) \\ &= 18 + 2 \cdot (2 + 2 + 0 + 0) + 4 \cdot (3 + 12 + 0 + 0) = 86.\end{aligned}$$

This calculation can be done entirely before the route has begun to be traversed, and so once the vehicle sets out it is exactly known how long the trip will take including waiting at stations.

2.3. Problem output

The solution to the problem is a function Υ that, upon arrival of a vehicle at time t wanting to travel between an OD pair j_t, takes the current battery levels of the stations and the already made reservations and outputs a route that minimizes:

$$\mathbb{E}_t[\psi(\Upsilon(R_t, S_t), S_t) + \mathbb{E}_{t+1}[\psi(\Upsilon(R_{t+1}, S_{t+1}), S_{t+1}) + \mathbb{E}_{t+2}[\cdots + \mathbb{E}_T[\psi(\Upsilon(R_T, S_T), S_T)]]]].$$

Thus the objective is to not only minimize the value of the trip the next vehicle is about to take, but also the value of the trips future vehicles may take. Notice that this is equivalent to minimizing the delay penalties, the differences between the trip values and the best case trip values, on each trip between t and T:

$$\mathbb{E}_t\left[\left(\psi(\Upsilon(R_t, S_t), S_t) - L_{j_t}\right) + \mathbb{E}_{t+1}\left[\left(\psi(\Upsilon(R_{t+1}, S_{t+1}), S_{t+1}) - L_{j_{t+1}}\right) + \mathbb{E}_{t+2}\left[\cdots + \mathrm{E}_T\left[\psi(\Upsilon(R_T, S_T), S_T) - L_{j_T}\right]\right]\right]\right].$$

We will use this equivalent objective function. Unfortunately finding the exact solution is intractable for all but the smallest cases, and so we will provide an approximate solution.

3. Solution theory

3.1. Markov chance-decision process

3.1.1. Description

For a vehicle wanting to travel between an OD pair there may be many different routes the vehicle should take depending on the station conditions. When a vehicle arrives all of the reservations that have already been made by other vehicles are known, so the actual routes the previous vehicles have taken are irrelevant. That is to say the best route for the vehicle to take only depends on the current reservations and battery levels at the station at the time of routing and not on previous decisions made. In this way the system is Markovian. Thus, we will model the system using a specialized Markov Decision Process (MDP) called a Markov Chance-Decision Problem (MCDP). While MCDPs and MDPs have the same expressive power, in certain situations the MCDP has easier computations. The MCDP model fits especially well into the framework of this problem since the chance and the decision have clear analogs to the vehicles being routed.

A Markov Chance-Decision Problem (MCDP) is formally defined as a tuple $(\mathcal{S}, S_0, \Xi, Z, C)$ where:

1. $\mathcal{S}$ is a finite set of states;
2. $S_0 \in \mathcal{S}$ is the initial state;
3. $\Xi = (\xi_t)_{t \geqslant 0}$ is the input process. It is a Markov chain with a state set J, an initial probability distribution $\mu \in prob(J)$, and a $J \times J$ transition matrix $\mathcal{U}$;
4. $Y : \mathcal{S} \times J \to 2^{\mathcal{S}}$ is a transition mapping with $Y(S, j) \neq \emptyset$ for all $S \in \mathcal{S}$ and $j \in J$; and
5. $C : S \times J \times S \to \mathbb{R}$ is the cost map.

In a MCDP the agent starts at initial state S_0, and at the start of each time period t, the agent is in state S_t. At time t the Markov chain Ξ transitions from state $j_{t-1} \in J$ by one step to a new state $j_t \in J$ using matrix $\mathcal{U}$. The agent then has the choice to move to any state $S_{t+1} \in Y(S_t, j_t)$ and receives a cost $C(S_t, j_t, S_t + 1)$. A visual representation of the progression of an MCDP can be seen in Fig. 3. In this figure the agent starts in a state S_0 and then a random outcome j_0 occurs. This leads to a set of new possible states $Y(S_0, j_0)$ of which $S_1 \in Y(S_0, j_0)$ is chosen for the next state. This repeats indefinitely for the infinite version of an MCDP, and for the finite time version of a MCDP, T transitions occur. The initial state of the input process is randomly distributed by μ. A *policy* for a MCDP is a mapping $\pi : \mathcal{S} \times J \to \mathcal{S}$ such that $\pi(S, j) \in Y(S, j)$ for all $S \in \mathcal{S}$ and $j \in J$. For a finite MCDP the objective is to find a policy that minimizes the expected sum of the costs taken from transitioning from S_0 over the course of the T transitions. Formally our objective is to find the policy π that is the solution to:

$$\min_{\pi}\left(\mathbb{E}_{j_1}\left[\min_{S_1 \in \pi(S_0 j_1)}\left(C(S_1, j_1, S_0) + \mathbb{E}_{j_2}\left[\min_{S_2 \in \pi(S_1 j_2)}\left(C(S_2, j_2, S_1) + \ldots \mathbb{E}_{j_T}\left[\min_{S_T \in \pi(S_{T-1} j_T)}(C(S_T, j_T, S_{T-1}))\right]\right)\right]\right)\right].$$

An optimal policy for a finite time MCDP can be found using *dynamic programming* (DP). In dynamic programming, the value of each state is determined by starting at the latest state in time and working backwards until the first time period. Define the value of a state $\mathcal{V}_t(S_t, j_{t-1})$ for states $S_t \in \mathcal{S}_t$ for $t = 1, \ldots, T-1$ as

$$\mathcal{V}_t(S_t, j_{t-1}) = \mathbb{E}_{j_t}\left[\min_{S_{t+1} \in Y(S_t, j_t)}(C(S_t, j_t, S_{t+1}) + \mathcal{V}_{t+1}(S_{t+1}))|j_{t-1}\right],$$

and for $S_T \in \mathcal{S}_T$ let $\mathcal{V}(S_T) = \mathbb{E}_{j_T}[\min_{S'_T \in Y(S_T, j_T)} C(S_T, j_T, S'_T)]$. These values can be calculated by first finding the values for $S_T \in \mathcal{S}_T$, then computing the values $S_{T-1} \in \mathcal{S}_{T-1}$ and so on. For a more in depth overview of dynamic programming see Bertsekas (2007).

3.1.2. MCDP for the network routing problem

In the case of this network routing problem, the state of the system depends on how many batteries are unavailable at each station during each time period future. Let t be the current time. At any time period $t, \ldots, T$ for a station b_i, between 0 and x_i batteries may be unavailable. As time passes less information needs to be stored since fewer time periods remain until the end of the day. Fig. 4 shows the state for a network with a single battery-exchange station with four batteries at time $t = 10$, and then the state after a vehicle makes a reservation and time passes to $t = 11$. In this figure it takes 2 time units to swap the battery and 6 to fully charge a used battery (so h is a constant function). Notice that what may have happened is a vehicle wanted a reservation at time 17, but the only two batteries available at time 17 would be needed by two other vehicles at times 19 and 20. Thus the reservation had to be postponed until time 22,

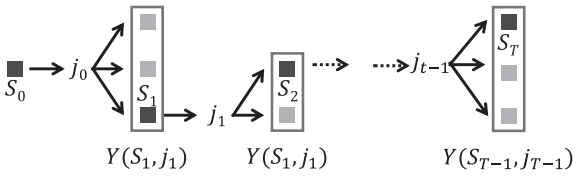

Fig. 3. The progression of states for a finite time MCDP.

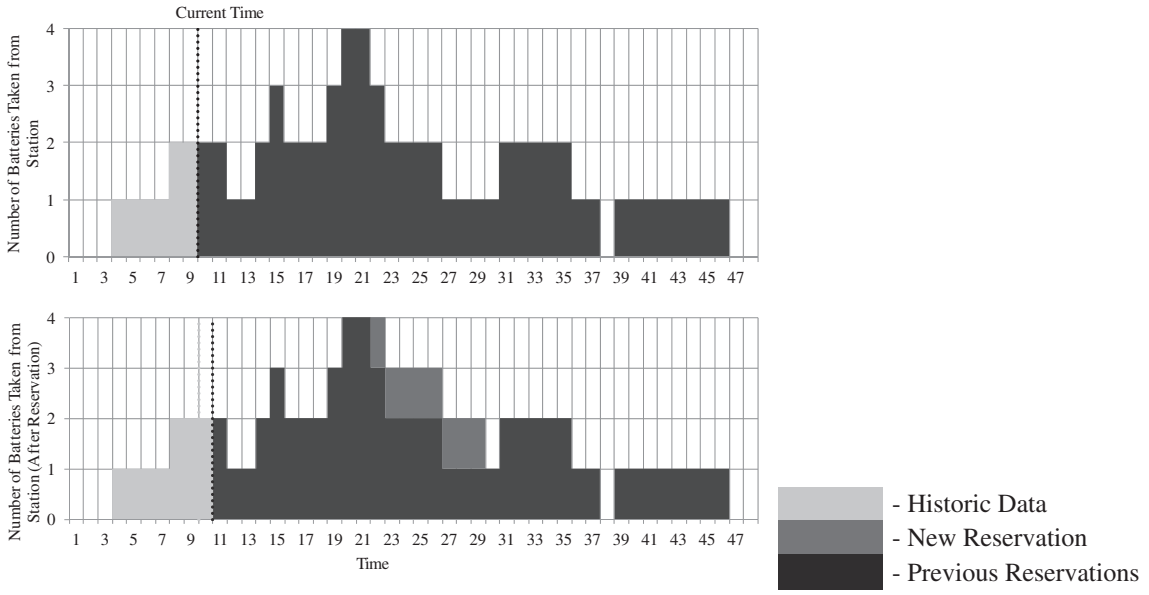

Fig. 4. An example of the state of a single station and the station after a vehicle makes a reservation and time steps by one unit.

and the vehicle would have to wait at the station for 5 extra time units. The reservation takes up 8 time units, 2 to actually do the swapping and 6 to charge.

At time period t for $t = 0, \ldots, T$, the possible states of station i are $\mathcal{S}_t^i = \{0, \ldots, x_i\}^{T-t}$, which represents the number of unavailable batteries at station i during each time period from t until the end of the day. We define the possible states of the whole system at time t for $t = 0, \ldots, T$ as $\mathcal{S}_t = \mathcal{S}_t^1 \times \mathcal{S}_t^2 \times \cdots \times \mathcal{S}_t^n$. The set $\mathcal{S} = \bigcup_{t=0}^{T} \mathcal{S}_t$ represents the possible states of the entire system. For convenience, for a state $S_t \in \mathcal{S}_t$ to refer to the number of batteries available at station i at time $\sqcup \geqslant t$ we use the notation $S_t(i, \sqcup)$. Also for convenience, define the function $\Lambda : \mathcal{S} \to \mathcal{S}$ as the function that maps $\mathcal{S}_t \to \mathcal{S}_{t+1}$ (progressing a state one step forward in time) by removing the first element of each $\mathcal{S}_t^i$. For an example state $(2, 3, 1, 0, 0, 1)$ where there is a single station and 7 time periods until the end of the day, $\Lambda((2, 3, 1, 0, 0, 1)) \mapsto (3, 1, 0, 0, 1)$.

The initial state S_0 represents the stations before any vehicles arrive, and thus S_0 is a tuple of $T \cdot n$ zeros. The input process $\Xi = (\xi_t)_t \geqslant 0$ determines which vehicles arrive at each time period. Thus the state set J is the possible desired OD pairs a vehicle could arrive wanting to travel: $J = \{0, 1, 2, \ldots, m\}$. Here the values $j = 1, \ldots, m$ represents a vehicle arriving wanting to travel from origin o_j to destination d_j, and the value $0 \in J$ represents no vehicle arriving at that time period. Since the vehicle arrivals at different times are independent, we can simply define the transition matrix $\mathcal{U}$ such that $\mathcal{U}_{j,j'} = p_j$ for all $j, j' \in 0, 1, \ldots, m$, and the μ vector is unnecessary.

The transition mapping Y will describe the possible ways a vehicle could be routed. An (S_t, j_t) pair represents the state of the system at time t and which vehicle has just arrived, and $Y(S_t, j_t)$ is the set of all possible states the system can transition to. Each $S_t + 1 \in Y(S_t, j_t)$ has a one-to-one correspondence with the routes in $\mathcal{R}_{j_t}$ since each possible route gives a new state that the system could transition to. In the case that $j_t = 0$, there is a single element in $Y(S_t, j_t)$, the state where no new reservations are made but time steps forward by one interval.

The exact mathematical formulation of transition function Y is cumbersome to define explicitly since it has to encompass the logic of when is the earliest a reservation could be made after a vehicle arrives as described in Section 2.2. It is however straightforward to write an algorithm that on input (S_t, j_t) gives set $Y(S_t, j_t)$. That algorithm can be seen in Algorithm 2. This algorithm also returns a matrix Z where each column corresponds to a route in $\mathcal{R}_{j_t}$. The column is a $(n+1)$-vector where the i th entry is in the column is the penalty incurred for waiting at station i and the $(n+1)$th entry is the time travelled by the vehicle plus the penalty for all of the battery swapping times. This matrix Z will be used by later calculations in the MCDP. In the algorithm, variable a tracks the time at which the vehicle arrives at each station in the route, w tracks how long the vehicle will wait at the current station. The variable $\bar{h}$ stores the amount of time to charge the battery that is dropped off.

The algorithm iterates through the stations along each route, and determining how long the vehicle will have to wait when it arrives at the station before it can leave with a new battery. Starting with the first station in the sequence, the algorithm checks if there will be no available batteries at any time between when the vehicle arrives assuming no wait, $(a + w)$ for $w = 0$, and up to when the battery the vehicle swapped out would be fully charged, which is $(a + w + g + \bar{h})$. If indeed there is a point in that interval where no batteries will be available then the vehicle must wait to get a battery. The value w' is added to the waiting time, where w' is the count of the number of instances in the time sequence where no batteries are available. The value w' provides an amount of time that is necessary, but not sufficient, to ensure the vehicle has waited enough for a battery to be available. The algorithm then checks if the updated wait time w is sufficient, and if not the process

is repeated until a valid waiting time is found. Once the algorithm determines how long the vehicle will wait at the station, the arrival time at the next station is calculated and the process continues for that station.

Algorithm 2. An algorithm for the mapping $Y : S \times J \to 2^s$ that provides the state of the system after a vehicle has been routed and also returns the route length and waiting times.

On input S_t, j_t
Initialize $Y \leftarrow \emptyset, a \leftarrow 0, w \leftarrow 0, i \leftarrow 0$,
Initialize S'_t as a $(T-t)$-tuple of zeros
Initialize Z as a $(n+1) \times |\mathcal{R}_{jt}|$-matrix
For each $R = \left(b_1^R, b_2^R, \ldots, b_{|R|}^R\right) \in \mathcal{R}_{j_t}$
  Set $i \leftarrow i+1; a \leftarrow l'\left(o_{j_t}, b_1^R\right); S_t \leftarrow S_t; Z(n+1,i) \leftarrow l'\left(o_{j_t}, b_1^R\right)$
  For $q \leftarrow 1, \ldots, |R|$
    Set $w \leftarrow 0$
    If $q = 1$ **then** $\bar{h} \leftarrow h\left(l'\left(o_{j_t}, b_1^R\right)\right)$ **else** $\bar{h} \leftarrow h\left(l'\left(b_{q-1}^R, b_q^R\right)\right)$
    While $S_t\left(b_q^R, y\right) \geqslant x_q^R$ **for any** $y \in \{a+w, \ldots, \min(a+w+g+\bar{h}-1, T)\}$
      Set $w \leftarrow w + w'$ where w' is the count of times $S_t\left(b_q^R, y\right) \geqslant x_q^R$ for $y \in \{a+w, \ldots, \min(a+w+g+\bar{h}-1, T)\}$
    End while
    Set $a \leftarrow a+w+g; Z(n+1,i) \leftarrow Z(n+1,i) + l'\left(b_q^R, b_{q+1}^R\right) + \rho_1 g;\ Z\left(b_q^R, i\right) \leftarrow \rho_2 w$
    For each $y \in \{a+w, \ldots, \min(a+w+g+\bar{h}-1, T)\}$
      Set $S'_t\left(b_q^R, y\right) \leftarrow S'_t\left(b_q^R, y\right) + 1$
    End for
    If $q < |R|$ **then**
      Set $a \leftarrow a + l'\left(b_q^R, b_{q+1}^R\right)$
      Set $Z(n+1,i) \leftarrow Z(n+1,i) + \left(b_y^R, b_{y+1}^R\right)$
    End if
  End for
  Set $Z(n+1,i) \leftarrow Z(n+1,i) + l'\left(b_y^R, d_{j_t}\right)$
  Replace column i of L with Z
  Set $Y \leftarrow Y \cup \wedge(S'_i)$
End for
Return Y, Z

We define the cost function $C : S \times J \times S \to \mathbb{R}$, which describes the cost of taking a transition from (S_t, j_t) to $Y(S_t, j_t)$. For (S_t, j_t) where $S_t \in \mathcal{S}_t$ and $j_t \in \{0, \ldots, m\}$, let $S'_t \in Y(S_t, j_t)$ correspond to taking path R which is column i in Z. The value of the route, denoted $\psi(R_t, S_t)$ is the value in the Algorithm 2 found by summing the column i of matrix Z, since the first n entries are the weighted wait times at the stations and the $(n+1)$th entry is the weighted time travelling and swapping. The cost function mapping is $C(S_t, j_t, S'_t) \mapsto \psi(R_t, S_t) - L_{j_t}$, which represents the delay penalty the vehicle will take if that route is assigned. If the vehicle were to take the shortest possible route and had no delays, the resulting cost would be the minimum value 0. Technically if $S'_t \notin Y(S_t, j_t)$ we then assign the cost $C(S_t, j_t, S'_t) \mapsto \infty$, although this is important only for ensuring C is a proper mapping. Further we also define cost function $C^1 : S \times J \times S \to \mathbb{R}$ as $C^1(S_t, j_t, S'_t) \mapsto \psi^{\text{drive}}(R_t, S_t) + \sum_i \psi_i^{\text{swap}}(R_t, S_t) - L_{j_t}$: the cost attributed to the vehicle taking a longer route and making addition battery swaps beyond the minimum (but not associated with waiting). Also define the cost functions $C_i^2 : S \times J \times S \to \mathbb{R}$ as $C_i^2(S_t, j_t, S'_t) \mapsto \psi_i^{\text{wait}}(R_t, S_t)$ for $i = 1, \ldots, n$: the cost attributed to the vehicle having to wait at station i. Thus cost C is equal to the summation $C^1 + C_1^2 + C_2^2 + \ldots + C_n^2$, which is the cost of travelling time and swapping time, plus the waiting incurred at each station. Each of these costs is stored in a different row in Z; the cost C^1 is the $(n+1)$th entry of Z and C_i^2 is the ith row of Z for $i = 1, \ldots, n$ The component cost functions $C^1, C_1^2, C_2^2, \ldots, C_n^2$ will be used for calculations later, but with only C we now have a full description of the MCDP for the network routing problem. The optimal policy for the MCDP will correspond to the function Υ that takes as input the state of the system and the vehicle that arrives and outputs the optimal route for that vehicle to take.

3.2. MCDP approximation using linear temporal differencing

As stated previously, for a Markov decision process the methods for finding the optimal policy typically rely on dynamic programming. These techniques compute the value of being in a particular state, where the value of a state is the expected

cost incurred after arriving in that state and having the agent continue in the Markov decision process. Once the values of each state are found, each decision can be made to minimize the sum of the expected value of the action plus the expected value of the next state. While value iteration and other techniques can be used on MCDPs as well as MDPs, in both cases the algorithms that find the optimal policy become intractable for a large number of states. The problem of routing and reserving batteries for electric vehicles has an immense number of states. In fact at time period t there are $\prod_{i=1}^{n}(x_i+1)^{T-t+1}$ possible states, so for example in a system with 10 stations each having 5 batteries at 20 time periods before the end of the day there are $(6_{20})^{10} \approx 4.27 \cdot 10^{155}$ states.

Thus, we need to find an approach that will grant a solution that may not be optimal but is still better than using a naïve policy such as the greedy routing of each vehicle. For that we turn to approximate dynamic programming. Here we will still attempt to find the value of each state of the MCDP, only now we will accept approximate solutions for the values of states. Notice that in the case of the network routing and reservation problem the value of a state represents the estimated delay penalties of all of the future arrivals.

3.2.1. Approximate dynamic programming

In *Approximate Dynamic Programming* (ADP) (Powell, 2011), rather than computing the exact value for each state, the values of the states are approximated. Unlike the standard dynamic programming algorithm which computes the values of each state backwards, typically in approximate dynamic programming the state values are given approximations and using the approximations the algorithm steps forward in time. Here we present the Q-learning ADP algorithm (Russell and Norvig, 2009) but modified for MCDPs instead of the classic MDP. Let $\mathcal{V}_t(S_t,j_t)$ be the value of being in state S_t at time t and then having the random outcome j_t occur. This represents the value of the state after the uncertain outcome j_t has happened. We could also measure the value of the state before the uncertainty (and decision making), which we will denote $\mathcal{V}_t(S_t|j_{t-1})$, since it depends on the previous state of the Markov chain. Let $\overline{\mathcal{V}}_t^{q-1}(S_t,j_t)$ be an approximation of $\mathcal{V}_t(S_t,j_t)$, and let $(j_0^q,j_2^q,\ldots,j_{T-1}^q)$ be a randomly generated realization of Ξ. Then we can compute a new and hopefully closer approximation $\overline{\mathcal{V}}_t^q(S_t,j_t)$ for $\mathcal{V}_t(S_t,j_t)$ by finding for each $t=0,\ldots,T$:

$$\hat{v}_t^q = \min_{S_{t+1}^q \in Y(S_t^q j_t^q)} \left(C(S_t^q, j_t^q, S_{t+1}) + \sum_{i\in J} \overline{\mathcal{V}}_{t+1}^{q-1}(S_{t+1},i) P(i|j_t^q) \right),$$

and letting S_{t+1}^q be the value that minimizes S_{t+1}. Then using a stepsize parameter $0<\alpha\leqslant 1$,

$$\overline{\mathcal{V}}_t^q(S_t,j_t) = \begin{cases} (1-\alpha_q)\overline{\mathcal{V}}_t^{q-1}(S_t,j_t) + \alpha_q \hat{v}_t^q & S_t = S_t^q \\ \overline{\mathcal{V}}_t^{q-1}(S_t,j_t) & \text{otherwise.} \end{cases}$$

The algorithm works by using an approximation of the value of the possible next states to determine which action to take. Since in an MCDP which chance outcome will occur at the next state is unknown, the possible outcomes are averaged using the probabilities of the outcomes. The values of the states before the uncertainty can then be computed as $\overline{\mathcal{V}}_t^q(S_t|j_{t-1}) = \sum_{i\in J}\overline{\mathcal{V}}_{t+1}^q(S_{t+1},i)P(i|j_{t-1})$. As new approximations are generated for the values of the states the approximations should hopefully get more accurate. The value q represents the iteration of the approximation.

The values of the states correspond to the expected costs that will be incurred until the final state is reached. In the case of the electric vehicle routing and reservation problem, these values are the sums of the expected delays that will be incurred by future vehicles arriving. Thus being in a state with a higher value is worse than a lower one, since it means there are more delays that are expected in the future.

3.2.2. Temporal difference updates

Suppose that the system is in state S_t^q (having previously seen j_{t-1}^q) and we are already given a policy π. Also suppose there is a given future set of outcomes $(j_0^q, j_{t+1}^q, \ldots, j_{T-1}^q)$. One way to compute the value of the state S_t^q would be to follow the policy until the end of the horizon at time T, this would give us an approximation for $\mathcal{V}_t(S_t^q|j_{t-1}^q)$ of

$$\hat{v}_t^q = C(S_t^q, j_t^q, S_{t+1}^q) + C(S_{t+1}^q, j_{t+1}^q, S_{t+2}^q) + \cdots + C(S_{T-1}^q, j_{T-1}^q, S_T^q),$$

where for $\ell = t,\ldots,T$, $S_{\ell+1}^q = \pi(S_\ell^q, j_t^q)$. Here the value $\hat{v}_t^q$ is sum of all of the future costs that will be incurred during the time horizon. The value of the $\mathcal{V}_t^{q+1}\left(S_t|j_t^{q+1}\right)$ could be set directly by $\overline{\mathcal{V}}_t^{q+1}(S_t^q|j_t^q) = (1-\alpha_{q+1})\overline{\mathcal{V}}_t^q(S_t^q|j_t^q) + \alpha_{q+1}\hat{v}_t^{q+1}$. Alternatively, define the *temporal difference* as

$$\delta_\ell^{\pi,q} = C(S_\ell^q, j_\ell^q, S_{\ell+1}^q) + \mathcal{V}_{\ell+1}^{q-1}(S_{\ell+1}^q|j_\ell) - \overline{\mathcal{V}}_\ell^{q-1}(S_\ell^q|j_{\ell-1}^q).$$

Then because the sum of the temporal differences are telescoping, it is equivalent to write

$$\hat{v}_t^q = \mathcal{V}_t^{q-1}(S_t^q|j_t^q) + \sum_{\ell=t}^{T}\delta_\ell^{\pi,q}$$

So the updated value is the sum of all of the differences to the current values. However this weighs the changes to the approximation that occur far into the future just as heavily as the changes that occur immediately after t. Since the earlier

changes are likely more dependent on the particular action we take (since the decision has more of a direct impact), the differences can be geometrically discounted with rate λ and thus:

$$\overline{V}_t^q(S_t^q|j_t^q) = \overline{V}_t^{q-1}(S_t^q|j_t^q) + \alpha_q \sum_{\ell=t}^{T} \lambda^{T-\ell} \delta_\ell^{\pi,q}$$

This method of updating the approximations is the classic temporal difference algorithm TD(λ) (Powell, 2011), with the slight modification due to using an MCDP instead of an MDP. We will use this algorithm for our particular MCDP. The value of λ is a parameter to the model that can be tuned to adjust the model effectiveness; many different ways of picking a lambda are discussed in Chapter 11 of Powell (2011). The λ parameter does not change the true values of the states, it only allows for the approximations of those values to hopefully converge to the correct values more quickly.

3.2.3. Temporal differences with a linear model

While it is possible to find the optimal policy to a finite horizon MCDP using dynamic programming TD(λ), the amount of states in the network routing MCDP makes this technique still intractable. Dynamic programming requires the calculation of the value of the system in each possible state, and each state can have a unique and independently calculated value. It is likely the case however that the values of the states are closely related. For instance in the case of the network routing problem the value of states S_t and $\Lambda(S_t)$ (i.e. no vehicle arriving and time progressing by one) should be fairly close to each other. If the values of the states are approximated in such a way that we utilize the similarities of the states, the calculation should become tractable. Thus by approximating the values of each state future by using linear functions the problem becomes feasible to solve.

For simplicity, since in our model the random variable ξ_t does not depend on ξ_{t-1}, we will denote the value of a state as $\overline{V}_t^q(S_t)$ instead of $\overline{V}_t^q(S_t|j_{t-1})$. For our problem we will approximate the state values using a linear model. The approximations of the state values will take the form of a linear model

$$\overline{V}_t^q(S_t) = \sum_{f\in\mathcal{F}} \theta_{tf}^q \phi_f(S_t),$$

where $\phi_f(S_t)$ are *basis functions* used to describe the state and θ_{tf}^q is a set of coefficients for the functions, which depend on the time t of the system and the approximation iteration q. The set $\mathcal{F}$ is the indexes all of the basis functions. Now instead of finding the best approximation for each state at each time, we are instead interested in finding the optimal coefficients $\theta_{f,t}^q$ which best approximate each state at each time. Notice that because this is a finite horizon model, there is a different set of coefficients for each time interval and hence the index t.

For each station $b_i \in B$ and for each value $j = 1, \ldots, x_i$, define the function $\phi_{(i,\,j)}$ which maps state $S \in \mathcal{S}$ to the number of time periods in which station b_i has at least j batteries reserved. Functions $\phi_{(i,1)}, \phi_{(i,2)}, \ldots, \phi_{(i,x_i)}$ capture the total minutes that the batteries at station i are reserved for. For example, if at station b_2 has two batteries in it ($x_2 = 2$), and at time $t = 30$ there is exactly one battery reservation which makes the battery unavailable during times 41, 42, ..., 48, then $\phi_{(i,\,1)} = 8$ and $\phi_{(i,\,2)} = 0$. Also define the function $\phi_0(S_t) = T - t$, which determines the number of time periods left until the end of the day. Therefore for this network problem $\mathcal{F} = \{0, (1,1), \ldots, (1,x_1), (2,1), \ldots, (n,x_n)\}$. Additionally, since we know that the costs associated with routing a vehicle increase when some batteries are unavailable, as the values for these basis functions increase so too should the value of the state. Having fewer batteries available is always worse than having more batteries available, so we can restrict the coefficients so that $\theta_{tf}^q \geqslant 0$ for all $t \in T, f \in \mathcal{F}$ and all q.

When using the functions to describe a station, we lose exactly when the batteries are reserved: we may know that there are 8 time periods in the future where 1 battery is unavailable, but we do not know when those 8 periods fall (and the time may be split into multiple segments). The advantage of using these basis functions is that they dramatically lower the dimensionality of the state space. Instead of a state having a value for each station and for each time period until the end of the day there are only $|\mathcal{F}| = 1 + \sum_{i=1}^{n} x_i$ values that need to be stored.

This selection of basis functions used to describe the states have that added property that they are scale appropriately with time. For example if between a certain time t and the end of the day there are 20 time periods where station 3 has all 5 of its batteries reserved, what the particular value of t is does not matter for the sum of the vehicle delays that are expected to occur. Thus, instead of computing parameters θ_{tf}^q for each $t \in \{0, \ldots, T\}$ and $f \in \mathcal{F}$ we can instead compute a single value θ_f^q for each $f \in \mathcal{F}$ and assume that $\theta_{tf}^q = \theta_f^q$ for all $t \in \{0, \ldots, T\}$. This greatly reduces the number of coefficients that need to be calculated.

There is one further complication to the computation. The equation $\overline{V}_t^q(S_t) = \sum_{f\in\mathcal{F}} \theta_f^q \phi_f(S_t)$ estimates the entire value of the state using each of the basis functions. While these basis functions should work, the network routing problem has an added advantage that much of the value of the state can be attributed to certain battery-exchange stations (and thus to particular basis functions). The value of a state is the amount of expected delay that will be incurred in the future, and thus that expected delay can be split by where the vehicles will be waiting. Recall that the cost of a transition is due to the amount of time travelling and swapping batteries beyond the minimum time for any possible route in an empty system, and the amount of time waiting at a station for a battery. Define *base value* $\overline{V}_t^{q,1}(S_t)$ as the value of the state S_t at time t due to vehicle travel time and swap time. Define the *sub-values* $\overline{V}_{t,i}^{q,2}(S_t)$ for $i = 1, \ldots, n$ as the value of the state S_t at time t due to vehicles waiting at station i. Thus,

$$\overline{V}_t^q(S_t) = \overline{V}_t^{q,1}(S_t) + \sum_{i=1}^{n} \overline{V}_{t,i}^{q,2}(S_t).$$

Further, we can estimate these values by using:

$$\overline{V}_t^{q,1}(S_t) = \sum_{f\in\mathcal{F}} \theta_f^{q,1}\phi_f^q(S_t)$$

$$\overline{V}_{t,j}^{q,2}(S_t) = \sum_{z=0}^{x_i} \theta_{(i,z)}^{q,2}\phi_{(i,z)}^q(S_t) \quad \text{for all } i = 1,\ldots,n.$$

Here $\theta_f^{q,1}$ and $\theta_f^{q,2}$ are non-negative real numbers for each $f \in \mathcal{F}$ and we force $\theta_0^{q,2} = 0$ since it is not used in any of the sub-value approximations. The base value is still an approximation using all of the basis functions and coefficients $\theta_f^{q,1}$. The sub-value for each station i is approximated using only the basis functions related to that station, i.e. the functions that count the number of minutes at each battery level at that station. By using these multiple values we decrease the likelihood that a delay caused by a station would be attributed to the battery level at a station on the other side of the network that just happened to also have a high battery level at that time. Splitting the value by the different stations increases the number of coefficients that need to be stored to $1 + 2\sum_{i=1}^{n} x_i$.

Algorithm 3. A TD(λ) algorithm with linear approximations for finding a policy for the network routing problem.

Initialize $\theta_f^{0,1} \leftarrow 0$ for $f \in \mathcal{F}$

Initialize $\theta_f^{0,2} \leftarrow 0$ for $f \in \mathcal{F}$

For $q \leftarrow 1,\ldots,Q$

  For each $u \in 1,\ldots,U$

    Set $y \leftarrow \emptyset$

    Generate random sequence of car arrivals $(j_0^u,\ldots,j_{T-1}^u)$

    Set S_0 to be the state where all of the stations are empty

    For $t \leftarrow 1\ldots T$

      Set $S_t^u \leftarrow \underset{S_t'\in Y\left(S_{t-1},j_{t-1}^u\right)}{\operatorname{argmin}} \left(C\left(S_{t-1}^u,j_{t-1}^u,S_t'\right) + \sum_{f\in\mathcal{F}}\left(\theta_f^{q-1,1} + \theta_f^{q-1,2}\right)\phi_f(S_t')\right)$

    End for

    For $t \leftarrow 0,\ldots,T-1$

      Set $\delta_t^{1,u} \leftarrow C^1\left(S_t^u,j_t^u,S_{t+1}^u\right) + \sum_{f\in\mathcal{F}}\theta_f^{q-1,1}\phi_f(S_{t+1}^u) - \sum_{f\in\mathcal{F}}\theta_f^{q-1,1}\phi_f\left(S_t^u\right)$

      For $j = 1,\ldots,\beta$

        Set $\delta_{t,j}^{2,u} \leftarrow C_j^2\left(S_t^u,j_t^u,S_{t+1}^u\right) + \sum_{z=0}^{n_j}\theta_{(j,z)}^{q-1,2}\phi_{(j,z)}\left(S_{t+1}^u\right) - \sum_{z=0}^{n_j}\theta_{(j,z)}^{q-1,2}\phi_{(j,z)}\left(S_t^u\right)$

      End for

    End for

    For $t \leftarrow 0,\ldots,T-1$

      Set $y_t^{u,1} \leftarrow \sum_{f\in\mathcal{F}}\theta_f^{q-1,1}\phi_f(S_t^u) + \sum_{\tau=t}^{T}(\lambda_q)^{T-\tau}\delta_\tau^{1,u}$

      For $j \leftarrow 1,\ldots,\beta$

        Set $y_{t,j}^{u,2} \leftarrow \sum_{z=0}^{n_j}\theta_{(j,z)}^{q-1,2}\phi_{(j,z)}^q(S_t^u) + \sum_{\tau=t}^{T}(\lambda_q)^{T-\tau}\delta_{\tau,j}^{2,u}$

      End for

    End for

  End for

  Set

$$\theta^{q,1} \leftarrow \underset{\theta\geqslant 0}{\operatorname{argmin}} \sum_{u=1}^{U}\sum_{t=0}^{T-1}\left(y_t^{u,1} - \sum_{f\in\mathcal{F}}\theta_f\phi_f(S_t^u)\right)^2$$

  For $j \leftarrow 1,\ldots,\beta$

    Set

$$\theta_j^{q,2} \leftarrow \underset{\theta\geqslant 0}{\operatorname{argmin}} \sum_{u=1}^{U}\sum_{t=0}^{T-1}\left(y_t^{u,2} - \sum_{z=0}^{n_j}\theta_{(j,z)}^{q-1,2}\phi_{(j,z)}(S_t^u)\right)^2$$

  End for

  Set $\theta^{q,1} \leftarrow \alpha_q\theta^{q,1} + (1-\alpha_q)\theta^{q-1,1}$

  Set $\theta^{q,2} \leftarrow \alpha_q\left[0,\theta_1^{q,2},\theta_2^{q,2},\cdots,\theta_\beta^{q,2}\right]' + (1-\alpha_q)\theta^{q-1,2}$

End for

The algorithm to compute the values $\theta_f^{q,1}$ and $\theta_f^{q,2}$ can be seen in Algorithm 3. The algorithm runs U progressions through the horizon, and each state taking the action that provides the lowest cost using the previous approximation. The temporal differences are then calculated for the base value and sub-values separately, and a new set of coefficients are generated by finding that minimize the L_2 distance of the approximate values from the TD values of each state, while ensuring that the coefficients are non-negative. The coefficients are generated for the base values and then for each of the sub-values separately, although they are all stored in a single vector. Thus we are computing $n+1$ non-negative least squares regressions, one for the base value and then one for each sub-value associated with a station. The non-negativity ensures that highly correlated values do not cause any of the coefficients to be negative, which would suggest that having fewer batteries available *decreases* the travel time. For example, if in most of the runs a station i rarely has exactly 4 batteries taken, if negative values were allowed it would be possible for $\theta^{q,2}_{t(i,4)} \ll 0$ and $\theta^{q,2}_{t(i,5)} \gg 0$. This also intuitively makes sense, since decreasing the number of available batteries should never make a state more attractive. The non-negative least squares problem can be solved using standard algorithms (Lawson and Hanson, 1974).

The parameters α and λ may change between different iterations of the main for loop, and thus are denoted as α_q and λ_q. The algorithm requires parameters Q, U, α_q, and λ_q to be tuned manually. Without parallelization, runtime of the algorithm grows linearly with both Q and U, since $Q \times U$ is the number of simulations we need to run of the system for the approximate dynamic programming. Higher values of U increase the accuracy of the estimates of the θ parameters, since the least squares fits will have more data and there is less of a chance of being biased by outliers. The U progressions through the time horizon may all be run in parallel, which would greatly speed up the algorithm and makes the value of U less important to the runtime. Higher values of Q allow for more time for the θ coefficients to converge, which may be required if the λ values are too low. The decisions behind what values to select for parameters is discussed in more detail in Powell (2011).

With the successful implementation of the algorithm in Algorithm 3, we will be provided a set of coefficients $\theta_f^{Q,1}$ and $\theta_f^{Q,2}$ for $f \in \mathcal{F}$. These coefficients can be used by the computer routing the vehicle to determine the best route to take. When the state of the system is S_t and a vehicle turns on at time t wanting to travel OD pair j_t taking one of the possible routes in $\mathcal{R}_{j_t}$, then for each route $R \in \mathcal{R}_{j_t}$ the computer should calculate

$$v_t = \psi_t(R, S_t) + \sum_{f \in \mathcal{F}} \left(\theta_f^{Q,1} + \theta_f^{Q,2} \right) \phi_f(S_{t+1}),$$

where S_{t+1} is the state of the system after the vehicle takes route R. The route selected should be the one that minimizes v_t. Finding the minimum v_t can be done extremely quickly since it only requires one run of the algorithm in Algorithm 2 for each route and a few summations.

4. Results and discussion

We tested the algorithm in Algorithm 3 to determine the amount of savings the algorithm would provide compared to the greedy policy and the run time of the algorithm. The test data was the Arizona state highway network from Upchurch et al. (2009), shown in Fig. 5. In that paper Upchurch, Kuby, and Lim had a charging station located in each of the 25 cities in the network plus an addition 25 stations located on longer roads between cities. They also used a gravity based demand model to determine the amount of vehicles wanting to traverse each OD pair between cities. The gravity was a function of the population of the OD cities and the length of the shortest path between them. They assumed that the electric vehicles would have a battery that allows them to travel 100 miles before recharging.

4.1. Network parameters

For this analysis we assumed that there were 18 h of demand in the day (since people would not be driving late at night) and split the time span into 10 s intervals. Therefore our implementation had $T = 6480$. We assumed that each vehicle travels at 65 miles per hour and thus could travel for 554 time units before requiring a recharge. We assumed that each battery would take 120 s to swap, since the Tesla S sedan was shown to take 90 s to swap, plus we included an additional 30 s for the driver to get out of the car and pay. The battery charging time from empty was set to 4 h based on charging a Nissan Leaf to full using a 220 volt outlet (Edmunds.com, 2013). The charging time of a partially full battery is assumed to be proportional to the amount of battery consumed before being dropped off.

Each of the battery-exchange stations located in a city had 48 batteries stored in it. Each of the stations located along side of the highway had only 12 batteries. Given that a vehicle arrived in the system in need of a route, the probability that a vehicle would arrive for a particular OD pair was proportional to the gravity between the two cities used in the model from Upchurch et al. For each OD pair j the set of possible routes $\mathcal{R}_j$ was generated using the algorithm from Algorithm 1, where the shortest path between each pair of cities was found and all possible routes with at most the same number of stops were added to the appropriate $\mathcal{R}_j$ set.

The probability of a vehicle arriving in each time period was taken to be values in $\{0.05, 0.075, \ldots, 0.2\}$ and the algorithm was run on each value. The reason for testing many arrival rates was that if the arrival probability was sufficiently low then the greedy policy would be the optimal policy, since the stations would never run out of batteries (as was the case when the arrival probability was 0.05). Similarly, if the arrival probability was too high then regardless of routing policy the vehicles

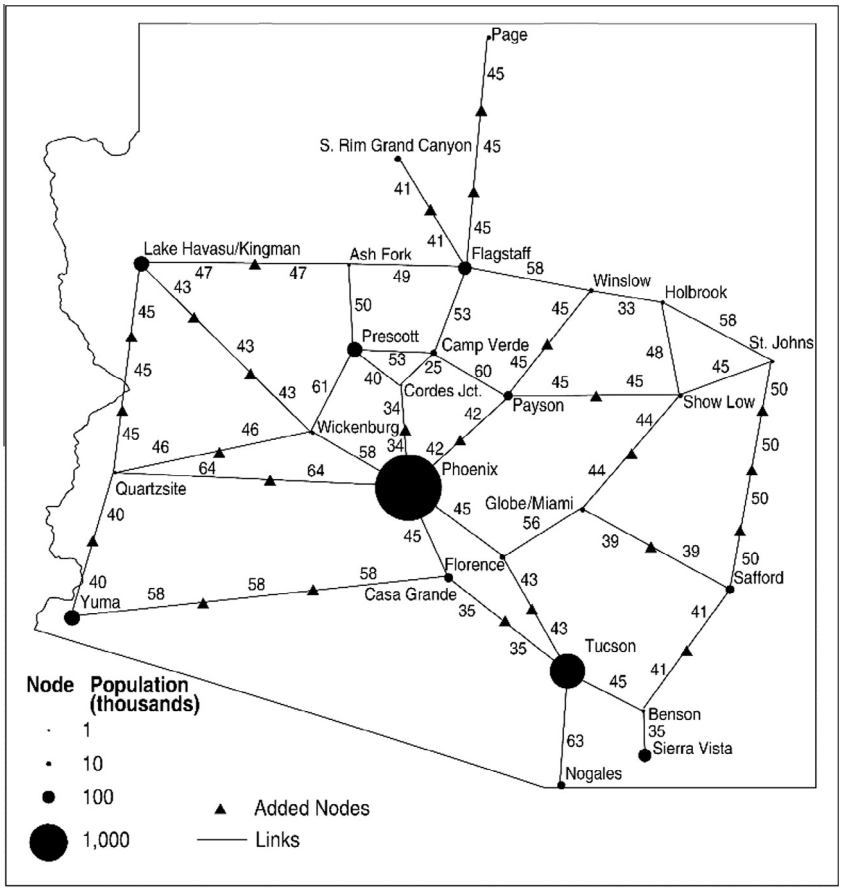

Fig. 5. The Arizona road network with battery-exchange stations from Upchurch et al. (2009).

would be waiting at the stations until the end of the day, and so the greedy policy would just as effective (which occurred when the arrival probability was 0.2). By varying the arrival probability we were able to find at what point the MCDP based policy is the most effective.

For each of the seven arrival probabilities the algorithm was run to find approximations for the value coefficients. The algorithmic policy was there compared to the greedy policy by simulating 10 days of vehicles traversing the network, and for each day using the greedy policy and the policy generated by the algorithm. The difference in the amount of delays between the two policies was used to determine which was more effective.

4.2. Parameter values

The four parameters for the algorithm were

- Q, the number of times the θ^q coefficients were tuned;
- U, the number of runs through the full time horizon for each run in Q;
- α_q the smoothing parameter for θ^q;
- and λ_q the decay parameter for the temporal differencing.

For our trials we set $Q = 24,\ U = 8$, $\alpha_q = \frac{20}{20+q}$, $\lambda_q = \alpha_{q-1}\left(\frac{1-\alpha_q}{\alpha_q}\right)$. There were many possible reasonable choices, especially for function definitions for α_q and λ_q, however in our experiments these seemed to prove sufficient while balancing the overall time it takes to compute the result.

4.3. Results and discussion

The calculations were done on a computer with an Intel i5-4670k quad core processor with 16 GB of RAM running Windows 8.1 Professional. The algorithm was coded and run in MATLAB 2013a. The non-negative least squares calculation was done using the "nnls" package by Whiten on the MATLAB Central File Exchange (Whiten, 2012).

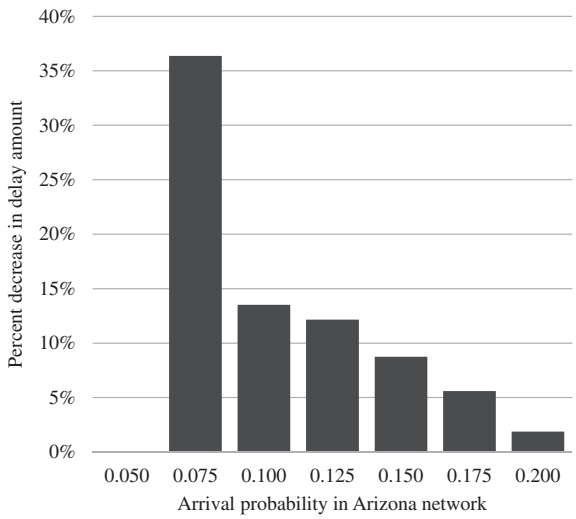

Fig. 6. A comparison of the algorithm vs. the greedy policy runs on different random networks.

Fig. 6 shows a comparison of the amount of delay incurred due to detours and waiting in the policy generated by the algorithm versus the greedy policy. For arrival probabilities between 0.075 and 0.125 the algorithm substantially lowered the amount of delays. When the arrival probability was 0.05 there were no delays at all (and thus the greedy policy was optimal). When the arrival probability was 0.2 there were too many cars so all of the vehicles had to wait until the end of the day before a battery was ready, and thus the algorithm was ineffective.

The results are further described in Table 1. In the table each column corresponds to the Arizona network with a different arrival probability. The last three rows shows the runtime in seconds of the algorithm, splitting by the amount of time required to compute the coefficients and the amount of time to route the vehicles. In all of the cases it took less than forty minutes to generate the coefficients. The θ^q coefficients would be pre-computed so that when the vehicles are routed they only need to find the value of each possible route by computing the value of the resulting state. If the system were stationary throughout the day, these coefficients would only need to be calculated once. If the system were to change unexpectedly, such as if the arrival rates diverged from what was expected due to a storm, then the coefficients may need to be recomputed during the day itself. The forty minutes required to compute the coefficients shown here is not onerous, since the code to calculate the coefficients could be sped up dramatically by better optimizing the code and utilizing the fact that U

Table 1
The algorithm and greedy policy results for each randomly generated network over ten runs.

	Arrival probability in each time period						
	0.050	0.075	0.100	0.125	0.150	0.175	0.200
Average number of vehicles in day	337.9	482.1	639.2	817.5	971.9	1134.5	1301.7
Greedy Policy							
Average total route value of all vehicles (weighted hours)	750	1063	1423	1858	2253	2932	5442
Average route value of each vehicle (weighted hours)	2.22	2.21	2.23	2.27	2.32	2.58	4.17
Average total delay for all vehicles (weighted hours)	0	0	17	51	89	403	2555
Average number of cars with any delays	0	0.6	53.7	184.8	301.9	483.4	697.2
Algorithm Policy							
Average total route value of all vehicles (weighted hours)	750	1063	1421	1852	2245	2910	5394
Average route value of each vehicle (weighted hours)	2.22	2.21	2.22	2.27	2.31	2.56	4.13
Average total delay for all vehicles (weighted hours)	0	0	15	45	81	380	2507
Average number of cars with any delays	0	0.5	56.5	182.1	301.3	472.3	682.1
Comparison							
Average number of times the policies differed	0	98.9	137.5	181.1	282	327.1	364.5
Percent decrease in delay time	–	36.4%	13.5%	12.2%	8.8%	5.6%	1.9%
Runtimes							
Runtime to generate coefficients (seconds)	1785	1845	1918	2014	2042	2147	2221
Routing runtime per vehicle using algorithm (seconds)	0.014	0.010	0.009	0.008	0.006	0.006	0.006
Routing runtime per vehicle using greedy (seconds)	0.014	0.011	0.009	0.008	0.006	0.006	0.006

simulations of the system can be run in parallel. Further, the coefficients could still be pre-computed for different scenarios of arrivals, so that the company is prepared for possible different situations.

The amount of time it took to route each individual vehicle during the 10 simulated days is shown in last two rows for both when the greedy policy and the policy from the algorithm were used. While the computation of the coefficients took a substantial amount of time, actually routing the cars would take fractions of a second, so the vehicles could easily be routed using these coefficients. Routing the vehicles when the greedy policy was used was no faster than routing with using the policy generated by the algorithm; the time added by using this model is only incurred before routing.

5. Conclusions

This research provides a novel framework for routing many electric vehicles through a network with battery-exchange technology allowing for battery reservations. We have provided an online algorithm which routes each vehicle in a manner which tries to minimize the total route value of all vehicles then makes the reservations at the stations to be visited. This is done using an approximation of the expected future costs associated with taking each route possible route using an MCDP with linear temporal differencing. The algorithm has been shown to successfully decrease the amount of delays vehicles will have which should lead to better adoption of electric vehicle technology.

There are several ways in which the model could be improved in future work. Better approximation functions for the state values could be found to improve the benefit of the algorithm. The model currently has a fixed demand rate for each OD pair throughout the day, however in real-world systems the demand is time dependent. The algorithm assumes that transit times are fixed and all vehicles take their assigned routes, both assumptions that could be relaxed. Finally, whether or not to charge a non-full battery at a station can be modeled as a decision, rather than assuming the batteries will always be charged whenever possible. This could potentially reduce costs since less electricity may be needed during peak hours.

An additional large area for future research would be in using this model as a framework for optimizing station locations and number of batteries in stock. While the research in this paper assumes that the stations have already been located and that the number of batteries at the stations is fixed, the routing policies used found by this model may not need all of the batteries at the stations. Thus, scenarios may be run with different amounts of batteries and stations in the system to find an arrangement that satisfies the customers at a lower cost to the company.

Acknowledgements

This work was supported by the National Science Foundation grant #1234584 as well as the U.S. Department of Transportation Federal Highway Administration through the Dwight David Eisenhower Transportation Fellowship Program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsor organizations. We would also like to thank Dr. Subbarao Kambhampati for this feedback on the draft of this paper.

References

Adler, J.D., Mirchandani, P.B., Xue, G., Xia, M., 2014. The electric vehicle shortest-walk problem with battery exchanges. Networks and Spatial Economics.

Bakker, J., 2011. Contesting range anxiety: The role of electric vehicle charging infrastructure in the transportation transition. alexandria.tue.nl. Eindhoven University of Technology.

Bertsekas, D.P., 2007. Dynamic Programming and Optimal Control, 4th ed. Athena Scientific, Belmont.

Better Place, 2013. Better Place: global progress [WWW Document]. URL <http://www.betterplace.com/global/progress>.

Botsford, C., Szczepanek, A., 2009. Fast charging vs. slow charging: pros and cons for the new age of electric vehicles. In: EVS24 International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium. pp. 1–9.

De Weerdt, M.M., Gerding, E.H., Stein, S., Robu, V., Jennings, N.R., 2013. Intention-aware routing to minimise delays at electric vehicle charging stations. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. AAAI Press.

Edmunds.com, 2013. 2013 Nissan Leaf Hatchback Review [WWW Document]. URL <http://www.edmunds.com/nissan/leaf/2013/?sub=hatchback>.

Hacker, F., Harthan, R., Matthes, F., Zimmer, W., 2009. Environmental impacts and impact on the electricity market of a large scale introduction of electric cars in Europe, European Topic Centre on Air and Climate Change.

He, F., Wu, D., Yin, Y., Guan, Y., 2013. Optimal deployment of public charging stations for plug-in hybrid electric vehicles. Transportation Research Part B 47, 87–101. http://dx.doi.org/10.1016/j.trb.2012.09.007.

Ichimori, T., Ishii, H., Nishida, T., 1981. Routing a vehicle with the limitation of fuel. Journal of the Operations Research Society of Japan 24, 277–281.

Kershner, I., 2013. Israel venture meant to serve electric vehicles is ending its run [WWW Document]. New York Times. URL <http://www.nytimes.com/2013/05/27/business/global/israeli-electric-car-company-files-for-liquidation.html>.

Kim, J.-G., Kuby, M., 2012. The deviation-flow refueling location model for optimizing a network of refueling stations. International Journal of Hydrogen Energy 37, 5406–5420. http://dx.doi.org/10.1016/j.ijhydene.2011.08.108.

Kim, J.-G., Kuby, M., 2013. A network transformation heuristic approach for the deviation flow refueling location model. Computers & Operations Research 40, 1122–1131. http://dx.doi.org/10.1016/j.cor.2012.10.021.

Kuby, M., 2005. The flow-refueling location problem for alternative-fuel vehicles. Socio-Economic Planning Sciences 39, 125–145. http://dx.doi.org/10.1016/j.seps.2004.03.001.

Kuby, M., Lim, S., 2007. Location of alternative-fuel stations using the flow-refueling location model and dispersion of candidate sites on arcs. Networks and Spatial Economics 7, 129–152. http://dx.doi.org/10.1007/s11067-006-9003-6.

Lam, S.-W., Lee, L.-H., Tang, L.-C., 2007. An approximate dynamic programming approach for the empty container allocation problem. Transportation Research Part C: Emerging Technologies 15, 265–277. http://dx.doi.org/10.1016/j.trc.2007.04.005.

Laporte, G., Pascoal, M.M.B., 2011. Minimum cost path problems with relays. Computers & Operations Research 38, 165–173. http://dx.doi.org/10.1016/j.cor.2010.04.010.

Lawson, C.L., Hanson, R.J., 1974. Solving Least Squares Problems. Prentice-Hall, Englewood Cliffs, NJ.

Lee, T.C., Hersh, M., 1993. A model for dynamic airline seat inventory control with multiple seat bookings. Transportation Science 27, 252–265.
Mak, H.-Y., Rong, Y., Shen, Z.-J.M., 2013. Infrastructure planning for electric vehicles with battery swapping. Management Science 59, 1557–1575. http://dx.doi.org/10.2139/ssrn.2022651.
Maxwell, M.S., Restrepo, M., Henderson, S.G., Topaloglu, H., 2010. Approximate dynamic programming for ambulance redeployment. INFORMS Journal on Computing 22, 266–281. http://dx.doi.org/10.1287/ijoc.1090.0345.
Motavalli, J., 2013. Tesla fast-tracks battery swapping while fighting a legislative attack [WWW Document]. New York Times. URL <http://wheels.blogs.nytimes.com/2013/06/21/tesla-fast-tracks-battery-swapping-while-fighting-a-legislative-attack/>.
Naveteur, J., Cœugnet, S., Charron, C., Dorn, L., Anceaux, F., 2013. Impatience and time pressure: Subjective reactions of drivers in situations forcing them to stop their car in the road. Transportation Research Part F: Traffic Psychology and Behaviour 18, 58–71. http://dx.doi.org/10.1016/j.trf.2012.12.008.
Nie, Y., Ghamami, M., 2013. A corridor-centric approach to planning electric vehicle charging infrastructure. Transportation Research Part B 57, 172–190. http://dx.doi.org/10.1016/j.trb.2013.08.010.
Powell, W.B., 2011. Approximate Dynamic Programming: Solving the Curses of Dimensionality, 2nd ed. John Wiley & Sons Inc., Hoboken, NJ.
Powell, W.B., Simao, H.P., Bouzaiene-ayari, B., 2012. Approximate dynamic programming in transportation and logistics: a unified framework. EURO Journal on Transportation and Logistics 1, 237–284. http://dx.doi.org/10.1007/s13676-012-0015-8.
Russell, S., Norvig, P., 2009. Artificial Intelligence: A Modern Approach, 3rd ed. Prentice Hall, Upper Saddle River, NJ.
Sachenbacher, M., Leucker, M., Artmeier, A., Haselmayr, J., 2011. Efficient energy-optimal routing for electric vehicles. In: AAAI, pp. 1402–1407.
Shemer, N. 2012. Better Place Unveils Battery-Swap Network. Jerusalem Post.
Simão, H.P., Day, J., George, A.P., Gifford, T., Nienow, J., Powell, W.B., 2009. An approximate dynamic programming algorithm for large-scale fleet management: a case application. Transportation Science 43, 178–197. http://dx.doi.org/10.1287/trsc.1080.0238.
Smith, O.J., Boland, N., Waterer, H., 2012. Solving shortest path problems with a weight constraint and replenishment arcs. Computers & Operations Research 39, 964–984. http://dx.doi.org/10.1016/j.cor.2011.07.017.
Topaloglu, H., Powell, W.B., 2006. Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems. INFORMS Journal on Computing 18, 31–42. http://dx.doi.org/10.1287/ijoc.1040.0079.
Upchurch, C., Kuby, M., Lim, S., 2009. A model for location of capacitated alternative-fuel stations. Geographical Analysis 41, 85–106. http://dx.doi.org/10.1111/j.1538-4632.2009.00744.x.
Van Hentenryck, P., Mercier, L., Upfal, E., 2009. Markov chance-decision processes. In: Online Stochastic Combinatorial Optimization. The MIT Press, pp. 194–218.
Whiten, B., 2012. nnls – non negative least squares [WWW Document]. MATLAB Central File Exchange. URL <http://www.mathworks.com/matlabcentral/fileexchange/38003-nnls-non-negative-least-squares>.
Worley, O., Klabjan, D., 2011. Optimization of battery charging and purchasing at electric vehicle battery swap stations. In: 2011 IEEE Vehicle Power and Propulsion Conference. IEEE, pp. 1–4. http://dx.doi.org/10.1109/VPPC.2011.6043182.
Yu, A.S.O., Silva, L.L.C., Chu, C.L., Nascimento, P.T.S., Camargo, A.S., 2011. Electric vehicles: struggles in creating a market. In: Technology Management in the Energy Smart World. Portland, OR, pp. 1–13.
Zhang, D., Adelman, D., 2009. An approximate dynamic programming approach to network revenue management with customer choice. Transportation Science 43, 381–394. http://dx.doi.org/10.1287/trsc.1090.0262.