Complexity of Scheduling Charging in the Smart Grid

Mathijs de Weerdt[1], **Michael Albert**[2], **Vincent Conitzer**[2], and **Koos van der Linden**[1]
[1] Delft University of Technology, Delft, The Netherlands
[2] Duke University, Durham, North Carolina, USA
M.M.deWeerdt@tudelft.nl, malbert@cs.duke.edu, conitzer@cs.duke.edu, J.G.M.vanderLinden@tudelft.nl

Abstract

The problem of optimally scheduling the charging demand of electric vehicles within the constraints of the electricity infrastructure is called the *charge scheduling problem*. The models of the charging speed, horizon, and charging demand determine the computational complexity of the charge scheduling problem. We show that for about 20 variants the problem is either in P or weakly NP-hard and dynamic programs exist to compute optimal solutions. About 10 other variants of the problem are strongly NP-hard, presenting a potentially significant obstacle to their use in practical situations of scale. An experimental study establishes up to what parameter values the dynamic programs can determine optimal solutions in a couple of minutes.

1 Introduction

Renewable sources for the generation of electricity are intermittent, but the amount of power generated needs to equal the amount of power consumed at all times. Because it is expensive to store electricity, there is an incentive to make part of the demand flexible and controllable. For example, electric vehicle owners can get a discount on their electricity bill if they allow their provider to charge their car flexibly [García-Villalobos *et al.*, 2014]. Specifically, car owners may have a deadline by which they would like the vehicle charged (say, by 8:00 in the morning), and they may allow the provider to charge anytime before the deadline. Meanwhile, the supply (bought by the provider) or the network capacity for providing electricity at a given time may be limited [de Nijs *et al.*, 2015; Philipsen *et al.*, 2016], requiring providers to intelligently utilize capacity over time.

The problem of deciding when to charge under a common constraint gives rise to a new class of scheduling problems. The defining difference from the traditional scheduling literature [Brucker, 2007; Pinedo, 2012] is that such charging jobs are more flexible: not only can they be shifted in time, but the charging *speed* can also vary over time. Additionally, the charging resources (“the machines” in ordinary scheduling) may vary over time. Further, providers that control flexible demand will need to solve such scheduling problems repeatedly. Therefore, it is important to understand when such problems can be solved optimally within the time limits required, and what aspects make the problem intractable. We refer to this class of problems as the *charge scheduling problem*.

In this paper, over 30 variants are identified, their computational complexity is proven, and for the easy problems a polynomial algorithm is provided.

First, we provide a general model of scheduling flexible demand in a smart grid. Then we discuss the relation to the traditional scheduling literature. Our main results for classifying all restrictions of the general model can be found in the section thereafter. Finally, we demonstrate the performance of the new algorithms on problem instances of various sizes.

2 The Charge Scheduling Problem

We consider a supply of perishable resources (e.g., network capacity or available power) which varies over time. We discretize time into intervals $T = \{1, \ldots, |T|\}$. The availability of the joint resource supply at time (period) $t \in T$ is represented by a value $m_t \in \mathbb{R}$. This resource supply is allocated to a set of n agents, and the allocation to agent i is denoted by a function $a_i : T \to \mathbb{R}$ such that $\sum_i a_i(t) \leq m_t$ for every $t \in T$. The value of an agent i for such an allocation is denoted by $v_i : [T \to \mathbb{R}] \to \mathbb{R}$. This value function for schedules can represent both (time-of-use) prices of charging in certain time slots as well as user preferences for when their vehicle is charged. In this paper, we focus on problems where the valuation function of agent i can be represented by triples of a value $v_{i,k}$, a deadline $d_{i,k}$ and a resource demand $w_{i,k}$, such that the value $v_{i,k}$ is obtained if and only if the demand $w_{i,k}$ is met by the deadline $d_{i,k}$. This allows the agent (app) to represent user preferences such as: “I value being able to go to work at \$100, I must leave for work at 8am, and it requires 25 kWh to complete the trip.” By adding a second deadline, the user could express: “I might suddenly fall ill, so I value having at least the option to take my car to urgent hospital care at 10pm at \$20, and it would require 10 kWh to complete that trip.” If so, the scheduler could make sure that the car is charged up to 10 kWh before 10pm and complete the remaining 15 kWh of charge in the rest of the night, for the full \$120 of value; alternatively, the scheduler may decide that \$20 is too low given others’ high evening demands and charge the full 25 kWh later in the night for just \$100.

We denote the total amount of resources allocated to an agent i up to an interval t by $\bar{a}_i(t) = \sum_{t'=1}^{t} a_i(t')$. Then the valuation function $v_i(a_i) = \sum_k v_i(a_i, v_{i,k}, d_{i,k}, w_{i,k})$, for value-deadline-demand triples k, where

$$v_i(a_i, v_{i,k}, d_{i,k}, w_{i,k}) = \begin{cases} v_{i,k} & \text{if } \bar{a}_i\,(d_{i,k}) \geq w_{i,k} \\ 0 & \text{otherwise} \end{cases}$$

When we say (for some variants) that the resource supply m_t, values $v_{i,k}$, or demands $w_{i,k}$ are *polynomially bounded* by the size of the input, we mean that there exists a polynomial function $p(\cdot)$ such that for all t, $m_t \leq p(n, |T|)$, or, for all i, k, $v_{i,k} \leq p(n, |T|)$, or $w_{i,k} \leq p(n, |T|)$, respectively. We aim to find an allocation that maximizes social welfare subject to the resource constraints, i.e.,

$$\begin{array}{ll} \max & \sum_i v_i(a_i) \\ \text{subject to} & \sum_i a_i(t) \leq m_t \text{ for every } t \end{array}$$

The inequality in the constraint implies free disposal, which in many situations, such as network capacity, is realistic. This problem has $n{\cdot}T$ decision variables. In this paper we consider variants of this problem along the following dimensions.

Each agent i has a maximum *charging speed* s_i and for all t and i, $a_i(t) \leq s_i$. We consider three variants of such a constraint, namely fixed / unbounded / gaps: *fixed* means that the maximum charging speed is the same at all times, *unbounded* means that there is no bound on the charging speed for each individual agent, and *gaps* means that the only bound on maximum charging speed is 0 for some time steps, for example because other use of the household connection (e.g. electric cooking, heat pump) prohibits charging for some periods.

The number of *periods* T may be constant or polynomially bounded: *constant* means that there is an a-priori known number of periods for all instances of the problem, denoted by $O(1)$, while *polynomially bounded* means that the number of periods may be large, but is bounded by a polynomial function of the input size, denoted by $O(n^c)$.

The model of the *demand* $w_{i,k}$ may be one of constant / polynomial / unbounded, where *constant* means that $w_{i,k} \leq D$ for all i, k, and that this D is an a-priori known constant, *polynomial* means that each $w_{i,k}$ is bounded by a polynomial function of the input size, and unbounded means that there is no bound on the demand size.

We can have either a *single deadline* per agent, $k = 1$, or *multiple* deadlines where there may be more than one value-demand-deadline triple which combine into a total value in the way discussed earlier. In the case of $k = 1$ we simply write v_i, d_i, w_i to denote $v_{i,1}, d_{i,1}, w_{i,1}$.

3 Relation to Known Scheduling Problems

An important generalization of scheduling problems allows for including resource constraints as well, i.e., the so-called resource-constrained project scheduling problem (RCPSP) [Hartmann and Briskorn, 2010]. The charge scheduling problem can be seen as a special case if additionally the problem is extended to deal with continuously divisible resources [Błażewicz *et al.*, 2007, Ch.12.3], a varying availability of resources with time [Klein, 2000], and the possibility to schedule subactivities of the same activity in parallel, called fast tracking [Vanhoucke and Debels, 2008].

An initial investigation shows that for one resource, without preemption, the problem of minimizing the make-span is strongly NP-hard even if there are only three processors available [Blazewicz *et al.*, 1983]. This NP-hardness result, however, does not apply to the charge scheduling problem, mainly because of the difference in the objective, but also because in charge scheduling preemption is allowed.

However, there are some directly applicable results on some variants of the single-deadline charge scheduling problem. First, we observe that with a unit charging speed and a unit supply of resources this problem is equivalent to single machine scheduling with preemption. This variant is in P when values are assumed to be polynomially bounded [Lawler, 1990].

For non-unit (but fixed, $m_t = m$) supply, the charge scheduling problem is equivalent to a multi-machine problem where all agents have an identical charging speed $s_i = s$ and the supply is a multiple of the charging speed. From this we immediately obtain a dynamic program and that this variant is weakly NP-hard [Lawler, 1983; Lawler and Martel, 1989].

When there is some control of the supply, the problem is known as speed scaling or sprinting [Barcelo *et al.*, 2015]. In this setting processors may run at higher speed until the device becomes too hot. However, there are a number of differences that make it impossible to map scheduling with speed scaling to charge scheduling, or vice versa: First, speed scaling involves deciding on when to sprint which processors, so hardness results for speed scaling do not translate to charge scheduling. Second, in charge scheduling the speed of a task is determined independently of the available resources (as long as their sum is within the limit), while in speed scaling the speed of a task depends on the speed of the processor it is assigned to. Algorithms for speed scaling therefore do not directly work for charge scheduling.

In summary, when supply (i.e., the number of machines) varies over time, or when charging speeds (i.e., the maximum number of machines allowed for a single job) differ per agent, the existing literature does not readily provide an answer to the question of the charge scheduling problem complexity.

4 Complexity of Charge Scheduling

First we analyze the complexity of the charge scheduling problem with single deadlines (Table 1). The verification of a schedule can in all cases be done in polynomial time, so:

Proposition 1. *(The decision version of) the charge scheduling problem is in NP for all variants.*

By a reduction from the knapsack problem (reducing the knapsack to the available capacity at time 1), we argue that single-deadline charge scheduling is weakly NP-hard.

Proposition 2. *The (single-deadline) charge scheduling problem with unbounded demand is (weakly) NP-hard, even when* $|T| = 1$.

We next consider single-deadline charge scheduling for multiple periods T, a supply per period t of $m_t \leq M$, n agents, and demand (and/or charging speed) per agent of at most L. The optimal solution for this problem is denoted by $OPT(m_1, m_2, \ldots, m_{|T|}, n)$, and this is defined by the following recursive function that returns the best we can do with

	gaps			*fixed charging speed*			*unbounded charging speed*		
$\lvert T\rvert$	demand constant	demand polynomial	demand unbounded	demand constant	demand polynomial	demand unbounded	demand constant	demand polynomial	demand unbounded
$O(1)$	P^{T1}	P^{T1}	weak NP-c^{T1}	P^{T1}	P^{T1}	weak NP-c^{T1}	P^{T1}	P^{T1}	weak NP-c^{T1}
$O(n^c)$	strong NP-c^{T3}	strong NP-c^{T3}	strong NP-c^{T3}	?	?	? NP-c^{P2}	P^{T2}	P^{T2}	weak NP-c^{T2}

Table 1: Problem complexity of variants with a single deadline per agent, where Tx refers to Theorem x, Px to Proposition x, and '? NP-c' means that the problem variant is NP-complete, but it is an open problem whether this is strong or weak.

the first i agents only.

$$DP1: \quad OPT(m_1, m_2, \dots, m_{|T|}, i) = \begin{cases} 0 & \text{if } i = 0 \\ \max\left\{OPT(m_1, m_2, \dots, m_{|T|}, i-1), o\right\} & \text{otherwise} \end{cases}$$

where $o =$

$$\max_{a_1,.,a_{|T|}} \left\{OPT(m_1 - a_1, \dots, m_{|T|} - a_{|T|}, i-1) + v_i\left(a\right)\right\}.$$

For ease of notation, instead of considering allocation functions $a_i(t)$, we use here a to denote the complete allocation to all agents i and over all time steps $1, \dots, |T|$, and a_t to denote the vector of allocations to all agents at time t. In this formulation, gaps and charging speed limits can be incorporated with additional constraints on the possible allocation a. A dynamic programming implementation of this recursive function gives an algorithm that solves this problem in polynomial time if both maximum supply and maximum demand are polynomially bounded and the number of periods is constant. We denote this dynamic program by DP1.[1]

Proposition 3. *The single-deadline charge scheduling problem can be solved in $O\left(M^{|T|}\right)$ space and $O\left(n \cdot M^{|T|} \cdot L^{|T|}\right)$ time using DP1 where $M \leq n \cdot L$ is the maximum supply, and L is the maximum demand (or maximum charging speed).*

In the gaps and fixed charging speed problem variants, the maximum charging speed may be very large, and therefore the reduction from knapsack (Proposition 2) applies. However, when the maximum charging speed is polynomially bounded, then also the number of alternatives a_t for a single period t is polynomially bounded, and so is the maximum supply. Moreover, if either demand or supply is polynomially bounded then effectively the other is as well. Therefore, in that case, and with a constant number of periods, the problem is in P.

Theorem 1. *With a constant number of periods, the single-deadline charge scheduling problem is weakly NP-complete. If furthermore the demand, supply, or maximum charging speed is polynomially bounded, the charge scheduling problem is in P.*

[1]An iterative implementation needs only the results for the previous agent ($i-1$) to compute the optimal values for the ith agent, so a factor of n in space can be saved by maintaining optimal values of subproblems for at most two agents.

Next, we provide an algorithm for single-deadline scheduling for a polynomially bounded number of periods, but only for the case where there is no bound on the charging speed, i.e., a charging task can complete in one period if sufficient supply is available, denoted by DP2.

1. Sort all charging task triples on deadline (increasing, with arbitrary tie-breaking).
2. Let $M_1, M_2, \dots, M_n$ be the *cumulative supply* at the deadlines of tasks $1, 2, \dots, n$—that is, $M_i = \sum_{t=1}^{d_i} m_t$—and let $M_0 = 0$.
3. Run a DP based on the following recursion (where m denotes the remaining cumulative supply available for the first i tasks):

$$OPT(m, i) = \begin{cases} 0 & \text{if } i = 0 \\ OPT\left(\min\left\{m, M_{i-1}\right\}, i-1\right) & \text{if } m < w_i \\ \max\left\{OPT\left(\min\left\{m, M_{i-1}\right\}, i-1\right), \right. & \\ \quad \left. v_i + OPT\left(\min\left\{m - w_i, M_{i-1}\right\}, i-1\right)\right\} & \text{otherwise} \end{cases}$$

   where the first call is $OPT(M_n, n)$.
4. Recover the set of tasks that get allocated and match this to resources to find a concrete possible allocation.

This DP is similar to the standard one for knapsack in the special case in which all deadlines are equal.

The runtime of DP2 for general single-deadline charging problems depends mostly on the cumulative supply. The memory bound is a factor n less, using the same optimization as for DP1.

Proposition 4. *The single-deadline charge scheduling problem with unbounded charging speed can be solved in $O\left(M_n\right)$ space and $O\left(n \cdot M_n\right)$ time using DP2.*

Combining the above results, we conclude the following.

Theorem 2. *The single-deadline charge scheduling problem with unbounded charging speed is weakly NP-complete. If furthermore the demand or supply are polynomially bounded, the charge scheduling problem is in P.*

However, gaps make the problem hard.

Theorem 3. *The single-deadline charge scheduling problem with polynomially-bounded periods and gaps is strongly NP-complete, even with constant demand and all deadlines at T.*

Proof. This proof is based on the following reduction from exact cover by 3-sets. Let a set X, with $|X| = 3q$ and $q \in \mathbb{N}$, and a collection C of 3-element subsets of X be given. Assume w.l.o.g. that the elements in X are $\{1, 2, \ldots 3q\}$. Define the following charge scheduling problem: let $|T| = 3q$ and the supply be 1 per time period. For each $c_i \in C$ with $c_i = \{x_1, x_2, x_3\}$, define an agent with a value of $v_i = 1$, a deadline $d_i = T$, and demand $w_i = 3$, who can only charge during times x_1, x_2 and x_3 (by making all other time slot gaps, i.e., setting the charging speed to 0 at those slots). It is possible to attain an objective value of q if and only if C contains q non-overlapping subsets, for the following reasons. If it contains q such subsets, we can satisfy the corresponding agents' demands for an objective value of q. Conversely, to obtain q objective value, we need to satisfy q agents, who must correspond to nonoverlapping subsets for us to be able to simultaneously satisfy them. Since exact cover by 3-sets is strongly NP-hard and the reduction is polynomial, the charge scheduling problem with unbounded periods, gaps, and constant demand is also strongly NP-hard. This trivially extends to polynomially bounded and unbounded demand. □

The above results are summarized in Table 1. Furthermore, when supply is polynomially bounded, the variants with unbounded demand that are weakly NP-complete attain membership in P. Only for the variants with non-constant numbers of periods T and fixed charging speed is the complexity still open. Dynamic program DP1 can be used for these cases, but has exponential runtime $\Omega(M^{|T|})$; DP2, however, does not apply, because it schedules taking only supply constraints into account, ignoring any charging speed constraints. Conversely, the hardness result from Theorem 3 relies on being able to set the charging speed to zero in selected periods, which is not possible with fixed charging speed.

4.1 Multiple Deadlines

In this section, we consider the variants with multiple deadlines; see Table 2. Obviously, this is a harder setting than the single-deadline case.

Proposition 5. *Any problem with multiple deadlines is as hard as the corresponding variant with a single deadline.*

In fact, any problem variant with more than two deadlines and a non-constant number of periods is strongly NP-hard, which we show by another reduction from exact cover by 3-sets. Intuitively, this hardness is due to the fact that the charges for the different deadlines are not independent: charging ahead of the first deadline (for an unexpected evening trip) also contributes to a second deadline (having the car ready in the morning). This is why one cannot for example separate the deadlines into multiple agents.

Theorem 4. *The charge scheduling problem with multiple deadlines and polynomially bounded periods is strongly NP-hard, even with just three deadlines per agent, constant demand, and no bound on charging speeds.*

Proof. Let a set X, with $|X| = 3q$, and a collection C of 3-element subsets of X be given. Assume w.l.o.g. that the elements in X are $\{1, 2, \ldots 3q\}$. Define the following charging problem: let $|T| = 3q$ and the supply be 1 per time period. For each $c_i \in C$ with $c_i = \{x_1, x_2, x_3\} \subseteq X$, define a valuation function v_i such that a value of $3q - x_1 + 1$ is obtained if a charge of 1 takes place before time x_1, an additional $3q - x_2 + 2$ if an additional charge of 1 takes place before time x_2, and an additional value of $3q - x_3 + 3$ if an additional charge of 1 takes place before time x_3. Then:

Lemma. *Any feasible schedule has a value of at most $\frac{9}{2}q^2 + \frac{9}{2}q$, and this value is attained if and only if there are q agents that have all three of their deadlines met, each just in time (with the charge arriving exactly at the deadline).*

Proof. For a slot i to contribute value, it needs to contribute to a deadline x_k with $i \leq x_k$. Letting $k \in \{1, 2, 3\}$ denote whether it is the corresponding agent's first, second, or third deadline, holding k fixed, the maximum value that i can contribute is if x_k is in $\arg\max_x \{3q - x + k \mid i \leq x\} = \{i\}$, for a value of $3q - i + k$. That is, ideally, every slot is used just in time for a deadline.

Furthermore, focusing on optimizing the k terms, there can be at most q slots that are used for a deadline with $k = 3$, because for each of these there must be one slot used for a deadline with $k = 2$ and one for a deadline with $k = 1$. Similarly, there can be at most $2q$ slots that are used for deadlines with $k = 3$ or $k = 2$. Hence, ideally, there are q agents that have all three of their deadlines met.

We thus conclude that all schedules have a value of at most $\sum_{i=1}^{3q}(3q - i) + \sum_{i=1}^{q}(1 + 2 + 3) = 9q^2 - \frac{1}{2}3q(3q + 1) + 6q = \frac{9}{2}q^2 + \frac{9}{2}q$, and this value is attained only under the conditions of the lemma. □

To continue our reduction, we show that the optimal charging schedule has value $\frac{9}{2}q^2 + \frac{9}{2}q$ if and only if C contains q non-overlapping subsets.

1. If C contains q non-overlapping subsets $\{x_1, x_2, x_3\}$, then for each of these, the respective agent's charges can be feasibly scheduled exactly in slots $\{x_1, x_2, x_3\}$, leading to a value of $(3q - x_1 + 1) + (3q - x_2 + 2) + (3q - x_3 + 3)$ and thus a total value of $\sum_{i=1}^{3q}(3q - i) + \sum_{i=1}^{q}(1 + 2 + 3) = \frac{9}{2}q^2 + \frac{9}{2}q$, which is optimal (according to the above lemma).
2. If the optimal charging schedule has value $\frac{9}{2}q^2 + \frac{9}{2}q$, then this can only be because q agents have been allocated three slots each, all exactly at their respective deadlines, according to the above lemma. Because all slots are allocated at the respective deadlines, and there is only one deadline per slot, the sets of deadlines are not overlapping, and hence the q agents whose deadlines are met correspond to an exact cover.

Since the reduction is polynomial and exact cover by 3-sets is strongly NP-hard, so is the charge scheduling problem with multiple deadlines, polynomially bounded periods, constant demand and no bound on charging speeds. This also directly

	gaps			*fixed charging speed*			*unbounded charging speed*		
$\|T\|$	demand constant	demand polynomial	demand unbounded	demand constant	demand polynomial	demand unbounded	demand constant	demand polynomial	demand unbounded
$O(1)$	P^{C1}	P^{C1}	weak NP-c^{C1}	P^{C1}	P^{C1}	weak NP-c^{C1}	P^{C1}	P^{C1}	weak NP-c^{C1}
$O\left(n^c\right)$	strong NP-c^{T4}	strong NP-c^{T4}	strong NP-c^{T4}	strong NP-c^{T4}	strong NP-c^{T4}	strong NP-c^{T4}	strong NP-c^{T4}	strong NP-c^{T4}	strong NP-c^{T4}

Table 2: Problem complexity of variants with multiple deadlines per agent, where Tx refers to Theorem x and Cx refers to Corollary x.

implies strong NP-hardness in the case where demand is polynomial or unbounded and in the case where charging speeds may have gaps or are fixed. □

While Theorem 4 indicates that (assuming P≠NP) it is impossible to solve the charge scheduling problem with multiple deadlines over an arbitrary horizon in polynomial time, in many cases, it may be sufficient to examine only a relatively short period. Specifically, a grid manager may only be able to optimize over a single upcoming day due to uncertainty in longer term power production and uncertainty in consumers' preferences over longer horizons. The following corollary indicates that in this setting, solving the charge scheduling problem may still prove feasible in practice, using DP1.

Corollary 1. *With constant number of periods, the charge scheduling problem with multiple deadlines is weakly NP-complete. If the demand, supply, or maximum charging speed is polynomially bounded, then the problem is in P.*

With this result, we have established the computational complexity of all problem variants with multiple deadlines, as can be seen in Table 2.

5 Experimental Evaluation

The theoretical complexity results do not tell us exactly what the runtimes of the dynamic programs are for concrete problem instances and how they depend on the "constants" in the analysis, such as the time horizon or maximum supply. In particular, in the following experimental analysis we aim to establish when runtimes are sufficiently large that there is significant value in researching faster (but possibly non-optimal) algorithms instead of straightforwardly using the proposed dynamic programs.

5.1 Experimental Setup

We identify the maximum size of optimally solvable problems using the proposed dynamic programs on a standard desktop computer (in our case containing an i7-6700 3.4Ghz quad core processor with 32GB of memory) within limited time (e.g., in about 300 seconds). We implemented both DP1 and DP2 in python using a single threaded tabulation approach. In the experiments, we simulate the charge scheduling problem of a number of electric vehicles behind the same substation. We assume that all other demand is inflexible and that an upper bound on the aggregated load profile over time is given, together with the constant limit on the capacity of the substation. The supply is then defined by the remaining capacity of the substation after subtracting the aggregated load.

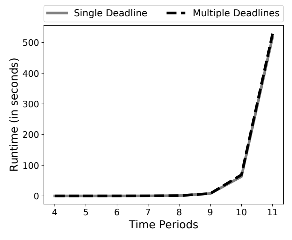
The value for charging demand is generated randomly: for each agent we generate a (number of) triple(s) of demand—uniformly random between 1 and a maximum demand; value (similarly uniform); and a random deadline within the time horizon. In the experiments the maximum charging speed of all electric vehicles is assumed to be the same (with DP1), or unbounded (with DP2).

5.2 Bounded Charging Speeds

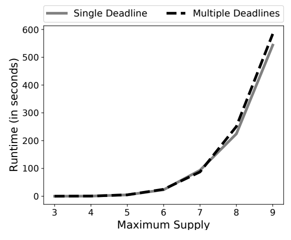
For the first set of experiments, where charging speed is bounded and where we expect DP1 to scale exponentially, we start with relatively small problem instances, using the following default settings (unless stated otherwise): there are 4 agents, the horizon is 6 time steps, supply at each time step is scaled into the range of 3 times the maximum charging speed, and the maximum demand per vehicle is equivalent to charging 3 time steps at full speed. The number of deadlines is either 1 (single) or 3 (multiple deadlines). We evaluate the effect of the time horizon, the maximum supply, the number of agents, and the number of deadlines on the runtime.

As expected, DP1 scales exponentially in both horizon (Figure 1a) and supply (Figure 1b), both for instances with 1 and with 3 deadlines. These results show that problem instances up to 11 time steps (with a maximum supply of 3) or up to a maximum supply of 9 (with 6 time steps) can be solved in about 5 minutes, but any increase beyond this approximately doubles the required computation time. Although it follows directly from the theoretical analysis of the runtime, given this exponential behavior in terms of supply and horizon it is good to be able to confirm (from Figure 1c) that the number of agents influences the runtime only linearly.

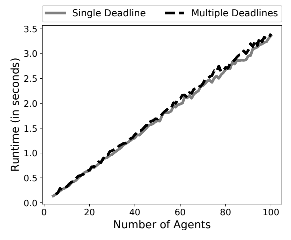
In practical settings, the maximum supply needs to be around the same order of magnitude as the number of agents; in fact, the total supply over the horizon needs to be at least the total demand, i.e., the number of agents times the average demand. A maximum supply (per time step) of 9, with 6 time steps can thus accommodate approximately 20 agents, with a runtime of approximately 1000 seconds with our implementation on our machine. We therefore conclude that this DP1 could be a good basis for implementing charge scheduling of about 20 electric vehicles (e.g., in a rolling horizon setting of 6 time steps). This implies optimal solutions can be



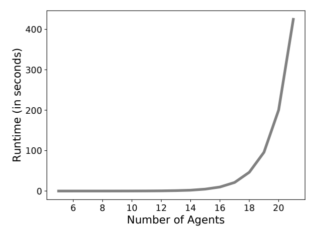
(a) The runtime for optimal solutions of variants with bounded charging speed and single or multiple deadline(s) (DP1) scales exponentially with the time horizon.



(b) The runtime for optimal solutions of variants with bounded charging speed and single or multiple deadline(s) (DP1) scales exponentially with the maximum supply.



(c) The runtime for optimal solutions of variants with bounded charging speed and single or multiple deadline(s) (DP1) scales linearly with the number of agents.



(d) When demand and supply increase exponentially (i.e. for n agents, max supply is 2^n), DP2 can solve instances up to 20 agents within 5 minutes.

Figure 1: Runtime analysis with varying parameters

found quickly only if the percentage of (participating) electric vehicles in a neighborhood is relatively small. For example, a typical scenario would be a constraint in a substation in the range of 300–1000kVA for a neighborhood with 200–500 houses and prices that differ per 15 minutes. So if more than 10% of the households owns an electric car, or if we aim to optimize with a look-ahead of more than one-and-a-half hour, this would require significant changes to the algorithm.

5.3 Results with Unbounded Charging Speeds

When the constraint on the charging speeds is ignored, with a constant horizon, the problem remains in the same complexity class. However, the runtime of DP2 depends linearly on the cumulative supply, instead of the supply to the power of the horizon in DP1. Ignoring the charging speed may obviously lead to infeasible solutions, but these may still be interesting starting points for producing solutions for instances where it would be impossible to use DP1.

For this second set of experiments we use parameter values that are more realistic: unless stated otherwise, we include 100 agents, a horizon of 64 time steps (e.g., modeling 16 hours ahead with 15 minute resolution), and a maximum demand for vehicles of 70 (kWh). The supply assumes a network capacity limit of 1000 (kVA) from which inflexible demand is subtracted: to model this, we use load measurements of a household with 15 minute resolution from the PecanStreet dataset [Pecan Street Inc., 2017]. The deadlines for the demand triples are chosen randomly using a normal distribution with standard deviation of 2 hours and mean of 7am, while the starting time of the simulation is 5pm the day before.

Again, we evaluate the effect of both the time horizon and the number of agents on the runtime. These results confirm the runtime depending linearly on these parameters, consistently with very little noise across a large set of instances. For example, problem instances with 64 time steps and 100 agents take about 1.2s and with 500 agents around 6.8s. With 100 agents and 300 time steps they takes 6.3s. In summary, problems with hundreds of agents and time steps can be solved in a few seconds.

However, Theorem 2 indicates that the problem is (weakly) NP-hard when the supply is not polynomially bounded. The aim of the final experiment is to establish this experimentally. We set the maximum supply and demand both to be 2^n. The runtime for a range of values for n can be found in Figure 1d. This indeed confirms the exponential behavior, but also shows that the runtime is reasonably within 5 minutes, up to demand and supply in the order of 2^{20}.

Because of the nature of the iterative implementation of the dynamic programs (nested for-loops) the runtime results only rely on the size of the input, not on the values themselves (and all computed outcomes are optimal). We therefore conclude that DP2 is capable of handling realistically-sized problem instances. However, as noted, charging speeds of the resulting schedules may exceed physical limits.

6 Discussion and Future Work

The detailed analysis of the complexity of charge scheduling and the dynamic programs provide an important step towards practical applicability. However, a number of questions are still open.

First, the hardness results for instances with three deadlines extend to any constant number of deadlines at least three, and we have separate results for most of the single-deadline cases. However, it is open exactly what happens with two deadlines (except that it is at least as hard as with one deadline).

Second, as we have demonstrated, if the number of time periods is constant and valuations are polynomially bounded, the charge scheduling problem is computationally tractable. However, as shown in the experiments, if these constants are too large, the dynamic program DP1 does not scale to realistically-sized problem instances, while DP2 does, but without taking charging speed limits into account. Therefore, a relevant direction for future work is to develop other approaches, for example using disjunctive MIP models, constraint programming [Ku and Beck, 2016], or fast heuristic algorithms, e.g., based on DP2.

Third, the results presented are realistic if good predictions for future supply and demand are available (such as based on weather predictions and historical charging patterns, which can be quite accurate for larger groups). If these predictions are only somewhat good, it becomes important to explicitly think about charge scheduling as an online problem [Albers, 2009], where we will want to re-solve the offline problem at each point in time. Although our hardness results still apply, this opens new questions, such as the competitive performance of on-line algorithms (compared to off-line). For deterministic algorithms we can conclude from [Bar-Noy *et al.*, 1995] that if realized charging speeds needs to be either the maximum or 0, no constant competitive algorithm exists. However, algorithms exist for other variants, e.g., without the supply constraint [Tang *et al.*, 2014] or with a weak supply constraint [Yu *et al.*, 2016], and there exist randomized algorithms for other variants of on-line scheduling [Koren and Shasha, 1995]. It remains an open question which of these can be effectively applied to an on-line variant of the charge scheduling problem and how they would perform against simpler heuristics.

Acknowledgements

This work is part of the URSES+ research programme, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO). Conitzer is thankful for support from NSF under awards IIS-1527434 and CCF-1337215.

References

[Albers, 2009] Susanne Albers. Online Scheduling. In Yves Robert and Frederic Vivien, editors, *Introduction to Scheduling*, chapter 3, pages 51–73. CRC Press, 2009.

[Bar-Noy *et al.*, 1995] Amotz Bar-Noy, Ran Canetti, Shay Kutten, Yishay Mansour, and Baruch Schieber. Bandwidth allocation with preemption. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 616–625, 1995.

[Barcelo *et al.*, 2015] Neal Barcelo, Peter Kling, Michael Nugent, Kirk Pruhs, and Michele Scquizzato. On the complexity of speed scaling. In Italiano, Pighizzini, and Sannella, editors, *Mathematical Foundations of Computer Science 2015*, pages 75–89, 2015.

[Blazewicz *et al.*, 1983] Jacek Blazewicz, Jan Karel Lenstra, and A. H. G. Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983.

[Błażewicz *et al.*, 2007] Jacek Błażewicz, Klaus H. Ecker, Erwin Pesch, Günter Schmidt, and Jan Weglarz. *Handbook on scheduling: from theory to applications*. Springer Science & Business Media, 2007.

[Brucker, 2007] Peter Brucker. *Scheduling algorithms*. Springer Verlag, 2007.

[de Nijs *et al.*, 2015] Frits de Nijs, Matthijs T. J. Spaan, and Mathijs M. de Weerdt. Best-response planning of thermostatically controlled loads under power constraints. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 615–621, 2015.

[García-Villalobos *et al.*, 2014] García-Villalobos, Zamora., San Martín., Asensio, and Aperribay. Plug-in electric vehicles in electric distribution networks: A review of smart charging approaches. *Renewable and Sustainable Energy Reviews*, 38:717–731, 2014.

[Hartmann and Briskorn, 2010] Sönke Hartmann and Dirk Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14, 2010.

[Klein, 2000] Robert Klein. Project scheduling with time-varying resource constraints. *International Journal of Production Research*, 38(16):3937–3952, 2000.

[Koren and Shasha, 1995] Gilad Koren and Dennis Shasha. D^{over}: An Optimal On-Line Scheduling Algorithm for Overloaded Uniprocessor Real-Time Systems. *SIAM Journal on Computing*, 24(2):318–339, 1995.

[Ku and Beck, 2016] Wen-Yang Ku and J. Christopher Beck. Mixed integer programming models for job shop scheduling: A computational analysis. *Computers & Operations Research*, 73:165 – 173, 2016.

[Lawler and Martel, 1989] Eugene L. Lawler and Charles U. Martel. Preemptive scheduling of two uniform machines to minimize the number of late jobs. *Operations Research*, 37(2):314–318, 1989.

[Lawler, 1983] E. L. Lawler. Recent results in the theory of machine scheduling. In *Mathematical programming the state of the art*, pages 202–234. Springer, 1983.

[Lawler, 1990] Eugene L. Lawler. A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Annals of Operations Research*, 26(1):125–133, 1990.

[Pecan Street Inc., 2017] Pecan Street Inc. Dataport, 2017.

[Philipsen *et al.*, 2016] Rens Philipsen, Germán Morales-España, Mathijs de Weerdt, and Laurens de Vries. Imperfect Unit Commitment decisions with perfect information: A real-time comparison of energy versus power. In *Power Systems Computation Conference*, pages 1–7. IEEE, 2016.

[Pinedo, 2012] Michael Pinedo. *Scheduling: theory, algorithms, and systems*. Springer Science+ Business Media, 2012.

[Tang *et al.*, 2014] Wanrong Tang, Suzhi Bi, and Ying Jun Angela Zhang. Online coordinated charging decision algorithm for electric vehicles without future information. *IEEE Transactions on Smart Grid*, 5(6):2810–2824, 2014.

[Vanhoucke and Debels, 2008] Mario Vanhoucke and Dieter Debels. The impact of various activity assumptions on the lead time and resource utilization of resource-constrained projects. *Computers & Industrial Engineering*, 54(1):140–154, 2008.

[Yu *et al.*, 2016] Zhe Yu, Shiyao Chen, and Lang Tong. An intelligent energy management system for large-scale charging of electric vehicles. *CSEE Journal of Power and Energy Systems*, 2(1):47–53, 2016.