Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Relating Electric Vehicle Charging to Speed Scaling with Job-Specific Speed Limits

Leoni Winschermann, Antonios Antoniadis, Marco E. T. Gerards, Gerwin Hoogsteen, Johann Hurink

To cite this article:
Leoni Winschermann, Antonios Antoniadis, Marco E. T. Gerards, Gerwin Hoogsteen, Johann Hurink (2026) Relating Electric Vehicle Charging to Speed Scaling with Job-Specific Speed Limits. Operations Research 74(3):1165-1186.
https://doi.org/10.1287/opre.2024.1044

https://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Vol. 74, No. 3, May–June 2026, pp. 1165–1186
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Contextual Areas

Relating Electric Vehicle Charging to Speed Scaling with Job-Specific Speed Limits

Leoni Winschermann,[a,*] Antonios Antoniadis,[a] Marco E. T. Gerards,[a] Gerwin Hoogsteen,[a] Johann Hurink[a]

[a] Department of Electrical Engineering, Mathematics and Computer Science, University of Twente, 7522 NB Enschede, Netherlands
*Corresponding author

Contact: l.winschermann@utwente.nl, https://orcid.org/0000-0003-3306-6762 (LW); a.antoniadis@utwente.nl, https://orcid.org/0000-0003-2152-7883 (AA); m.e.t.gerards@utwente.nl, https://orcid.org/0000-0002-3870-4826 (METG); g.hoogsteen@utwente.nl, https://orcid.org/0000-0002-5396-3682 (GH); j.l.hurink@utwente.nl, https://orcid.org/0000-0001-6986-5633 (JH)

Received: May 23, 2024
Revised: April 24, 2025
Accepted: August 27, 2025
Published Online in Articles in Advance: September 24, 2025

Area of Review: Energy and Environment

https://doi.org/10.1287/opre.2024.1044

Abstract. Because of the ongoing electrification of transport in combination with limited power grid capacities, efficient ways to schedule the charging of electric vehicles (EVs) are needed for the operation of, for example, large parking lots. Common approaches such as model predictive control repeatedly solve a corresponding offline problem. In this work, we first present and analyze the flow-based offline charging scheduler (FOCS), an offline algorithm to derive an optimal EV charging schedule for a fleet of EVs that minimizes an increasing and strictly convex function of the corresponding aggregated power profile. To this end, we relate EV charging to processor speed scaling models with job-specific speed limits. Experiments based on real-world EV charging data show that FOCS takes only 2.5 seconds to schedule 400 EVs in 15-minute granularity. Furthermore, we analyze the online algorithms Average Rate and Optimal Available and show that they are, respectively, $2^{\alpha-1}\alpha^{\alpha}$ and α^{α} competitive, where α is typically two in energy applications. Further numerical experiments show that, for the real-world EV charging use case, both algorithms achieve approximation ratios of less than 1.3. Furthermore, they significantly improve on the uncontrolled default often applied in practice.

Funding: This work was supported by the Ministerie van Economische Zaken en Klimaat [Grant MOOI32005] and the Ministerie van Binnenlandse Zaken en Koninkrijksrelaties [Grant MOOI32005].

Supplemental Material: All supplemental materials, including the code, data, and files required to reproduce the results, are available at https://doi.org/10.1287/opre.2024.1044.

Keywords: speed scaling • scheduling • electric vehicle

1. Introduction

Because of the ongoing electrification of transport in combination with limited power grid capacities (Eising et al. 2014) and synchronization effects (Turitsyn et al. 2010), efficient ways to schedule the charging of electric vehicles (EVs) are needed for the operation of, for example, large parking lots. In practice, however, individual vehicles come with uncertainty in their availability and energy demand (Vecchio and Tricarico 2019). To bridge this information gap, model predictive control (MPC) can be applied (van Kriekinge 2021). Such MPC frameworks introduce a (predictive) model to the scheduler that based on all information available at the current moment in time derives a control action for the next time step. Basic examples for such models are predictions based on historical data (Ireshika and Kepplinger 2024) or the introduction of deterministic charging guarantees of the form that everyone receives x units of energy within y hours (Winschermann et al. 2023a). Another possible model is centered around the prediction of fill levels that dictate the targeted aggregated (e.g., transformer level) power profile (SchootUiterkamp et al. 2018). The resulting planning may either be updated periodically, for example every 15 minutes, or rescheduling may occur based on events, for example the arrival or (early) departure of an EV. As a result, MPCs repeatedly solve an offline problem. This problem is characterized by the EVs' arrival times, departure times, energy demand, and EV-specific maximum charging

rates. EVs can charge simultaneously, and charging of a single EV may be preempted.

To account for the limited grid capacity and a quadratic relation between charging powers and energy losses, one natural objective in EV scheduling problems is to minimize the sum of squares of the aggregated power profile of, for example, a parking lot hosting multiple EVs. EV scheduling problems with this objective naturally reduce to (processor) speed scaling problems with job-specific speed limits. Hereby, as opposed to the classical model, multiple jobs may run simultaneously. In speed scaling, tasks are scheduled on a processor within their respective availability such that a (typically increasing and strictly convex) function of the processing speed is minimized. One such function may correspond to the ℓ_2-norm, a well-studied objective function in both processor scheduling and energy research. Speed scaling problems without speed limits are well studied, with the Yao-Demers-Shenker (YDS) algorithm being one of the core approaches (Yao et al. 1995). Already before YDS, Vizing et al. (1982) studied the same problem and came up with a similar solution as early as 1981. An extension of YDS considering continuous speed limits for the aggregated speed profile is given by Antoniadis et al. (2017). Another variant considers job-specific speed functions and uses a maximum flow formulation to find an optimal solution for both a single-processor and multiprocessor setup (Shioura et al. 2017). Zhang et al. (2011) investigate a model where changes in global speed are associated with additional cost. However, to the best of our knowledge, the use case with job-specific speed limits, as is relevant to EV scheduling with EV-specific maximum charging powers, has not yet been studied. Here, job-specific speed limits correspond to the maximum charging powers of the individual EVs. As summarized in recent reviews by, for example, Elghanam et al. (2024), Al-Alwash et al. (2024), and Singh et al. (2024), centralized EV scheduling problems like the one discussed here are typically solved to optimality by formulating them as mathematical programs (e.g., linear or quadratic programs) or approximated with heuristics. The overview by Elghanam et al. (2024) illustrates that the applied methods to solve mathematical programs and the used heuristics are usually generic methods that do not consider the explicit problem structure or constructively derive an optimal schedule.

In this work, we make the following contributions to the understanding of both the offline and online versions of the EV scheduling problem sketched above.

- We discuss the relation between the classical speed scaling model and the extension based on EV scheduling.
- We present and analyze a novel offline algorithm to constructively derive an optimal charging schedule for a fleet of EVs, minimizing the integral of an increasing and strictly convex function of the aggregated speed profile.
- We derive necessary and sufficient conditions for solutions to the offline problem to be optimal.
- We derive competitive ratios for two natural algorithms for the online problem known from speed scaling without job-specific speed limits. In particular, we show that scheduling according to the Average Rate (AVR) algorithm (Yao et al. 1995) is directly applicable to the extended model and has a competitive ratio of $2^{\alpha-1}\alpha^{\alpha}$ for a given α-dependent objective function. The second algorithm considered is an adapted version of Optimal Available (OA) as previously studied for speed scaling without job-specific speed limits by Yao et al. (1995). We show it to be α^{α} competitive for the same objective function. Both competitive ratios match those for the corresponding algorithms applied to speed scaling without job-specific speed limits.
- We prove that given the aggregated speed profile of a feasible solution, there exists no deterministic online scheduling rule that reliably finds a feasible solution following the given profile.
- We put the theoretical results in perspective using real-world data to empirically quantify the performance of AVR and OA for an EV scheduling case.

The remainder of the paper is organized as follows. Section 2 formally describes the extended speed scaling model. After that, in Section 3, we analyze the offline problem and present the *flow-based offline charging scheduler* (FOCS), which is an offline algorithm that uses maximum flows to compute an optimal solution for the given problem. Then, Section 4 extends the problem to online scheduling, considering the competitive ratios of two natural algorithms. Furthermore, we prove that, given the aggregated speed profile of a feasible solution, there exists no deterministic online scheduling rule that reliably finds a feasible solution that follows the given profile. Finally, we compare theoretical competitive ratios to empirical results in Section 5 using numerical experiments based on real-world EV charging data. Section 6 presents the conclusion of the paper.

2. Problem Statement

In this section, we describe the considered speed scaling models for processor scheduling with and without job-specific speed limits. Note that the used notation follows scheduling convention to emphasize the relation to classical results. In sentences that discuss energy, we caution the reader to carefully consider the context since the term refers to two related but different concepts in respectively processor scheduling and EV research.

The *deadline-based speed-scaling with speed limits* (*DSL*) problem is defined as follows. Consider a set $\mathcal{J} := \{1, \dots n\}$ of jobs that have to be scheduled on a speed-scalable processor. Each job $j \in \mathcal{J}$ is characterized by its

workload $p_j \in \mathbb{R}_{\geq 0}$, *release time* $r_j \in \mathbb{R}_{\geq 0}$, *deadline* $d_j \geq r_j$, and a *job-specific speed limit* $\ell_j \in \mathbb{R}_{\geq 0}$. A *schedule* is given by a function $s : \mathbb{R}_{\geq 0} \to (\mathbb{R}_{\geq 0})^n$, such that $s(t)$ is a vector describing at what speed each job is processed at time t. Let $s_j(t)$ be the jth entry of that vector.

Definition 1. A schedule for a given set of jobs $\mathcal{J}$ is said to be feasible for DSL if

i. Every job $j \in \mathcal{J}$ is fully processed within $[r_j, d_j)$, that is, $\int_{r_j}^{d_j} s_j(t)\,dt \geq p_j$,

ii. Speed $s_j(t) = 0$ for $t \notin [r_j, d_j)$, and

iii. Each job respects its speed limit, that is, $s_j(t) \leq \ell_j$ for all $t \in \mathbb{R}_{\geq 0}$.

Note that jobs may be preempted and (in contrast to the classical speed scaling model) run simultaneously.

Definition 2. For a schedule s, let $\mathrm{P}_{\mathrm{F}_s} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be the speed profile of s defined by $\mathrm{P}_{\mathrm{F}_s}(t) = \sum_j s_j(t)$.

For any schedule s, we consider an objective function

$$F(s) = \int_0^\infty \bar{F}(\mathrm{P}_{\mathrm{F}_s}(t))\,dt, \tag{1}$$

of the aggregated speed profile to quantify the *intensity* of schedule s. The aim of the optimization problem discussed in this paper is to minimize this function F. Here, function $\bar{F}$ as used in the definition of F in (1) is a strictly convex and increasing function. Note that, in power applications, energy losses are quadratically correlated with the speed profile, and for dynamic voltage and frequency scaling, that relation is cubic. Therefore, a natural choice for objective function F is the *energy consumption* of a schedule, given by

$$E(s) = \int_0^\infty (\mathrm{P}_{\mathrm{F}_s}(t))^\alpha\,dt, \tag{2}$$

where $\alpha > 1$ is a constant.

In the following, we assume that all considered DSL instances are *feasible*, that is, they satisfy

$$p_j \leq \ell_j (d_j - r_j) \ \forall j \in \mathcal{J}. \tag{3}$$

DSL is closely related to the *deadline-based speed-scaling* (*DS*) problem described by Yao et al. (1995). The only difference is that inputs to DS omit the speed limits, and at most, one job can run at any given time. More formally, the following conditions apply to a feasible DS schedule.

Definition 3. A schedule is said to be feasible for DS if

i. Every job $j \in \mathcal{J}$ is fully processed within $[r_j, d_j)$, that is, $\int_{r_j}^{d_j} s_j(t)\,dt \geq p_j$,

ii. Speed $s_j(t) = 0$ for $t \notin [r_j, d_j)$, and

iii. At most one job runs at any time, that is, $|\{j | s_j(t) > 0\}| \leq 1$ for all $t \in \mathbb{R}_{\geq 0}$.

The rest of the problem definition carries over from that of DSL.

3. Offline Scheduling

As mentioned above for the MPC context, in the operational reality of EV charging, DSL may be solved repeatedly. Therefore, in this section, we analyze the offline DSL problem. In particular, we analyze the relation between feasible schedules for DSL and DS (Section 3.1), derive necessary and sufficient optimality conditions for offline algorithms (Section 3.2), and introduce and analyze the FOCS, an offline algorithm that solves DSL to optimality (Sections 3.3 and 3.4).

3.1. Preliminaries Offline Algorithms

Let T be the set of all time points that are either a release time or a deadline of a given problem instance. Formally, $T := \{t | \exists j \in \mathcal{J} : t = r_j \text{ or } t = d_j\}$. We refer to the elements of T as breakpoints. Let $t_1 < t_2 < \cdots < t_{m+1}$ be the sorted elements of T, whereby $1 \leq m \leq 2n - 1$, and intervals $\{[t_i, t_{i+1}) | i = 1, \ldots m\}$ partition the time between the earliest release time and latest deadline into subintervals. We refer to those subintervals as *atomic intervals* and denote atomic interval $[t_i, t_{i+1})$ as M_i. Hereby, the length $t_{i+1} - t_i$ of interval M_i is denoted as $|M_i|$. Furthermore, we introduce an operator L for the combined length of a set of atomic intervals; that is, if $\mathcal{T}$ is a set of indices, then $L(\mathcal{T}) = \sum_{i \in \mathcal{T}} |M_i|$. We denote the set of all indices corresponding to atomic intervals by $\mathcal{M} = \{1, \ldots, m\}$.

Given that the close relationship between DSL and DS plays a central role in our results, it is useful to derive notation for the inputs, outputs, and algorithms for each problem.

Definition 4. Let $\mathcal{I}_{\mathrm{DSL}}$ (respectively, $\mathcal{I}_{\mathrm{DS}}$) be the set of all possible inputs to DSL (respectively, DS), with $I \in \mathcal{I}_{\mathrm{DSL}}$ being described as $I = \langle \vec{r}, \vec{\mathbf{d}}, \vec{\mathbf{p}}, \vec{\ell} \rangle$ (respectively, $I \in \mathcal{I}_{\mathrm{DS}}$, with $I = \langle \vec{r}, \vec{\mathbf{d}}, \vec{p} \rangle$) where $\vec{\mathbf{r}}, \vec{\mathbf{d}}, \vec{p}$, and $\vec{\ell}$ refer to a vector of the respective release times, deadlines, processing volume, and (if applicable) speed limits describing the job set $\mathcal{J}$. We say that an instance $I = \langle \vec{\mathbf{r}}, \vec{d}, \vec{\mathbf{p}}, \vec{\ell} \rangle \in \mathcal{I}_{\mathrm{DSL}}$ augments an instance $I' = \langle \vec{\mathbf{r}}', \vec{\mathbf{d}}', \vec{\mathbf{p}}' \rangle \in \mathcal{I}_{DS}$ if and only if $\vec{\mathbf{r}} = \vec{\mathbf{r}}'$, $\vec{d} = \vec{\mathbf{d}}'$ and $\vec{\mathbf{p}} = \vec{\mathbf{p}}'$. We express this as $I \in a(I')$ and call I and I' corresponding. Here, $a(I')$ is the set of all DSL instances that augment I'.

We note that a feasible schedule for an input $I' \in \mathcal{I}_{\mathrm{DS}}$ is also feasible for input $I \in a(I') \subset \mathcal{I}_{\mathrm{DSL}}$ if and only if it satisfies all job-specific speed limits.

Lemma 1. *Any feasible schedule s for an instance $I = \langle \vec{\mathbf{r}}, \vec{\mathbf{d}}, \vec{\mathbf{p}}, \vec{\ell} \rangle \in \mathcal{I}_{\mathrm{DSL}}$ of the DSL problem, can be transformed into a feasible schedule s' for the corresponding augmented instance $I' \in \mathcal{I}_{\mathrm{DS}}$ with $I \in a(I')$, such that both schedules have the same speed profile (i.e., $\mathrm{P}_{\mathrm{F}_s} = \mathrm{P}_{\mathrm{F}_{s'}}$ and therefore also $F(s) = F(s')$).*

Proof of Lemma 1. Consider a schedule s that is feasible for instance $I = \langle \vec{\mathbf{r}}, \vec{\mathbf{d}}, \vec{\mathbf{p}}, \vec{\ell} \rangle \in \mathcal{I}_{\mathrm{DSL}}$, with associated

job set $\mathcal{J}$. We show how to transform s into a schedule s' such that $P_{F_s} = P_{F_{s'}}$ and s' is feasible for the corresponding augmented instance $I' \in \mathcal{I}_{DS}$. To this end, we consider atomic intervals $M_i = [t_i, t_{i+1}) \in \mathcal{M}$ separately.

Schedule s' is obtained by simply scheduling within each interval M_i and for each job $j \in \mathcal{J}$ an amount of processing volume equal to $\int_{t_i}^{t_{i+1}} s_j(t)dt$. All these volumes are scheduled within M_i according to earliest deadline first (EDF) and with the same speed profile P_{F_s}.

By the definition of s', it is straightforward that $P_{F_s} = P_{F_{s'}}$, and therefore it remains to argue that s' is a feasible schedule for I'. Indeed, properties (i) and (ii) as defined in Definition 3 hold because, for any atomic interval M_i, it is the case that $\int_{t_i}^{t_{i+1}} s_j(t)\,dt = \int_{t_i}^{t_{i+1}} s'_j(t)\,dt$, and furthermore by definition, the interior of M_i contains no release time or deadline. Property (iii) follows directly by the definition of EDF. □

Figure 1 illustrates the method applied in the proof for instances $I' = \langle (0,0),(1,2),(1,2)\rangle$ and $I = \langle (0,0),(1,2),(1,2),(2,2)\rangle \in a(I')$. Here, breakpoints t_0, t_1, and t_2 are indicated with dashed lines.

Figure 2 gives an example showing that the converse statement to that of Lemma 1 is not true. It shows an instance $I = \langle \vec{\mathbf{r}}, \vec{\mathbf{d}}, \vec{\mathbf{p}}, \vec{\boldsymbol{\ell}}\rangle$, the corresponding instance I', and a feasible schedule s' for DS instance I', for which no feasible schedule for I with the exact same speed profile $P_{F_{s'}}$ exists. Moreover, even if given a speed profile P_{F_s} corresponding to a feasible schedule s, EDF does not necessarily result in a feasible schedule, even if it respects job-specific maximum speeds. See Figure 3 for an example of this phenomenon. The above lemma implies the following corollary.

Corollary 1. *The intensity of an optimal schedule s for DSL instance $I = \langle \vec{\mathbf{r}}, \vec{\mathbf{d}}, \vec{\mathbf{p}}, \vec{\boldsymbol{\ell}}\rangle$ is at least as high as the intensity of an optimal schedule s' for the corresponding DS instance $I' = \langle \vec{\mathbf{r}}, \vec{\mathbf{d}}, \vec{\mathbf{p}}\rangle$, that is, $F(s) \geq F(s')$.*

3.2. Optimality Conditions

In this section, we provide a convex programming formulation for the considered offline EV scheduling problem to derive necessary and sufficient optimality conditions. To this end, we extend the mathematical program given in Bansal et al. (2007).

First, we introduce some additional notation for the offline model. For a given atomic interval M_i, we denote by $J(i)$ the jobs that are available in interval M_i, that is,

$$J(i) = \{j \in \mathcal{J} \mid (r_j \leq t_i) \wedge (t_{i+1} \leq d_j)\}.$$

Similarly, for a job $j \in \mathcal{J}$, we define by $J^{-1}(j)$ the set of indices i for which job j is available in interval M_i. Finally, as atomic intervals are in general not unit sized, we introduce maximum work limits $p_{i,j}^{\max} = \ell_j |M_i|$ per job j and interval M_i.

As decision variables, let $p_{i,j}$ be the work scheduled for job j during atomic interval M_i, that is, $p_{i,j} = \int_{t_i}^{t_{i+1}} s_j(t)\,dt$, where schedule s is yet to be determined. Given those $p_{i,j}$ values, a schedule s follows naturally by scheduling job j at speed $p_{i,j}/|M_i|$ throughout M_i. Therefore, decision variables $p_{i,j}$ naturally correspond to a discretized EV charging schedule where all $s_j(t)$ are *step functions*, that is, the speed between two breakpoints is constant. Moreover, note that by Jensen's inequality (Jensen 1906)

$$\bar{F}\left(\frac{\int_{t_i}^{t_{i+1}} P_{F_s}(t)\,dt}{|M_i|}\right)|M_i| \leq \int_{t_i}^{t_{i+1}} \bar{F}(P_{F_s}(t))\,dt, \qquad (4)$$

Figure 1. (Color online) Transformation Applied in Proof of Lemma 1

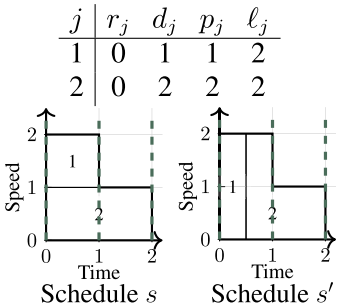


Figure 2. Example of a DSL Instance for Which the Optimal Speed Profile Differs from That of the Corresponding DS Instance Under Objective Function (2) with $\alpha = 2$

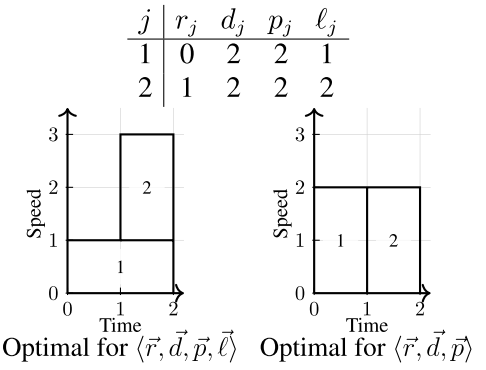
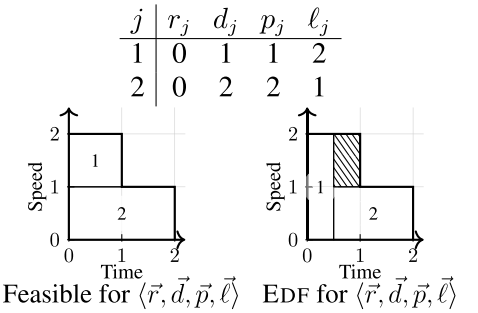


Figure 3. Example of an Instance Where EDF Results in an Infeasible Schedule for a DSL Instance Even When Respecting Speed Limits



for any schedule s and atomic interval M_i. Furthermore, by additivity of the integral, we may consider the intensity function F per atomic interval. Therefore, the right-hand side of (4) is M_i's contribution to the intensity of the schedule according to schedule s. Furthermore, the left-hand side of (4) gives the intensity of a schedule that has constant speed $\int_{t_i}^{t_{i+1}} \mathrm{Pf}_s(t)\,dt/|M_i|$ throughout M_i. Because by strict convexity of $\bar{F}$, the optimal speed profile of an instance is unique, this implies that the optimal speed profile is constant within atomic intervals. Because we do not consider vehicle-to-grid applications in this work, we require that $p_{i,j} \geq 0$. Together with the feasibility conditions for DSL introduced in Section 2, we summarize the mathematical model for DSL as follows:

$$\sum_{i \in J^{-1}(j)} p_{i,j} \geq p_j \qquad \forall j \in \mathcal{J}, \quad (5a)$$

$$p_{i,j} \geq 0 \qquad \forall j \in \mathcal{J}, i \in J^{-1}(j), \quad (5b)$$

$$p_{i,j} \leq p_{i,j}^{\max} \qquad \forall j \in \mathcal{J}, i \in J^{-1}(j). \quad (5c)$$

Note that these constraints are the same as those used by Bansal et al. (2007) (up to notation), extended by Inequality (5c), which models the job-specific speed limits.

From a grid perspective, the aggregated power level resulting from an EV schedule is of interest. For a given schedule, the average aggregated speed in atomic interval M_i is given by $\sum_{j \in J(i)} p_{i,j}/|M_i|$.

Next, we consider the Karush-Kuhn-Tucker (KKT) conditions corresponding to the problem. Generally, for a convex program

$$\begin{aligned} \min \quad & \phi(x) \\ \text{s.t.} \quad & \psi_k(x) \leq 0 \qquad k = 1, \ldots, N, \end{aligned}$$

with differentiable functions ψ_k, they are expressed using the KKT multipliers λ_k associated with ψ_k. These necessary and sufficient conditions for optimality of solutions x and λ (Boyd and Vandenberghe 2004) are

$$\psi_k(x) \leq 0 \qquad k = 1, \ldots, N, \quad (6a)$$

$$\lambda_k \geq 0 \qquad k = 1, \ldots, N, \quad (6b)$$

$$\lambda_k \psi_k(x) = 0 \qquad k = 1, \ldots, N, \quad (6c)$$

$$\nabla \phi(x) + \sum_{k=1}^{N} \lambda_k \nabla \psi(x) = 0. \quad (6d)$$

In this section, we consider the general form F of the objective function as defined in (1). Applying (6) to the discretized formulation of DSL (5), and introducing dual variables denoted by δ_j for (5a), $\gamma_{i,j}$ for (5b), and $\zeta_{i,j}$ for (5c), KKT Condition (6d) leads to

$$\begin{aligned} 0 = {} & \nabla F(s) \\ & + \sum_{j=1}^{n} \delta_j \nabla \left(p_j - \sum_{i \in J^{-1}(j)} p_{i,j} \right) \\ & - \sum_{i \in \mathcal{M}} \sum_{j \in J(i)} \gamma_{i,j} \nabla p_{i,j} \\ & - \sum_{i \in \mathcal{M}} \sum_{j \in J(i)} \zeta_{i,j} \nabla (p_{i,j}^{\max} - p_{i,j}). \end{aligned} \quad (7)$$

First, note that $F(s)$ is the integral of a strictly convex and increasing function $\bar{F}$ of the aggregated speed profile and that the optimal speed profile is constant within atomic intervals and assumes value $\sum_{j \in \mathcal{J}} p_{i,j}/|M_i|$. On top of that, by additivity of the integral in F, decision variable $p_{i,j}$ only effects the part of the schedule's intensity associated with atomic interval M_i. In the following, we use this fact to gain insights on speed levels of atomic intervals based on an analysis of the components of the gradient in (7) with respect to $p_{i,j}$. Note that the component of this gradient that corresponds to the partial derivative with respect to $p_{i,j}$ is

$$0 = \frac{\partial F}{\partial p_{i,j}} - \delta_j - \gamma_{i,j} + \zeta_{i,j}. \quad (8)$$

We analyze Condition (8) for components corresponding to partial derivatives with respect to $p_{i,j}$, where job $j \in J(i)$. We consider three cases in our analysis.

First, consider $0 < p_{i,j} < p_{i,j}^{max}$. In this case job j charges in interval i, but not at full power. Complementary slackness (see (6c)), now implies that $p_{i,j}\gamma_{i,j} = 0$ and $(p_{i,j} - p_{i,j}^{\max})\zeta_{i,j} = 0$. In the considered case, this implies that $\gamma_{i,j} = \zeta_{i,j} = 0$. Therefore, (8) simplifies to

$$\begin{aligned} 0 &= -\delta_j + \frac{\partial F}{\partial p_{i,j}} \\ \iff \delta_j &= \frac{\partial F}{\partial p_{i,j}}. \end{aligned} \quad (9)$$

This shows that the dual variable δ_j is the derivative of the intensity function F with respect to $p_{i,j}$. Because δ_j does not depend on i, if there is another atomic interval $i' \in J^{-1}(j)$ where $0 < p_{i',j} < p_{i',j}^{\max}$, we have $\partial F/\partial p_{i,j} = \partial F/\partial p_{i',j}$. Substituting with the definition of F (cf. (1)) and using additivity of the integral and the fact that the aggregated speed outside of M_i is independent of $p_{i,j}$, this yields

$$\frac{\partial \left(\int_{t_i}^{t_{i+1}} \bar{F}(\mathrm{P}_{\mathrm{F}_s}(t))\,dt \right)}{\partial p_{i,j}} = \frac{\partial \left(\int_{t_{i'}}^{t_{i'+1}} \bar{F}(\mathrm{P}_{\mathrm{F}_s}(t))\,dt \right)}{\partial p_{i',j}}.$$

Note that previously we derived that the aggregated speed within atomic intervals is constant. Therefore, we may substitute $\mathrm{P}_{\mathrm{F}_s}(t)$ by $\sum_k pi,k/|M_i|$ on domain M_i of

the integral and find that

$$|M_i| \frac{\partial\left(\bar{F}\left(\frac{\sum_k p_{i,k}}{|M_i|}\right)\right)}{\partial p_{i,j}} = |M_{i'}| \frac{\partial\left(\bar{F}\left(\frac{\sum_k p_{i',k}}{|M_{i'}|}\right)\right)}{\partial p_{i,j}}.$$

Here, we consider the numerator to be a composition of functions. By applying the chain rule, we find that

$$\frac{\partial(\bar{F})}{\partial p_{i,j}}\left(\frac{\sum_k p_{i,k}}{|M_i|}\right) = \frac{\partial(\bar{F})}{\partial p_{i,j}}\left(\frac{\sum_k p_{i',k}}{|M_{i'}|}\right),$$

where we now evaluate the partial derivatives of $\bar{F}$ in $\sum_k p_{i,k}/|M_i|$ and $\sum_k p_{i',k}/|M_{i'}|$, respectively. By strict convexity of $\bar{F}$, we conclude that $\sum_k p_{i,k}/|M_i| = \sum_k p_{i',k}/|M_{i'}|$; that is, the aggregated speeds of any such two intervals M_i and $M_{i'}$ where job j charges at a rate strictly between zero and its power limit are the same.

Next, consider the case where $0 = p_{i,j} < p_{i,j}^{\max}$. Complementary slackness gives $\zeta_{i,j} = 0$, leaving us with

$$\begin{aligned} 0 &= -\delta_j + \frac{\partial F}{\partial p_{i,j}} - \gamma_{i,j} \\ \Leftrightarrow \gamma_{i,j} &= -\delta_j + \frac{\partial F}{\partial p_{i,j}}. \end{aligned} \tag{10}$$

Using nonnegativity of $\gamma_{i,j}$ (see (6b)), it follows that $\partial F/\partial p_{i,j} \geq \delta_j$. As above, δ_j is independent of i and characterizes $\partial F/\partial p_{i',j}$ for intervals with index i' where $0 < p_{i',j} < p_{i,j}^{\max}$. Using the same arguments as above and the fact that the derivative of a strictly convex and increasing function is increasing, we conclude that the aggregated speed during interval M_i where by assumption job j does not process at positive speed is at least as high as during intervals where job j does process at a (positive) speed below its maximum.

Lastly, consider the case where $0 < p_{i,j} = p_{i,j}^{\max}$. Complementary slackness gives us $\gamma_{i,j} = 0$, leaving us with

$$\begin{aligned} 0 &= -\delta_j + \frac{\partial F}{\partial p_{i,j}} + \zeta_{i,j} \\ \Leftrightarrow \zeta_{i,j} &= \delta_j - \frac{\partial F}{\partial p_{i,j}}. \end{aligned} \tag{11}$$

Applying similar reasoning as in the previous cases and considering that the signs in the right-hand sides of (10) and (11) are reversed, we conclude that the speed in any interval M_i where job j is executed at maximum speed is at most as high as during intervals where j is available and is either processed at a (positive) power below its maximum or is available and not processed at all.

From the above analysis, the following necessary and sufficient conditions for a schedule to be optimal follow.

Condition KKT1. The aggregated speed in all intervals where j is scheduled but does not reach its speed limit is the same.

Condition KKT2. The aggregated speed in intervals where j could but does not run is at least as high as in intervals where j actually runs.

Condition KKT3. The aggregated speed in intervals where j runs at maximum speed is smaller or equal than in intervals where j runs below its speed limit.

The first two conditions are similar to those derived by Bansal et al. (2007), whereas the last results from the addition of job-specific speed limits.

In Section 3.4.2, we show that the output of the FOCS algorithm introduced in Section 3.3.2 is a feasible schedule that satisfies said conditions. For such a schedule, we can solve System (9), (10), and (11), proving optimality of the derived primal solution.

3.3. Offline Algorithm Using Flows

In this section, we present an iterative offline algorithm to determine an optimal schedule for DSL instances, minimizing the integral of an increasing and strictly convex function of the aggregated speed profile. First, note that, because of the convexity of the objective function and the finite number of release times and deadlines, the aggregated speed profile of any optimal solution is a step function. Moreover, the aggregated speed within any atomic interval M_i is constant for such a schedule. Similarly to YDS, the algorithm presented here uses the notion of critical intervals. These intervals are exactly those intervals that in an optimal solution require the highest aggregated power. Formally, these intervals are defined as follows.

Definition 5 (Critical Intervals). An atomic interval M_i is critical if for any optimal schedule s its average aggregated speed $(1/|M_i|)\int_{t_i}^{t_{i+1}} \mathrm{PF}_s(t)\,dt$ is larger or equal to the average aggregated speed $(1/|M_{i'}|)\int_{t_{i'}}^{t_{i'+1}} \mathrm{PF}_s(t)\,dt$ for any $i' \in \mathcal{M}$.

Note that there may be multiple critical (atomic) intervals. Furthermore, one major difference with critical intervals as defined for YDS is that jobs do not have to be fully contained within a (set of) critical interval(s) in order to be scheduled there. This difference with YDS follows from the job-specific speed limits. The speed profile that YDS assigns to what they call a critical interval when solving DS instances is not necessarily feasible in DSL, the setting with speed limits (Figure 2). Compared with YDS, determining critical intervals and their power level is more involved. In the algorithm presented in Section 3.3.2, determining critical intervals is based on the computation of multiple maximum flows. To be able to compute the flows and to keep track of the developments over the iterations of the proposed algorithm, we follow the DSL notation introduced thus far and introduce some additional notation.

3.3.1. Flow Formulation. For the proposed algorithm, we use a network $G = (V, D)$. The network is initialized as follows (Figure 4). The vertex set V consists of source and sink vertices v_0 and v_t, as well as two sets of vertices representing job vertices and atomic interval vertices, respectively, that is, $V = \{v_0, v_t\} \cup \mathcal{J} \cup \{M_i | i \in \mathcal{M}\}$. Furthermore, the edge set D consists of the union of the following three sets:

$$D_0 = \{(v_0, j) | j \in \mathcal{J}\}$$
$$D_1 = \{(j, M_i) | j \in \mathcal{J}, i \in J^{-1}(j)\}$$
$$D_t = \{(M_i, v_t) | i \in \mathcal{M}\},$$

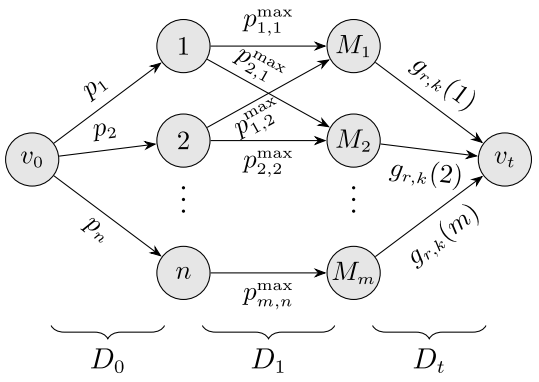
with respective edge capacities

$$c_{u,v} = \begin{cases} p_v & \text{if } u = v_0,\ v \in \mathcal{J} \\ p^{max}_{i,u} & \text{if } u \in \mathcal{J},\ v = M_i,\ i \in J^{-1}(u) \\ g_{r,k}(u) & \text{if } u \in \mathcal{M},\ v = v_t \end{cases}.$$

Note that the function $g_{r,k}$ is not defined yet. The algorithm works with rounds (indexed by r), each of which executes iterations (indexed by k). Intuitively, $g_{r,k}$ is a lower bound on the flatness of the aggregated speed profile. It varies over the execution of the algorithm and is discussed in more detail in Section 3.3.2.

Given a flow f in network G, we denote the flow value as $|f|$ and call an edge (u, v) *saturated* if $f(u, v) = c_{u,v}$. Note that a flow in G corresponds to an EV schedule. Here, a job j is scheduled to process $f(j, M_i)$ units of work in interval M_i or equivalently an EV j charges $f(j, M_i)$ in interval M_i and the capacities on edges in D_1 model the job-specific speed limits. Furthermore, any flow for which the edges in D_0 are saturated corresponds to a feasible EV charging schedule and the flow through D_t models the aggregated speed in the atomic intervals of the charging schedule corresponding to f. Note that the capacities and flow through D_t are expressed in terms of the aggregated work processed. By normalizing for the length of each atomic interval, we can deduce the aggregated speed profile. Based on this correspondence, we may use the network structure to not only derive a feasible, but an optimal schedule for objective function $F(s)$.

Figure 4. Schematic of Flow Network Structure of DSL



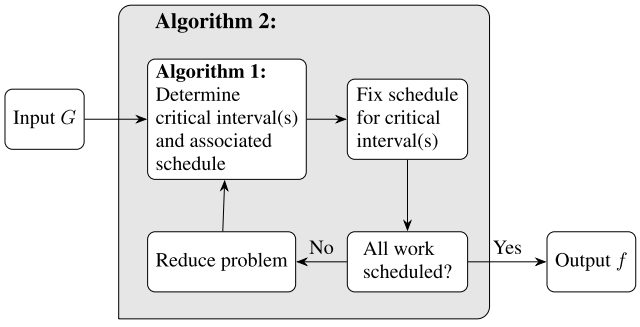
Note. Edge annotations denote edge capacities.

3.3.2. Algorithm Formulation. In the following, we use network G defined in Section 3.3.1 to derive an iterative algorithm that gives an optimal schedule and power profile for (aggregated) EV charging with objective function $F(s) = \int_0^\infty \bar{F}(\mathrm{P}_{\mathrm{F}_s}(t))\,dt$ where $\bar{F}$ is strictly convex and increasing (cf. (1)).

Before going into detail, we provide some intuition and a rough overview of the workings of the algorithm. Intuitively, the edge set D_0 can be interpreted as the processing work of the jobs. For any feasible DSL schedule, those demands have to be met. The flow through edge set D_1, on the other hand, is what we are trying to determine: the schedule itself. For any interval node M_i, the incoming flow corresponds to the load scheduled in that interval. In particular, $f(j, M_i)$ is the work processed for job j in interval M_i. Whereas the capacities of edges in D_0 and D_1 are determined by the instance, edge capacities of edges in D_t are not. However, the flow through D_t directly corresponds to the value of the objective function. Therefore, the algorithm presented in this section defines edge capacities for D_t such that they are a lower bound on the highest aggregated speed contributing to the objective function, that is, a lower bound on the outgoing flow of nodes M_i, where M_i is a critical interval. If given those capacities, we find a maximum flow that saturates all edges in D_0, we have found a feasible solution with this maximum speed and use this to determine the partial schedule for any critical interval M_i. This partial schedule corresponds to the incoming flow at each such node M_i. Conversely, we adapt the lower bound and repeat the process until we do find such a maximum flow and partial schedule.

In Figure 5, we provide a rough outline of an algorithm that exploits the bottleneck function of the critical intervals. In both the algorithm formulation and analysis, we distinguish between iterations and rounds of the algorithm. In Figure 5, a new round starts every time that Algorithm 1 is called. To determine a (set of) critical

Figure 5. Schematic Overview of Focs



interval(s) (see Definition 5), we may require multiple iterations in which we adapt the lower limit. Given the dynamic nature of this lower limit, we denote it as $g_{r,k}$, where r and k denote the current round and iteration, respectively. At the end of a round, we have determined a (set of) critical interval(s). We determine the schedule for those intervals to be the incoming flow at the corresponding interval nodes. For noncritical intervals, there is no schedule yet. Their schedules will be determined in the next rounds. In that fashion, we will construct a schedule for the entire instance. To keep track of what has yet to be scheduled, we introduce the notion of *active intervals*. At the beginning of a round, interval M_i is active if it has not yet been scheduled (i.e., has not yet been critical) in previous rounds. Let $\mathcal{M}_a$ be the set of indices of active intervals, which we initialize to be all atomic intervals, that is, $\mathcal{M}_a = \mathcal{M}$.

The first iteration of the first round goes as follows. Given that we have to schedule a certain amount of energy and that the objective function is increasing, the most optimistic lower bound on the aggregated power is a constant profile over all intervals. Therefore, we initialize the capacities of the edges in D_t by

$$g_{1,1}(i) = \frac{\sum_{j=1}^{n} p_j}{L(\mathcal{M}_a)} |M_i| \ \forall i \in \mathcal{M}_a,$$

which is the aggregated energy charged in atomic interval M_i given that in all intervals the same aggregated charging power is used, and all energy requirements are met. In that way, the edge capacities of D_t act as lower bounds to the highest aggregated power level. They are dynamic and will be increased over iterations. Given the capacities, we determine a maximum flow $f_{1,1}$ for this instance. If the flow value $|f_{1,1}|$ of $f_{1,1}$ is $\sum_{j=1}^{n} p_j$, we have found a feasible schedule, and all active intervals are critical. If not, then there is at least one nonsaturated edge (M_i, v_t) with M_i an active interval. We call the intervals corresponding to such edges *subcritical*. Note that those intervals will not be critical in this round. We therefore temporarily remove them from the set of active intervals and add them to what we call the collection of *parked intervals* $\mathcal{M}_p$. At the beginning of each round, this collection is initialized to be empty. This is the end of the first iteration.

From here, we structurally increase the edge capacities of edges in D_t and again compute a maximum flow until all edges in D_0 are saturated, and we find a feasible EV schedule. To this end, first note that after the first iteration,

$$\sum_{j=1}^{n} c_{v_0,j} - |f_{1,1}| = \sum_{j=1}^{n} p_j - |f_{1,1}| > 0,$$

if there were subcritical intervals. In particular, this means that there are jobs j for which additional work still needs to be scheduled. Among the interval vertices, the only candidates for additional flow are those vertices M_i for which edge (M_i, v_t) was saturated in $f_{1,1}$, that is, the remaining active intervals. Keeping the objective in mind, we therefore proportionally increase the capacities at the remaining active intervals to

$$g_{1,2}(i) = g_{1,1}(i) + \frac{\sum_{j=1}^{n} p_j - |f_{1,1}|}{L(\mathcal{M}_a)} |M_i| \ \forall i \in \mathcal{M}_a.$$

We repeat this process until we find a flow with flow value $\sum_{j=1}^{n} p_j$. Such a flow leads to a feasible EV schedule for which the maximum aggregated power is minimal. Say this happens after K_1 iterations. The remaining active intervals in that iteration make up the set of critical intervals in the corresponding round. In Figure 5, this case corresponds to the first time we leave the box of Algorithm 1 and move on to fix parts of the schedule we aim to compute.

We generalize the steps discussed thus far to an arbitrary round r and iteration k with $1 \le k < K_r - 1$, where K_r is the number of iterations in round r. This yields

$$g_{r,1}(i) = \frac{\sum_{j=1}^{n} c_{v_0,j}}{L(\mathcal{M}_a)} |M_i| \qquad \forall i \in \mathcal{M}_a$$

$$g_{r,k+1}(i) = g_{r,k}(i) + \frac{\sum_{j=1}^{n} c_{v_0,j} - |f_{r,k}|}{L(\mathcal{M}_a)} |M_i| \quad \forall i \in \mathcal{M}_a,$$

given that flow $f_{r,k}$ is the maximum flow in round r and iteration k and that between iterations active intervals and flow networks are updated. We end the round when we find a maximum flow with flow value $\sum_{j=1}^{n} c_{v_0,j}$.

After each round r, we fix the part of the schedule associated with the critical interval(s) (top right box in Figure 5) to correspond to the flow incoming at the respective (critical) interval nodes and reduce the remainder of the problem by constructing a new network G_{r+1} (bottom left box in Figure 5) as follows. First, we exploit the acyclic topology of the network to define a flow $f_r|_{M_r^*}$ of the determined maximum flow f_r, where $M_r^* = \{i \in \mathcal{M} | M_i \text{ iscriticalinround r}\}$ is the set of indices of critical intervals and

$$f_r|_{M_r^*}(M_i, v_t) = \begin{cases} f(M_i, v_t) & \text{if } i \in M_r^* \\ 0 & \text{otherwise} \end{cases}$$

$$f_r|_{M_r^*}(j, M_i) = \begin{cases} f(j, M_i) & \text{if } i \in M_r^* \\ 0 & \text{otherwise} \end{cases}$$

$$f_r|_{M_r^*}(v_0, j) = \sum_{i \in J^{-1}(j)} f_r|_{M_r^*}(j, M_i).$$

Note that this definition backpropagates flow from the sink to the source. Intuitively, $f_r|_{M_r^*}$ denotes the flow that goes through critical intervals. In the YDS sense,

$f_r|_{M_r^*}(v_0, j)$ is the critical load of job j in round r. Now, G_{r+1} is the network obtained by removing edges (M_i, v_t) with $i \in M_r^*$ from G_r and updating edge capacities to be $c_{u,v} - f_r|_{M_r^*}(u,v)$. From here, we start the next round of the algorithm and initialize a new flow $f_{r+1,1}$. For convergence, between iterations within a round, we similarly construct $G_{r,k+1}$ based on the subcritical flow $f_{r,k|\mathcal{M}_p}$. Alternatively, we can require for $k \geq 1$ that we initialize flow $f_{r,k+1}$ with $f_{r,k}$ and augment it to a maximum flow using for example shortest augmenting path algorithms.

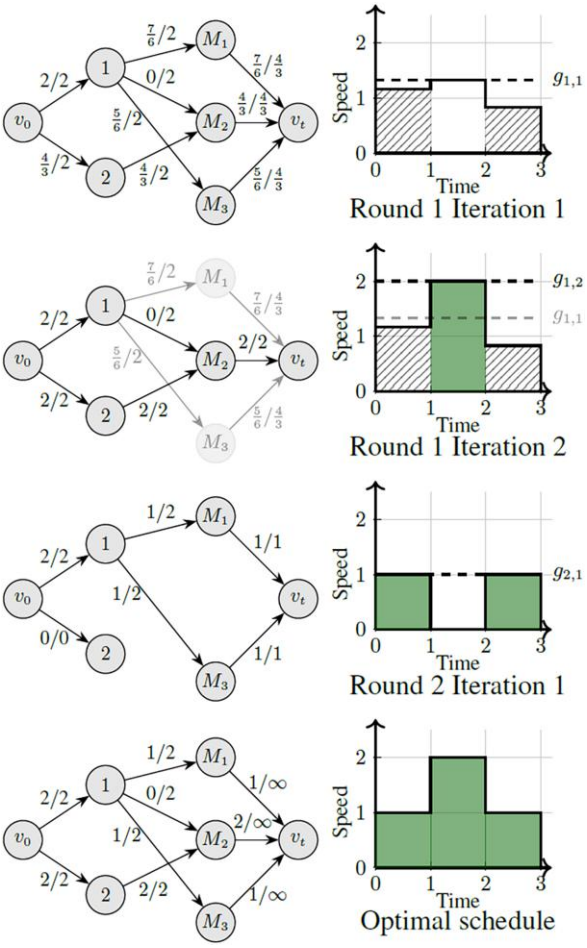
The optimal flow output by the algorithm is $f = \sum_r f_r|_{M_r^*}$. Implicitly, we use that augmenting paths in future rounds will not reshuffle the already determined subschedule induced by the critical intervals. We will come back to that in Lemma 4. For more information about augmenting paths and their relation to maximum flows, please refer to Edmonds and Karp (1972).

The algorithm to derive a feasible schedule within a round is summarized in Algorithm 1. This algorithm is then embedded in the global algorithm (Algorithm 2) described in this section, outputting a flow f corresponding to an optimal EV charging schedule. We refer to this algorithm as FOCS. To illustrate, Figure 6 displays both the flow and aggregated power profile of an example instance over the rounds and iterations of the algorithm. Here, the first three flows display $f_{r,k}$, whereas the last flow is the optimal flow f. In the respective power profiles corresponding to the flow-induced schedules, shaded intervals are parked, and solidly shaded intervals are critical. In general, maximum flows are not unique. To illustrate that, the first flow is deliberately chosen such that the flow through $(1, M_1)$ differs from that through $(1, M_3)$. Note how the optimal power profile in the bottom graph is a sum of the solidly shaded components at the end of each round of the algorithm. Note that Step 10 in Algorithm 2 can be reformulated as a recursion by calling FOCS (G_r).

Algorithm 1 (Round)
Input: G_r, r, $\mathcal{M}_a$
Output: feasible flow f_r, critical sets M_r^*
1: Initialize: $\mathcal{M}_p = \emptyset$, $k = 0$, $G_{r,k} = G_r$
2: $c_{M_i, v_t} = g_{r,k}(i)$ $\forall i \in \mathcal{M}_a$
3: Determine a maximum flow $f_{r,k}$
4: $\mathcal{M}_p = \mathcal{M}_p \cup \{i \in \mathcal{M}_a \mid i \text{ subcritical in } f_{r,k}\}$
5: $\mathcal{M}_a = \mathcal{M}_a \setminus \mathcal{M}_p$
6: **if** $|f_{r,k}| = \sum_{j=1}^{n} c_{v_0,j}$ **then**
7: **return** $f_r = f_{r,k}$, $M_r^* = \mathcal{M}_a$
8: **else**
9: $G_{r,k+1} = G_{r,k}$ with capacities reduced by $f_{r,k|\mathcal{M}_p}$ and vertices M_i removed for subcritical M_i
10: $k = k + 1$ and repeat from Step 2
11: **end if**

Figure 6. (Color online) Intermediate States of FOCS for an Example Instance Tracked over Rounds and Iterations

j	r_j	d_j	p_j	ℓ_j
1	0	3	2	2
2	1	2	2	2



Algorithm 2 (FOCS)
Input: G
Output: optimal flow f
1: Initialize: $\mathcal{M}_a = \mathcal{M}$, $\mathcal{M}_p = \emptyset$, $r = 0$, $G_r = G$, f
2: $f_r, M_r^* = \text{ROUND}(G_r, r, \mathcal{M}_a)$
3: $\mathcal{M}_a = \mathcal{M}_a \setminus M_r^*$
4: $f = f + f_r|_{M_r^*}$
5: $G_{r+1} = G_r$ with capacities reduced by $f_r|_{M_r^*}$ and vertices M_i removed for $i \in M_r^*$
6: $r = r + 1$
7: **if** $\mathcal{M}_a = \emptyset$ **then**
8: **return** f
9: **else**
10: Repeat from Step 2
11: **end if**

3.4. Algorithm Analysis

In this section, we analyze FOCS, the algorithm presented above. In particular, Section 3.4.1 shortly

discusses its time complexity and properties, after which its optimality is proved in Section 3.4.2.

3.4.1. Properties and Time Complexity. In this section, we discuss some properties and lemmas that apply to the flow model and algorithm. In particular, we establish some building blocks that enable us to prove optimality of the algorithm in Section 3.4.2.

Lemma 2. *If an instance has a feasible schedule, FOCS terminates and outputs a feasible schedule. Its time complexity is bound by $\mathcal{O}(n^2\mu)$, where $\mathcal{O}(\mu)$ is the time complexity of the used maximum flow algorithm.*

Proof of Lemma 2. First, we argue that FOCS terminates and analyze its time complexity. We do this by arguing that the number of iterations in Algorithm 1, and the number of rounds in Algorithm 2 calling Algorithm 1 are finite.

Any feasible schedule s can be directly translated to a feasible flow for G_r by sending $\int_{t_i}^{t_{i+1}} s_j(t)\,dt$ units of flow through edge (j, M_i) for each job j and $i \in J^{-1}(j)$. The flows through edges in D_0 and D_t follow directly by flow conservation. Therefore, there exists a maximum flow for the input to Algorithm 1 that saturates all edges in D_0. The algorithm (and therefore the current round) finishes once the if condition in Step 6 is satisfied, that is, if we find a maximum flow that saturates all edges outgoing of sink node v_0 when using edge capacities $g_{r,k}$ for the network. If Step 6 is false, there exists at least one subcritical interval M_i for which the flow through (M_i, t) is strictly below capacity $g_{r,k}(i)$. In each iteration, at least one such interval is removed from the network, until $g_{r,k}$ is increased sufficiently to find a maximum flow satisfying the if condition. The number of intervals is finite, and therefore the number of iterations in Algorithm 1 is finite. In particular, we can bound this number to at most $2n-1$ iterations per round, based on the fact that the number m of atomic intervals is bound by the number of jobs j, implying that $m \le 2n$. Note that there are efficient algorithms available to solve maximum flow problems, for example, Ford and Fulkerson (1956), Edmonds and Karp (1972), Dinitz (1970), and Karzanov (1974). Furthermore, a comprehensive overview of traditional polynomial time maximum flow algorithms is given by Goldberg and Tarjan (1988). Denoting their time complexity by μ, we find that Algorithm 1 has a time complexity of $\mathcal{O}(n\mu)$.

As at the end of each round at least one interval is critical and therefore removed from $\mathcal{M}_a$, the finite number of intervals implies that the if condition in Step 7 of Algorithm 2 is satisfied after at most $2n-1$ rounds and hence FOCS terminates. This implies that the time complexity of FOCS is bound by $\mathcal{O}(n^2\mu)$.

Note that the time complexity for the EV charging setting may be reduced further by exploiting the underlying structure of EV charging schedules and by considering the decrease in network size over the rounds of the algorithm. In particular, we may initialize the flow of any iteration with the flow found in the previous iteration of the same round. Furthermore, there are maximum flow algorithms that are cubic in the number of nodes (Goldberg and Tarjan 1988). As the largest flow network that is considered in FOCS (the network in the initial round) has $n+m+2 \le 3n+2$ nodes, a rough upper bound of the time complexity of maximum flows in FOCS is given by $\mu \le n^3$.

Finally, feasibility of the output follows from the defined edge capacities of the considered network. Following the order of DSL feasibility conditions listed in Definition 1, we conclude the following.

i. Every job j is fully processed within its availability since by flow conservation the exact amount of work done per job within its availability is the flow through edge (v_0, j). The algorithm only terminates once that edge is saturated, that is, if $f(v_0, j) = p_j$.

ii. An edge (j, M_i) is in D_1 if and only if $j \in J(i)$. Therefore, assuming the default value is zero, all decision variables $e_{i,j}$ for which $i \in \mathcal{M} \setminus J^{-1}(j)$ are zero, implying that the speed of j outside its availability is zero.

iii. Each job respects its speed limit by the capacities defined for edges in D_1.

Therefore, the output of FOCS is feasible. □

Next, we extend on the concept of work transferability as described by Antoniadis et al. (2017) to integrate job-specific speed limits.

Definition 6 (Work Transferability). If for a given schedule and atomic intervals M_i and $M_{i'}$, there exists a job $j \in J(i) \cap J(i')$ such that $p_{i,j} > 0$ and $p_{i',j} < p^{\max}_{i',j}$, we state that the work-transferable relation $i \to i'$ holds. Furthermore, let $\twoheadrightarrow$ be the transitive closure of $\to$.

Intuitively, if we have work transferability from one atomic interval M_i to another atomic interval $M_{i'}$, then we can transfer some work that was scheduled during M_i to $M_{i'}$. In EV charging terms, this implies that we can advance or delay some charging from one period in time to another. Applying the concept to flows, we can make the following statement.

Lemma 3 (Work Transferability in Flows). *For a given schedule and atomic intervals M_i and $M_{i'}$, we have $i \to i'$ if and only if there exists a path $(M_i, j, M_{i'})$ in the residual graph corresponding to the schedule, where $j \in \mathcal{J}$. Similarly, we have $i \twoheadrightarrow i''$ if and only if in the residual network corresponding to the schedule there exists an $(M_i$-$M_{i''})$ path through interval and job vertices only.*

Proof of Lemma 3. We show only the first statement as the extension follows naturally using concatenations of

paths. Assume that $i \to i'$. Then there exists a job j such that $j \in J(i) \cap J(i')$ with $p_{i,j} > 0$ and $p_{i',j} < p^{\max}_{i',j}$. The former implies that edge (M_i, j) exists in the residual graph. As $c_{j,M_{i'}} = p_{i',j} < p^{\max}_{i',j}$, edge $(j, M_{i'})$ is in the residual graph. This proves existence of path $(M_i, j, M_{i'})$ in the residual graph.

For the opposite direction, assume the existence of a path $(M_i, j, M_{i'})$. Because $j \in \mathcal{J}$, we know the edge capacity $c_{j,M_{i'}}$ in the original network to be $p^{\max}_{i',j}$. The existence of the edge in the residual graph implies that for the flow going through this edge that is defined by the schedule to be $p_{i',j}$, we have $p_{i',j} < p^{\max}_{i,j}$. Furthermore, existence of edge (M_i, j) in the residual graph indicates positive flow through (j, M_i) in the original network, implying $p_{i,j} > 0$. From the presence of both edges, it follows that $j \in J(i) \cap J(i')$, proving that $i \to i'$. □

Figure 7 illustrates the concept of work transferability. Here, dashed edges are those that are not in the original network but might be present in the residual network. Lemma 3 translates work transferability to the existence of paths oscillating between the interval and job layers in the residual network.

Next, we consider two lemmas that have a more direct relation to the algorithm.

Lemma 4 (Isolation of Critical Intervals). *If M_i is a critical interval in round r and if the round consists of multiple iterations whereby $M_{i'}$ was subcritical in one of those iterations, then there is no work-transferable relation between i and i' in the schedule corresponding to the flow at the end of round r, that is, $i \nrightarrow i'$ with respect to flow f_r.*

Proof of Lemma 4. We prove the lemma by constructing an augmenting path (Figure 8). Assume in round r interval $M_{i'}$ was parked in iteration k and let $f_{r,k}$ be the flow at the end of iteration k. Because $M_{i'}$ is subcritical, we have $|f_{r,k}| < \sum_{j=1}^{n} p_j$, implying that for the next iteration the lower bound $g_{r,k}$ will be increased to

$$g_{r,k+1}(i'') = g_{r,k}(i'') + \frac{\sum_{j=1}^{n} c_{v_0,j} - |f_{r,k}|}{L(\mathcal{M}_a)} |M_{i''}| \quad \forall i'' \in \mathcal{M}_a.$$

Figure 7. (Color online) Work Transferability Represented in Flows

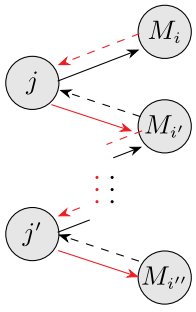
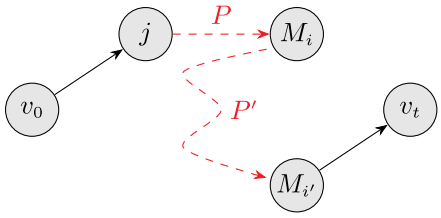


Figure 8. (Color online) Augmenting Path in Proof of Lemma 4



By criticality of M_i, the interval is active at the end of the iteration, implying $g_{r,k+1}(i) > g_{r,k}(i)$. Furthermore, criticality implies that there is no iteration in this round where M_i is subcritical. Combing those facts, the flow through (M_i, t) increases in iteration $k+1$ compared with iteration k. This is only possible if there is a job j such that (v_0, j) is not saturated and there exists a (j-M_i) path P in the residual graph. Furthermore, note that because $M_{i'}$ is being parked in iteration k, edge $(M_{i'}, t)$ is not saturated and therefore exists in the residual graph. Now, assume $i \nrightarrow i'$. By Lemma 3, there exists an (M_i-$M_{i'}$) path P' that passes only through job and interval vertices. This implies that $P'' = (v_0, P, P', v_t)$ exists in the residual graph and contains an (v_0-v_t) path, proving existence of an augmenting path in $f_{r,k}$. This contradicts maximality of the flow, implying $i \nrightarrow i'$. □

Intuitively, this lemma says that we cannot push any charging from (high power) critical intervals to (low power) subcritical intervals. This is in line with the notion of critical intervals as introduced for the YDS algorithm and will be a key element in the optimality proof in Section 3.4.2. Furthermore, this particular lemma justifies that we fix the schedule of critical intervals at the end of each round.

For the next lemma, we first introduce the notion of ranks.

Definition 7 (Rank). The rank $r(i)$ of an atomic interval M_i is defined as the round $r(i)$ in which M_i was critical, that is, $i \in M^*_{r(i)}$.

Lemma 5 (Monotonicity). *For the schedule corresponding to output flow f of algorithm FOCS and atomic intervals M_i and $M_{i'}$ where $r(i) < r(i')$, the aggregated power in M_i is strictly larger than in $M_{i'}$, that is,*

$$\frac{f(M_i, t)}{|M_i|} > \frac{f(M_{i'}, v_t)}{|M_{i'}|}.$$

Proof of Lemma 5. We prove the lemma by contradiction, whereby we consider flows at the end of rounds. Let interval M_i be the lowest ranked interval such that its aggregated power level in the flow output by the algorithm is larger than that in intervals with rank $r(i) - 1$. As the algorithm does not change schedules at critical intervals, this already occurs at

round $r(i)$ itself. Given that M_i was subcritical in the previous round, the speed level for M_i increased. In particular, there is a job j for which the speed during interval M_i increased compared with the previous round. Furthermore, we know that the flow through (j, M_i) is positive in round $r(i)$, implying that edge (M_i, j) is in the residual graph. However, applying Lemma 4 to the previous round, the amount of work scheduled for job j remains the same. Therefore, there is an interval $M_{i'}$ for which the flow through $(j, M_{i'})$ decreased compared with the previous round. As a consequence, the flow in round $r(i)$ does not saturate the edge, implying that edge $(j, M_{i'})$ is in the residual graph. Combining these findings, the path $(M_i, j, M_{i'})$ is in the residual graph, implying $i \to i'$, and thus contradicting Lemma 4. □

The lemma shows that the average aggregated speed of atomic intervals is decreasing in their rank. Therefore, critical intervals as determined using the method presented in this paper share the monotonicity property known for YDS for corresponding DS instances. We first find those intervals with the highest intensity, and then iteratively determine the next highest speeds.

We also note that, similarly to YDS, the power profile output by FOCS is unique if the objective function is strictly convex. However, this does not necessarily apply to the schedule.

3.4.2. Optimality Proof. In this section, we prove that Algorithm 2 as described in Section 3.3.2 computes a solution s that is optimal under objective function $F(s)$. To this end, we first prove some auxiliary lemmas that show compliance with the sufficient conditions derived in Section 3.2.

Lemma 6. *The output of Algorithm 2 complies with Condition* KKT1.

Proof of Lemma 6. If in the final output of the algorithm there are two distinct atomic intervals M_i and $M_{i'}$ such that for job j we have $0 < p_{i,j}/|M_i| = p_{i',j}/|M_{i'}| < \ell_j$, then by definition of work transferability, we have $i \to i'$ and $i' \to i$. By Lemma 4 and the strict monotonicity in Lemma 5, this implies that the aggregated speed in both intervals is the same. □

Lemma 7. *The output of Algorithm 2 complies with Condition* KKT2.

Proof of Lemma 7. Let $i \in J^{-1}(j)$ be such that $p_{i,j} = 0$ in the output of the algorithm. Assume that there is an interval $M_{i'}$ with $i \neq i'$ and $i' \in J^{-1}(j)$ for which the aggregated power in $M_{i'}$ is strictly greater than in M_i, that is, $\sum_{j=1}^{n} p_{i,j}/|M_i| < \sum_{j=1}^{n} p_{i',j}/|M_{i'}|$. By Lemma 5, we have $r(i') < r(i)$, implying by Lemma 4 that $i' \nrightarrow i$. Applying the definition of work transferability, it follows that $p_{i',j} = 0$, proving compliance with Condition KKT2. □

Lemma 8. *The output of Algorithm 2 complies with Condition* KKT3.

Proof of Lemma 8. Let job j run at maximum speed in M_i in the schedule found by FOCS. Assume that there is an interval $M_{i'}$ with $i \neq i'$ and $i' \in J^{-1}(j)$, such that the aggregated speed in M_i is strictly greater than in $M_{i'}$. By Lemma 5, we know that $r(i) < r(i')$. Therefore, by Lemma 4, there is no work-transferable relation between i and i' ($i \nrightarrow i'$). From the definition of work transferability, it now follows directly that $p_{i',j} \geq p^{\max}_{i',j}$, proving compliance with Condition KKT3. □

Combining all discussed above, we conclude optimality of the algorithm output.

Theorem 1 (Optimality). *For any feasible input instance, the schedule produced by Algorithm 2 is an optimal solution minimizing the integral of any strictly convex and increasing function of the aggregated output powers.*

Proof of Theorem 1. The proof follows directly from the KKT conditions derived in Section 3.2, the inherent feasibility of the output and Lemmas 6–8. □

To summarize, this section considered the offline DSL problem, applicable to EV scheduling in MPC settings. In particular, we analyzed the relation between solutions of DSL instances and their corresponding DS instances. Furthermore, we derived necessary and sufficient optimality conditions for DSL schedules and presented an offline algorithm that determines an optimal schedule in $\mathcal{O}(n^2\mu)$ time where $\mathcal{O}(\mu)$ is the complexity of an efficient maximum flow algorithm. Lastly, we provided proof of the optimality of the output of the algorithm.

4. Online Scheduling

As discussed in the Introduction, MPC is a much-deployed framework for coordinated EV charging, especially due to its usability to bridge data gaps. However, it is an interesting and important question to ask how close to an optimal solution such frameworks can get, and in particular, how close to optimal an MPC, or intraday controller, can get, assuming perfect knowledge on an EV's characteristics upon arrival. Note that in that case, the model component of the MPC was clairvoyant.

Also from a theoretical point of view, considering DSL in an online setting is a natural next step. Therefore, in this section, we are interested in schedules constructed *online*, that is, schedules where jobs are released one by one, and the algorithm only gets to know their characteristics at their respective release times.

4.1. Preliminaries Online Algorithms

We define the online variant of a job scheduling problem to be such that the existence and characteristics of

jobs become known at their respective release times. In this section, we analyze online algorithms for DSL in terms of their respective *competitive ratio*.

Definition 8. Given a deterministic algorithm ALG that for any DSL instance $I \in \mathcal{I}_{DSL}$ determines a feasible schedule $s^{\text{ALG}}(I)$, and given an optimal solution $s^*(I)$, the competitive ratio of the algorithm is defined as

$$\sup_{I \in \mathcal{I}_{DSL}} \frac{E(s^{\text{ALG}}(I))}{E(s^*(I))}. \tag{12}$$

The definition carries over to DS instances.

Two classical online approaches for DS are AVR and OA (Yao et al. 1995). Given the connection between DSL and DS, we first provide a short description of those two algorithms, before relating them to DSL.

4.1.1. AVR. AVR for DS works in two steps. First, upon release, job j is scheduled at speed $p_j/(d_j - r_j)$ throughout its availability; that is, each job is scheduled at the constant speed corresponding to the average speed it needs to complete its work between release time and deadline. The resulting schedule may not be feasible for DS because jobs may run simultaneously, but it gives a useful initial speed profile. Therefore, in a second step, EDF is applied using the speed profile resulting from the initial schedule.

4.1.2. OA. OA reoptimizes the remaining problem instance each time a new job is released. In particular, let s be an optimal schedule for jobs $\mathcal{J} = \{1, \dots n\}$ and instance $I = \langle \vec{r}, \vec{d}, \vec{p} \rangle$. Let t' be the first point in time where a new job $n+1$ is released. We then define the remaining instance at point t' according to schedule s as $I' = \langle \vec{r}', \vec{d}', \vec{p}' \rangle$, where

$$\vec{r}'_j = t' \qquad \forall j \in \mathcal{J} \cup \{n+1\}$$
$$\vec{d}'_j = d_j \qquad \forall j \in \mathcal{J} \cup \{n+1\}$$
$$\vec{p}'_j = \begin{cases} p_j - \int_{r_j}^{t'} s_j(t)\,dt & \text{if } j \in \mathcal{J} \\ p_{n+1} & \text{if } j = n+1. \end{cases}$$

For convenience, if a job has remaining workload 0 or if its deadline is at most t', we remove it from the remaining problem instance. We now determine an optimal schedule s' for I'. The updated schedule $\bar{s}$ for OA at time t' is such that

$$\bar{s}_j(t) = \begin{cases} s_j(t) & \text{if } j \in \mathcal{J} \wedge t < t' \\ s'_j(t) & \text{if } j \in \mathcal{J} \wedge t \geq t' \\ s'_{n+1}(t) & \text{if } j = n+1. \end{cases}$$

Iteratively repeated over the time horizon every time a new job is released, this results in a schedule s_{OA} for OA.

Note that OA may be applied to either DS or DSL, with the difference being the algorithm applied in the optimization subroutine. Optimization for DS may be done by applying YDS, whereas for DSL FOCS is a suitable optimization algorithm.

4.2. AVR for DSL

In the following, we discuss the application of AVR to DSL instances. As remarked earlier, EDF does not necessarily result in a feasible schedule for DSL instances, even if it follows a profile for which there exists a feasible schedule (Figure 3). However, because feasible schedules for DSL allow more than one job to be processed at any time and because assumption (3) holds, we can adapt AVR to DSL instances by skipping the last step (therefore not applying EDF) to find a feasible schedule. In other words, upon release, we schedule any job j at speed $p_j/(d_j - r_j)$ for the next $d_j - r_j$ units of time.

We analyze the performance guarantee of applying AVR to DSL instances by relating the resulting schedules to those resulting from applying AVR to DS instances as follows. Assume we are given an instance $I = \langle \vec{r}, \vec{d}, \vec{p}, \vec{\ell} \rangle$. Let $s^{\text{DSL, AVR}}$ be the schedule for I found by AVR without EDF and let $s^{\text{DSL},*}$ be an optimal schedule. Furthermore, let $s^{\text{DS, AVR}}$ be the schedule for the corresponding augmented DS instance I' (i.e., $I \in a(I')$) found by AVR with EDF, and let $s^{\text{DS},*}$ be an optimal schedule for I'. Note that $\text{Pf}_{s^{\text{DS, AVR}}} = \text{Pf}_{s^{\text{DSL, AVR}}}$, and therefore their objective values are the same. Thus,

$$E(s^{\text{DSL, AVR}}) = E(s^{\text{DS, AVR}}) \tag{13a}$$
$$\leq 2^{\alpha-1}\alpha^\alpha E(s^{\text{DS},*}) \tag{13b}$$
$$\leq 2^{\alpha-1}\alpha^\alpha E(s^{\text{DSL},*}). \tag{13c}$$

Here, (13a) follows from the fact that the AVR (respectively, with and without EDF) schedules $s^{\text{DS, AVR}}$ and $s^{\text{DSL, AVR}}$ have the same speed profile, (13b) follows from the known tight competitive ratio for AVR with EDF for DS instances (Bansal et al. 2007) and Lemma 1, and (13c) follows from Corollary 1.

Furthermore, we can conclude that the upper bound on the competitive ratio for AVR without EDF for DSL instances is the same as the upper bound for the competitive ratio for AVR with EDF for DS instances. In particular, assume that $I' = \langle \vec{r}, \vec{d}, \vec{p} \rangle$ was an instance for which the inequality in (13b) is an equality. Based on this, we can construct a DSL instance $I \in a(I')$ by taking $\ell_j = \max_t \text{P}_{s^{\text{DS},*}}(t)$. Then, the energy of the respective optimal schedules of I and I' are the same.

4.3. OA for DSL

In this section, we adapt the potential function approach that Bansal et al. (2007) used to analyze the competitive ratio of OA for DS instances to analyze OA for DSL instances. In particular, we show the competitive ratio of α^α to be tight, where α is the same as in the

energy function defined in (2). Notably, the competitive ratio for DS and DSL instances is the same.

First, we remark that, although in each reoptimization step of OA, there is a unique aggregated speed profile, the schedule is not necessarily unique. Therefore, we explicitly note that throughout the upcoming analysis, we consider a fixed (arbitrary) realization of OA.

Next, we introduce some notation, before giving the potential function, and deriving the competitive ratio. Let $\mathrm{P}_{\mathrm{F}_{\mathrm{OA}}}(t)$ be the aggregated speed at which OA runs at time t, and similarly let $\mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t)$ be the aggregated speed at which OPT runs at time t, where OPT is an optimal algorithm leading to an optimal (offline) schedule s_{OPT}. Note that these are the speed profiles as realized at the end of the time horizon and that OA recomputes a schedule every time a new task is released. Therefore, for current time t_0, we introduce schedule s and corresponding speed $\mathrm{P}_{\mathrm{F}_s}(t)$, at which OA runs at time $t \geq t_0$ if no new jobs are released after t_0. Although it holds that $\mathrm{P}_{\mathrm{F}_s}(t_0) = \mathrm{P}_{\mathrm{F}_{\mathrm{OA}}}(t_0)$, this is generally not the case for $t > t_0$ because a job may be released in interval (t_0, t).

At this point t_0, OA may be interpreted to solve a DSL instance where all tasks have the same release time t_0. Therefore, the resulting speed profile $\mathrm{P}_{\mathrm{F}_s}(t)$ for $t > t_0$ is a nonincreasing step function. Let t_1 be the first point in time after t_0 where a step occurs and the speed profile changes, that is,

$$t_1 = \sup\{t \geq t_0 \mid \mathrm{P}_{\mathrm{F}_s}(t) = \mathrm{P}_{\mathrm{F}_s}(t_0)\}.$$

Applying this inductively, we find breakpoints t_i for $i \geq 0$, such that $[t_i, t_{i+1})$ are the inclusion maximal intervals with constant speed profiles. As Bansal et al. (2007), we call such intervals *critical*, and define $M_i := [t_i, t_{i+1})$. Note that both criticality and the intervals M_i are being redefined here compared with their use in their section 3.

We define $w_{\mathrm{OA}}(t, t')$ as the work done by OA in interval $(t, t']$ that is already available (and unfinished) at current time t_0. Moreover, let $w_{\mathrm{OA}}(t, t')/(t' - t)$ be the density of interval $(t, t']$. We note that this definition of w is different from the one given by Bansal et al. (2007) to account for the methods needed to solve DSL. For critical intervals M_i, we can relate this workload to the speed profile by

$$\mathrm{P}_{\mathrm{F}_s}(t) = \frac{w_{\mathrm{OA}}(t_i, t_{i+1})}{t_{i+1} - t_i},$$

for $t \in M_i$. For notational purposes, we shorten $w_{\mathrm{OA}}(t_i, t_{i+1})$ to $w_{\mathrm{OA}}(i)$ in places where t_i and t_{i+1} are clear from the context.

Similarly to w_{OA}, we define $w_{\mathrm{OPT}}(t, t')$ to be the work done by optimal schedule OPT in interval $(t, t']$ that is already available at current time t_0. Note that as opposed to OA, OPT is aware of tasks that will be released in the future. Therefore, the speed profile induced by w_{OPT} is not necessarily nonincreasing, as shown in Figure 9. The figure shows the part of the optimal speed profile corresponding to jobs that are already available. In the middle, there is a valley in the profile, where the second (not yet available) job will be scheduled. Moreover, note that the breakpoints determined based on $\mathrm{P}_{\mathrm{F}_s}(t)$ above do not necessarily align with changes in speed in the (partial) speed profile of OPT.

Finally, we define the potential function at current time $t = t_0$ to be

$$\Phi(t) = \alpha \sum_{i \geq 0} \mathrm{P}_{\mathrm{F}_s}(t_i)^{\alpha - 1} (w_{\mathrm{OA}}(i) - \alpha w_{\mathrm{OPT}}(i)).$$

Note that breakpoints t_i on the right-hand side of the formula depend on function input t. Furthermore, note that the form of the introduced potential function is similar to the potential function used by Bansal et al. (2007) to derive the competitive ratio for OA under DS. The main differences lie in the problem definition and in the definition of w_{OA} and w_{OPT}. The proof for DSL follows the same structure as that for DS presented by Bansal et al. (2007). However, it is not evident that the same form of potential function applies to DSL. Therefore, we work out the details in places where speed limits play a role and include the proof for DSL despite the similarities.

In the following, we show the following.

Lemma 9. *The potential function $\Phi(t)$ satisfies the following conditions:*

1. *Boundary property: $\Phi(t) = 0$ for any t before the first release time and for any t after the last deadline.*

2. *Job release and completion property: At any time t where a new task is released, or a task is completed by s_{OPT} or s_{OA}, the potential function is nonincreasing, that is,*

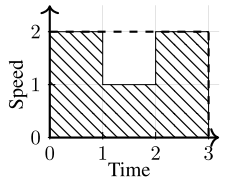
$$\lim_{t' \uparrow t} \Phi(t') \geq \lim_{t' \downarrow t} \Phi(t'). \tag{14}$$

3. *General property: For a time t at which no job is released, we have*

$$\mathrm{P}_{\mathrm{F}_{\mathrm{OA}}}(t)^{\alpha} + \frac{d\Phi(t)}{dt} \leq \alpha^{\alpha} \mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t)^{\alpha}. \tag{15}$$

Figure 9. Example of an Instance Where the Speed Profile Corresponding to Available Work w_{OPT} at Time $t_0 = 0$ (Profile of Shaded Area) Is Not a Nonincreasing Step Function

j	r_j	d_j	p_j	ℓ_j
1	0	3	5	2
2	1	2	1	2



Note. The dashed line corresponds to the optimal speed profile at the end of the time horizon.

Bansal et al. (2010) have previously proven similar properties for the potential function of another speed scaling model to derive the corresponding competitive ratio. The proof of Lemma 9 uses the following lemma taken from Bansal et al. (2007). We refer to the original paper for its proof.

Lemma 10 (Bansal et al. 2007, Lemma 3.3). *Let* $q, r, \delta \geq 0$, *and* $\alpha \geq 1$. *Then* $(q+\delta)^{\alpha-1}(q-\alpha r-(\alpha-1)\delta)-q^{\alpha-1}(q-\alpha r)\leq 0$.

Proof of Lemma 9. We consider each property separately.

For Property 1, note that for any t before the first release, or after the last deadline, there will be no tasks to schedule for OA, resulting in speed $\mathrm{P}_{\mathrm{F}_s}(t_i)=0$ for all $i\geq 0$. Hereby, $\Phi(t)=0$ for such t.

For Property 2, we first consider the case where a new job is released. Assume task j is released at time t_0 with deadline $d_j\in M_i$, requiring an amount x of work. This release may affect the breakpoints of the critical intervals. Note that the speed profile $\mathrm{P}_{\mathrm{F}_s}(t)$ before the release was a nonincreasing step function. To add a job, we add its associated work to the latest intervals for which the job is available. We consider the addition of work in increments; that is, we add some $x'\leq x$ of work until one of the following cases occurs.

- The speed in M_i increases to the speed in M_{i-1}. In particular $\mathrm{PF}_s(t_i)=(w(t_i,t_{i+1})+x')/(t_{i+1}-t_i)=\mathrm{PF}_s(t_{i-1})$.
- Two critical intervals M_i and M_{i+1} merge into one new critical interval.
- The speed at which job j runs in interval M_i reaches the job-specific speed limit ℓ_j.
- The interval M_i splits into two critical intervals $M_{i'}$ and $M_{i''}$. This may occur due to deadlines that do not match the already existing breakpoints or due to speed limits being reached in parts of recently merged critical intervals.
- The job is completely scheduled (i.e., $x'=x$).

Because of the speed limits, we have to carefully keep track of the added work x' thus far, before adding the next part of the work, until the whole amount of work x is scheduled.

We start with cases that do not change the structure of critical intervals. Those are the cases where M_i increases to the speed in M_{i-1}, where in M_i job j reaches its job-specific speed limit ℓ_j, and where the remaining work can be scheduled within M_i without triggering any of the other events.

By definition, OA schedules all additional work x' during M_i. Therefore, the only values associated with OA that change are $\mathrm{P}_{\mathrm{F}_s}(t)$ for $t\in M_i$ and $w_{\mathrm{OA}}(i)$. For the optimal schedule OPT, no such claim can be made. Therefore, we denote $x'_{i'}\geq 0$ to be the work scheduled for interval $M_{i'}$ for $0\leq i'\leq i$ where $\sum_{i'=0}^{i}x'_{i'}=x'$.

We initially consider the ith term of the potential function separately. Speed function $\mathrm{P}_{\mathrm{F}_s}(t)$ changes by $x'/(t_{i+1}-t_i)$ for $t\in M_i$. To compare the change in potential function, we denote the new speed as

$$\mathrm{P}_{\mathrm{F}_{s'}}(t)=\begin{cases}\dfrac{w_{\mathrm{OA}}(i)+x'}{t_{i+1}-t_i} & \text{if } t\in M_i\\ \mathrm{P}_{\mathrm{F}_s}(t) & \text{otherwise.}\end{cases}\tag{16}$$

We further note that $w_{\mathrm{OA}}(i)$ increases by x', and $w_{\mathrm{OPT}}(i)$ increases by x'_i. That gives a total change of

$$\begin{aligned}&\mathrm{P}_{\mathrm{F}_{s'}}(t_i)^{\alpha-1}(w_{\mathrm{OA}}(i)+x'-\alpha(w_{\mathrm{OPT}}(i)+x'_i))\\&-\mathrm{P}_{\mathrm{F}_s}(t_i)^{\alpha-1}(w_{\mathrm{OA}}(i)-\alpha w_{\mathrm{OPT}}(i)),\end{aligned}\tag{17}$$

for the ith term. In the term for $i'\in\{0,\dots,i-1\}$, the values $w_{\mathrm{OA}}(i')$ do not change after adding work x' because the work is added to a different interval, namely M_i. However, $w_{\mathrm{OPT}}(i')$ increases by $x'_{i'}$. If we sum the change for all such i', we find the following expression:

$$\begin{aligned}&\sum_{i'=0}^{i-1}(\mathrm{P}_{\mathrm{F}_s}(t_{i'})^{\alpha-1}(w_{\mathrm{OA}}(i')-\alpha(w_{\mathrm{OPT}}(i')+x_{i'}')))\\&-\mathrm{P}_{\mathrm{F}_s}(t_{i'})^{\alpha-1}(w_{\mathrm{OA}}(i')-\alpha w_{\mathrm{OPT}}(i'))\\&=\sum_{i'=0}^{i-1}\mathrm{P}_{\mathrm{F}_s}(t_{i'})^{\alpha-1}(-\alpha x_{i'}').\end{aligned}\tag{18}$$

To show that (14) holds in the release case, we denote $\Delta\Phi(t)=\lim_{t'\uparrow t}\Phi(t')-\lim_{t'\downarrow t}\Phi(t')$. We bring all terms together and conclude

$$\Delta\ \Phi(t_0)\tag{19}$$

$$=\sum_{i'=0}^{i-1}\mathrm{P}_{\mathrm{F}_s}(t_{i'})^{\alpha-1}(-\alpha x'_{i'})\tag{20}$$

$$\begin{aligned}&+\mathrm{P}_{\mathrm{F}_{s'}}(t_i)^{\alpha-1}(w_{\mathrm{OA}}(i)+x-\alpha(w_{\mathrm{OPT}}(i)+x'_i))\\&-\mathrm{P}_{\mathrm{F}_s}(t_i)^{\alpha-1}(w_{\mathrm{OA}}(i)-\alpha w_{\mathrm{OPT}}(i))\\&\leq\mathrm{P}_{\mathrm{F}_{s'}}(t_i)^{\alpha-1}\left(w_{\mathrm{OA}}(i)+x-\alpha\left(w_{\mathrm{OPT}}(i)+\sum_{i'=0}^{i}x'_{i'}\right)\right)\end{aligned}\tag{21}$$

$$\begin{aligned}&-\mathrm{P}_{\mathrm{F}_s}(t_i)^{\alpha-1}(w_{\mathrm{OA}}(i)-\alpha w_{\mathrm{OPT}}(i))\\&=\mathrm{P}_{\mathrm{F}_{s'}}(t_i)^{\alpha-1}(w_{\mathrm{OPT}}(i)+x-\alpha(w_{\mathrm{OPT}}(t_i,t_{i+1})+x'))\\&-\mathrm{P}_{\mathrm{F}_s}(t_i)^{\alpha-1}(w_{\mathrm{OA}}(i)-\alpha w_{\mathrm{OPT}}(i))\\&\leq 0.\end{aligned}\tag{22}$$

Here, (20) follows from the derivations in (17) and (18). In (21), we made use of the fact that for all $0\leq i'<i$, we can lower bound the speed $\mathrm{P}_{\mathrm{F}_s}(t_{i'})$ by the new speed $\mathrm{P}_{\mathrm{F}_{s'}}(t_i)$. By doing so, we can exploit that $\sum_{i'=0}^{i}x'_{i'}=x'$ in the next step. Moreover, (22) is exactly the case discussed in the proof of Theorem 3.4 by Bansal et al. (2007) where they do inequality manipulations, apply Lemma 10 (Lemma 3.3 in Bansal

et al. (2007)), and conclude nonpositivity. For the details, we refer the reader to that work.

For the cases where interval M_i either splits in two or merges with another, at that point, the densities between old and new intervals are constant, leaving the potential function unchanged.

We conclude that the potential function does not increase if a new task is released.

Next, consider the case where OA finishes at least one task at time t. Either

$$\lim_{t' \uparrow t} \mathrm{P}_{\mathrm{F}_s}(t') = \lim_{t' \downarrow t} \mathrm{P}_{\mathrm{F}_s}(t'), \tag{23}$$

in which case the task finished strictly within the critical interval, leaving $\mathrm{P}_{\mathrm{F}_s}(t)$ unaffected and continuously reducing $w_{\mathrm{OA}}(t_0, t_1)$ by which (14) holds by continuity in t, or the equality in (23) does not hold, in which case we transition from one critical interval to the next. Then, indices shift by one, $\mathrm{P}_{\mathrm{F}_s}(t_1)$ becomes $\mathrm{P}_{\mathrm{F}_s}(t_0)$, and so on. For the potential function, the only change is the formerly first term $\alpha \mathrm{P}_{\mathrm{F}_s}(t_0)^{\alpha-1}(w_{\mathrm{OA}}(t_0, t_1) - \alpha w_{\mathrm{OPT}}(t_0, t_1))$ disappearing. However, although $\mathrm{P}_{\mathrm{F}_s}(t_0)$ remained constant, both $w_{\mathrm{OA}}(t_0, t_1)$ and $w_{\mathrm{OPT}}(t_0, t_1)$ approached zero. The first term's contribution therefore approaches zero from above as the critical interval draws to an end. The change in potential function in such a point is therefore continuous and (14) holds.

For the case where at time t, OPT finishes a task, we note that the potential function is independent of $\mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t)$, and the change in $w_{\mathrm{OPT}}(t_0, t_1)$ is continuous.

Combining the observations above, we have shown that $\Phi(t)$ satisfies Property 2.

Lastly, we show that $\Phi(t)$ has Property 3 by showing that

$$\mathrm{P}_{\mathrm{F}_{\mathrm{OA}}}(t)^{\alpha} - \alpha^{\alpha}\, \mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t)^{\alpha} + \frac{d\Phi(t)}{dt} \leq 0. \tag{24}$$

We consider the working case where in the next infinitesimally small dt time units no new task is released or completed by either OA or OPT. Furthermore, we do assume that there are tasks available and that therefore $\mathrm{P}_{\mathrm{F}_{\mathrm{OA}}}(t_0) = \mathrm{P}_{\mathrm{F}_s}(t_0) > 0$ and $\mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t_0) > 0$ for current time t_0. As OA runs, $w_{\mathrm{OA}}(t_0, t_1)$ is reduced at rate $\mathrm{P}_{\mathrm{F}_s}(t_0)$. For $i > 0$, the value of $w_{\mathrm{OA}}(i)$ remains unchanged. For the work done by OPT, we remark once more that OPT's speed profile does not necessarily align with breakpoints t_i. However, it is easy to verify that, if at any point $t > t_0$, the speed $\mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t)$ increases, at least one new task will be released at time t. Therefore, assuming that no new tasks are released in the next dt units of time, we can assume $\mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t)$ to be a nonincreasing step function over interval $[t_0, t_0 + dt)$. From this, we use that $\mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t_0)$ is an upper bound on the rate at which OPT reduces $w_{\mathrm{OPT}}(t_0, t_1)$ throughout the next dt units of time. Therefore, for (24) to hold, it suffices to show that the following inequality holds:

$$\begin{aligned}&\mathrm{P}_{\mathrm{F}_s}(t_0)^{\alpha} - \alpha^{\alpha}\mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t_0)^{\alpha} - \alpha \mathrm{P}_{\mathrm{F}_s}(t_0)^{\alpha-1}\, \mathrm{P}_{\mathrm{F}_s}(t_0)\\ &+ \alpha^2 \mathrm{P}_{\mathrm{F}_s}(t_0)^{\alpha-1} \mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t_0) \leq 0.\end{aligned} \tag{25}$$

Substituting $z = \frac{\mathrm{P}_{\mathrm{F}_s}(t_0)}{\mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t_0)}$ results in

$$\mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t_0)^{\alpha}((1-\alpha)z^{\alpha} + \alpha^2 z^{\alpha-1} - \alpha^{\alpha}) \leq 0,$$

where we note that in the working case $\mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t_0) > 0$. Therefore, consider the polynomial

$$u(z) = (1-\alpha)z^{\alpha} + \alpha^2 z^{\alpha-1} - \alpha^{\alpha}. \tag{26}$$

Evaluating this interval at the domain boundaries, we note that $\lim_{z \downarrow 0} u(z) = -\alpha^{\alpha}$ and $\lim_{z \uparrow \infty} u(z) = -\infty$ for $\alpha > 1$. For (24) to hold, it now suffices to show that the maximum of (26) does not exceed zero. To this end, we differentiate the polynomial with respect to z, finding

$$\frac{du}{dz}(z) = (\alpha - \alpha^2)z^{\alpha-1} + (\alpha^3 - \alpha^2)z^{\alpha-2}.$$

Given that $z \neq 0$, this derivative assumes its unique zero in $z = \alpha$. Substituting this value into (26), we find the maximum value of $u(\alpha) = 0$, thereby proving (24) for the working case.

Lastly, we note that the arguments above also apply to the case where no new task arrives, but a task j is completed by OPT or OA at time t_0. This is because speed profiles $\mathrm{P}_{\mathrm{F}_{\mathrm{OA}}}(t)$ and $\mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t)$ are unaffected by the completion of j, allowing us to apply the working case arguments.

This concludes the proof of Lemma 9. □

Theorem 2. *OA is α^{α}-competitive for DSL.*

Proof of Theorem 2. We first note that if we can upper bound the competitiveness by α^{α}, then this bound is tight. This follows directly from lemma 3.2 in Bansal et al. (2007), where a DS instance is presented. A corresponding DSL instance given sufficiently large speed limits (e.g., speed limits $\ell_j = w_j = (1/(n-j))^{\frac{1}{\alpha}}$), yields the same upper bound on the objective value and on the competitive ratio. For the details, we refer to Bansal et al. (2007).

As for the upper bound on the competitive ratio, we integrate (15) with regard to time to find that for any DSL instance I and corresponding OA schedule s_{OA} and optimal schedule s_{OPT}:

$$\int_t \mathrm{P}_{\mathrm{F}_{\mathrm{OA}}}(t)^{\alpha} + \int_t \frac{d\Phi(t)}{dt} \leq \alpha^{\alpha} \int_t \mathrm{P}_{\mathrm{F}_{\mathrm{OPT}}}(t)^{\alpha}, \tag{27}$$

$$E(s_{\mathrm{OA}}) + \Phi(\max_j d_j) - \Phi(\min_j r_j) \leq \alpha^{\alpha} E(s_{\mathrm{OPT}}), \tag{28}$$

$$E(s_{\mathrm{OA}}) \leq \alpha^{\alpha} E(s_{\mathrm{OPT}}), \tag{29}$$

where all integrals in (27) are taken over the positive range $\mathbb{R}_{\geq 0}$. Furthermore, in (28), we use the definition of the objective function, the fundamental theorem of

calculus, and Property 2 of Lemma 9. Finally, (29) follows from Property 1 of the same lemma, that is, from $\Phi(\max_j d_j) = \Phi(\min_j r_j) = 0$. □

4.4. Exact Scheduling Rules

As noted before, applying EDF may result in infeasible schedules for DSL instances (Figure 3). This section takes this observation a step further, concluding that there exists no deterministic online scheduling rule that given any speed profile corresponding to a feasible schedule can guarantee to find such a schedule. In this, we assume that the scheduling rule only becomes aware of job j at its release time r_j, whereas the speed profile assumes implicit knowledge of all jobs released over the time horizon.

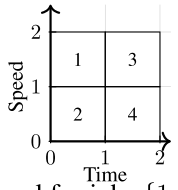
Theorem 3. *Let I be a DSL instance and let $\mathbf{P_F}$ be a speed profile for which there exists a feasible schedule s for I such that $P_{F_s} = \mathbf{P_F}$. There exists no deterministic online scheduling rule that reliably finds a feasible schedule s' for any such pair (I, P_F) for which $P_{F_{s'}} = \mathbf{P_F}$.*

Proof of Theorem 3. We prove this by providing a speed profile and two DS instances corresponding to that profile, such that for any initial scheduling decision, one of the instances cannot be feasibly scheduled. A job set with five jobs is illustrated in Figure 10. We now consider two DSL instances that are subsets of those jobs. Here, the first instance I considers jobs $\mathcal{J} = \{1,2,3,4\}$ and the second instance I' considers job set $\mathcal{J}' = \{1,2,3,5\}$. Note that the optimal speed profile for both instances is constant with speed 2 throughout the time horizon $[0, 2]$. The respective optimal schedules are shown in the figure. Furthermore, at time $t = 0$, for both instances, only the first two jobs have been released.

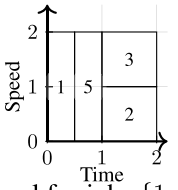
Based on these two instances, we argue that for any scheduling decision made by an online algorithm at $t = 0$, we can choose an instance such that the resulting schedule cannot feasibly follow the speed profile.

Figure 10. Example Instances for the Proof That for DSL Instances No Online Scheduling Rule Can Reliably Find a Feasible Schedule Based on a Speed Profile

j	r_j	d_j	p_j	ℓ_j
1	0	1	1	2
2	0	2	1	1
3	1	2	1	1
4	1	2	1	1
5	$\frac{1}{2}$	1	1	2



Optimal for jobs $\{1,2,3,4\}$



Optimal for jobs $\{1,2,3,5\}$

First, consider the case where at $t = 0$ the algorithm decides to only run Job 1. In that case, we reveal instance I. Limited by its speed limit, Job 2 cannot finish its processing before time $t = 1$ when Jobs 3 and 4 are released. Next, consider the case where on time interval $[0,1/2]$ Job 1 and Job 2 are processed at strictly positive speed for an $\epsilon > 0$ amount of time. If we now reveal instance I', it is not possible anymore to follow the speed profile while completing both Jobs 1 and 5 before their respective deadlines. This illustrates that no deterministic online scheduling rule can reliably follow a given speed profile, even if there does exist a feasible schedule. □

In particular, the result implies that there exists no one consistent learning augmented online scheduling algorithm for DSL that relies on predictions of the aggregated speed profile. Even if given the optimal speed profile, the proof above indicates that there exists no deterministic scheduling rule that can reliably find an optimal schedule.

5. Numerical Experiments

In this section, we relate the theory developed in the previous sections to the EV scheduling application. To this end, we evaluate the discussed algorithms based on numerical experiments with real-world data.[1] In particular, we numerically evaluate the computational complexity of the offline algorithm in Section 5.1 and compare the competitive ratios of the discussed online algorithms with simulation results based on real-world data in Section 5.2.

The real-world data underlying the experiments was collected at an office parking lot in Utrecht, the Netherlands, between September 1, 2022, and June 16, 2023, resulting in a total of 13,694 charging sessions.[2] A maximum of 113 charging sessions was recorded in a single day. Each recorded EV charging session is described by the EV's arrival time, departure time, and the total amount of energy charged. The charging stations at the parking lot are two-plug installations that support charging with at most 11 kW or 22 kW, depending on whether one or two EVs are plugged into the same charging station. In the experiments, we chose between these two values for the individual EV-specific maximum charging rates, depending on the average power required to charge the recorded amount of energy within the EV's availability in the parking lot. Currently, there are around 250 chargers installed in the office parking lot, and this number is expected to increase to more than 400 in the next few years.

5.1. Computational Efficiency of Offline Algorithm

In this section, we numerically evaluate the computational complexity of the offline algorithm FOCS. Research

has shown that for some algorithms theoretical and empirical running times may differ. For example, the simplex algorithm, although theoretically exponential in running time, performs well on real-world instances. The other way around, ellipsoid methods for linear programs are known to be polynomial on paper, whereas performing poorly in practice. This motivates us to investigate the empirical running time of the novel algorithm FOCS based on real-world EV charging data. We solve instances based on real-world data using a proof-of-concept implementation of FOCS and compare it to results achieved with the commercial solver Gurobi (Gurobi Optimization LLC 2023).

5.1.1. Software Implementation. The proof-of-concept FOCS implementation used for the empirical running time analysis of FOCS has been developed specifically as a proof of concept for this paper. The code is written in python, heavily relying on the `networkx` package (Hagberg et al. 2008). This package is chosen for its user-friendly flow models and because various maximum flow algorithms are readily available within the package. An overview of the considered algorithms is provided in Table 1. The last column of the table specifies the computational complexity of the respective algorithms when substituting the network size of the FOCS network ($\leq 3n+1$ nodes and $\leq 2n^2+2n-1$ edges; see Section 3.3.1) into the complexity as reported by Networkx[3] (third column in Table 1). We find that `shortest_augmenting_path()`, `edmonds_karp()`, `preflow_push()`, and `dinitz()` have a respective theoretical complexity of $\mathcal{O}(n^4)$, $\mathcal{O}(n^5)$, $\mathcal{O}(n^3)$, and $\mathcal{O}(n^4)$ on FOCS networks. Note that the implementations available in `networkx` likely do not strictly follow the original name-sake algorithms (Ford and Fulkerson 1956, Dinitz 1970, Edmonds and Karp 1972, Karzanov 1974/Goldberg and Tarjan 1988, respectively). Since their publication, various improvements and variants have been introduced that are usually referred to by the same name.

Based on the problem constraints (5) and the quadratic objective function (2) with $\alpha = 2$, it is also possible to use the commercial solver Gurobi (Gurobi Optimization LLC 2023) to solve DSL instances. In this paper, we use Gurobi for comparison of FOCS with state-of-the art solvers.

5.1.2. Experimental Setup. In this section, we describe the experimental setup used to numerically evaluate the computational efficiency of the offline algorithm FOCS compared with the benchmark results by the commercial solver Gurobi.

The main parameters to classify the experiments are as follows:

- Instance size n, that is, the number of EV charging sessions
- Time granularity, that is, either 1 minute, 15 minutes, or 1 hour
- Used maximum flow method in FOCS

For a given set of parameters for the experiment, we randomly sample n sessions from the data set described in Section 5 and solve the instance using FOCS and Gurobi. We record their CPU running times using the function `time.process_time()` from the python package `time`. To get meaningful results, we repeat this process 500 times, starting at the instance sampling. The values reported in this work are the median running times over those 500 runs. All experiments are run on an Intel Xeon E5-2630 v3 processor.

5.1.3. Results. In this section, we present and analyze the results of the running time experiments. In particular, we are interested in the efficiency and therefore usability of the offline algorithm FOCS for (large) EV parking lots. To this end, we focus on the dependency of the running time on the instance size n. We discuss the impact that different maximum flow algorithms have on the performance of FOCS and compare the impact of various time granularities under invariant maximum flow methods.

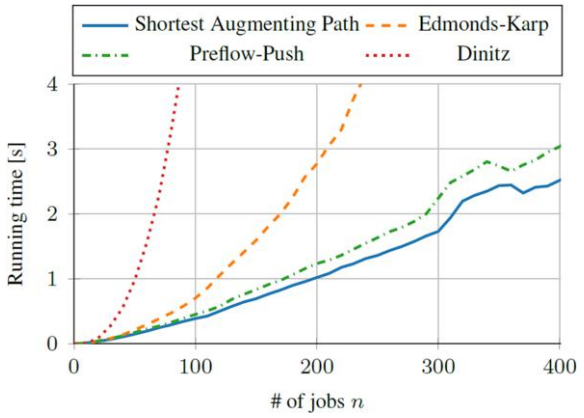
As discussed in Lemma 2, FOCS derives a solution in $\mathcal{O}(n^2\mu)$ time, where $\mathcal{O}(\mu)$ is the time complexity of the used maximum flow algorithm. Figure 11 presents the running time of FOCS for a full day and quarterly time granularity for four maximum flow methods. We see a clear difference in performance between methods that rely on augmenting paths (`shortest_augmenting_path()`, `edmonds_karp()`, and `dinitz()`).

Table 1. Complexity Table for Four Maximum Flow Algorithms

Maximum flow algorithm	Theoretical complexity	Networkx complexity	FOCS network
`shortest_augmenting_path`	$\mathcal{O}(NMU)$	$\mathcal{O}(N^2M)$	$\mathcal{O}(n^4)$
`edmonds_karp`	$\mathcal{O}(M^2N)$ or $\mathcal{O}(M^2\log U)$	$\mathcal{O}(M^2N)$	$\mathcal{O}(n^5)$
`preflow_push`	$\mathcal{O}(M\mathrm{N}\log\left(\frac{N^2}{M}\right))$	$\mathcal{O}(N^2\sqrt{M})$	$\mathcal{O}(n^3)$
`dinitz`	$\mathcal{O}(N^2M)$ or $\mathcal{O}(M\mathrm{N}\log U)$	$\mathcal{O}(N^2M)$	$\mathcal{O}(n^4)$

Notes. Here, N, M, and U are, respectively, the number of nodes, number of edges, and maximum capacity in a given flow network. The theoretical complexity is taken from Cruz-Mejía and Letchford (2023).

Figure 11. (Color online) Running Times for Solving Only for Quarterly Granularity, Comparing Maximum Flow Algorithms Applied in FOCS



Although the time taken to solve FOCS using `shortest_augmenting_path()` seems to grow almost linearly for instance sizes up to 400, running times with `edmonds_karp()` and `dinitz()` clearly increase nonlinearly. Finally, the preflow-push method behaves similarly to `shortest_augmenting_path()`.

If we compare the above to the theoretical complexity of the various maximum flow methods embedded in `networkx` (Table 1), the most striking observation is that, although `preflow_push()` has the smallest theoretical running time on FOCS networks ($\mathcal{O}(n^3)$), the theoretically quartic running time of `shortest_augmenting_path()` ($\mathcal{O}(n^4)$) outperforms the other tested methods. `Edmonds_karp()` shows the highest theoretical complexity and empirically is the second slowest method, overtaken only by `dinitz()`.

Overall, `shortest_augmenting_path()` appears to be the dominating method in our experiments with respect to running time. Therefore, for the remainder of the section, we focus on results generated with `shortest_augmenting_path()` as a subroutine in FOCS. The experiments further indicate that the proper choice of maximum flow method greatly influences the usability of FOCS for EV scheduling applications.

In the following, we compare FOCS to the Gurobi implementation. The total running time relative to instance size n for the three time granularities is depicted in Figure 12. First, we note that across subfigures, for small instances, the quadratic program solved in Gurobi either outperforms FOCS or runs at similar speed. However, for Gurobi, we observe a gradually steeper increase in running time relative to instance size than for FOCS. As a result, the median running time of FOCS outperforms Gurobi starting from instance sizes of approximately 90, 120, and 80 for time granularities of 1 minute, 15 minutes, and 1 hour, respectively. Furthermore, we observe a sudden steep increase in the running time of the Gurobi implementation at instance sizes from around 90, 130, and 230 in Figure 12, (a), (b), and (c), respectively. This behavior possibly reflects that a certain memory threshold is reached after which solving becomes more costly for Gurobi.

Overall, as expected, a finer time granularity for a given instance size results in a larger running time for both solvers. Notably, the slope of the running time for FOCS seems to be almost linear for the instance sizes considered, except in Figure 12(a). There, a nonlinear increase can be observed. To give an indication of this growth, the median running times for FOCS and Gurobi with instance size 400 under time granularity of one minute are, respectively, 36 and 62 seconds.

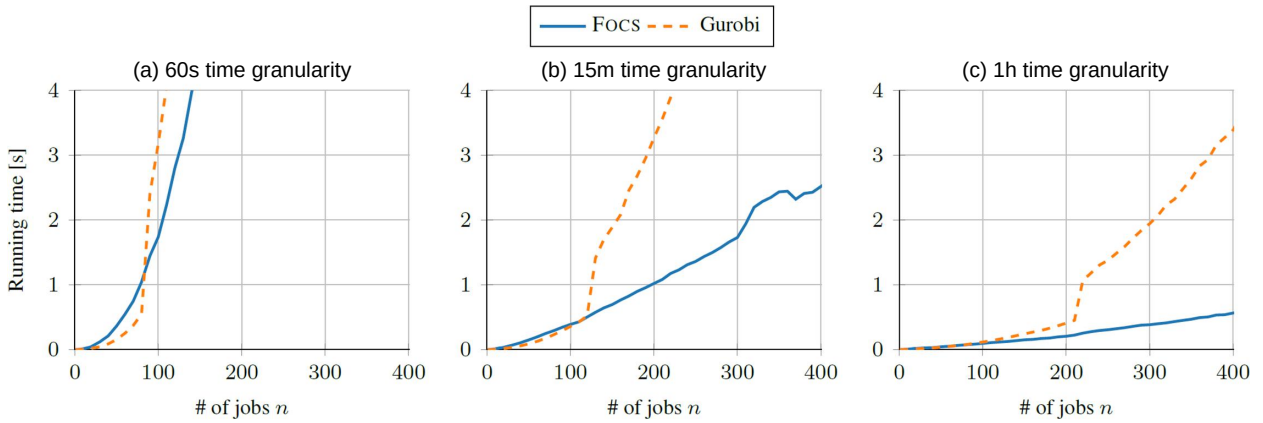
We conclude that even for a proof-of-concept implementation of FOCS, in terms of running time, the method is competitive with existing commercial solvers and therefore promising for application in the field. In practice, schedules are often made in 15-minute granularity, for which FOCS computed an optimal schedule in 2.5 seconds (median) for instance sizes up to 400. Based on the evaluation of the impact of the chosen maximum flow algorithm, we advise careful consideration of the applied maximum flow algorithm when implementing FOCS for practical applications.

5.2. Competitiveness of Online Algorithms

As theoretical competitive ratios may be based on worst case instances that may be very unrealistic to occur in practice, we compare the competitive ratios for the online algorithms AVR and OA presented in Section 4 with simulation results based on real-world data. We define the *empirical ratios* observed in the simulations in accordance with the definition of the competitive ratio in (12) to be $E(s^{\mathrm{ALG}}(I))/E(s^*(I))$, where I is the considered instance, the numerator is the objective value of the considered (online) algorithm, and the denominator the objective value of the optimal solution.

For the numerical experiments, we randomly sampled 400 charging sessions and combined them into one instance. We solve this instance using FOCS, AVR, and OA. Because FOCS is an offline solver that results in an optimal solution (see Section 3), we use its objective value for the calculation of the empirical ratios for online algorithms AVR and OA. Note that this means that FOCS derives a schedule using perfect knowledge on all jobs, whereas OA and AVR only become aware of jobs as they are released. We consider Objective Function (2) with $\alpha = 2$ and repeat the sampling and solving process 500 times, each time recording the objective value and power profile for the three algorithms. For comparison, we also evaluate a greedy algorithm that schedules each job at its maximum speed upon release until its processing requirement is met. In practice, that corresponds to EVs arriving and charging uncontrolled at their maximum charging power until their charging

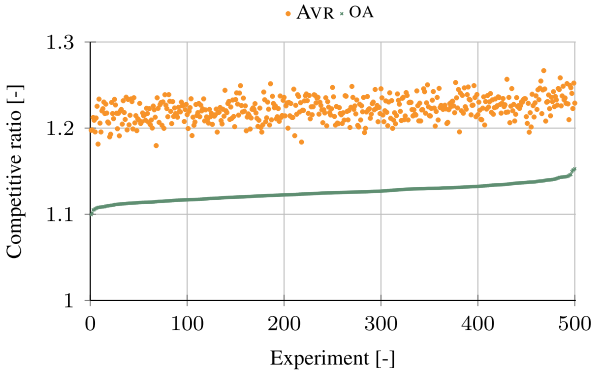
Figure 12. (Color online) Running Times of FOCS (Using `shortest_augmenting_path()`) and Gurobi Relative to Instance Size for Various Time Granularities



demand is met, which is the default if no scheduling is applied.

The results are summarized in Figures 13 and 14. First, Figure 13 shows the empirical ratios for AVR (circles) and OA (squares) for 500 randomly sampled instances. The instances have been sorted based on the empirical ratio for OA, resulting in the squared dots forming an increasing sequence. Notably, this ordering has not translated to the empirical ratios of AVR, meaning that the ordering of two DSL instances based on the objective value for OA in general does not say anything about their objective values for AVR. For AVR, the minimum and maximum empirical ratios recorded in the experiments were 1.18 and 1.27, respectively, as opposed to the theoretical bound $2^{\alpha-1}\alpha^{\alpha} = 8$. For OA, the minimum and maximum empirical ratios were 1.10 and 1.15, respectively, as opposed to the theoretically tight competitive ratio $\alpha^{\alpha} = 4$. Note that the maximum empirical ratio for OA is smaller than the minimum for AVR. That implies that in terms of objective value, OA dominates AVR. For comparison, the empirical ratio for greedy assumes values between 1.68 and 2.06.

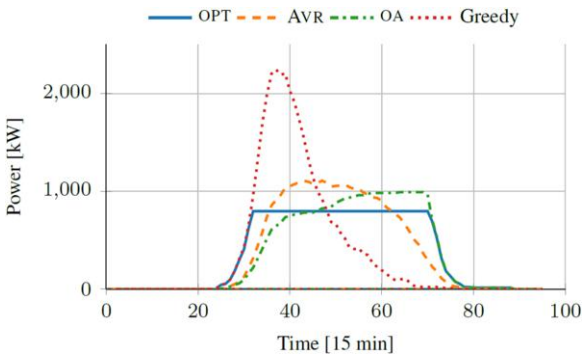
Figure 13. (Color online) Empirical Ratios for AVR and OA for 500 Instances of Size 400



Notes. Their respective theoretical bounds are eight and four. Results sorted in increasing order based on their empirical ratios for OA.

For the first of the 500 sampled experiments, Figure 14 shows the four power profiles resulting from FOCS (solid), AVR (dashed), OA (dashdotted), and greedy (dotted). The speed profile of AVR is more impacted by the high simultaneity of arrivals and great variance of departure times in this particular office building (Winschermann et al. 2023b). The effect is graphically reflected by the slightly left-leaning form of the curve. Its relative smoothness can be attributed to the long dwell times and simultaneity in office parking lots. OA, on the other hand, shifts a lot of the work to the end of the time horizon, as it is oblivious to each subsequent new arrival. From a user perspective, having a gradual charging process over the day invokes less anxiety and mistrust as opposed to charging later in the day. Furthermore, the AVR solution is more robust to early departure by individual EVs. This last observation falls outside of the DSL problem statement but becomes relevant when considering for example, charging guarantees as deterministic input to the optimization. We note that from a parking lot perspective, both AVR and OA clearly outperform the uncontrolled greedy approach. From a parking lot–level perspective, a reduction of the power peak to about half the original peak is highly desirable.

Figure 14. (Color online) Aggregated Speed Profiles for OPT, OA, and AVR for One Sampled Instance Based on Real-World Data



6. Conclusion

In this work, we consider an EV scheduling problem with an objective to minimize the integral of a strictly convex and increasing function of the aggregated power profile. In particular, we relate EV scheduling to speed scaling with job-specific speed limits and present FOCS, an exact offline algorithm that determines an optimal schedule in $\mathcal{O}(n^2\mu)$ time, where $\mathcal{O}(\mu)$ is the complexity of the used maximum flow algorithm. Numerical experiments based on real-world data show that a proof-of-concept implementation of FOCS has a median running time of 2.5 seconds to solve instances with 400 EVs over a full day in 15-minute granularity. Given that, within the energy domain, it is common to consider 15-minute planning intervals and that the efficiency of the software implementation used for our experiments can still be improved, this indicates usability of FOCS as optimization subroutine in the field.

Next to the offline algorithm, we analyze two online algorithms and their respective competitive ratios for a class of objective functions depending on a parameter α, where in energy applications α typically equals two. AVR is shown to be $2^{\alpha-1}\alpha^\alpha$ competitive, and OA has a tight competitive ratio α^α. These competitive ratios match those for the classical speed scaling model that does not consider job-specific speed limits and only processes one job at a time. In our experiments with real-world EV charging data, both AVR and OA approximate the optimal offline solution with a factor of less than 1.3. This performance is significantly better than the uncontrolled default often used in practice where EVs charge at their maximum power starting from their arrival until their energy charging demand is met.

Future work may investigate additional problem constraints such as global power limits. Furthermore, numerical experiments are of interest, especially their integration with control strategies such as model predictive control or fill-level algorithms. Lastly, given that optimal schedules are not necessarily unique, scheduling rules resulting in a robust output should be explored.

Endnotes

[1] The code used for the simulations is available under https://github.com/lwinschermann/FlowbasedOfflineChargingScheduler (commit 7506297).

[2] A more extensive version of the data set is available in de Bont et al. (2024).

[3] The website was last accessed on December 19, 2023.

References

Al-Alwash HM, Borcoci E, Vochin M-C, Balapuwaduge IAM, Li FY (2024) Optimization schedule schemes for charging electric vehicles: Overview, challenges, and solutions. *IEEE Access* 12:32801–32818.

Antoniadis A, Kling P, Ott S, Riechers S (2017) Continuous speed scaling with variability: A simple and direct approach. *Theoretical Comput. Sci.* 678:1–13.

Bansal N, Kimbrel T, Pruhs K (2007) Speed scaling to manage energy and temperature. *J. ACM* 54(1):1–39.

Bansal N, Kimbrel T, Pruhs K, Stein C (2010) Speed scaling for weighted flow time. *SIAM J. Comput.* 39(4):1294–1308.

Boyd S, Vandenberghe L (2004) *Convex Optimization* (Cambridge University Press, Cambridge, UK).

Cruz-Mejía O, Letchford AN (2023) A survey on exact algorithms for the maximum flow and minimum-cost flow problems. *Networks* 82(2):167–176.

de Bont K, Hoogsteen G, Hurink J, Kokhuis R, Kusche F, Nijenhuis B, Ozceylan B, et al. (2024) Electric vehicle charging session data of large office parking lot (version1). [dataset]. *4TU.ResearchData*. Accessed December 10, 2024, https://data.4tu.nl/datasets/80ef3824-3f5d-4e45-8794-3b8791efbd13/1.

Dinitz Y (1970) Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Math. Dokl.* 11:1277–1280.

Edmonds J, Karp RM (1972) Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM* 19(2):248–264.

Eising JW, van Onna T, Alkemade F (2014) Towards smart grids: Identifying the risks that arise from the integration of energy and transport supply chains. *Appl. Energy* 123:448–455.

Elghanam E, Abdelfatah A, Hassan MS, Osman AH (2024) Optimization techniques in electric vehicle charging scheduling, routing and spatio-temporal demand coordination: A systematic review. *IEEE Open J. Vehicular Tech.* 5:1294–1313.

Ford LR, Fulkerson DR (1956) Maximal flow through a network. *Canadian J. Math.* 8:399–404.

Goldberg AV, Tarjan RE (1988) A new approach to the maximum-flow problem. *J. ACM* 35(4):921–940.

Gurobi Optimization LLC (2023) Gurobi Optimizer reference manual. Accessed December 19, 2023, https://www.gurobi.com.

Hagberg AA, Schult DA, Swart PJ (2008) Exploring network structure, dynamics and function using Networkx. Varoquaux G, Vaught T, Millman J, eds. *Proc. 7th Python Sci. Conf.*, 11–15.

Ireshika MAST, Kepplinger P (2024) Uncertainties in model predictive control for decentralized autonomous demand side management of electric vehicles. *J. Energy Storage* 83:110194.

Jensen JLWV (1906) Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Math.* 30:175–193.

Karzanov A (1974) Determining the maximal flow in a network by the method of preflows. *Dokl. Math.* 15:434–437.

Schoot Uiterkamp MHH, Gerards MET, Hurink JL (2018) Fill-level prediction in online valley-filling algorithms for electric vehicle charging. *Proc. IEEE PES Innovative Smart Grid Technol. Conf. Eur. (ISGT-Eur.)* (IEEE, Piscataway, NJ), 1–6.

Shioura A, Shakhlevich NV, Strusevich VA (2017) Machine speed scaling by adapting methods for convex optimization with submodular constraints. *INFORMS J. Comput.* 29:724–736.

Singh AK, Chaturvedi DK, Kumar M, Kumar R, Naseerudin I (2024) Electric vehicle charging scheduling: A review and optimization framework. *Proc. 1st Internat. Conf. Sustainable Comput. Integrated Comm. Changing Landscape AI*, 1–10.

Turitsyn K, Sinitsyn N, Backhaus S, Chertkov M (2010) Robust broadcast-communication control of electric vehicle charging. *Proc. 1st IEEE Internat. Conf. Smart Grid Comm.* (IEEE, Piscataway, NJ), 203–207.

van Kriekinge G, de Cauwer C, Sapountzoglou N, Coosemans T, Messagie M (2021) Peak shaving and cost minimization using model predictive control for uni- and bi-directional charging of electric vehicles. *Energy Rep.* 7:8760–8771.

Vecchio G, Tricarico L (2019) "May the force move you": Roles and actors of information sharing devices in urban mobility. *Cities* 88:261–268.

Vizing VG, Komzakova NL, Tarchenko AV (1982) An algorithm for selecting the execution intensity of jobs in a schedule. *Cybernetics* 17:646–649.

Winschermann L, Hoogsteen G, Hurink J (2023a) Integrating guarantees and VetoButtons into the charging of electric vehicles at office buildings. *Proc. IEEE PES Innovative Smart Grid Technol. Conf. Eur. (ISGT-Eur.)* (IEEE, Piscataway, NJ), 1–5.

Winschermann L, Bañol Arias N, Hoogsteen G, Hurink J (2023b) Assessing the value of information for electric vehicle charging strategies at office buildings. *Renewable Sustainable Energy Rev.* 185:113600.

Yao F, Demers A, Shenker S (1995) A scheduling model for reduced CPU energy. *Proc. IEEE 36th Ann. Sympos. Foundations Comput. Sci.* (IEEE, Piscataway, NJ), 374–382.

Zhang Z, Li F, Aydin H (2011) Optimal speed scaling algorithms under speed change constraints. *Proc. IEEE Internat. Conf. High Performance Comput. Comm.* (IEEE, Piscataway, NJ), 202–210.

Leoni Winschermann is preparing to defend her PhD in the Mathematics of Operations Research and the Computer Architecture for Embedded Systems research groups at University of Twente. Her work focuses on the intersection between theoretical scheduling and energy applications, in particular, for electric vehicle charging.

Antonios Antoniadis is a member of the Discrete Mathematics and Mathematical Programming group at the University of Twente. His broad research interests lie in the areas of discrete mathematics, combinatorial optimization, theory of algorithms, and complexity theory. His current focus lies in the areas of approximation and online algorithms, energy-efficient algorithms, and scheduling theory.

Marco E. T. Gerards is an associate professor within the Computer Architecture for Embedded Systems group. His research interests are robust decentralized and distributed algorithms and decision making under uncertainty. He applies his developed methodologies in energy management and sustainable computing.

Gerwin Hoogsteen is a tenured assistant professor in the field of smart grids within the Mathematics of Operations Research and Computer Architecture for Embedded Systems chairs. His research interest is in energy management for smart grids, particularly where it concerns multidisciplinary research and cyber-physical systems. Current research directions include artificial intelligence in smart grids, digital twins, distributed coordination, and cyber-security of smart grids.

Johann Hurink is a full professor at the University of Twente, Enschede, Netherlands, and the chair of the Applied Mathematics Department of University of Twente, Netherlands. His current research mainly focuses on optimization and control problems for energy management and smart grids.