Utrecht University
Master's Programme: Computing Science

Scheduling of Flexible EV Charging under Grid Constraints and Uncertainty

Author: Max van Huffelen

Student Number: 1714236

Supervisors:
dr. ir. J.M. van den Akker
R.J.J Brouwer MSc.

Netherlands
November 7th, 2023

Abstract

In this thesis, we study the problem of the real-time scheduling of flexible-rate Electric Vehicle charging under resource constraints and uncertainty. We present a novel Schedule Generation Scheme (SGS) which repeatedly mutates schedules to find potential improvements. This method is evaluated on various network layouts using a discrete-event simulation and compared to other well-known SGSs which have been adapted to this problem context. We show how our SGS can yield schedules with reduced customer delays, even when schedules are updated less frequently than other methods.

Acknowledgments

The creation of this thesis would not have been possible without the support of the people around me, to whom I owe a heartfelt thanks. First, my daily supervisor, Roel Brouwer, whose guidance has been instrumental in every step of this thesis and without whose wisdom I no doubt would have called it quits long ago.

Secondly, I would like to thank my first supervisor, Marjan van den Akker, whose critical questions have always pushed me to look at this problem from every angle and to not get lost in my own way of seeing things.

Finally, on a more personal note, this thesis would not have been possible without the warm support and care of the people around me; my amazing girlfriend, friends, and family. Thank you for all the many stimulating conversations, the interest you have shown in my thesis, the numerous hugs and the countless hours we have enjoyed wasting together. I could not have reached this point without you.

1 Introduction

As a part of the ongoing energy transition, Electric Vehicles (EVs) have steadily become more widely used. However, as the usage of EVs increases, so too does the demand for charging these vehicles. This is often done at car parks which have parking places for a number of EVs to park and charge. These car parks are connected to the local power grid from where they obtain their power. They may additionally be equipped with renewable energy sources such as solar panels. Still, if we try to charge a large number of cars at high rates simultaneously, these measures may not be able to supply enough power to the parking place without overloading one or more segments of the cable grid supplying the car park. As such, it is paramount to properly schedule the charging of each car. If the charging is scheduled well, this allows the customers to use their EVs when they want, thus further supporting the energy transition. On the other hand, if this scheduling is done poorly either customers suffer delays and may opt to use less renewable methods of transportation or we overload the energy network and potentially cause damages. As such, it is important to use the limited available resources at charging parks efficiently in order to maximize usage of (clean) energy to optimize customer satisfaction by means of minimizing delays. This can be seen as a variation of the Job Scheduling problem where schedules need to be created and updated under resource constraints and uncertainty in realtime. While a great body of literature has been published on the topic of Job Scheduling and its variants, to the author's knowledge, no research has been published on the realtime scheduling of charging under uncertainty and resource constraints as described in Section 3.

This thesis will contribute to this by presenting and evaluating two novel variations of classic scheduling methods and a novel method of improving such schedules significantly, all of which may be employed at EV charging stations. By finding strategies that construct schedules which minimize charging delays more EVs can be supported and used by customers. For this purpose, the following research question has been formulated which we aim to answer in this thesis:

> *Which scheduling strategies can produce effective schedules for EV charging with the least delay suffered by customers when considering grid constraints, flexible charging rates and uncertain future demand and renewable energy yield?*

To answer this question, we will need to find which methods can construct schedules and incorporate new car arrivals in a way that yields the lowest average delay. Furthermore, we also investigate the effect of decreasing the frequency at which schedules are updated to permit usage of methods with greater runtime. To do this, we will investigate a number of things:

- The efficacy of scheduling using single-pass schedule generation schemes with priority rules
- How well these schedules can be improved by creating mutations of them in a similar fashion to, for example, local search methods.
- The impact of updating the schedule less often.

To do this, we will first take an in-depth look at the literature related to this problem and the state-of-the-art solution methods in Section 2. Then, Section 3 details the exact problem which will be investigated in this thesis, after which Section 4 will describe our approach in detail. The results of this will be presented and analyzed in Section 5. Conclusions are finally drawn in Section 6, with recommendations for future work presented in Section 7.

2 Literature Study

The problem of EV vehicle charging scheduling can be modeled as a Job Scheduling problem. While this problem context has, to the author's knowledge, not been investigated previously, a broad range of other approaches and other problem settings have been. A selection of these will be described in this section. Firstly, a broader context for our problem will be given in Section 2.1 by describing some similar problems that have been approached previously. Then, we will continue to describe a number of solution methods that have been used for similar problems in Section 2.2. Section 2.2.1 details some of the numerous methods using priority rules and schedule generation schemes. Various online and packing algorithms that have been used are introduced in Section 2.2.2. Solution approaches using mixed-integer programming and other exact approaches are described in Section 2.2.3. Finally, Section 2.2.4 describes the solution approaches that use genetic algorithms. We round of this section with a small conclusion on our findings in Section 2.3.

2.1 Problem Context

In this thesis, we will search for a method to schedule the charging of EVs in a manner that services as many customers as possible with minimal delays. While this problem is complex and, to the authors best knowledge,

not previously studied, it does share similarities with a number of other problems that have been studied in the past. These problems include Resource-Constrained Project Scheduling, Bin Packing, (Network Constrained) Unit Commitment and Economic Dispatch.

The Resource-Constrained Project Scheduling Problem (RCPSP) is a problem first introduced in the work by Pritsker et al. [1969]. Here, a series of jobs that each require a certain amount of (renewable) resources must be non-preemptively scheduled. To do this, the limited amount of these resources must be respected. Since its introduction, many variations of RCPSP have been proposed and studied. Some of the different variations and extensions of RCPSP which have been studied include [Hartmann and Briskorn, 2010, 2022]:

- The inclusion of multi-modal jobs, where a job can be executed in one of multiple manners requiring different resource profiles, or even freely chosen resource consumption profiles. [Wall, 1996, Kis, 2006, Okubo et al., 2015]
- Modeling the time and/or processing rates as being continuous rather than being limited to integer amounts. [Koné et al., 2011, Schutt et al., 2013, Artigues and Lopez, 2014, Nattaf et al., 2019]
- The usage of nonrenewable, partially renewable or cumulative resources. Cumulative resources are those which can be consumed or stored like energy in a battery. [Kis, 2006, Okubo et al., 2015]
- RCPSP where the jobs have resource profiles that vary over time instead of having a constant resource usage. [Wall, 1996]
- The usage of different or additional temporal restrictions. While standard RCPSP may have precedence constraints, it is also possible to include, for example, release dates and deadlines or due dates on jobs. [Özdamar et al., 1998, Kolisch and Hartmann, 1999, Neumann and Zimmermann, 2000, Ballestín et al., 2006, Nakahira et al., 2017]
- Optimizing RCPSP for different objectives. While the traditional objective is to minimize the makespan of all the jobs, other possible objectives are to optimize the Net Present Value, keep a constant resource usage over time or to minimize the tardiness if the jobs have due dates. [Neumann and Zimmermann, 2000]
- Processing rates of jobs can be updated throughout their execution. [Nakahira et al., 2017]

Naturally, many of these extensions can be and have been combined together. Many different heuristics have been proposed for this problem, a number of which we will see in Section 2.2. The problem we will study shares many similarities with an RCPSP problem where the capacity of each cable segment is a limited resource, jobs can process at continuous rates, and the future demand and - to lesser extent[1] - availability of resources is uncertain. As each job has a due date, our objective is then to minimize total or average tardiness.

According to the literature overview by Pellerin et al. [2020], heuristic solution approaches can typically be subdivided into three classes; single-pass heuristics, multi-pass heuristics and metaheuristics. Single- and multi-pass heuristics use one or more priority rules in combination with a schedule generation scheme once or multiple times to construct a schedule whereas metaheuristics include methods such as tabu search, simulated annealing or genetic algorithms. For RCPSP, a standard method of creating test instances has been developed and described in the work by Kolisch et al. [1995].

The Rectangle Packing problem is one where we consider a bin of unit width and infinite height. Given a number of rectangles of given width and height, we are to pack these into the bin such that the height of the rectangles in the bin is minimized. A scheduling problem with a limited resource could be regarded as a bin packing problem where each rectangle represents a job. However, to the authors knowledge it is not possible to consider a scheduling problem with multiple different limited resources as a (two-dimensional) rectangle packing problem. This scheduling can be done in an offline case, where all jobs are known beforehand, or an online context where jobs are revealed one at a time, such as in the research by Ye and Zhang [2007]

The Unit Commitment problem is a problem primarily considered in electricity markets. As described in the works by Saravanan et al. [2013], Conejo and Baringo [2018], and Knueven et al. [2018], the Unit Commitment problem determines the commitment of power generating units ahead of time under a number of constraints, such as minimum up- and downtime of units and ramping constraints. This problem can be either considered as a deterministic problem [Moghimi Haji and Vahidi, 2012] or as a stochastic problem where future demand and production of renewable energy sources are not certain [Ackooij et al., 2018, Isuru et al., 2020]. If network constraints are also considered, this becomes known as the Network Constrained Unit Commitment problem. Once the commitment of units is decided, Economic Dispatch determines the exact amount to be generated by

[1] After all, we are guaranteed to have power available through the local transformer and only the power which will be yielded by solar panels is uncertain.

each unit to satisfy demand [Conejo and Baringo, 2018]. In practice, Economic Dispatch is commonly included as part of the Unit Commitment problem.

At first this might seem rather different from our problem of scheduling energy *consumption*. However, if we consider each car a generator in the Unit Commitment problem which needs to generate an amount of energy equal to its charging volume before its intended departure time it becomes clear how the two problems are related.

Finally, we can observe that the situation of our problem shares a number of similarities with microgrids, a particular application of scheduling under its own set of constraints. Microgrids are small, (semi-)isolated electrical systems with their own methods of generating, storing and consuming energy. These microgrids are typically connected to a larger grid to help with load balancing. While our problem does not account for the storage of energy, it is nonetheless a fairly similar context otherwise. As such, some of the methods used to manage microgrids may also be applicable to our problem. Examples of studies on microgrids include the works by Wang et al. [2016], Brouwer et al. [2018], Nasir et al. [2021], Gust et al. [2021], and Taghikhani and Zangeneh [2022].

2.2 Solution Approaches

Numerous different approaches to variations of the problem considered in this paper have been proposed previously. This subsection will list a number of useful overviews of this literature before describing several papers in greater detail.

An insight into the numerous variations and approaches of RCPSP can be found in the works by Kolisch and Hartmann [2006], Hartmann and Briskorn [2010], Pellerin et al. [2020], and Hartmann and Briskorn [2022]. An overview of smart charging, smart grids and the strategies applied to it can be found in the works by Wang et al. [2016] and Nasir et al. [2021]. In addition to this, a survey of the strategies for microgrid usage is done by Gust et al. [2021]. An introduction to (Network Constrained) Unit Commitment and Economic Dispatch problems can be found in the literature survey by Conejo and Baringo [2018]. Furthermore, Ackooij et al. [2018] provide a broad view of Uncertain Unit Commitment problem approaches.

2.2.1 Priority Rules and Schedule Generation Schemes

One approach to the scheduling problem is found in using priority rules in combination with schedule generation schemes. Here, the jobs are ranked in order of priority according to some priority rule. Then, the schedule generation scheme constructs a schedule based on the ordering of the jobs. A schedule generation scheme is a method of inserting a new job into the schedule, which is used to insert the jobs one by one to create a complete schedule. Three such schemes, the Flexible Serial, Flexible Parallel, and Rescheduling Schedule Generation Schemes, are detailed in Section 4.2.1. Note that the well-known List Scheduling problem where jobs are listed and scheduled in order of rules such as earliest due date, latest feasible start time or Smith's rule is a special case of using a priority rule and schedule generation scheme to schedule a series of jobs.

A number of papers compare the effectiveness of a variety of priority rules and/or schedule generation schemes under different constraints and with different objectives. The work by Li and Willis [1992] introduces a schedule generation scheme which iterates between using forward and backward scheduling techniques. Using various priority rules, the authors show how this schedule generation scheme can be used to reduce the makespan of a resource-constrained project scheduling problem. Each iteration of the algorithm the makespan (and thus the start or finish time) is reduced to what has been found in the previous iteration. Then, the algorithm is to find a schedule that fits within this makespan - and is hopefully even shorter to keep reducing the makespan. The authors compare this to having a box of wooden blocks of various lengths and sizes. By turning the box over a few times, various blocks can shift into alignment with each other so they will fit neatly.

More research into the usage of priority rules and schedule generation schemes is published by Özdamar et al. [1998]. This paper considers a case of RCPSP where both tardiness and Net Present Value (NPV) need to be optimized under precedence constraints. For this dual objective, hybrid priority rules are proposed in order to enhance both objectives. In effect, these hybrid rules consist of the weighted sums of each rule they are composed of (for simplicity the weights sum to 1). The authors experimentally demonstrate that hybrid rules can outperform traditional priority rules that only consider a single objective.

Nakahira et al. [2017] consider the problem where a number of EVs are to be charged at a single car park before their deadlines with limited power available. While uncertainty of future car arrivals is taken into account, there is no uncertainty of available power and preemption is allowed. For this problem, an online algorithm is presented that smoothly updates the charging rates of each EV throughout time based on its laxity. The laxity of an EV is a measure of how much time it has leftover before its deadline if it continues charging at its current rate.

Extensive research has also been done to experimentally demonstrate and compare scheduling using priority rules and schedule generation schemes to other common scheduling methods, such as genetic algorithms or

simulated annealing. This body of literature includes the works by Kolisch and Hartmann [1999], Hartmann and Kolisch [2000], Neumann and Zimmermann [2000], and Ballestín et al. [2006].

Kolisch and Hartmann [1999] consider RCPSP under general temporal constraints. Here, the researchers use priority rules as well as metaheuristic strategies including genetic algorithms, simulated annealing, tabu search, truncated branch-and-bound and disjunctive arc methods to order the jobs before a schedule generation scheme such as a parallel or serial schemes constructs a schedule. The effectiveness of the various priority rules and metaheuristics at minimizing the project makespan are compared, and it is shown that the metaheuristics tend to outperform the priority rules at this objective.

The research presented by Hartmann and Kolisch [2000] further compares priority rules with metaheuristic strategies. Here, in addition to the serial and parallel schedule generation schemes, X-pass schedule generation schemes are also investigated. X-pass schedule generation schemes include both single-pass schedule generation schemes, such as those investigated by Kolisch and Hartmann [1999] as well as multi-pass schedule generation schemes. Here, the same SGS is used multiple times with a different priority rule each time. There are numerous ways to apply this, some of which, including sampling methods, are investigated in this paper. Instead of assigning each job a priority value and selecting jobs to schedule in order of this value, sampling methods assign each job a selection probability. The order in which jobs are scheduled is random according to this probability. This means the same priority rule and schedule-generating scheme can be used many times to create different schedules, the best of which is then used. However, it is still found that the metaheuristic approaches yield better results, likely since they can 'remember' previously generated schedules whereas multi-pass methods start from scratch each iteration.

In their research, Neumann and Zimmermann [2000] consider RCPSP with minimal and maximum time lags. This is a generalization of the RCPSP with deadlines or release dates as we can simply generate a 'start job' with zero duration and give every other job a (zero or nonzero) minimum and maximum time lag with respect to this 'start job'. Two non-regular objective functions are considered in this paper: the resource leveling and net value problems. For the resource leveling problem, the cost is dependent on either the greatest amount of each resource that is used in any time period, or each greatest change in the amount of each resource used between any two consecutive time periods. The paper presents an algorithm which consists of a number of steps. Firstly, all jobs which have zero total float are scheduled (i.e. those which can only feasibly be scheduled at one point in time). Then, in order of priority, all jobs are scheduled at their earliest starting time, or at the finish time of another job. Once a time-feasible (but not necessarily resource-feasible) schedule is generated, it must be made resource-feasible. This is done by iteratively looking at the first point in time where the schedule is not resource-feasible and shifting jobs back in time to make it feasible. Finally, this feasible schedule is improved by means of tabu search. Since each job has quite a lot of neighbors (each other time it could start), firstly out of these only the most promising are selected to reduce the size of the neighborhood. This method has shown to be able to solve even large problem instances quickly and effectively.

Ballestín et al. [2006] consider RCPSP with due dates or deadlines. In order to find feasible solutions for these problems, a number of priority rules are proposed. These are used in combination with regret-based random sampling. The results of this algorithm are then compared to the results of a genetic algorithm applied to the same problems. From this, it is shown that the priority rule-based algorithm outperforms the genetic algorithm at finding better solutions for the problem with due dates and at finding more feasible solution for the problem with deadlines. This is in contrast to some of the previously mentioned papers which have found metaheuristics to perform better than priority rules and SGSs in other problem contexts. Since our problem incorporates similar aspects as this problem, such as due dates, this makes our outlook on using a similar approach to solve our problem more optimistic.

2.2.2 Online Algorithms and Packing Algorithms

A different variation of the problem is the online variant. Here, jobs arrive one by one and must be irrevocably scheduled before the next job arrives. While the problem studied in this thesis obviously has a measure of flexibility with respect to being able to change schedules later on not present here, this is otherwise nonetheless rather similar.

Ye and Zhang [2007] present a solution to the online parallel job scheduling problem. In this online problem, each job requires usage of a certain number of machines at the same time throughout its whole duration to be processed. This is analogous to different cars arriving that charge at different rates in our problem. However, in this online problem the objective is to minimize the makespan, which is different from our problem. The solution presented in this paper works on the basis of dividing the schedule into "rooms" and "walls". A job that uses many machines at once becomes (part of) a wall, a segment of the schedule that divides between rooms. Rooms are the segments of the schedule that are filled with the jobs that use less machines simultaneously. This way a 7-competitive approach is achieved on this problem.

Ye et al. [2018] consider a problem very similar to the one considered by Ye and Zhang [2007], except here jobs are malleable. This means that when the task is scheduled, we can choose how many machines work continuously on the task until it is finished. The processing time of each job is a function of the number of

machines assigned to it. This problem of malleable task scheduling is first reduced to a rigid task scheduling problem - a problem where the number of machines assigned to a job is fixed. Then, schedules are generated consisting of "shelves", a number of machines working on the same task(s). We aim to complete all jobs within a certain makespan and each time it turns out this is not possible, we multiply our attempted makespan with some constant to increase it. This way, a competitive ratio of 16.74 is achieved.

2.2.3 Mixed Integer Programming and Other Exact Approaches

A variety of mixed-integer programming approaches have been used to find solutions to manage and schedule energy consumption such as EV charging. Taghikhani and Zangeneh [2022] consider a microgrid connected to the main grid with renewable energy sources as well as a diesel generator and a battery. A hybrid mixed-integer program is used in combination with stochastic programming to model the uncertain renewable energy. As such, the profit of the model can be maximized. In the research performed by Isuru et al. [2020], the network-constrained unit commitment problem with wind power uncertainty is considered. The solution consists of a MIP using a Benders decomposition which iterates between the master and subproblems to converge to a solution.

Furthermore, Seddig et al. [2017] consider the charging of a fleet of EVs at a single car park equipped with renewable energy sources. The objective is to use as much renewable energy and draw as little power as possible from the grid to save costs. Various methodologies, including stochastic programming and heuristics, have been used to forecast the yield of the solar panels. These results are then used by a mixed-integer program to schedule the charging. In a Monte Carlo simulation it has been shown that in controlled charging renewable energy can be used very effectively when compared to uncontrolled charging.

Finally, the work by Kis [2006] considers RCPSP with renewable and non-renewable resources and precedence constraints where an activity may only be started after another activity has been completed for a certain percentage. Furthermore, in the considered problem, the rate at which each job is performed at each timestep may be chosen freely. This means that preemption is allowed, and the only upper bound on charging rate is charging the entire charging volume in a single timestep as a higher rate would be pointless. This problem is solved by formulating it as a MIP which is then solved using the cutting plane method.

A different exact solution approach is using branch-and-bound. Branch-and-bound is a method where the original problem 'branches' into different sub-problems. Repeating this for different possibilities in the solution space leads to a branching tree. Using the upper and lower bounds of each branch, we determine which branches are (and are not) worth considering. Klein and Scholl [2000] develop the PROGRESS procedure for solving generalized RCPSP using existing branch-and-bound rules combined with their own improvements and additions. It is shown that PROGRESS quickly yields great results for smaller problem instances, but becomes infeasibly computationally expensive for larger problem sizes.

Another approach is constraint propagation. Th usage of this method has been surveyed by Brucker [2002]. In this method, additional constraints to deduce the solution space are deduced based on what is already known. Then, more variables can be set, and more can be deduced again (this approach is similar to that of solving the popular logic puzzle 'sudoku' [Simonis, 2005]). This approach can be combined with other methods such as branch-and-bound.

Somewhat similarly, Okubo et al. [2015] consider a problem dubbed '$RCPSP/\pi RC$' - the resource-constrained project scheduling problem with partially renewable resources and resource consumption during setup operations. The authors consider multi-modal jobs with precedence relations. To solve this problem, the authors make use of a constraint programming method for which a mask calculation algorithm is developed. This method is compared to an integer programming method which has also been developed by the authors. It is shown through experimental results that the computational programming model is effective, and even outperforms the integer programming model.

2.2.4 Genetic Algorithms

Next, we look at the solutions that have been designed using genetic algorithms. The algorithm by Wall [1996] considers a problem with multiple resource types where jobs may have different execution modes with non-uniform resource usage, may be preempted, and where temporal and precedence constraints may be present for both tasks and resources. Schedules can be optimized for a variety of objectives, including minimization of total tardiness. Note that while the multi-modal non-uniform resource usage profiles would not entirely allow us to model our freely varying rates of charge, it does come closer than most solution methods we have seen so far as we could use a set of modes to model a number of possible charging profiles. The author proceeds to find solutions to the problem by using integer arrays to represent the start time and execution modes of the jobs. A variety of genetic algorithms are applied to these to find suitable schedules. This has yielded good but computationally expensive results, finding solutions within 2% of the published best in 60% of the project scheduling problems it was applied to.

Another approach using genetic algorithms is presented in the paper by Moghimi Haji and Vahidi [2012]. Here, a so-called 'imperialistic competition algorithm' is designed. In this algorithm, solutions are grouped into populations called 'empires'. Iteratively, solutions ('colonies') in an empire move towards the best solution in their respective empires ('imperialists'), and imperialists (try to) steal the colonies of other empires. The authors show this can be used generate schedules for unit commitment problems.

2.3 Conclusion

In this literature section we have seen that the problem we study shares similarities with a number of other previously studied problems, such as RCPSP, Rectangle Packing and Unit Commitment and Economic Dispatch. Out of these, our problem can best be considered as a resource-constrained project scheduling problem. For such problems, a great deal of solutions have been presented. Out of these methods, priority rules in combination with schedule generation schemes seem promising for our problem while having little literature dedicated to it yet. As such, this method will be implemented and evaluated in future sections.

3 Problem Description

We consider a problem where Electric Vehicles (EVs) continuously arrive at a set of car parks where they wish to recharge their batteries by a certain amount of energy. Cars indicate a due date at which they will leave, unless they are not fully charged then, in which case they wait to charge fully before leaving. We are to schedule the charging of the EVs such that the average tardiness - the average time of completion of recharging each car past the customer's indicated preferred departure time - is minimized. A number of constraints must be taken into account for this. These constraints originate from:

- Limited power available at different locations.
- Limited capacity to transfer power between locations through cables.

In more detail; the car parks are connected to the energy supply by a network of cables. Each segment of these cables is rated for a specific power capacity which may not be exceeded. An example of a grid is shown in Figure 1. It is, however, supplemented by a number of solar panels at various car parks which provide additional power throughout the day. As such, the scheduling is constrained by a number of 'resources'; the capacities of each cable segment and the power supplied by solar panels and at the grid source.

We can change the schedule of each EV at any point in time as we see fit, though of course events that have already happened cannot be altered. However, once a car starts charging, it cannot be preempted - it must continuously charge at rates between the car's lower and its upper charging rate limits until it has completed charging.

There are a number of constraints that must be taken into account. Each of the car parks has a limited number of parking places; if an EV arrives at a car park where the maximum number of vehicles is already parked, they must move to another car park. It is assumed that if a car has visited three different filled car parks in the simulation, they will move to park somewhere else, effectively leaving the simulation unserviced.

As such, the charging at each car park is limited by the available power. The power to each car park is supplied from the local transformer through a series of cables. One cable may be used to supply power to multiple car parks simultaneously. Each of these cables have a load limit that may not be exceeded.

To supply additional power, some car parks may be supplied with solar panels. It is assumed that, on average, solar panels generate a certain fraction of their peak power of 200kW. This fraction depends on the time of day. These fractions throughout the day are shown in Table A1 in the appendix. Note that in this thesis fractions representing distributions during summer are used, other fractions representing different seasons can be used. Suggested distributions for winter conditions are included in the aforementioned appendix. The amount generated at any time is assumed to be drawn from a normal distribution with this average and a standard deviation equal to 15% of this value. However, this, in addition to the uncertainty of future car arrivals, introduces uncertainty into the system. Effective scheduling methods will need to be able to handle this. We assume cars arrive according to a Poisson process. The arrival rates at each hour of the day as well as the distributions of charging rates, charging volumes and connection times can be found in Tables A2-A5 in the appendix. Note that when a car arrives, the time it is parked is equal to or greater than the time it takes to charge its charging volume at the car's minimum charging rate so that it can actually reasonably be charged before the due date. The objective of the problem is then to schedule the vehicles such that the total tardiness is minimized.

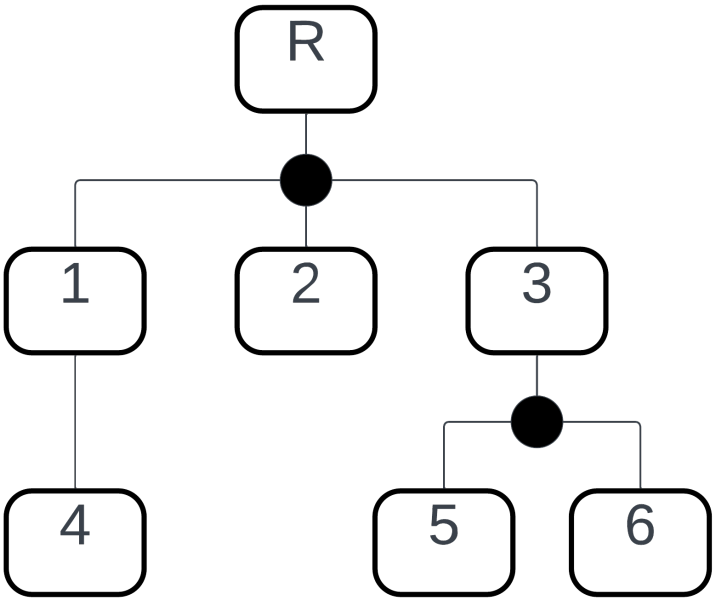


Figure 1: Example of a grid. Here R denotes the power source at the root of the grid, car parks are denoted 1-6 and circles indicate junctions connecting cable segments.

4 Methodology

In order to compare methods of scheduling jobs under the constraints of a situation as described in Section 3, a discrete-event simulation model has been developed to test the efficacy of various approaches. This section will describe the simulation model and how it can be used to test different scheduling approaches in Section 4.1. The various scheduling approaches will be described in greater detail in Section 4.2. Finally, how the various scheduling approaches are evaluated will be described in Section 4.4.

The processor of the system used to run the tests on was an Intel(R) Xeon(R) Gold 6130 CPU with a clock rate of 2.10GHz[2].

4.1 Discrete-Event Simulation Model

In a discrete-event simulation model, the current situation is stored in the state. Changes to this state and how it changes over time are modeled through events. When running the model, the first event is iteratively removed from the schedule, it is calculated how the state has changed since the event that happened before it, changes that happened in that event are applied and new events are scheduled as appropriate. Then, this process is repeated with the next event in the schedule.

The problem context and different scheduling approaches can be modeled in a discrete-event simulation model by storing all parked cars and car parks in the state. Then, events can be used to model the arrival and departure of cars. The various approaches to scheduling the charging of the cars is also modeled through events. We simply schedule events representing a car starting to charge at a certain rate or finishing its charging as dictated by our scheduling method.

The model used to test and evaluate scheduling strategies is represented in Figure 2[3]. Here, immutable events are those of which we cannot choose to change, and mutable events are those we can change when this is beneficial. While unusual for discrete-event simulation models, the ability to change scheduled events is necessary here to be able to update schedules. Note that no information about future events is used when finding schedules.

[2]We would like to thank the research programme "Energie: Systeem Integratie en Big Data" with project number 647.003.005, which is financed by the Dutch Research Council (NWO), for financing the machines used for the computational experiments.

[3]Note that a minor exception to this graph will be made for the Simplified Parallel Schedule Generation Scheme. Here, 'Car stops charging' schedules an 'Update Schedule' event, with all else remaining the same. This will be detailed further detailed in Section 4.2.1

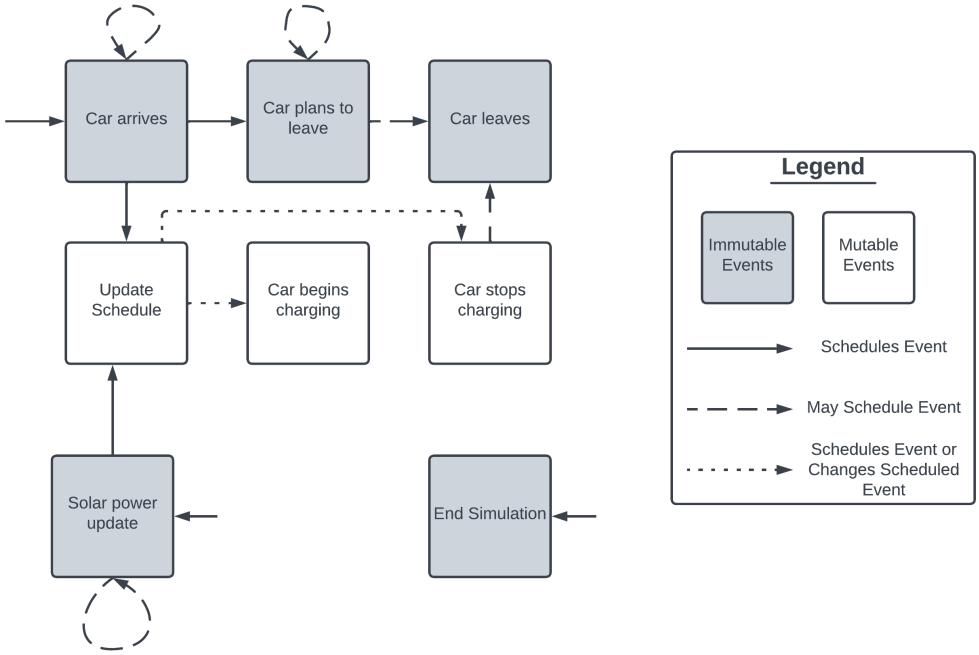


Figure 2: The Event Graph used to model the situation and evaluate scheduling strategies for event-triggered scheduling

To test different scheduling approaches using this model, all one needs to do is use a different method for the 'update schedule' event - the rest can be kept the same. Thus, different methods of scheduling the cars' charging can be fairly compared.

In the event graph shown in Figure 2, every car arrival means the schedule will be updated. While this 'event-triggered scheduling' is flexible with respect to new arrivals, it also requires a lot of computing for new schedules which may be adhered to for only a few minutes until the next car arrives. As such, an alternative is 'interval-based scheduling'. Here, the schedule is only updated when either the solar power is updated or when a car arrives whose priority value is higher than some predefined percentage of cars present the last time the schedule was updated. Both these strategies will be evaluated and compared in the remaining sections of this thesis.

One common caveat of digital simulations are floating point errors. In a simulation model like the one used here, these small deviations can cause large issues. For example, a floating point error erroneously increasing the completion time of a job by a fraction of a second might lead to a very short spike in power consumption in the simulation where this job overlaps with the one after it. This can then cause various issues such as brief cable overloads or making it difficult or even impossible to schedule jobs here without preemption. To prevent jobs from accidentally overlapping, the starting and finishing times of jobs in the schedule are rounded up and down, respectively, to the third decimal. This may cause a very marginal reduction in delays of jobs. As such, the accuracy of reported delays suffered by users can not be guaranteed beyond the first decimal.

4.2 Scheduling Approaches

The family of methods that we will design, implement and test in this thesis are Schedule Generation Schemes (SGSs) in combination with priority rules. Here, the jobs are ranked in order of priority according to some priority rule. Then, the schedule generating scheme constructs a schedule based on the priority assigned to each job. Notably, a special case of this is the well-known List Scheduling problem where a series of jobs must be ordered to be scheduled according to some rule.

Usage of SGSs for job scheduling problems where all jobs are known beforehand and are processed at fixed rates has been studied extensively. Commonly used Schedule Generation Schemes for this purpose include the Serial SGS and Parallel SGS. These schemes first order all jobs in order of their priority. The Serial SGS then repeatedly takes the highest priority unscheduled job and schedules it at the earliest time it can be scheduled. This repeats until all jobs are scheduled. The Parallel SGS instead moves through the time of the schedule. Starting at the earliest point in time and moving forward, at each point in time it is checked if jobs can be scheduled. If this is the case, the highest priority job that can be scheduled here is scheduled. Then, if more jobs can be scheduled at this point in time, the highest priority job is scheduled here again until no more jobs can be scheduled. Then, the scheme repeats this process at the next point in time until all jobs are scheduled. Figure 3 gives an example of the serial and parallel schedule generation schemes applied to a small set of three

jobs.

However, comparatively little research has been done into the usage of Schedule Generation Schemes for dynamically scheduling jobs which can be processed at flexible rates. As such, this thesis presents a number of novel SGSs designed for this problem. These will be described in Section 4.2.1. An overview of the priority rules that will be tested for use by these SGSs is then presented in Section 4.2.2.

4.2.1 Schedule Generation Schemes

An overview of the SGSs that have been designed for and will be evaluated in this thesis is given in here. The first two SGSs are flexible variations of the serial and parallel SGSs, respectively. These are adapted to function well in the dynamic environment where rescheduling and flexible rates of charging are possible. Then, Rescheduling SGSs are introduced which can improve upon the results of Flexible Serial and Parallel SGSs by changing how some jobs are scheduled.

Flexible Serial SGS

The Flexible Serial SGS is quite like the regular Serial SGS. Jobs are still scheduled in order of priority at the first available point in time. However, it is possible that jobs are already being processed when we generate a new schedule. Since these cannot be preempted, they must be processing at the start of the new schedule. As such, initially, all jobs that are already being processed are scheduled to charge continuously at their lowest processing rates. Then, we iterate through all jobs in order of priority. When we consider a job that is not already scheduled, we schedule it at the earliest point in time it can process. At each point in time, the job is scheduled at either its maximum processing rate or the highest rate permitted due to resource constraints. If the job we consider is already scheduled, we instead increase its processing rate from the minimum at each point in time to the maximum. This means the job may be finished earlier, freeing up resource capacities there. If we want to allocate these resources to jobs with the highest priority (instead of the next job that could use it), it would be necessary to remove and reschedule all jobs with higher priority. Since SGSs are heuristics, these are both possible and valid variants of the Flexible Serial SGS. Due to time constraints, this thesis will only consider the variant where all previously-scheduled jobs are kept as-is, with the other variant being left as future work.

(Simplified) Flexible Parallel SGS

Similar to its Serial counterpart, the Flexible Parallel SGS considers the jobs in order of priority. First, already-started jobs are scheduled to charge at their minimum charging rates. Then, the algorithm iterates through each point in time. Here, in order of priority, each job is scheduled to charge at the highest rate possible (which may be zero, if too much available power has been consumed by higher priority jobs).

Due to the dynamic nature of our problem, it is unnecessary to create a full schedule when using a Parallel SGS. Since the schedule can be updated at any point in time, it is only needed to look at which jobs need to process immediately, and at what rates. It is not necessary to consider the scheduling of jobs that start or change processing rate at a later point in time yet. This is because relevant events that change the demand or availability of power such as a car arriving or finishing charging will always trigger the generation of a new schedule. As such, scheduling this far ahead is not necessary as schedules at that point in time will be created at a later point in time regardless.

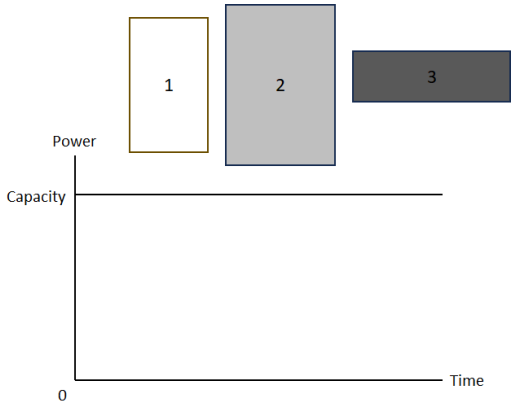
It can easily be seen that, in the Parallel SGS, jobs that are scheduled to start later cannot have an impact on how jobs charge before it. As such, we can make a Simplified Flexible Parallel SGS where we stop the schedule generation after we have considered charging jobs at the current time. This will lead to jobs charging in the same manner, but saves on processing time.

Rescheduling SGSs

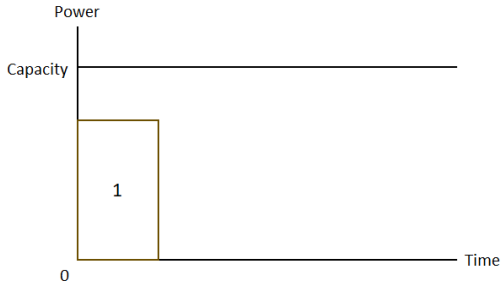
The previously described SGSs are so-called 'single-pass schedule generation schemes'. Here, to create a schedule, the SGS iterates only once over each job in the list of jobs. On the other hand, here we will use schedule generation schemes that will iterate over ordered lists of jobs multiple times. This is done by scheduling and then iteratively removing and re-scheduling jobs multiple times. Note that the rescheduling is done instantaneously after removal of jobs and already-started jobs will always be rescheduled in such a way that no jobs are preempted. By using this approach, mutations of the original schedule are created and improvements can be found in a manner similar to, for example, local search and hillclimber algorithms. The rescheduling SGS implemented in this thesis first creates an initial schedule using either the Flexible Parallel SGS or the Flexible Serial SGS with a priority rule. Once a schedule is created, a number of jobs will be removed. This is done in one of three ways:

1. All jobs are given identical unit weights to be removed. Random jobs are selected and removed until enough jobs are removed.

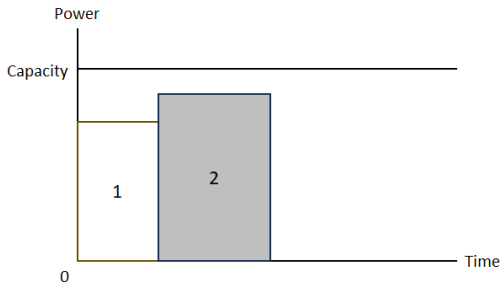
2. A small fraction of jobs are randomly chosen to be removed. All jobs are assigned a weight equal to the number of 'adjacent' jobs in the schedule have been removed. Two jobs are considered adjacent if their scheduled processing times overlap and they are either in the same car park or in different car parks which



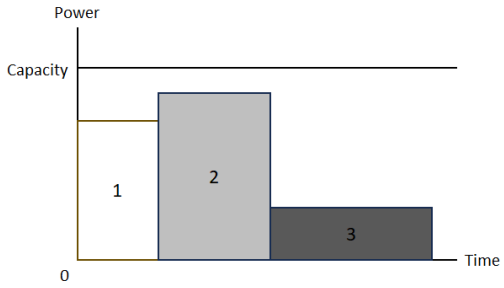
The initial situation of the serial scheduling scheme; no jobs (shown above schedule) are scheduled yet.



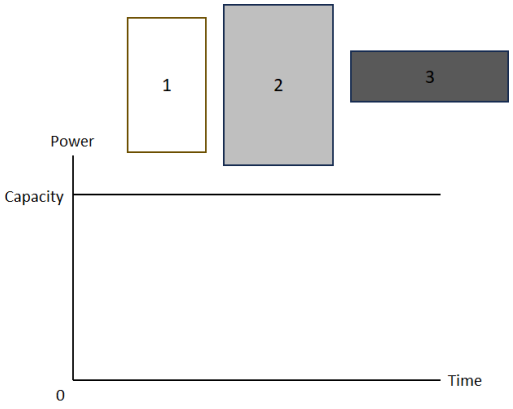
Job 1 is placed at the first possible time.



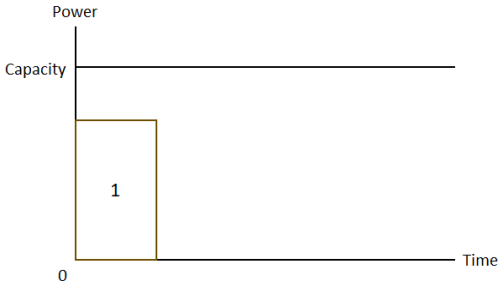
Job 2 is scheduled at the first possible time.



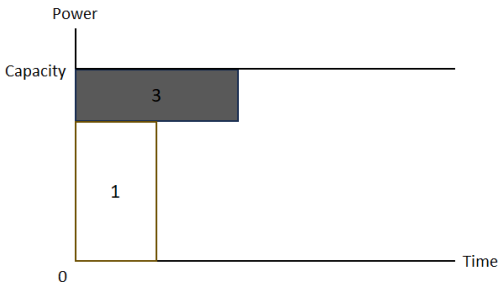
Job 3 is scheduled at the first possible time.



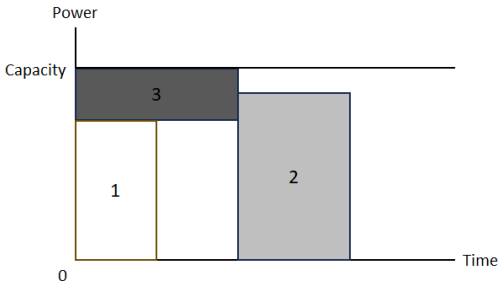
The initial situation of the parallel scheduling scheme; no jobs (shown above schedule) are scheduled yet.



At time 0, the highest priority job that can charge is scheduled.



At time 0, the highest priority job that can charge is again scheduled. This is job 3 as job 2 requires too much power.



The algorithm continues iterating through time until the first point in time where the remaining job can be scheduled, which is then scheduled.

Figure 3: An example of the serial (left) and parallel (right) schedule generation schemes applied to a set of three jobs. Here job 1 is the highest priority and job 3 the lowest.

have been defined as being 'adjacent'[4]. Then, random jobs are removed and the weights are updated until a predefined fraction of all jobs have been removed.

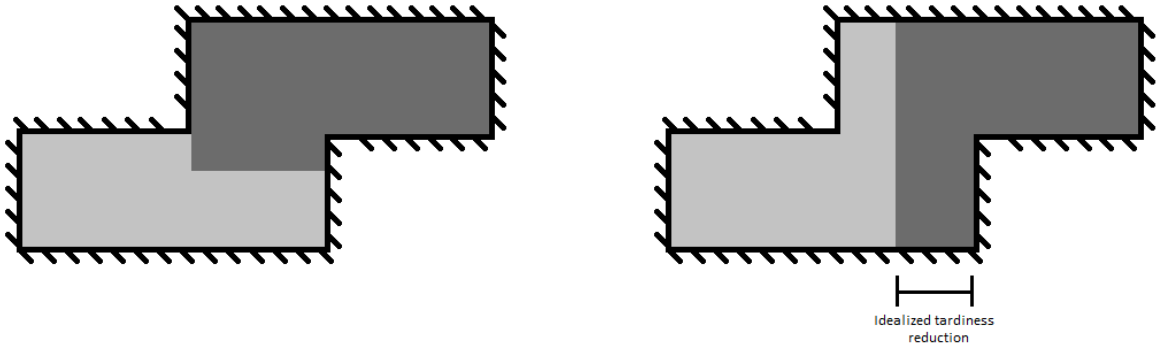
3. Similarly to the previous approach, a small fraction of jobs is initially removed randomly and all remaining jobs are assigned a weight. Then, one by one jobs are removed and the weights are updated until a predefined fraction of all jobs are removed. However, now whenever a job is removed, instead of a unit increase, the weight of each adjacent job is increased by the idealized reduction in delays that could be achieved if only these two jobs were removed. The idealization made is that the two jobs can charge without minimum or maximum charging rate constraints. Then, the jobs can freely be scheduled to use the power consumed by the other job in the original schedule, to potentially reduce delays. This is illustrated in Figure 4. Given two jobs, j_1 and j_2, where j_1 is chosen to denote the job with the earliest due date, this reduction in tardiness can be calculated using:

$$\Delta\Sigma_j T_j = T^*_{j_1} - T_{j_1} + T^*_{j_2} - T_{j_2} = \max(0, C^*_{j_1} - d_{j_1}) - \max(0, C_{j_1} - d_{j_1}) \quad \text{where } d_{j_1} \leq d_{j_2} \tag{1}$$

Where $\Delta\Sigma_j T_j$ denotes the total change in tardiness. d_j, T_j, T^*_j, C_j, and C^*_j denote the due date, tardiness, idealized tardiness when rescheduled with another job, completion time and idealized completion time when rescheduled with another job of job j, respectively. Note that it can trivially be seen that the job with the latest deadline will have the same completion time; $C_{j_2} = C^*_{j_2}$, and thus a zero reduction in tardiness. $C^*_{j_1}$, on the other hand, can be found by solving equation:

$$\int_t^{C^*_{j_1}} c_{j_1,t'} + c_{j_2,t'} dt' = v_{j_1} - w_{j_1,t} \tag{2}$$

Where t denotes the current time in the simulation, $c_{j,t'}$ denotes the charging rate of job j at time t' in the original schedule, v_j denotes the charging volume of job j and $w_{j,t}$ finally denotes the amount already charged by job j at time t. Note that Equation 2 simply ensures that all power originally assigned to the two jobs is used entirely to completely charge the first job until its completion. After this, the rest can be used to charge the second job, which will then logically finish at the same time as in the original, non-idealized schedule.



(a) The initial situation of two jobs in a schedule.

(b) The idealized situation where the jobs are rescheduled to use each other's power without charging rate restrictions. For the sake of example, we assume the first job has a due date before its original completion time.

Figure 4: An example of two jobs in a schedule (left) and the idealized schedule where tardiness is reduced (right)

Once jobs are removed, they must again be reinserted one by one. This is done using the serial SGS with a different priority rule. When the jobs are reinserted in some order, it is hard to guarantee that when we try to reinsert an already-started job we can do so without causing preemption. After all, it is possible that all available power at some point in time has been consumed by other jobs making it impossible for charging to continue beyond that point. Fortunately, we can reduce the probability of such a thing occurring by first scheduling all already-started jobs and only reinserting the other jobs later. After all, this way the jobs that are not already started cannot be scheduled in a way to block the scheduling of an already-started job. If the already-started jobs are reinserted successfully, all jobs can be rescheduled without preemption. Note, however,

[4]This has been done based on which car parks are likely to have more power available if less power would be used in some other car park. For exact details, see Section 4.3.

that reinserting all already-started jobs first in some order is no guarantee of feasible rescheduling, it just decreases the probability of a job being unable to be reinserted into the schedule without preemption to some specific scenarios.

The exact implementation of this consists of two steps:

1. Firstly, in order of priority, we attempt to temporarily schedule all already-started jobs as late and at as low a rate as possible without preempting the jobs. Note that this disregards the due dates of these jobs. By giving these jobs a valid place in the schedule, we ensure that no jobs are preempted. If rescheduling any already-started jobs without preemption is not possible, this mutation is discarded and we return to the previous schedule.

2. Secondly, we iterate through all jobs in order of priority. When we encounter a job that has not already begun charging, we schedule it to finish as soon as possible. If the job has already started, we instead first remove its temporary schedule before scheduling it to finish as soon as possible. We are guaranteed to be able to schedule this job in the final planning as the temporary schedule created earlier functions as a lower bound which we can reuse. In the worst case, we can always use this slowest schedule to continue charging the job without causing preemptions or overloads.

Since our implementation for rescheduling perturbs our schedules, we effectively explore the search space of possible solutions, akin to local search. We keep making mutations of the best schedule found so far and stop iterating after a predefined number of successive mutations fail to improve sufficiently. Other implementations where mutations are made analogous to, for example, methods such as simulated annealing (where we continue mutating on a worse solution with some probability), tabu search (where certain mutations are forbidden) or evolutionary algorithms (where other solutions are combined to create new ones) are left for future research.

4.2.2 Priority Rules

A number of priority rules are used to assign priority values to the different jobs. Then, the jobs are ordered in ascending order such that they can be used by the SGSs. This means that a lower priority value always indicates a higher priority. The used priority rules calculate the priority value of a job based on a number of parameters of the jobs. For some job j, these are its due date d_j, charging volume v_j, minimum charging rate $c_{j,min}$, maximum charging rate $c_{j,max}$, amount charged at the current time in the simulation w_j and the current time of the simulation t. The priority rules and their formulae are:

- First Come First Served (FCFS): j[5].
- Earliest Due Date (EDD): d_j
- Latest Starting Time (LST): $d_j - v_j/c_{j,max}$
- Updated Latest Starting Time (LSTU): $d_j - (v_j - w_j)/c_{j,max}$
- Adapted Latest Starting Time (LSTA):

$$\begin{cases} d_j - (v_j - w_j - (d_j - t) * c_{j,min})/(c_{j,max} - c_{j,min}) & \text{if car has started charging} \\ d_j - v_j/c_{j,max} & \text{otherwise} \end{cases}$$

- Minimum Greatest Slack Time (MinGST): $d_j - t - (v_j - w_j)/c_{j,max}$
- Least Work Remaining (LWKR): $v_j - w_j$
- Most Work Remaining (MWKR): $w_j - v_j$
- Least Flexible Resource Demand (LFRD): $c_{j,max} - c_{j,min}$

Note that the Updated Latest Starting Time priority rule simply calculates the latest time a job could start charging at its maximum charging rate to finish charging its remaining charging volume at its due date, disregarding that this might preempt the charging. The Adapted Latest Starting Time instead returns the standard LST priority value if the job has not started charging yet, and otherwise finds the point in time up to which point the job could charge at its minimum charging rate before it would need to switch to its maximum charging rate to finish on time. Note that, if the job cannot finish on time the equation will return a value before the current time, as the job would have needed to start charging earlier at a higher rate to be able to finish on time.

[5]Jobs are indexed in order of arrival, i.e. the first job to arrive will be indexed 1, the second 2, etc

4.3 Instances

The approaches described in Section 4.2 will be tested on a number of different instances. These instances comprise of combinations of five different grids and a range of different daily car arrivals from 750 to 3000 average daily car arrivals. These grid instances are graphically represented here using rounded squares to denote the transformer (indicated by the letter 'R') and the car parks (indicated using two numbers, the top of which denotes the park's car capacity and the lower of which indicates the fraction of cars which will arrive at this car park). Car parks with solar panels are filled in gray. Cables are drawn as lines between car parks, with their power capacity (in kW) indicated. The instances on which we will test are:

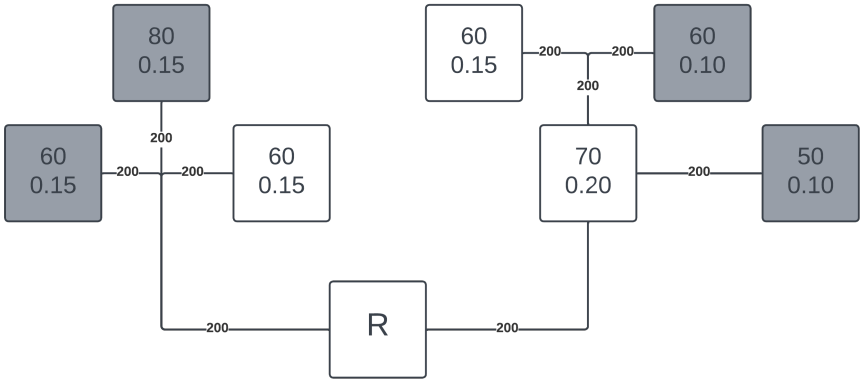


Figure 5: Grid 1: a small tree-like network with uniform cable capacities. The top number in each car park represents its car capacity; the lower the fraction of cars which arrive at this car park.

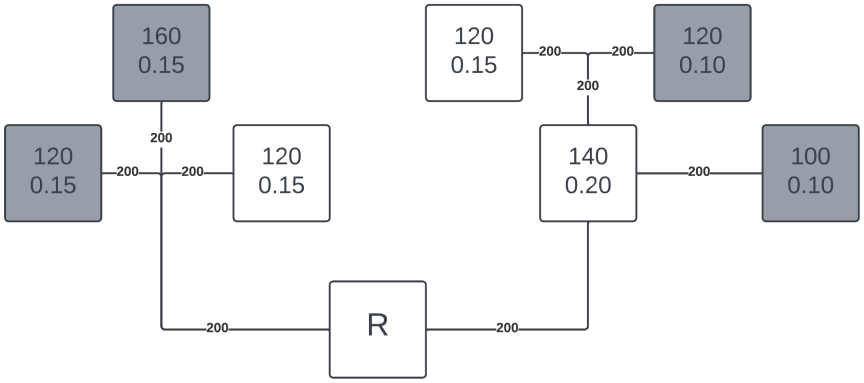


Figure 6: Grid 2: identical to grid 1, except the car park capacities are doubled. The top number in each car park represents its car capacity; the lower the fraction of cars which arrive at this car park.

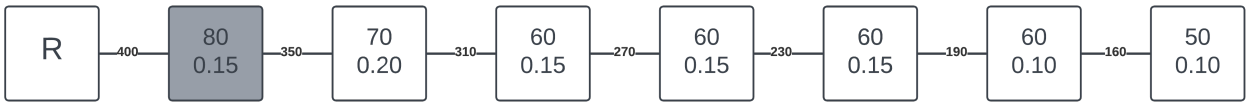


Figure 7: Grid 3: a grid with all car parks arranged in series. The top number in each car park represents its car capacity; the lower the fraction of cars which arrive at this car park.

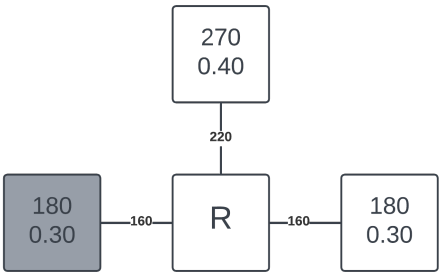


Figure 8: Grid 4: a small grid where all car parks are arranged parallel to each other. The top number in each car park represents its car capacity; the lower the fraction of cars which arrive at this car park.

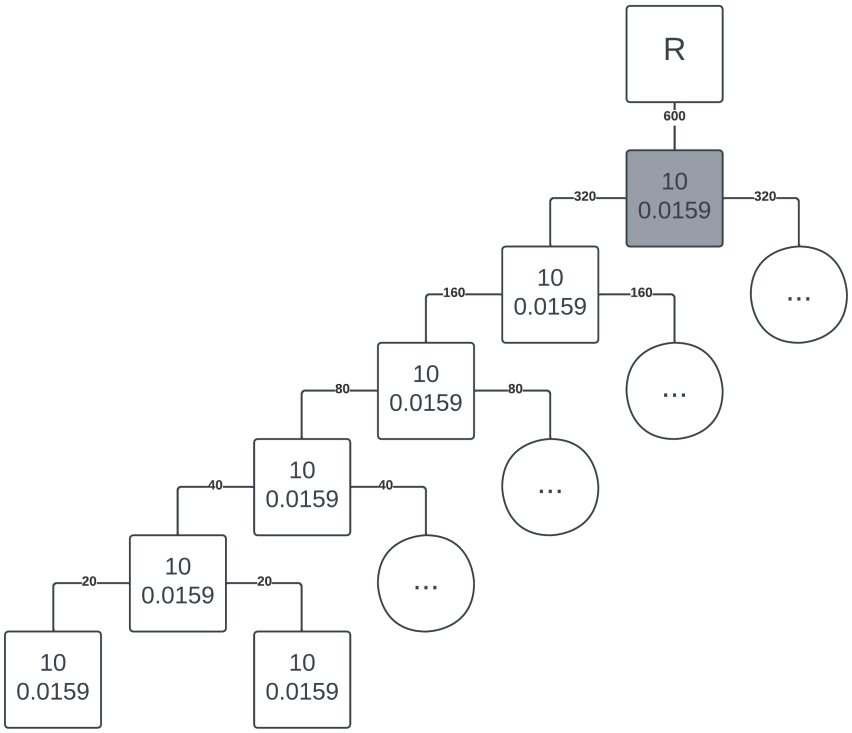


Figure 9: Grid 5: a large, binary tree-like grid. Here, the bubbles with '...' indicate a subtree identical to their sibling. The top number in each car park represents its car capacity; the lower the fraction of cars which arrive at this car park.

In Figure 9, each bubble with '...' indicates a subtree which is identical to their sibling. As such, the network shown here resembles a (symmetrical) binary tree. In Grids 1 and 2, each car park is adjacent to all other other car parks in the same 'halve' of the network, i.e. each car park it is connected to without passing through the grid root. In Grid 3 all car parks are adjacent. In Grid 4, no car parks are adjacent. Finally, in Grid 5, each car park is adjacent to all its ancestors and descendants.

4.4 Evaluation Metrics

The performance metrics we measured for each run of the simulations are:

- Average tardiness[6]
- The greatest delay experienced by a user
- Runtime of the method
- Number of cars serviced
- Percentage of cars serviced without a delay

[6]Cars that are not delayed are included in this as '0'.

- Number of cars that could not find a parking spot

The primary metrics among these are the average delay or tardiness and the greatest delay experienced by users . A low average delay across multiple testing runs means that the scheduling method is able to generate schedules that service customers quickly. Customers are less likely to choose a charging option where they risk large delays, even if the average delay is low. as such, it is of importance to minimize the maximum delay.

The other metrics are secondary. While the time needed to generate a schedule does not directly influence the schedule's quality, it is nonetheless of some importance. After all, if the runtime is so long that it is not possible to update the schedule as frequently as we might like, we might end up using older schedules that are less relevant to the current situation, thus yielding a less optimal result.Furthermore, in order to find a completer context for our metric, we also look at the total number of cars that were serviced on time, were serviced with a delay and which could not find a parking spot.

5 Experimental Results

An extensive number of simulations have been performed on the various aforementioned grids. This way, the quality of the various methods can be evaluated. The seeds for generating the arrivals of cars have been fixed in the NumPy.Random (the NumPy Community) Python package to enable an equal comparison between methods. This section will present and discuss a selection of the results; the full results can be found in Appendix C and `https://github.com/MaxVanHuffelen/ThesisData`.

For each of the various problem contexts and scheduling strategies discussed, a number of different numbers are presented. These are:

- The maximum delay in seconds (abbreviated 'max delay')
- The mean delay in seconds (abbreviated 'mean delay')
- The percentage of cars that are delayed (abbreviated '% delayed')
- The amount of cars served (shortened to 'served')
- The amount of cars that could not be served as they did not find a place to park (shortened to 'non-served')
- The runtime of the method(in seconds)
- The number of simulation runs (shortened to 'runs') performed

Note that for all numbers (except the number of runs performed) for each seed on which one or more runs have been performed, the average of all runs on the same seed is taken first. Then, the average is calculated of these average results, and those values are presented here. This is done to prevent any one seed from being overrepresented in the results in case it has been evaluated more often than others for a strategy.

For readability purposes, the names of methods have been abbreviated in this thesis. A table detailing the full names of methods as used in the GitHub repository can be found in Appendix B. The abbreviated names consist of a number of hyphenated elements. The first element is the name of the schedule generation scheme. Here, 'P' and 'S' are used to denote the parallel and serial SGSs, respectively. Similarly, 'PR' and 'SR' denote the usage of the rescheduling SGS with the parallel or serial SGS as a basis, respectively. The second element is used to denote the priority rule used to create the initial schedule. If the rescheduling SGS is used, the third element will denote the priority rule used for rescheduling. Finally, further elements may be included to denote variations in the approach, which will be explained throughout this section as they are introduced.

5.1 Search for Effective Methods

Our initial base scenario has been Grid 1 with 750 expected daily arrivals. A wide variety of methods has been applied to this grid in order to evaluate which methods are worth investigating further. The data of some of the best methods on this grid as well as S-EDD, which we regularly use as a baseline in this section, are shown in Table 1. As can be seen here, many methods have little to no difficulty creating effective schedules with minimal delays. We furthermore note that if an initial schedule created using the flexible serial or parallel SGS contains delays, these can typically be reduced or even removed entirely by using this schedule as a basis for a rescheduling SGS. While being able to solve scheduling issues with little delays is of course great, we would like our scenarios to be a little more challenging to be able to compare methods effectively. As such, in the second scenario we decided to increase the arrival rate by 50% while using the same grid to have a harder problem to solve. All priority rules described in Section 4.2.2 have been used to create schedules in this scenario. The results of the most interesting methods in this scenario with 1125 expected daily car arrivals can be seen in Table 2. A broader view of the qualities of methods is provided in Figure 10, which shows the average delays of all combinations of scheduling methods and priority rules, evaluated with the most effective found combination

of hyperparameters on Grid 1 with 1125 expected daily arrivals. For the sake of clarity, the delays of the serial SGS using the FCFS priority rules have been omitted; this method has an average delay of 1119.6 seconds, and has not been evaluated further due to its poor quality. In Table 2, we observe a number of things. A number of methods, especially PR-EDD-LSTU, perform particularly well, yielding infrequent and low delays - though this comes at the cost of having a higher runtime. When runtime is a constraint, PR-LSTU-EDD-IB has a more competitive runtime while still yielding good results. This is the interval-based scheduling variant of PR-LSTU-EDD, as denoted with '-IB'. The lowest maximum delay is achieved by P-LSTU. This method does, however, come at the cost of a greater mean delay experienced by users. While in many cases, using interval-based scheduling leads to an increase in delays, it can also lead to a reduction of delays as demonstrated by the results of SR-EDD-LSTU and its interval-based scheduling variant SR-EDD-LSTU-IB, where the latter has a lower mean delay and less delayed cars. In all cases, however, we see that interval-based scheduling leads to a significant reduction in computing time compared to event-triggered scheduling. This is consistent with what we would expect to see, as re-calculating schedules less frequent logically leads to less operations and thus less computing time. Furthermore, when comparing methods such as SR-EDD-LSTU and SR-EDD-LSTU-IB to S-EDD, we see how rescheduling can lead to significant improvements in every metric except runtime.

The methods yielding the best results, and those we will continue using in later subsections, are SR-EDD-LSTA, SR-EDD-LSTU, PR-EDD-LSTU, PR-LSTU-EDD, PR-LSTU-LST, S-EDD, P-EDD, and P-LSTU. Here, S-EDD is chosen for its simplicity and will be used as baseline method to which we compare other methods in Section 5.4.

Table 1: Results of some of the most effective methods on Grid 1 with 750 expected daily car arrivals

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LSTU	0.0	0.0	0.0	3787.4	1007.8	21030.1	5
PR-LSTU-EDD	0.0	0.0	0.0	3787.4	1007.8	17287.1	5
PR-LSTU-LST	0.0	0.0	0.0	3787.4	1007.8	18802.9	5
P-LSTU	7.3	0.0	0.0	3787.4	1007.8	4407.8	5
S-EDD	5298.2	2.7	0.1	3787.4	1007.8	3922.3	5

Table 2: Results of the discussed methods on Grid 1 with 1125 expected daily car arrivals: results of effective methods

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LSTU	2290.3	32.4	0.6	4321.6	3464.7	47212.8	15
SR-EDD-LSTU-IB	3232.2	29.9	**0.5**	4321.0	3465.3	11413.7	15
PR-EDD-LSTU	1777.4	**26.0**	0.5	**4321.7**	**3464.6**	44395.5	15
PR-LSTU-EDD-IB	2244.1	29.1	0.6	4320.7	3465.6	13418.8	15
P-LSTU	**809.9**	42.6	0.9	4321.5	3464.7	8105.1	15
S-EDD	16156.5	67.3	1.1	4320.2	3466.1	7250.3	15

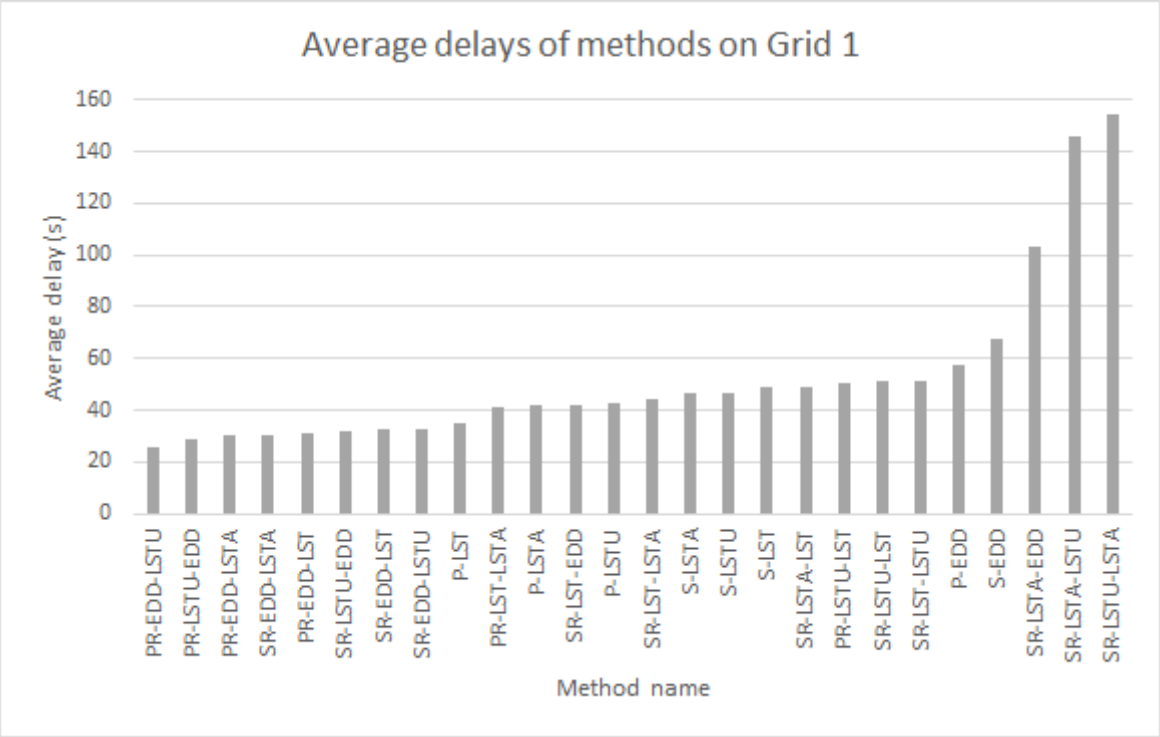


Figure 10: Average delays of all evaluated combinations of priority rules with default settings on Grid 1 with 1125 expected daily arrivals

Besides the priority rules, hyperparameters related to how the rescheduling is performed are interesting to look at and tweak. From the results, of which those of the SR-EDD-LSTA and SR-EDD-LSTU methods and their variants**?** are shown in Table 3, we see that a larger rescheduling fraction based on a smaller fraction of initial removals tends to lead to better results than otherwise identical methods with lower rescheduling fractions (as indicated with '-LR' in the strategy name, denoting a rescheduling fraction of 0.15 instead of the default 0.5) or greater rescheduling cluster fractions (as indicated with '-GC' in the strategy name, denoting a rescheduling cluster fraction of 0.15 instead of the default 0.05). In other words, the best results are achieved when we initially remove only a small amount of jobs (rescheduling cluster fraction), and then a great number of adjacent jobs (rescheduling fraction) to achieve large clusters of jobs to reschedule. From this we observe that it is likely that the removal fraction can be tuned to optimize the balance between effectively implementing the best traits of the new priority rule we reschedule to without discarding the old one. Where this balance lies exactly is likely dependent on many other factors, such as network layout, the used priority rules, the initial schedule generation scheme, the removal scheme and so on. The default values used and described throughout this thesis have been shown to yield effective results, however. The hyperparameters used in this thesis have been selected based on a preliminary investigation on a wide spread of permutations of hyperparameter settings, though finding exactly where this balance lies falls outside the scope of this thesis. An explanation for this phenomenon can be found in the fact that it is generally hard to reschedule isolated jobs effectively since, in a busy schedule where all available power is used, removing a single job creates a gap in the schedule that is exactly that job's size. Then, scheduling another job in that gap might either not fit (if the new job is larger than the old one or charges at higher rates) or leave an amount of power unused that can be hard to assign to another job (if the new job is smaller than the old one or charges at lower rates). Naturally, as the rescheduling fraction increases, the probability of such an occurrence decreases as the probability of adjacent jobs in a schedule being removed increases. As expected, then, we see that the random removal scheme (as indicated with '-RR' in the strategy name) also suffers from this, typically yielding more delays than other methods. The random removal scheme does, however, yield lower runtimes as the others require calculating which jobs are adjacent to which, a process which is not needed here. The data also shows that the idealized reduction-based removal scheme (as indicated with '-IR' in the strategy name) consistently yields more and greater delays while serving less customers than the simpler rule based on the number of adjacent jobs which is used by default.

Table 3: Simulation results on Grid 1 with 1125 expected daily car arrivals of SR-EDD-LSTA and SR-EDD-LSTU and their variants

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LSTA	2096.2	30.6	0.6	4322.3	3464.0	48698.7	15
SR-EDD-LSTA-LR	4662.8	32.7	0.6	4320.9	3465.3	19262.2	15
SR-EDD-LSTA-LR-IR	16400.9	92.8	1.4	4311.0	3469.4	6987.8	7
SR-EDD-LSTA-GC	3899.0	73.0	1.3	4312.7	3467.7	48036.6	7
SR-EDD-LSTA-RR	5960.4	95.8	1.6	4319.6	3459.6	17864.8	5
SR-EDD-LSTU	2290.3	32.4	0.6	4321.6	3464.7	47212.8	15
SR-EDD-LSTU-LR	4929.4	34.5	0.7	4321.0	3465.3	18605.3	15
SR-EDD-LSTU-LR-IR	16400.9	92.8	1.4	4311.0	3469.4	7161.9	7
SR-EDD-LSTU-GC	3800.5	68.1	1.2	4311.9	3468.6	44307.7	7
SR-EDD-LSTU-RR	5425.5	79.9	1.3	4317.8	3461.4	16965.5	5

5.2 Flexible Charging vs Fixed-Rate Charging

While we have seen the impact the previously discussed scheduling methods can have on the scheduling of flexible jobs, we cannot yet conclude if or how powerful the ability to charge jobs flexibly is compared to only being able to charge jobs at fixed rates. However, for practical applications it is relevant to know whether implementing this option is worthwhile. As such, we have also evaluated some of the most promising methods found in Section 5.1 in the same scenario of 1125 expected daily arrivals on Grid 1, but with the ability to charge flexibly removed. To do this, all cars can now only be charged at a fixed rate of 9kW; the average maximum charging rate of jobs in all other scenarios. Then, the results of our most promising methods are as can be seen in Table 4

Table 4: A comparison of results of methods when charging rates are fixed at 9kW (top) versus when charging can be done flexibly (bottom)

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LSTA[7]	66453.2	25014.0	99.8	3483.6	4295.6	126078.9	5
SR-EDD-LSTA	2096.2	30.6	0.6	4322.3	3464.0	48698.7	15
SR-EDD-LSTU[7]	65028.6	24825.8	99.7	3489.0	4290.2	133900.4	5
SR-EDD-LSTU	2290.3	32.4	0.6	4321.6	3464.7	47212.8	15
PR-LSTU-LST[7]	43284.6	24942.5	99.9	3515.2	4264.0	134910.3	5
PR-LSTU-LST	992.9	50.6	1.0	4320.4	3465.9	44580.4	15
P-EDD[7]	62744.7	24903.8	99.7	3480.8	4298.4	7059.3	5
P-EDD	15360.3	57.4	0.9	4319.2	3467.1	6997.2	15
S-EDD[7]	62744.7	24903.8	99.7	3480.8	4298.4	26646.4	5
S-EDD	16156.5	67.3	1.1	4320.2	3466.1	7250.3	15

As can be seen, the results are significantly worse than when flexible charging is possible; all scheduling methods yield significant delays for nearly all cars while servicing less customers. It should, however, be noted that due to the inability to charge flexibly, it is not possible for the scheduling methods in this scenario to use solar power - after all, if they did and solar power decreases, it would not be possible to prevent cable overloads without preemption. As such, it is evident that in our problem context the ability to charge flexibly is a great boon. In other scenarios where preemption or cable overloads might be tolerated but disincentivized the difference might not be quite so staggering as it is here. Nonetheless, being able to adaptively use up to all available power while being able to avoid the penalties associated with overloads or preemption will likely still be a great benefit.

5.3 Increased Parking Capacity

We notice that on Grid 1 with 1125 expected daily arrivals, a sizable fraction of the arrivals is unable to find a place to park and leaves the simulation unserved. As such, we investigated how well our methods would fare if more cars could find a place to park by creating a second grid, called Grid 2, where the parking capacities of all car parks are doubled. A selection of these results (top) compared to those of the same method on Grid 1 (bottom) are shown in Table 5.

[7] With charging rates fixed at 9kW

Table 5: A selection of results on Grid 2 with 1125 expected daily car arrivals (top) compared to those same methods on Grid 1 (bottom)

Strategy	grid	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LSTA	2	108715.6	49577.8	97.4	4574.8	3221.2	1625104.9	6
SR-EDD-LSTA	1	2096.2	30.6	0.6	4322.3	3464.0	48698.7	15
SR-EDD-LSTU	2	107356.6	49685.0	97.4	4576.0	3220.0	1666256.3	6
SR-EDD-LSTU	1	2290.3	32.4	0.6	4321.6	3464.7	47212.8	15
PR-EDD-LSTU	2	111858.5	49240.1	98.5	4589.3	3206.7	1609724.3	6
PR-EDD-LSTU	1	1777.4	26.0	0.5	4321.7	3464.6	44395.5	15
PR-LSTU-LST-IB	2	87223.2	50440.3	96.6	4580.5	3215.5	252475.2	6
PR-LSTU-LST-IB	1	1264.5	50.2	1.0	4320.9	3465.3	13266.4	15
P-EDD	2	112715.1	49031.4	98.5	4600.0	3196.0	26086.1	6
P-EDD	1	15360.3	57.4	0.9	4319.2	3467.1	6997.2	15
S-EDD	2	109012.6	49584.5	97.4	4573.0	3223.0	245441.6	6
S-EDD	1	16156.5	67.3	1.1	4320.2	3466.1	7250.3	15

Compared to those on Grid 1, the results on Grid 2 are vastly different. The increase in parking capacity leads to significant delays with all methods. Among these, we observe that the usage of interval-based scheduling and different hyperparameters such as a smaller removal fraction can yield some minor improvements, though. A small increase in the number of cars serviced is also observed.

The reason for these increases can likely be found in the fact that a greater parking capacity leads to more cars being able to park (until all car parks are at capacity), and thus placing a greater demand for power on the already-limited supply. This leads to there generally being more cars waiting to be served - and thus also more cars experiencing delays until eventually nearly all cars will need to wait (too) long before they can be serviced. This queue of cars waiting to be served does, however, mean that there are almost always customers available to service meaning we never waste available power, leading to a small increase in customers served.

The results on this grid, though somewhat limited due to the extensive runtimes, show that PR-LSTU-EDD yields the least delayed cars and P-LSTU yields the lowest maximum delay. The lowest average delay has been achieved by SR-EDD-LSTU-LR, though this is most likely caused by the fact that this method serves significantly fewer cars than other methods rather than it yielding better schedules. P-EDD then not only manages to serve the greatest amount of customers, it also does so with a competitive mean delay. Rescheduling did not manage to improve these metrics here. In all likelihood, in an (over)strained grid such as this one, rescheduling may lead to too many jobs being attempted to process simultaneously, leading to them all charging at low rates and having a late completion time and all the while leading to a reduction in the flexibility we have when scheduling other jobs.

5.4 Variations of Grid Layout

After having seen the impact a change in grid layout can have on the efficacy of the various scheduling approaches, these methods have been evaluated on a number of other grids and compared to a baseline approach of S-EDD. This way, we can ensure that our rescheduling SGSs offer improvements over the baseline in all scenarios. Firstly, the most promising methods found on Grid 1 have been evaluated on Grid 3. Grid 3 is quite similar to Grid 1, boasting the same total parking capacity across the same number of car parks and the same amount of power supplied from the root node. However, less solar power is available as the only solar panels are placed at the first car park from the root, and all car parks are placed in one long line. Since many cable segments, especially those near the root, are used to supply power to all car parks, scheduling approaches will need to balance where the power sent through these cables is directed. Too much power consumption at one point in the grid might otherwise deprive other car parks of power. The results of the methods evaluated on this grid are shown in Table 6

Table 6: An overview of results of the discussed approaches on Grid 3 with 1125 expected daily arrivals

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LSTA	49262.0	8030.2	71.2	3892.3	3886.9	286444.1	6
SR-EDD-LSTU-IB	47937.8	**7884.5**	71.0	3909.4	3869.8	29920.6	5
PR-LSTU-LST	21633.4	8844.0	81.6	3864.2	3915.0	551593.4	5
P-EDD	50319.1	8047.7	70.5	**3915.0**	**3864.2**	7202.9	5
S-EDD	53371.3	8397.6	**70.4**	3894.6	3884.6	31455.4	5
S-LST	22745.2	9189.5	81.6	3853.2	3926.0	31380.2	5
S-LSTU	**18408.7**	8607.6	82.8	3884.2	3895.0	33215.9	5

When looking at the results on this grid, we note that effective charging at 1125 expected daily arrivals proves to be challenging. While some methods provide significantly better results than others, all yield significant and frequent delays. Out of these results, SR-EDD-LSTU-IB yields a competitively low mean delay which is suffered by relatively few customers. On the other hand, P-EDD manages to serve the most customers. The lowest maximum delay, on the other hand, is found when using the S-LSTU method. Compared to our baseline of S-EDD, we see that both rescheduling approaches and different choices in priority rules can lead to improvements in all metrics besides the fraction of delayed cars. The improvement in mean delay yielded by rescheduling methods over S-EDD is especially pronounced, with nearly all rescheduling methods having a lower mean delay. SR-EDD-LSTA has been included in Table 6 as a second example besides, SR-EDD-LSTU-IB which has achieved the greatest reduction. As this is the metric for which these methods optimize, this demonstrates how these methods retain their quality and efficacy on this grid.

Worth noting is that, despite sharing many similarities, methods applied here have significantly longer runtimes and worse results than those same methods applied on Grid 1 with the same arrival rate. This difference is especially pronounced on the (event-triggered) rescheduling methods, whereas it is less significant on the flexible parallel SGS. This promotes the belief that having too many car parks supplied through the same cables leads to less effective scheduling as each car scheduled deprives many other cars, including those in other car parks, of power. Furthermore, this means that many cars are adjacent in the schedules, leading to more computation power being required for rescheduling, thus increasing computing time.

Finally, we also see that a large fraction of car arrivals go unserved as the car parks are full. As such, when we increase the difficulty of the scenario by increasing the arrival rate from 1125 to 1500 expected daily car arrivals, the vast majority of the extra car arrivals leave the simulation unserved. The amount of cars served increases by only minute amounts, often less than one percent. Nonetheless, we see that delays increase both in frequency and severity. Thus, we see that an increase of arrivals can lead to a significant reduction of performance in an overstrained grid. This reduction in performance appears to affect rescheduling methods more strongly than simpler scheduling methods, who may outperform the rescheduling methods here. Just as we have observed on Grid 2, the most likely reason for this is that in an (over)strained grid, rescheduling methods may try to charge too many jobs simultaneously, leading to all of them being processed at a low rate and finishing late while causing a reduction in the flexibility of scheduling.

Then, based on this observation one might expect to see that a grid with parallel parking places and little depth would perform well. As such, these same methods will be evaluated on Grid 4, where no cables are shared by car parks. It should be noted that Grid 4 has access to a greater amount of power from the root node, 540 kW versus the 400 available to Grid 3, as well as a parking capacity, 630 places total versus Grid 3's 440. As per our earlier observations, we thus expect results in this grid might have an increase in the amount of cars processed (since more power is available supplied through cables that are not shared by car parks) as well as the length of delays experienced by users (as the number of parking places is increased). Indeed, when looking at the results of this grid we see that the amount of cars serviced and the mean and maximum delays experienced are consistently higher than those measured on Grid 3 with the same arrival rate, which further supports the observations we made earlier on how grid layout affects performance. However, the percentage of cars that are delayed is significantly reduced here compared to the previous grid. Thus, it is evident delays are typically less frequent but larger than those on Grid 3. This is consistent with what we would expect as, outside of the most busy hours cars can be processed faster, and thus more cars can be served before their due date. During those busiest hours, however, waiting times are longer as the expanded parking capacity facilitates a longer queue.

When comparing the results on this grid, as seen in Table 7, we again see that S-EDD is outperformed by other scheduling methods in every metric. PR-LSTU-EDD serves the most customers with the lowest mean delay and the second-lowest percentage of cars delayed - only SR-EDD-LSTU manages to have a marginally lower fraction of delayed cars. The lowest maximum delay is achieved by P-LSTU, with a number of rescheduling methods also managing to improve on the maximum delay achieved by S-EDD. We see clearly that on this grid, too, rescheduling methods succeed in improving on the standard set by our baseline approach for every metric.

Table 7: An overview of results of the discussed approaches on Grid 4 with 1500 expected daily arrivals

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LSTU	65582.4	13735.0	**66.4**	5171.8	5285.4	497858.2	5
SR-EDD-LSTU-IB	65182.6	13383.1	67.8	5175.6	5281.6	35718.3	5
PR-LSTU-EDD	66529.8	**13302.4**	66.4	**5202.8**	**5254.4**	1172933.3	5
P-EDD	64884.1	13541.9	68.0	5179.6	5277.6	13852.0	5
P-LSTU	**32909.2**	14194.6	72.6	5184.0	5273.2	18082.4	5
S-EDD	64737.5	14053.0	67.5	5165.0	5292.2	56851.4	5

Table 8: An overview of results of the discussed approaches on Grid 5 with 1125 expected daily arrivals

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-LST-EDD-LR	21166.6	**171.9**	4.0	5620.6	2158.6	57226.5	5
SR-LST-LSTU-LR	21166.6	**171.9**	4.0	5620.6	2158.6	58917.5	5
SR-LSTU-EDD-LR	**12007.8**	229.6	5.5	5619.4	2159.8	66585.2	5
PR-EDD-LSTU-IB	26010.8	212.0	3.3	5620.6	2158.6	16135.1	5
PR-LSTU-EDD-IB	13595.0	199.2	5.0	**5621.8**	**2157.4**	15046.5	5
PR-LSTU-LST-IB	13595.0	199.2	5.0	**5621.8**	**2157.4**	17219.9	5
P-EDD	23984.6	208.3	**3.1**	5619.0	2160.2	43313.4	5
S-EDD	25269.8	231.7	3.4	5618.2	2161.0	67306.5	5
S-LST	21166.6	**171.9**	4.0	5620.6	2158.6	69965.1	5

So far, we have primarily considered relatively small grids of at most seven car parks. As such, we have also evaluated our approaches on the significantly larger Grid 5 with an arrival rate of 1125 expected daily cars to see how well our approaches handle more sizable grids. Due to the size of this grid, runtimes of the rescheduling methods increased drastically. For this reason rescheduling approaches have only been used with interval-based scheduling or reduced rescheduling fractions. Despite these restrictions, however, we see that these methods still performed well as can be seen in Table 8. The lowest values for the mean delay are yielded by S-LST, SR-LST-EDD-LR, and SR-LST-LSTU-LR, which all produce the same value for the lowest mean delay. While the schedules produced by the rescheduling methods did not manage to reduce delays compared to those yielded by S-LST, they did both manage achieve a lower runtime. Tests have shown that a notable measure variance is present in the runtimes of simulations, which leads us to believe that this improvement is due to random variance rather than a property of the method. As the rescheduling method first creates a base schedule using the serial or parallel schedule generation scheme before creating mutations, it thus should not typically be faster than the method creating just those base schedules. Nonetheless, this does show that when the rescheduling scheme does not improve on the quality of schedules compared to the schedule it is based on, it also does not increase the runtime of the method significantly.

The lowest maximum delay is yielded by SR-LSTU-EDD-LR and the lowest fraction of cars delayed is yielded by P-EDD. Similarly, PR-EDD-LSTU-IB improves on the fraction of cars delayed by S-EDD in a shorter runtime and also serves more cars than both P-EDD and S-EDD, though at the cost of somewhat higher delays compared to P-EDD and a greater maximum delay than both. Finally, the schedules with the greatest amount of cars served are yielded by both PR-LSTU-EDD-IB and PR-LSTU-LST-IB. These two approaches yield results with identical metrics, except for runtime; PR-LSTU-EDD-IB produces its schedules more quickly than any other evaluated method, whereas PR-LSTU-LST-IB is the third fastest approach out of those tested.

Again, the results clearly show that a variety of the methods evaluated can offer notable improvements compared to our baseline method, S-EDD.

5.5 Comparison of Schedule Updating Frequencies

On the various grids, we have evaluated a number of methods using both event-triggered scheduling and interval-based scheduling. When using event-triggered scheduling, the schedule is updated whenever either a new car arrives or the solar powers are updated (which happens hourly). Interval-bases scheduling updates the schedule much less frequently; it does so only when the solar powers are updated or when a car arrives who's priority value is higher than that of 80% of vehicles in the schedule. If our schedules are effective and robust, we expect to see that the lower frequency of updating the schedules (and thus being less adaptive to incorporating new arrivals) does not lead to a significant reduction in quality of our method compared to the event-triggered variant.

The differences in results between interval-based and event-triggered scheduling can be seen in Tables 9 to 11. Comparisons for Grids 2 and 5 have been omitted as, due to long runtimes, results of some methods are not present or only present with a lower rescheduling fraction, making a fair comparison impossible.

Table 9: Comparison of event-triggered and interval-based scheduling methods on Grid 1 with 1125 average daily arrivals

	max delay	mean delay	% delayed	served	non-served	runtime
event-triggered	1777.4	33.7	0.6	4321.6	3464.6	46122.1
interval-based	2338.1	36.9	0.7	4321.2	3465.0	12419.8
% increase IB	31.5	9.7	7.4	-0.0	0.0	-73.1

Table 10: Comparison of event-triggered and interval-based scheduling methods on Grid 3 with 1125 average daily arrivals

	max delay	mean delay	% delayed	served	non-served	runtime
event-triggered	43957.3	8328.6	73.5	3883.2	3896.0	444395.8
interval-based	44148.3	8318.5	73.3	3894.9	3884.3	32503.4
% increase IB	0.4	-0.1	-0.3	0.3	-0.3	-92.7

Table 11: Comparison of event-triggered and interval-based scheduling methods on Grid 4 with 1500 average daily arrivals

	max delay	mean delay	% delayed	served	non-served	runtime
event-triggered	59976.1	13651.4	68.0	5182.6	5274.6	737654.0
interval-based	59086.2	13701.1	68.9	5183.4	5273.8	49081.2
% increase IB	-1.5	0.4	1.5	0.0	-0.0	-93.3

As can be seen from these results, when we compare the quality of interval-based to event-triggered scheduling we indeed see that using interval-based scheduling typically leads to at most a minor decrease in quality and at best even an improvement in some metrics. As such, we conclude that the quality of the schedules created with our methods is such that even when they are updated much less frequently, the reduction in charging efficacy is only minor. Clearly, our methods are indeed an effective way of scheduling EV charging.

6 Conclusion

The research question that has been formulated at the start of this thesis and which we have aimed to find an answer to is:

> *Which scheduling strategies can produce effective schedules for the scheduling of EV charging with the least delay suffered by customers when considering grid constraints, flexible charging rates and uncertain future demand and renewable energy yield?*

The results of our experiments, as discussed in Section 5 provide an answer to this question. The ability to charge jobs at flexible rates allows us to easily adapt to new arrivals or changes in available solar power and thus ensure constraints are always met while using solar power, yielding significantly improved results. Furthermore, we see that the simpler flexible and parallel schedule generating schemes can occasionally yield decent results, in particular regularly yielding low maximum delays. However, the best solutions are often found by using the rescheduling schedule generation scheme. Through rescheduling, these simple schedules can often be improved further, typically yielding improvements in the metric for which we optimize as well others. Furthermore, these improvements can be extended to include the runtime of methods by using interval-based scheduling instead of event-triggered scheduling without incurring significant reductions in performance - and occasionally even increasing the performance quality. The performance quality of a rescheduling SGS is strongly influenced by the choice of hyperparameters. Here we see that rescheduling fewer, larger clusters of adjacent jobs tends to yield the best results. This can be done by using the adjacency-based removal scheme to remove large clusters of jobs based on a small number of initial job removals. Though it differs per scenario which combination of priority rules yields the best results, methods using Earliest Due Date and variations of the Latest Starting Time priority rules have consistently lead to good solutions being found with the rescheduling SGS. The best of these methods have been found to be SR-EDD-LSTA, SR-EDD-LSTU, PR-EDD-LSTU, PR-LSTU-EDD, PR-LSTU-LST. Furthermore, we have observed that the LSTU priority rule consistently leads to low maximum delays experienced by users.

Finally, the grid layout also has a significant effect on performance quality. It has been observed that having too large parking places can lead to long delays. Furthermore, supplying too many parking places through one cable segment may lead to longer computation times and slower processing of parked cars.

To summarize, we have designed novel variations of the LST priority rules which, together with the EDD priority rule, have been used in our Rescheduling Schedule Generation Scheme to create good schedules for the charging of EVs at flexible rates. Furthermore, we have shown that these schedules are of sufficient quality that they maintain their efficacy well even when updated infrequently on five different grids.

7 Future Work

Naturally, there are numerous expansions of the research presented here that we would have liked to look into, but were unable to do due to time constraints. The topics discussed in this section are suggested for future research.

One limitation of the current implementation of the simulation is that the arrival rates are treated as being identical throughout the day for every day. However, in practice, it is common for these distributions to vary throughout the week; especially in the weekend arrival rates differ from weekdays. It would be interesting to see how the efficacy of the tested methods changes when this is taken into account. Since the arrival rates repeat each day in the used simulation model, this could be implemented by making a 'day' have the length of one week (or another period of time after which arrival rates should repeat). This is done simply providing the arrival fractions for each hour of the week, setting the expected 'daily arrivals' to be the arrivals we expect in a week, and changing the constants in the code of '24 hours' to '168 hours' as relevant.

Furthermore, this thesis has only investigated a limited subset of the infinite number of grids which can be designed and built. It is very conceivable that certain strategies might perform better on grid layouts that have been designed to make maximally efficient use of them. As such, a possible future avenue of research would be an investigation of how the efficacy of the different methods changes on different grid layouts with different capacities. One possible feature of grids that has not been explored yet is using a cable with sufficiently large power capacity to supply a number of smaller car parks such that it is not necessary to always charge cars at every car park using all power available. Then, the capacity of the shared cable can be assigned to car parks flexibly depending on where power is needed, possibly reducing the amount of cable capacity needed.

All grids investigated thus far had the same amount of solar panels. However, some methods might be able to handle this uncertainty better than others. An investigation into which methods can most effectively utilize the uncertain yield of solar panels and thus minimize the use of power from external sources (which might be generated in a less sustainable manner) might yield very interesting and useful results.

In a similar vein, the solar distributions can also be changed to represent, for example, different seasons. The current simulation has only used a distribution representing summer conditions, though in simulations of other distributions the results of the investigated methods might change. It would be interesting to see how these changes in scenario would affect the efficacy of methods.

Regarding the scheduling methods, further research into the optimal values and combinations of hyperparameters can most likely yield improvements to the results of the investigated methods. Similarly, the creation and investigation of more priority rules may result in better results being found. In particular, since it is observed that rescheduling methods sometimes schedule too many jobs at low rates leading to jobs finishing late, implementing priority rules that give a higher priority to jobs that are already started may prove to be fruitful.

The current rescheduling method only allows the usage of two priority rules; one for the initial schedule and one to reschedule with. It would theoretically be possible to reschedule first with one rule, then a second, and so on until we return to the first rule in a cyclical pattern. Work on this topic can be continued by implementing and evaluating this method. This way, many rules can be used to hopefully exploit the best features of each.

Finally, the current method has only been used to optimize schedules for the total delay suffered by users. However, the same and/or aforementioned methods can theoretically also be used to optimize schedules for a different metric or even use a linear combination of metrics as an objective. To do this, only the quality measure function in would need to be changed. A potential avenue of research would then be evaluating the efficacy and practical applications of such an approach here.

References

W. Ackooij, I. Danti Lopez, A. Frangioni, F. Lacalandra, and M. Tahanan. Large-scale unit commitment under uncertainty: an updated literature survey. *Annals of Operations Research*, 271(1):11–85, December 2018. doi: 10.1007/s10479-018-3003-z. URL `https://ideas.repec.org/a/spr/annopr/v271y2018i1d10.1007_s10479-018-3003-z.html`.

Christian Artigues and Pierre Lopez. Energetic reasoning for energy-constrained scheduling with a continuous resource. *Journal of Scheduling*, page 36 p., 2014. doi: 10.1007/s10951-014-0404-y. URL `https://hal.science/hal-01108964`.

Francisco Ballestín, Vicente Valls, and Sacramento Quintanilla. *Due Dates and RCPSP*, pages 79–104. Springer US, Boston, MA, 2006. ISBN 978-0-387-33768-5. doi: 10.1007/978-0-387-33768-5_4. URL `https://doi.org/10.1007/978-0-387-33768-5_4`.

R.J.J. Brouwer, J.M. van den Akker, F.P.M. Dignum, and W. Vermeiden. Forecast-based optimization of islanded microgrids. In *2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6, 2018. doi: 10.1109/ISGTEurope.2018.8571679.

Peter Brucker. Scheduling and constraint propagation. *Discrete Applied Mathematics*, 123(1):227–256, 2002. ISSN 0166-218X. doi: https://doi.org/10.1016/S0166-218X(01)00342-0. URL `https://www.sciencedirect.com/science/article/pii/S0166218X01003420`.

Antonio J. Conejo and Luis Baringo. *Unit Commitment and Economic Dispatch*, pages 197–232. Springer International Publishing, Cham, 2018. ISBN 978-3-319-69407-8. doi: 10.1007/978-3-319-69407-8_7. URL https://doi.org/10.1007/978-3-319-69407-8_7.

Gunther Gust, Tobias Brandt, Salman Mashayekh, Miguel Heleno, Nicholas DeForest, Michael Stadler, and Dirk Neumann. Strategies for microgrid operation under real-world conditions. *European Journal of Operational Research*, 292(1):339–352, 2021. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2020.10.041. URL https://www.sciencedirect.com/science/article/pii/S0377221720309188.

Sönke Hartmann and Dirk Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14, 2010. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2009.11.005. URL https://www.sciencedirect.com/science/article/pii/S0377221709008558.

Sönke Hartmann and Dirk Briskorn. An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 297(1):1–14, 2022. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2021.05.004. URL https://www.sciencedirect.com/science/article/pii/S0377221721003982.

Sönke Hartmann and Rainer Kolisch. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 127(2):394–407, 2000. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(99)00485-3. URL https://www.sciencedirect.com/science/article/pii/S0377221799004853.

Mohasha Isuru, Matthias Hotz, H.B. Gooi, and Wolfgang Utschick. Network-constrained thermal unit commitment for hybrid ac/dc transmission grids under wind power uncertainty. *Applied Energy*, 258:114031, 2020. ISSN 0306-2619. doi: https://doi.org/10.1016/j.apenergy.2019.114031. URL https://www.sciencedirect.com/science/article/pii/S0306261919317180.

Tamás Kis. *RCPS with Variable Intensity Activities and Feeding Precedence Constraints*, pages 105–129. 01 2006. ISBN 978-0-387-33643-5. doi: 10.1007/978-0-387-33768-5_5.

Robert Klein and Armin Scholl. Progress: Optimally solving the generalized resource-constrained project scheduling problem. *Mathematical Methods of Operations Research*, 52:467–488, 01 2000. doi: 10.1007/s001860000093.

Bernard Knueven, James Ostrowski, and Jean-Paul Watson. On mixed integer programming formulations for the unit commitment problem. *Optimization Online Repository*, 2018, 11 2018. ISSN 9999-0042. URL https://www.osti.gov/biblio/1492385.

Rainer Kolisch and Sönke Hartmann. *Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis*, pages 147–178. Springer US, Boston, MA, 1999. ISBN 978-1-4615-5533-9. doi: 10.1007/978-1-4615-5533-9_7. URL https://doi.org/10.1007/978-1-4615-5533-9_7.

Rainer Kolisch and Sönke Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174(1):23–37, 2006. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2005.01.065. URL https://www.sciencedirect.com/science/article/pii/S0377221705002596.

Rainer Kolisch, Arno Sprecher, and Andreas Drexl. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science*, 41(10):1693–1703, 1995. URL https://EconPapers.repec.org/RePEc:inm:ormnsc:v:41:y:1995:i:10:p:1693-1703.

Oumar Koné, Christian Artigues, Pierre Lopez, and Marcel Mongeau. Event-based milp models for resource-constrained project scheduling problems. *Computers Operations Research*, 38(1):3–13, 2011. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2009.12.011. URL https://www.sciencedirect.com/science/article/pii/S0305054809003360. Project Management and Scheduling.

K.Y. Li and R.J. Willis. An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research*, 56(3):370–379, 1992. ISSN 0377-2217. doi: https://doi.org/10.1016/0377-2217(92)90320-9. URL https://www.sciencedirect.com/science/article/pii/0377221792903209.

Moosa Moghimi Haji and Behrooz Vahidi. A solution to the unit commitment problem using imperialistic competition algorithm. *Power Systems, IEEE Transactions on*, 27:117 – 124, 03 2012. doi: 10.1109/TPWRS.2011.2158010.

Yorie Nakahira, Niangjun Chen, Lijun Chen, and Steven H. Low. Smoothed least-laxity-first algorithm for ev charging. In *Proceedings of the Eighth International Conference on Future Energy Systems*, e-Energy '17, page 242–251, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350365. doi: 10.1145/3077839.3077864. URL https://doi.org/10.1145/3077839.3077864.

Tehreem Nasir, Syed Sabir Hussain Bukhari, Safdar Raza, Hafiz Munir, Abrar Muhammad, Khawaja Abdul Muqeet, M. Kamran Bhatti, Jong-Suk Ro, and Rooha Masroor. Recent challenges and methodologies in smart grid demand side management: State-of-the-art literature review. *Mathematical Problems in Engineering*, 2021:1–16, 08 2021. doi: 10.1155/2021/5821301.

Margaux Nattaf, Markó Horváth, Tamás Kis, Christian Artigues, and Pierre Lopez. Polyhedral results and valid inequalities for the continuous energy-constrained scheduling problem. *Discrete Applied Mathematics*, 258:188–203, 2019. ISSN 0166-218X. doi: https://doi.org/10.1016/j.dam.2018.11.008. URL https://www.sciencedirect.com/science/article/pii/S0166218X18306164.

K. Neumann and J. Zimmermann. Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints. *European Journal of Operational Research*, 127 (2):425–443, 2000. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(99)00498-1. URL https://www.sciencedirect.com/science/article/pii/S0377221799004981.

Hironori Okubo, Toshiyuki Miyamoto, Satoshi Yoshida, Kazuyuki Mori, Shoichi Kitamura, and Yoshio Izui. Project scheduling under partially renewable resources and resource consumption during setup operations. *Computers Industrial Engineering*, 83:91–99, 2015. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2015.02.006. URL https://www.sciencedirect.com/science/article/pii/S0360835215000637.

Robert Pellerin, Nathalie Perrier, and François Berthaut. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 280(2):395–416, 2020. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2019.01.063. URL https://www.sciencedirect.com/science/article/pii/S0377221719300980.

A. Alan B. Pritsker, Lawrence J. Waiters, and Philip M. Wolfe. Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16(1):93–108, 1969. URL https://EconPapers.repec.org/RePEc:inm:ormnsc:v:16:y:1969:i:1:p:93-108.

B. Saravanan, Siddharth Das, Surbhi Sikri, and Dwarkadas Pralhaddas Kothari. A solution to the unit commitment problem—a review. *Frontiers in Energy*, 7:223 – 236, 2013.

Andreas Schutt, Thibaut Feydy, and Peter J. Stuckey. Explaining time-table-edge-finding propagation for the cumulative resource constraint. In Carla Gomes and Meinolf Sellmann, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 234–250, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-38171-3.

Katrin Seddig, Patrick Jochem, and Wolf Fichtner. Integrating renewable energy sources by electric vehicle fleets under uncertainty. *Energy*, 141:2145–2153, 2017. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2017.11.140. URL https://www.sciencedirect.com/science/article/pii/S0360544217319989.

Helmut Simonis. Sudoku as a constraint problem. In *CP Workshop on modeling and reformulating Constraint Satisfaction Problems*, volume 12, pages 13–27. Citeseer, 2005.

Mohammad Taghikhani and Behnam Zangeneh. Optimal energy scheduling of micro-grids considering the uncertainty of solar and wind renewable resources. *Journal of Scheduling*, 25:1–10, 06 2022. doi: 10.1007/s10951-022-00739-5.

the NumPy Community. Random sampling (numpy.random). URL https://numpy.org/doc/stable/reference/random/index.html.

Matthew Wall. A genetic algorithm for resource-constrained scheduling. 08 1996.

Qinglong Wang, Xue Liu, Jian Du, and Fanxin Kong. Smart charging for electric vehicles: A survey from the algorithmic perspective. *CoRR*, abs/1607.07298, 2016. URL http://arxiv.org/abs/1607.07298.

Deshi Ye and Guochuan Zhang. On-line scheduling of parallel jobs in a list. *Journal of Scheduling*, 10:407–413, 2007.

Deshi Ye, Danny Ziyi Chen, and Guochuan Zhang. Online scheduling of moldable parallel tasks. *Journal of Scheduling*, 21:647–654, 2018.

Linet Özdamar, Gunduz Ulusoy, and Mete Bayyiğit. A heuristic treatment of tardiness and net present value criteria in resource constrained project scheduling. *International Journal of Physical Distribution Logistics Management*, 28:805–824, 12 1998. doi: 10.1108/09600039810248181.

A Generation Distributions

Table A1: Average solar panel revenues as a fraction of peak revenue

Hour of day	Winter	Summer
0	0	0
1	0	0
2	0	0
3	0	0.001548952
4	0	0.017126799
5	0	0.05556752
6	0.000103127	0.13203421
7	0.00823918	0.22293125
8	0.05180765	0.31115308
9	0.11377241	0.38261825
10	0.16397867	0.42669293
11	0.1900904	0.446001
12	0.18747252	0.4401042
13	0.15443817	0.40363738
14	0.09591667	0.3424478
15	0.03456073	0.26145884
16	0.004871739	0.16952068
17	8.68E-06	0.08330672
18	0	0.026936987
19	0	0.005260382
20	0	0
21	0	0
22	0	0
23	0	0

Table A2: Average fraction of arrivals at each hour

Hour of day	Arrival fraction
0	0.012926435
1	0.004938874
2	0.002042621
3	0.001188988
4	0.000304869
5	0.000518277
6	0.007560745
7	0.027072345
8	0.058748209
9	0.049114356
10	0.035456236
11	0.040120728
12	0.045181549
13	0.048108289
14	0.050913082
15	0.057254352
16	0.072223408
17	0.103899271
18	0.125453492
19	0.071156367
20	0.056217798
21	0.052467913
22	0.047315631
23	0.029816164

Table A3: Distribution of charging rates

Minimum Charging Rate	Maximum Charging Rate	Probability
3	6	0.0625
3	7	0.0625
3	8	0.0625
3	9	0.0625
4	7	0.0625
4	8	0.0625
4	9	0.0625
4	10	0.0625
5	8	0.0625
5	9	0.0625
5	10	0.0625
5	11	0.0625
6	9	0.0625
6	10	0.0625
6	11	0.0625
6	12	0.0625

Table A4: Weights of charging volumes of car arrivals

Charging Volume (kWh)	Weight	Charging Volume (kWh)	Weight
0	0.029938112	51	0.004268163
1	0.026188226	52	0.003871833
2	0.035212341	53	0.004786439
3	0.03268193	54	0.003810859
4	0.043443797	55	0.003262096
5	0.050333831	56	0.003384043
6	0.071186854	57	0.003445017
7	0.055973903	58	0.003201122
8	0.048809488	59	0.002804793
9	0.032163654	60	0.002987714
10	0.021188378	61	0.002957227
11	0.023383433	62	0.002317003
12	0.020670102	63	0.002073108
13	0.019725008	64	0.002256029
14	0.018688455	65	0.001981647
15	0.018261638	66	0.001585318
16	0.01804823	67	0.001615804
17	0.017347032	68	0.001249962
18	0.017286058	69	0.001128014
19	0.018718941	70	0.000640224
20	0.018078717	71	0.000853633
21	0.018901863	72	0.000518277
22	0.017286058	73	0.000731685
23	0.017316545	74	0.000518277
24	0.017103137	75	0.000762172
25	0.016615347	76	0.000579251
26	0.014511753	77	0.000396329
27	0.015487333	78	0.000396329
28	0.014938569	79	0.000182921
29	0.013200817	80	0.000182921
30	0.011859395	81	9.15E-05
31	0.013322765	82	9.15E-05
32	0.010853328	83	9.15E-05
33	0.01112771	84	0.000274382
34	0.009603366	85	0.000243895
35	0.008444864	86	0.000152434
36	0.00829243	87	0.000182921
37	0.008079022	88	0.000182921
38	0.006585165	89	0.000182921
39	0.007042468	90	3.05E-05
40	0.006920521	91	0
41	0.006676626	92	0
42	0.006219323	93	3.05E-05
43	0.005762019	94	0
44	0.005822993	95	0
45	0.005487638	96	0
46	0.005640072	97	0
47	0.005883967	98	0
48	0.005121795	99	0
49	0.004115728	100	0
50	0.004115728	101	3.05E-05

Table A5: Weights of connection times of car arrivals

Connection time (hour)	Weight
0	0.075089174
1	0.089052163
2	0.084570592
3	0.069967379
4	0.05356544
5	0.036309869
6	0.025608975
7	0.02820036
8	0.041492637
9	0.03755983
10	0.03707204
11	0.040699979
12	0.046431511
13	0.055242218
14	0.049205817
15	0.034511143
16	0.02682845
17	0.022590775
18	0.019237218
19	0.016737295
20	0.013261791
21	0.010426511
22	0.008536325
23	0.006707113
24	0.005518124
25	0.004237676
26	0.00292674
27	0.001859699
28	0.001524344
29	0.001128014
30	0.000579251
31	0.000914606
32	0.001280449
33	0.001249962
34	0.001341423
35	0.001920673

Connection time (hour)	Weight
36	0.002256029
37	0.003231609
38	0.003536478
39	0.002804793
40	0.002408463
41	0.002865766
42	0.001890186
43	0.002286516
44	0.001890186
45	0.001310936
46	0.001432883
47	0.001341423
48	0.000945093
49	0.000548764
50	0.000518277
51	0.000274382
52	0.000121948
53	0.000152434
54	0.000426816
55	0.000243895
56	0.000213408
57	0.000457303
58	0.000396329
59	0.000274382
60	0.000670711
61	0.000945093
62	0.000792659
63	0.000884119
64	0.000853633
65	0.000640224
66	0.000396329
67	0.000609738
68	0.000548764
69	0.000457303
70	0.007987561

B Strategy Name Abbreviations

Table B1: Strategy name abbreviations

Full Strategy Name	Abbreviation
serialReschedule_EDD-LST-4-0.15-0.05-100-idealizedReduction	SR-EDD-LST-LR-IR
serialReschedule_EDD-LST-4-0.15-0.05-100-nAdjacent	SR-EDD-LST-LR
serialReschedule_EDD-LST-4-0.15-0.1-100-nAdjacent	SR-EDD-LST-LR-MC
serialReschedule_EDD-LST-4-0.25-0.05-100-nAdjacent	SR-EDD-LST-MR
serialReschedule_EDD-LST-4-0.5-0.05-100-nAdjacent	SR-EDD-LST
serialReschedule_EDD-LST-4-0.5-0.05-100-nAdjacent-rollingHorizon20	SR-EDD-LST-IB
serialReschedule_EDD-LST-4-0.5-0.15-100-nAdjacent	SR-EDD-LST-GC
serialReschedule_EDD-LST-6-0.15-0.05-100-nAdjacent	SR-EDD-LST-MI-LR
serialReschedule_EDD-LSTA-4-0.15-0.05-100-idealizedReduction	SR-EDD-LSTA-LR-IR
serialReschedule_EDD-LSTA-4-0.15-0.05-100-nAdjacent	SR-EDD-LSTA-LR
serialReschedule_EDD-LSTA-4-0.15-0.15-100-nAdjacent	SR-EDD-LSTA-RR
serialReschedule_EDD-LSTA-4-0.15-0.15-100-nAdjacent-rollingHorizon20	SR-EDD-LSTA-RR-IB
serialReschedule_EDD-LSTA-4-0.5-0.05-100-nAdjacent	SR-EDD-LSTA
serialReschedule_EDD-LSTA-4-0.5-0.05-100-nAdjacent-rollingHorizon20	SR-EDD-LSTA-IB
serialReschedule_EDD-LSTA-4-0.5-0.15-100-nAdjacent	SR-EDD-LSTA-GC
serialReschedule_EDD-LSTU-4-0.15-0.05-100-idealizedReduction	SR-EDD-LSTU-LR-IR
serialReschedule_EDD-LSTU-4-0.15-0.05-100-nAdjacent	SR-EDD-LSTU-LR
serialReschedule_EDD-LSTU-4-0.15-0.15-100-nAdjacent	SR-EDD-LSTU-RR
serialReschedule_EDD-LSTU-4-0.15-0.15-100-nAdjacent-rollingHorizon20	SR-EDD-LSTU-RR-IB
serialReschedule_EDD-LSTU-4-0.5-0.05-100-nAdjacent	SR-EDD-LSTU
serialReschedule_EDD-LSTU-4-0.5-0.05-100-nAdjacent-rollingHorizon20	SR-EDD-LSTU-IB
serialReschedule_EDD-LSTU-4-0.5-0.15-100-nAdjacent	SR-EDD-LSTU-GC
serialReschedule_GRD-LST-4-0.15-0.05-100-nAdjacent	SR-GRD-LST-LR
serialReschedule_LST-EDD-4-0.15-0.05-100-idealizedReduction	SR-LST-EDD-LR-IR
serialReschedule_LST-EDD-4-0.15-0.05-100-nAdjacent	SR-LST-EDD-LR
serialReschedule_LST-EDD-4-0.5-0.05-100-nAdjacent	SR-LST-EDD
serialReschedule_LST-EDD-4-0.5-0.15-100-nAdjacent	SR-LST-EDD-GC
serialReschedule_LST-LSTA-4-0.15-0.05-100-idealizedReduction	SR-LST-LSTA-LR-IR
serialReschedule_LST-LSTA-4-0.15-0.05-100-nAdjacent	SR-LST-LSTA-LR
serialReschedule_LST-LSTA-4-0.5-0.05-100-nAdjacent	SR-LST-LSTA
serialReschedule_LST-LSTA-4-0.5-0.05-100-nAdjacent-rollingHorizon20	SR-LST-LSTA-IB
serialReschedule_LST-LSTU-4-0.15-0.05-100-idealizedReduction	SR-LST-LSTU-LR-IR
serialReschedule_LST-LSTU-4-0.15-0.05-100-nAdjacent	SR-LST-LSTU-LR
serialReschedule_LST-LSTU-4-0.5-0.05-100-nAdjacent	SR-LST-LSTU
serialReschedule_LSTA-EDD-4-0.15-0.05-100-idealizedReduction	SR-LSTA-EDD-LR-IR
serialReschedule_LSTA-EDD-4-0.15-0.05-100-nAdjacent	SR-LSTA-EDD-LR
serialReschedule_LSTA-EDD-4-0.5-0.05-100-nAdjacent	SR-LSTA-EDD
serialReschedule_LSTA-LST-4-0.15-0.05-100-idealizedReduction	SR-LSTA-LST-LR-IR
serialReschedule_LSTA-LST-4-0.15-0.05-100-nAdjacent	SR-LSTA-LST-LR
serialReschedule_LSTA-LST-4-0.5-0.05-100-nAdjacent	SR-LSTA-LST
serialReschedule_LSTA-LSTU-4-0.15-0.05-100-idealizedReduction	SR-LSTA-LSTU-LR-IR
serialReschedule_LSTA-LSTU-4-0.15-0.05-100-nAdjacent	SR-LSTA-LSTU-LR
serialReschedule_LSTA-LSTU-4-0.5-0.05-100-nAdjacent	SR-LSTA-LSTU
serialReschedule_LSTU-EDD-4-0.15-0.05-100-idealizedReduction	SR-LSTU-EDD-LR-IR
serialReschedule_LSTU-EDD-4-0.15-0.05-100-nAdjacent	SR-LSTU-EDD-LR
serialReschedule_LSTU-EDD-4-0.5-0.05-100-nAdjacent	SR-LSTU-EDD
serialReschedule_LSTU-EDD-4-0.5-0.05-100-nAdjacent-rollingHorizon20	SR-LSTU-EDD-IB
serialReschedule_LSTU-EDD-4-0.5-0.15-100-nAdjacent	SR-LSTU-EDD-GC
serialReschedule_LSTU-LST-4-0.15-0.05-100-idealizedReduction	SR-LSTU-LST-LR-IR
serialReschedule_LSTU-LST-4-0.15-0.05-100-nAdjacent	SR-LSTU-LST-LR
serialReschedule_LSTU-LST-4-0.5-0.05-100-nAdjacent	SR-LSTU-LST
serialReschedule_LSTU-LST-4-0.5-0.05-100-nAdjacent-rollingHorizon20	SR-LSTU-LST-IB
serialReschedule_LSTU-LSTA-4-0.15-0.05-100-idealizedReduction	SR-LSTU-LSTA-LR-IR
serialReschedule_LSTU-LSTA-4-0.15-0.05-100-nAdjacent	SR-LSTU-LSTA-LR
serialReschedule_LSTU-LSTA-4-0.5-0.05-100-nAdjacent	SR-LSTU-LSTA
serialReschedule_LSTU-LSTA-4-0.5-0.15-100-nAdjacent	SR-LSTU-LSTA-GC

serialReschedule_LWKR-GRD-4-0.15-0.05-100-nAdjacent	SR-LWKR-GRD-LR
serialReschedule_LWKR-LST-4-0.15-0.05-100-nAdjacent	SR-LWKR-LST-LR
serialReschedule_LWKR-MWKR-4-0.15-0.05-100-nAdjacent	SR-LWKR-MWKR-LR
serialReschedule_MWKR-LST-4-0.15-0.05-100-nAdjacent	SR-MWKR-LST-LR
parallelReschedule_EDD-LST-4-0.5-0.05-100-nAdjacent	PR-EDD-LST
parallelReschedule_EDD-LST-4-0.5-0.05-100-nAdjacent-rollingHorizon20	PR-EDD-LST-IB
parallelReschedule_EDD-LSTA-4-0.5-0.05-100-nAdjacent	PR-EDD-LSTA
parallelReschedule_EDD-LSTA-4-0.5-0.05-100-nAdjacent-rollingHorizon20	PR-EDD-LSTA-IB
parallelReschedule_EDD-LSTU-4-0.15-0.15-100-nAdjacent	PR-EDD-LSTU-RR
parallelReschedule_EDD-LSTU-4-0.15-0.15-100-nAdjacent-rollingHorizon20	PR-EDD-LSTU-RR-IB
parallelReschedule_EDD-LSTU-4-0.5-0.05-100-nAdjacent	PR-EDD-LSTU
parallelReschedule_EDD-LSTU-4-0.5-0.05-100-nAdjacent-rollingHorizon20	PR-EDD-LSTU-IB
parallelReschedule_LST-LSTA-4-0.5-0.05-100-nAdjacent	PR-LST-LSTA
parallelReschedule_LST-LSTA-4-0.5-0.05-100-nAdjacent-rollingHorizon20	PR-LST-LSTA-IB
parallelReschedule_LSTU-EDD-4-0.15-0.15-100-nAdjacent	PR-LSTU-EDD-RR
parallelReschedule_LSTU-EDD-4-0.15-0.15-100-nAdjacent-rollingHorizon20	PR-LSTU-EDD-RR-IB
parallelReschedule_LSTU-EDD-4-0.5-0.05-100-nAdjacent	PR-LSTU-EDD
parallelReschedule_LSTU-EDD-4-0.5-0.05-100-nAdjacent-rollingHorizon20	PR-LSTU-EDD-IB
parallelReschedule_LSTU-LST-4-0.15-0.15-100-nAdjacent	PR-LSTU-LST-RR
parallelReschedule_LSTU-LST-4-0.15-0.15-100-nAdjacent-rollingHorizon20	PR-LSTU-LST-RR-IB
parallelReschedule_LSTU-LST-4-0.5-0.05-100-nAdjacent	PR-LSTU-LST
parallelReschedule_LSTU-LST-4-0.5-0.05-100-nAdjacent-rollingHorizon20	PR-LSTU-LST-IB
parallel_EDD	P-EDD
parallel_LST	P-LST
parallel_LSTA	P-LSTA
parallel_LSTU	P-LSTU
serial_EDD	S-EDD
serial_EDD-rollingHorizon20	S-EDD-IB
serial_FCFS	S-FCFS
serial_LST	S-LST
serial_LST-rollingHorizon20	S-LST-IB
serial_LSTA	S-LSTA
serial_LSTA-rollingHorizon20	S-LSTA-IB
serial_LSTU	S-LSTU
serial_LSTU-rollingHorizon20	S-LSTU-IB

C Experimental results

Table C1: Results on Grid 1 with 750 average daily arrivals

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LST-LR-IR	3334.6	1.4	0.1	3794.3	998.4	4273.5	7
SR-EDD-LST-LR	0.0	0.0	0.0	3787.4	1007.8	8261.8	5
SR-EDD-LST	0.0	0.0	0.0	3787.4	1007.8	21399.5	5
SR-EDD-LST-GC	0.0	0.0	0.0	3787.4	1007.8	22323.5	5
SR-EDD-LSTA-LR-IR	3334.6	1.4	0.1	3794.3	998.4	4186.8	7
SR-EDD-LSTA-LR	0.0	0.0	0.0	3787.4	1007.8	9218.0	5
SR-EDD-LSTA	0.0	0.0	0.0	3787.4	1007.8	22162.5	5
SR-EDD-LSTA-GC	0.0	0.0	0.0	3787.4	1007.8	22906.1	5
SR-EDD-LSTU-LR-IR	3334.6	1.4	0.1	3794.3	998.4	4077.9	7
SR-EDD-LSTU-LR	0.0	0.0	0.0	3787.4	1007.8	8742.0	5
SR-EDD-LSTU	0.0	0.0	0.0	3787.4	1007.8	21030.1	5
SR-EDD-LSTU-GC	0.0	0.0	0.0	3787.4	1007.8	21934.7	5
SR-LST-EDD-LR-IR	0.0	0.0	0.0	3794.3	998.4	4606.3	7
SR-LST-EDD-LR	0.0	0.0	0.0	3787.4	1007.8	10103.3	5
SR-LST-EDD	0.0	0.0	0.0	3787.4	1007.8	22121.7	5
SR-LST-EDD-GC	0.0	0.0	0.0	3787.4	1007.8	20923.6	5
SR-LST-LSTA-LR-IR	0.0	0.0	0.0	3794.3	998.4	4372.1	7
SR-LST-LSTA-LR	0.0	0.0	0.0	3787.4	1007.8	10032.9	5
SR-LST-LSTA	0.0	0.0	0.0	3787.4	1007.8	23587.9	5
SR-LST-LSTU-LR-IR	0.0	0.0	0.0	3794.3	998.4	4150.6	7

SR-LST-LSTU-LR	0.0	0.0	0.0	3787.4	1007.8	9736.9	5
SR-LST-LSTU	0.0	0.0	0.0	3787.4	1007.8	23500.9	5
SR-LSTA-EDD-LR-IR	187.5	0.0	0.0	3794.4	998.3	5939.4	7
SR-LSTA-EDD-LR	127.2	0.0	0.0	3787.2	1008.0	11783.5	5
SR-LSTA-EDD	118.9	0.0	0.0	3787.2	1008.0	27603.0	5
SR-LSTA-EDD-GC	120.8	0.0	0.0	3787.2	1008.0	28722.1	5
SR-LSTA-LST-LR-IR	187.5	0.0	0.0	3794.4	998.3	5536.3	7
SR-LSTA-LST-LR	127.2	0.0	0.0	3787.2	1008.0	10587.5	5
SR-LSTA-LSTU-LR-IR	187.5	0.0	0.0	3794.4	998.3	5025.0	7
SR-LSTA-LSTU-LR	127.2	0.0	0.0	3787.2	1008.0	12306.9	5
SR-LSTU-EDD-LR-IR	0.0	0.0	0.0	3794.3	998.4	4981.3	7
SR-LSTU-EDD-LR	7.6	0.0	0.0	3787.4	1007.8	10513.8	5
SR-LSTU-EDD	0.0	0.0	0.0	3787.4	1007.8	24184.0	5
SR-LSTU-EDD-GC	0.0	0.0	0.0	3787.4	1007.8	25291.3	5
SR-LSTU-LST-LR-IR	0.0	0.0	0.0	3794.3	998.4	4813.0	7
SR-LSTU-LST-LR	10.2	0.0	0.0	3787.4	1007.8	8992.6	5
SR-LSTU-LSTA-LR-IR	0.0	0.0	0.0	3794.3	998.4	4250.7	7
SR-LSTU-LSTA-LR	10.2	0.0	0.0	3787.4	1007.8	11102.2	5
PR-EDD-LSTU	0.0	0.0	0.0	3787.4	1007.8	17935.4	5
PR-EDD-LSTU-IB	0.0	0.0	0.0	3787.4	1007.8	6797.2	5
PR-LSTU-EDD	0.0	0.0	0.0	3787.4	1007.8	17287.1	5
PR-LSTU-EDD-IB	0.0	0.0	0.0	3787.4	1007.8	6662.3	5
PR-LSTU-LST	0.0	0.0	0.0	3787.4	1007.8	18802.9	5
PR-LSTU-LST-IB	0.0	0.0	0.0	3787.4	1007.8	7056.9	5
P-EDD	5267.9	2.6	0.1	3787.4	1007.8	4111.9	5
P-LST	0.0	0.0	0.0	3787.4	1007.8	4100.7	5
P-LSTA	323.1	0.1	0.0	3787.6	1007.6	6069.5	5
P-LSTU	7.3	0.0	0.0	3787.4	1007.8	4407.8	5
S-EDD	5298.2	2.7	0.1	3787.4	1007.8	3922.3	5
S-LST	0.0	0.0	0.0	3787.4	1007.8	4221.9	5
S-LSTA	127.2	0.0	0.0	3787.2	1008.0	5500.0	5
S-LSTU	10.2	0.0	0.0	3787.4	1007.8	4358.8	5

Table C2: Results on Grid 1 with 1125 average daily arrivals

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LST-LR-IR	16400.9	92.8	1.4	4311.0	3469.4	7761.5	7
SR-EDD-LST-LR	4905.3	33.6	0.7	4320.9	3465.3	17500.4	15
SR-EDD-LST-LR-MC	23977.4	436.0	6.7	4347.0	3427.0	36026.8	1
SR-EDD-LST-MR	26843.8	510.0	7.9	4350.0	3424.0	56791.1	1
SR-EDD-LST	2409.5	32.4	0.6	4322.1	3464.2	46263.3	15
SR-EDD-LST-IB	2887.6	35.8	0.7	4320.9	3465.3	13861.6	15
SR-EDD-LST-GC	3704.1	71.7	1.1	4311.9	3468.6	42754.2	7
SR-EDD-LST-MI-LR	23977.4	413.9	6.5	4340.0	3434.0	47433.0	1
SR-EDD-LSTA-LR-IR	16400.9	92.8	1.4	4311.0	3469.4	6987.8	7
SR-EDD-LSTA-LR	4662.8	32.7	0.6	4320.9	3465.3	19262.2	15
SR-EDD-LSTA-RR	5960.4	95.8	1.6	4319.6	3459.6	17864.8	5
SR-EDD-LSTA-RR-IB	8574.9	86.6	1.4	4316.4	3462.8	5735.2	5
SR-EDD-LSTA	2096.2	30.6	0.6	4322.3	3464.0	48698.7	15
SR-EDD-LSTA-IB	2406.0	38.2	0.7	4321.7	3464.6	11510.0	15
SR-EDD-LSTA-GC	3899.0	73.0	1.3	4312.7	3467.7	48036.6	7
SR-EDD-LSTU-LR-IR	16400.9	92.8	1.4	4311.0	3469.4	7161.9	7
SR-EDD-LSTU-LR	7537.6	82.6	1.4	4317.6	3471.0	19762.2	18
SR-EDD-LSTU-RR	5425.5	79.9	1.3	4317.8	3461.4	16965.5	5
SR-EDD-LSTU-RR-IB	9099.0	114.3	1.7	5111.8	4069.2	5484.9	6
SR-EDD-LSTU	2290.3	32.4	0.6	4321.6	3464.7	47212.8	15
SR-EDD-LSTU-IB	3232.2	29.9	0.5	4321.0	3465.3	11413.7	15
SR-EDD-LSTU-GC	3800.5	68.1	1.2	4311.9	3468.6	44307.7	7
SR-GRD-LST-LR	20251.8	999.8	16.3	4327.0	3447.0	137897.2	1
SR-LST-EDD-LR-IR	1912.5	100.8	2.0	4308.9	3471.6	8555.1	7
SR-LST-EDD-LR	2296.5	44.4	0.8	4322.0	3464.3	22078.4	15

SR-LST-EDD	3798.8	41.9	0.7	4321.3	3464.9	48208.6	15
SR-LST-EDD-GC	3816.3	77.8	1.3	4313.1	3467.3	49398.5	7
SR-LST-LSTA-LR-IR	1912.5	100.8	2.0	4308.9	3471.6	7576.2	7
SR-LST-LSTA-LR	1370.2	49.4	1.0	4320.3	3465.9	21161.1	15
SR-LST-LSTA	1159.0	44.1	1.0	4321.1	3465.2	48808.6	15
SR-LST-LSTA-IB	1482.5	48.9	1.1	4320.5	3465.7	10791.3	15
SR-LST-LSTU-LR-IR	1912.5	100.8	2.0	4308.9	3471.6	8201.1	7
SR-LST-LSTU-LR	1500.3	49.8	1.1	4320.1	3466.2	19430.7	15
SR-LST-LSTU	1255.0	51.5	1.0	4320.1	3466.1	48252.4	15
SR-LSTA-EDD-LR-IR	2315.5	100.4	2.0	4309.4	3471.0	10738.0	7
SR-LSTA-EDD-LR	1989.2	37.4	0.8	4321.4	3464.9	25711.1	15
SR-LSTA-EDD	6004.5	103.6	1.7	4347.0	3447.2	64867.8	5
SR-LSTA-LST-LR-IR	2315.5	100.4	2.0	4309.4	3471.0	9718.7	7
SR-LSTA-LST-LR	1342.4	41.8	0.9	4319.9	3466.3	23023.0	15
SR-LSTA-LST	1182.6	49.0	1.0	4321.3	3464.9	51611.4	15
SR-LSTA-LSTU-LR-IR	2315.5	100.4	2.0	4309.4	3471.0	9984.0	7
SR-LSTA-LSTU-LR	1245.9	48.0	1.0	4321.0	3465.3	23452.8	15
SR-LSTA-LSTU	3007.3	145.8	2.9	4346.2	3448.0	59529.4	5
SR-LSTU-EDD-LR-IR	1726.2	100.3	2.0	4308.9	3471.6	8935.9	7
SR-LSTU-EDD-LR	1991.8	34.4	0.7	4320.6	3465.7	23434.2	15
SR-LSTU-EDD	1988.9	31.5	0.6	4321.5	3464.7	48796.4	15
SR-LSTU-EDD-IB	2749.3	42.0	0.7	4322.2	3464.1	11963.7	15
SR-LSTU-EDD-GC	12813.4	211.3	3.8	4283.5	3486.0	79402.9	2
SR-LSTU-LST-LR-IR	1726.2	100.3	2.0	4308.9	3471.6	8119.3	7
SR-LSTU-LST-LR	1243.4	46.3	0.9	4320.5	3465.8	21591.7	15
SR-LSTU-LST	1252.9	51.2	1.0	4320.6	3465.7	49548.6	15
SR-LSTU-LST-IB	1171.8	47.6	1.0	4320.5	3465.8	11662.7	15
SR-LSTU-LSTA-LR-IR	1726.2	100.3	2.0	4308.9	3471.6	8459.4	7
SR-LSTU-LSTA-LR	1620.6	80.7	1.6	4330.3	3459.1	22976.2	9
SR-LSTU-LSTA	2974.4	154.2	2.8	4347.2	3447.0	60943.7	5
SR-LSTU-LSTA-GC	14211.8	800.7	15.1	4341.0	3433.0	111923.7	1
SR-LWKR-GRD-LR	239946.5	1529.5	7.3	4271.0	3503.0	20675.1	1
SR-LWKR-LST-LR	32021.5	765.1	10.8	4321.0	3453.0	65232.8	1
SR-LWKR-MWKR-LR	129151.3	1821.7	9.6	4265.0	3509.0	18028.4	1
SR-MWKR-LST-LR	20511.5	1036.2	18.4	4316.0	3458.0	230725.2	1
PR-EDD-LST	2371.8	31.1	0.5	4321.7	3464.5	43088.8	15
PR-EDD-LST-IB	3493.3	36.1	0.6	4321.7	3464.6	12455.6	15
PR-EDD-LSTA	2110.3	30.2	0.5	4321.2	3465.1	45955.4	15
PR-EDD-LSTA-IB	2424.8	34.6	0.6	4321.5	3464.8	13036.3	15
PR-EDD-LSTU-RR	5454.9	81.8	1.3	4320.0	3459.2	13670.6	5
PR-EDD-LSTU-RR-IB	8162.0	89.6	1.4	4318.6	3460.6	4519.9	5
PR-EDD-LSTU	1777.4	26.0	0.5	4321.7	3464.6	44395.5	15
PR-EDD-LSTU-IB	2543.6	37.4	0.6	4321.9	3464.4	12490.3	15
PR-LST-LSTA	1023.1	40.8	0.9	4321.1	3465.1	47939.6	15
PR-LST-LSTA-IB	1233.8	41.8	0.9	4320.9	3465.4	12183.6	15
PR-LSTU-EDD-RR	5498.5	101.6	2.2	4321.0	3458.2	16056.7	5
PR-LSTU-EDD-RR-IB	5600.7	103.4	2.2	4318.0	3461.2	5592.1	5
PR-LSTU-EDD	1730.0	28.8	0.5	4322.2	3464.1	45722.9	15
PR-LSTU-EDD-IB	2244.1	29.1	0.6	4320.7	3465.6	13418.8	15
PR-LSTU-LST-RR	2912.1	148.4	2.9	4317.8	3461.4	14116.6	5
PR-LSTU-LST-RR-IB	3028.9	162.7	3.0	4316.6	3462.6	4849.0	5
PR-LSTU-LST	992.9	50.6	1.0	4320.4	3465.9	44580.4	15
PR-LSTU-LST-IB	1264.5	50.2	1.0	4320.9	3465.3	13266.4	15
P-EDD	15360.3	57.4	0.9	4319.2	3467.1	6997.2	15
P-LST	1178.1	35.3	0.9	4320.4	3465.9	7420.1	15
P-LSTA	1681.2	41.6	1.0	4320.7	3465.6	9439.9	15
P-LSTU	809.9	42.6	0.9	4321.5	3464.7	8105.1	15
S-EDD	16156.5	67.3	1.1	4320.2	3466.1	7250.3	15
S-EDD-IB	16944.3	76.3	1.2	4321.1	3465.2	2422.8	15
S-FCFS	31397.1	1119.6	11.6	4263.2	3519.5	3092.2	5
S-LST	1371.9	48.8	1.1	4320.3	3466.0	8323.9	15
S-LST-IB	1696.8	66.8	1.2	4319.0	3467.3	1786.7	15

S-LSTA	1157.5	46.9	1.0	4320.5	3465.7	10038.1	15
S-LSTA-IB	1103.7	48.8	1.0	4320.9	3465.4	2364.2	15
S-LSTU	915.3	46.9	1.0	4320.3	3466.0	8572.7	16
S-LSTU-IB	1106.4	53.3	1.0	4321.1	3465.2	1354.8	15

Table C3: Results on Grid 1 with 1125 average daily arrivals which charge only at fixed rates of 9 kW

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LSTA	66453.2	25014.0	99.8	3483.6	4295.6	126078.9	5
SR-EDD-LSTA-IB	64785.3	24759.6	99.6	3488.4	4290.8	13478.5	5
SR-EDD-LSTU	65028.6	24825.8	99.7	3489.0	4290.2	133900.4	5
SR-EDD-LSTU-IB	64102.5	24667.5	99.7	3491.8	4287.4	13584.6	5
PR-EDD-LSTU	65880.3	24955.9	99.5	3484.8	4294.4	140508.6	5
PR-EDD-LSTU-IB	64115.0	25162.9	99.8	3484.2	4295.0	11690.7	5
PR-LSTU-EDD	63489.9	24621.0	99.5	3494.4	4284.8	330432.5	5
PR-LSTU-EDD-IB	65084.4	24141.5	98.1	3512.8	4266.4	32186.8	5
PR-LSTU-LST	43284.6	24942.5	99.9	3515.2	4264.0	134910.3	5
PR-LSTU-LST-IB	46674.5	25170.5	99.9	3500.0	4279.2	12182.3	5
P-EDD	62744.7	24903.8	99.7	3480.8	4298.4	7059.3	5
P-LSTU	43284.6	24942.5	99.9	3515.2	4264.0	7426.3	5
S-EDD	62744.7	24903.8	99.7	3480.8	4298.4	26646.4	5
S-FCFS	74014.8	23945.1	74.7	3507.2	4272.0	10046.0	5

Table C4: Results on Grid 2 with 1125 average daily arrivals

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LST-LR	103614.3	48607.6	98.1	4580.0	3212.0	617568.0	3
SR-EDD-LSTA	108715.6	49577.8	97.4	4574.8	3221.2	1625104.9	6
SR-EDD-LSTA-IB	108989.9	49300.0	97.5	4586.8	3209.2	93757.6	6
SR-EDD-LSTU-LR	103661.3	48790.1	97.8	4510.1	3162.5	418294.7	8
SR-EDD-LSTU	107356.6	49685.0	97.4	4576.0	3220.0	1666256.3	6
SR-EDD-LSTU-IB	110206.9	49393.6	97.5	4588.0	3208.0	97720.9	6
PR-EDD-LSTU	111858.5	49240.1	98.5	4589.3	3206.7	1609724.3	6
PR-EDD-LSTU-IB	110028.2	49303.8	98.5	4589.8	3206.2	97267.7	6
PR-LSTU-EDD-IB	110210.6	49145.9	95.8	4588.8	3207.2	299015.4	6
PR-LSTU-LST-IB	87223.2	50440.3	96.6	4580.5	3215.5	252475.2	6
P-EDD	112715.1	49031.4	98.5	4600.0	3196.0	26086.1	6
P-LST	82746.0	50486.5	99.3	4567.8	3228.2	25576.4	6
P-LSTA	83936.2	50272.6	99.5	4582.5	3213.5	26675.6	6
P-LSTU	80155.5	49818.2	99.4	4587.7	3208.3	32733.7	6
S-EDD	109012.6	49584.5	97.4	4573.0	3223.0	245441.6	6
S-LST	84722.6	50196.0	98.3	4577.2	3218.8	250599.5	6
S-LSTA	84103.3	50228.8	98.4	4574.7	3221.3	249130.9	6
S-LSTU	80439.0	50663.8	98.9	4558.7	3241.3	257208.4	3

Table C5: Results on Grid 3 with 1125 average daily arrivals

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LSTA	49262.0	8030.2	71.2	3892.3	3886.9	286444.1	6
SR-EDD-LSTA-IB	48431.8	8293.4	71.5	3897.2	3882.0	30896.1	6
SR-EDD-LSTU	50821.7	8251.2	71.4	3893.4	3885.8	272129.6	5
SR-EDD-LSTU-IB	47937.8	7884.5	71.0	3909.4	3869.8	29920.6	5
PR-EDD-LSTU	48924.3	8453.1	72.3	3877.0	3902.2	272433.7	5
PR-EDD-LSTU-IB	51176.2	8237.5	71.9	3903.0	3876.2	28719.8	5
PR-LSTU-EDD	49145.0	8064.2	71.0	3889.2	3890.0	839378.0	5
PR-LSTU-EDD-IB	50508.2	8298.3	71.5	3886.6	3892.6	45399.7	5
PR-LSTU-LST	21633.4	8844.0	81.6	3864.2	3915.0	551593.4	5
PR-LSTU-LST-IB	22687.3	8878.8	80.4	3878.4	3900.8	27580.7	5
P-EDD	50319.1	8047.7	70.5	3915.0	3864.2	7202.9	5
P-LSTU	19899.0	8505.0	82.4	3886.4	3892.8	8994.0	5
S-EDD	53371.3	8397.6	70.4	3894.6	3884.6	31455.4	5
S-LST	22745.2	9189.5	81.6	3853.2	3926.0	31380.2	5
S-LSTU	18408.7	8607.6	82.8	3884.2	3895.0	33215.9	5

Table C6: Results on Grid 3 with 1500 average daily arrivals

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LSTA	53649.9	9802.6	79.6	3927.7	6532.7	284237.8	3
SR-EDD-LSTA-IB	53307.6	9896.3	80.5	3943.4	6519.6	26725.7	5
SR-EDD-LSTU	55145.0	10085.6	80.8	3907.2	6550.0	293378.4	5
SR-EDD-LSTU-IB	55122.2	9479.2	79.5	3924.0	6533.2	28084.7	5
PR-EDD-LSTU	55020.7	9672.7	80.4	3933.6	6523.6	284391.4	5
PR-EDD-LSTU-IB	52916.4	9369.1	80.3	3933.4	6523.8	29272.5	5
PR-LSTU-EDD	56571.0	9771.7	79.7	3930.6	6526.6	734411.4	5
PR-LSTU-EDD-IB	55663.3	9651.4	78.6	3937.0	6520.2	50565.0	5
PR-LSTU-LST	21837.5	10372.6	89.1	3895.0	6562.2	599248.6	5
PR-LSTU-LST-IB	22217.4	10128.9	88.0	3881.0	6576.2	30648.4	5
P-EDD	54034.5	9372.4	78.2	3936.6	6520.6	7391.6	5
P-LSTU	19967.9	10300.4	90.5	3910.2	6547.0	9231.2	5
S-EDD	52610.1	9457.8	78.2	3937.2	6520.0	35210.2	5
S-LST	21237.8	10216.8	88.8	3898.2	6559.0	37136.0	5
S-LSTU	20381.4	10324.3	89.9	3891.0	6566.2	37238.7	5

Table C7: Results on Grid 4 with 1500 average daily arrivals

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LSTA	66489.9	13656.6	66.5	5178.4	5278.8	443772.8	5
SR-EDD-LSTA-IB	66639.3	13699.7	66.9	5183.8	5273.4	35846.1	5
SR-EDD-LSTU	65582.4	13735.0	66.4	5171.8	5285.4	497858.2	5
SR-EDD-LSTU-IB	65182.6	13383.1	67.8	5175.6	5281.6	35718.3	5
PR-EDD-LSTU	65556.4	13455.7	67.8	5187.2	5270.0	505396.5	5
PR-EDD-LSTU-IB	63066.3	13651.5	67.9	5190.4	5266.8	35478.9	5
PR-LSTU-EDD	66529.8	13302.4	66.4	5202.8	5254.4	1172933.3	5
PR-LSTU-EDD-IB	62678.2	13570.5	67.8	5179.4	5277.8	87114.0	5
PR-LSTU-LST	35722.2	14107.3	72.5	5172.8	5284.4	1068309.4	5
PR-LSTU-LST-IB	37864.6	14200.4	74.4	5187.6	5269.6	51248.8	5
P-EDD	64884.1	13541.9	68.0	5179.6	5277.6	13852.0	5
P-LSTU	32909.2	14194.6	72.6	5184.0	5273.2	18082.4	5
S-EDD	64737.5	14053.0	67.5	5165.0	5292.2	56851.4	5

Table C8: Results on Grid 4 with 3000 average daily arrivals

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LSTA	64296.8	17153.5	87.7	5133.0	15768.3	650835.2	4
SR-EDD-LSTA-IB	63096.8	17768.9	87.5	5072.2	15835.2	34567.2	10
SR-EDD-LSTU	66913.4	19044.9	88.2	4976.0	15942.0	630238.2	1
SR-EDD-LSTU-IB	62719.3	17428.1	87.0	5094.8	15812.6	35236.2	5
PR-EDD-LSTU	72220.0	17984.2	90.5	5061.0	15857.0	664469.8	1
PR-EDD-LSTU-IB	63376.2	17528.9	87.8	5092.0	15815.4	37325.3	5
PR-LSTU-EDD-IB	66031.8	18004.0	85.4	5064.8	15842.6	85350.1	5
PR-LSTU-LST-IB	38078.0	17701.1	92.2	5057.8	15849.6	54623.5	5
P-EDD	63871.7	17504.9	86.6	5072.8	15834.6	13077.1	5
P-LSTU	36385.1	18061.4	95.2	5053.0	15854.4	16271.9	5
S-EDD	66129.4	18275.6	87.2	5064.2	15843.2	70879.0	5

Table C9: Results on Grid 5 with 1125 average daily arrivals

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LST-LR	25269.8	231.7	3.4	5618.2	2161.0	68658.6	5
SR-EDD-LSTA-IB	26464.2	247.3	3.6	5619.6	2159.6	19475.6	5
SR-EDD-LSTU-LR	25269.8	231.7	3.4	5618.2	2161.0	66725.4	9
SR-EDD-LSTU-IB	26464.2	247.3	3.6	5619.6	2159.6	19078.8	5
SR-LST-EDD-LR	21166.6	171.9	4.0	5620.6	2158.6	57226.5	5
SR-LST-LSTU-LR	21166.6	171.9	4.0	5620.6	2158.6	58917.5	5
SR-LSTU-EDD-LR	12007.8	229.6	5.5	5619.4	2159.8	66585.2	5
PR-EDD-LSTU-IB	26010.8	212.0	3.3	5620.6	2158.6	16135.1	5
PR-LSTU-EDD-IB	13595.0	199.2	5.0	5621.8	2157.4	15046.5	5
PR-LSTU-LST-IB	13595.0	199.2	5.0	5621.8	2157.4	17219.9	5
P-EDD	23984.6	208.3	3.1	5619.0	2160.2	43313.4	5
P-LST	20663.4	174.8	3.9	5617.6	2161.6	42792.3	5
P-LSTA	28191.9	322.0	8.2	5608.3	2168.0	52559.9	3
P-LSTU	14243.2	230.0	5.6	5620.0	2159.2	37556.4	5
S-EDD	25269.8	231.7	3.4	5618.2	2161.0	67306.5	5
S-LST	21166.6	171.9	4.0	5620.6	2158.6	69965.1	5
S-LSTU	14919.5	377.1	8.9	5587.7	2189.7	74100.1	3

Table C10: Results on Grid 5 with 1500 average daily arrivals

Strategy	max delay	mean delay	% delayed	served	non-served	runtime	runs
SR-EDD-LST-LR	45882.8	2854.6	33.2	5830.0	4627.2	144483.1	5
SR-EDD-LSTU-LR	45882.8	2854.6	33.2	5830.0	4627.2	146922.4	10
SR-LST-EDD-LR	38545.1	3719.8	49.7	5765.2	4692.0	121992.2	5
SR-LST-LSTU-LR	38545.1	3719.8	49.7	5765.2	4692.0	106153.2	5
P-EDD	44319.4	2836.0	33.1	5830.4	4626.8	55813.4	5
P-LST	33945.3	3565.5	49.2	5782.6	4674.6	54993.9	5
P-LSTA	33530.8	3818.5	53.7	5764.8	4692.4	53805.3	5
P-LSTU	25084.3	3654.3	52.8	5786.6	4670.6	50765.8	5
S-EDD	45882.8	2854.6	33.2	5830.0	4627.2	142564.6	5
S-LST	38545.1	3719.8	49.7	5765.2	4692.0	144077.7	5
S-LSTU	26930.0	4165.1	64.6	5788.0	4656.0	108551.1	1