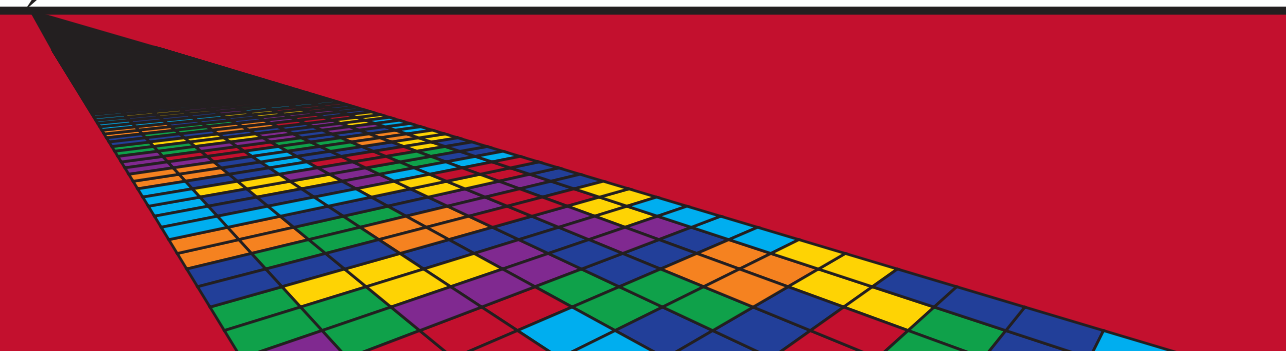
The Power of Scheduling: the Scheduling of Power

Roelof Jacob Jan Brouwer



The Power of Scheduling: the Scheduling of Power

Roelof Jacob Jan Brouwer

ISBN: 978-90-393-7685-0

DOI: https://doi.org/10.33540/2311

Printed by: Ipskamp Printing

This dissertation was printed on 100% recycled paper in an effort to minimize the environmental footprint.

The Power of Scheduling: the Scheduling of Power

Het Vermogen van Scheduling: Scheduling van Vermogen

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht
op gezag van de rector magnificus, prof.dr. H.R.B.M. Kummeling,
ingevolge het besluit van het college voor promoties
in het openbaar te verdedigen
op maandag 8 juli 2024 des ochtends te 10.15 uur

door

Roelof Jacob Jan Brouwer

geboren op 17 augustus 1994 te Rotterdam

Promotoren:

Prof. dr. H.L. Bodlaender

Dr. ir. J.M. van den Akker

Beoordelingscommissie:

Prof. dr. G.T. Barkema

Prof. dr. M. Gibescu

Dr. M.E. van Kooten Niekerk

Prof. dr. R. Leus

Prof. dr. M.M. de Weerdt

Contents

Acknowledgements

This thesis is the final product of six eventful years. I would not have reached this point without the help and support of numerous people. I want to use this opportunity to thank some of those people. Even if you are not mentioned explicitly, the fact that you are reading this paragraph probably means that this word of thanks is also meant for you. Thank you all for expressing interest in my work, providing support and helping me complete this lengthy process.

First and foremost, I would like to thank Marjan van den Akker. When I was writing my master's thesis under her supervision, seven years ago, I had not yet decided whether I wanted to pursue a PhD afterward. She pointed the call for a combined position at the central IT department out to me, and asked me to join a proposal by her and Frank Dignum to fill that position. The rest is history, as they say. She has supported me at all points along the way, and helped me to reorient the focus of my project after a difficult period. Without her support, I would not have completed this thesis.

Next, I want to acknowledge the significant contributions of my other co-authors, Han Hoogeveen and Emily van Huffelen, to the papers underlying many of the chapters in this thesis. Han is always able to ask the right questions to make me think about the approach we were developing. His feedback is always clear and extensive. I still have many previous drafts of papers with Han's handwritten comments lying around at home. Supervising the thesis of Emily was a very interesting experience for me. She nudged the research in a direction that I had not initially anticipated, thereby broadening my view on the subject. I also appreciate how she remained involved after the completion of her thesis, even though she did not want to pursue an academic career herself.

I would also like to thank my other promotor, Hans Bodlaender, for his help, especially in completing the final stretch of this marathon. In his feedback and during our meetings, he was always supportive, seemingly more confident in a positive result than I was myself. A special mention goes out to Frank Dignum, who was originally intended to be a co-supervisor for my project. I would like to thank him for his support in the initial stages of my PhD, and for his flexibility in changing his role when circumstances caused the project to go in a (partially) different direction than originally intended.

I want to thank Gerard Barkema, Madeleine Gibescu, Marcel van Kooten Niekerk, Roel Leus and Mathijs de Weerdt for joining the assessment committee and for their positive feedback on the manuscript.

Furthermore, I would like to thank all other colleagues at the department for the

fun conversations over lunch, and the interesting discussions on research topics. I have fond memories of the Algorithmic Game Evenings, especially when we were all working from home during the pandemic. I have enjoyed being part of the OR@UU group, which developed into a nice place to share work, and a fun group of people to visit conferences and spend coffee breaks with. Finally, the last group of people I want to specifically mention are those associated with the PhD-IT program. While our topics are diverse, we share the challenge of combining research with a job in the public sector. I am also grateful for the support Jet Haasbroek provided me in her role as PhD mentor and coach. Her positive attitude during our conversations was always encouraging.

I want to acknowledge the research program "Energie: Systeem Integratie en Big Data" with project number 647.003.005, which is financed by the Dutch Research Council (NWO), for financing the machine that was used for the computational experiments in Chapter 2, 3 and 5. I want to thank Laurens Stoop for allowing me to use it, and helping me do so. Additionally, the experiments presented in Chapter 4 were carried out on the Dutch national e-infrastructure with the support of SURF Cooperative.

I was able to pursue my PhD because of the funding provided by the Research IT program, managed by Menno Rasch, which allowed my to combine my research activities with my work as a Research Engineer at Information and Technology Services. I am very grateful to the late Folkert-Jan de Groot, my manager at ITS when I started, and Frank Heere, his successor, for providing me with the opportunity to combine these two roles, and for their flexibility regarding the way I divided my time over the different roles.

I have worked with many people over the past six years. I will refrain from naming everyone explicitly. But I have appreciated the conversations we had about my research, the interest you expressed in what I was doing and how I was doing, and the good working relationship we have had, and still have. It has been very interesting to witness the development of digital research support at the university, and in particular the research engineering team, over the past few years from the inside.

Then, there are many people left to thank, outside the professional context. I want to thank Thomas and my brother Kees for agreeing to be my paranymphs. I would like to thank my friends for sticking with me over the years. I am particularly grateful for the support of my family. My mother Heleen, who will always be willing to help me, even if I do not want her to; her husband Peter, who always knows how to make people smile; my grandmother Els, who regularly reminded me to finish my thesis as soon as possible. Finally, I am so happy that Hannah came into my life two years ago. I am grateful for her love, support and tolerance during a period of time that was already difficult for her without having me to deal with on top of everything.

Part I

Introduction

Contents

1

Introduction

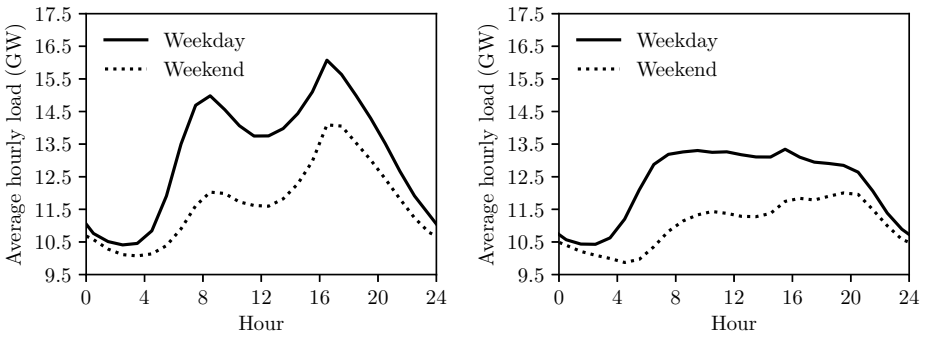
In the face of climate change, an energy transition is taking place, which affects the way the world produces and consumes energy. Electricity and heat generation are responsible for a large share of the world's total greenhouse gas emissions (World Resources Institute 2022). The electrification of heat demand and transportation is an important part of the global effort to reduce the carbon footprint of our energy use.

With the increasing reliance on electricity for our energy needs, the strain on the electrical grid increases as well. At the same time, the growing share of renewable sources in electricity generation changes the characteristics of the supply. These parallel developments require large investments in all levels of the electrical grid to accommodate the transition (Enexis 2024; Liander 2023; Stedin 2024; TenneT 2024a,b). Besides that, more efficient use of the grid is required.

It is essential to manage the supply and demand of electricity. Supply and demand should align as well as possible, both in time and location. We think *scheduling* approaches have a role to play in this. The efficient planning of scarce resources, which lies at the heart of operations research and scheduling, is a key issue here. In this thesis, we will consider the application of scheduling techniques to problems that are inspired by the challenges posed by the energy transition.

1.1. Energy transition

The main driver behind the energy transition is the commitment to reduce greenhouse gas emissions. A reduction of overall energy consumption contributes to this goal. Moreover, it requires a transition away from the use of fossil fuels, while the consumption of energy from renewable sources will grow. Especially for heating and transportation, fossil fuels are currently being used as the main source of energy, but increasingly a move towards electricity is taking



(a) January 2023 average

(b) July 2023 average

Figure 1.1: Average hourly demand in the Netherlands. Times are in Coordinated Universal Time (UTC). Data from ENTSO-E (2024).

place. This is exemplified by the increasing number of heat pumps installed, and the growth of the number of electric vehicles.

This development is clearly visible in the statistics on energy use in the Netherlands. While the total demand for energy is showing a downward trend (CBS 2023), the demand for electricity does not (CBS et al. 2024). This trend is expected to continue in the (near) future (PBL et al. 2022). An increasingly large share of our energy demand is met by renewable sources. In the Netherlands, this share rose from 13% in 2021 to 15% in 2022 (Brandenburg et al. 2023). If we only consider the generation of electricity, these percentages are even higher: 33.4% and 39.8%, respectively.

These developments come with a set of complicated challenges. They cause a change in the patterns of usage and generation. In Figure 1.1, we show the average electricity demand in the Netherlands for January and July. These patterns are representative of the average demand on winter and summer days, respectively. A coal- or gas-fired power plant can match these patterns closely. Some renewable energy sources, such as wind and solar power, that depend on the weather conditions for their production, cannot, however. See for example the production curve of solar panels in Figure 1.2. Whereas historically power production was largely controllable, this is no longer the case for an increasingly large share of the energy sources that are used to satisfy electricity demand. As a result, the peaks in electricity consumption and production do not match without intervention. The weather dependence of most renewable sources causes the supply to be uncertain and variable. Furthermore, the production of electricity is no longer limited to a few large power stations, but happens all across the grid. All of these factors together make clear that operating the electrical grid is becoming more and more challenging.

Grid operators are already forced to limit the number of new connections to the

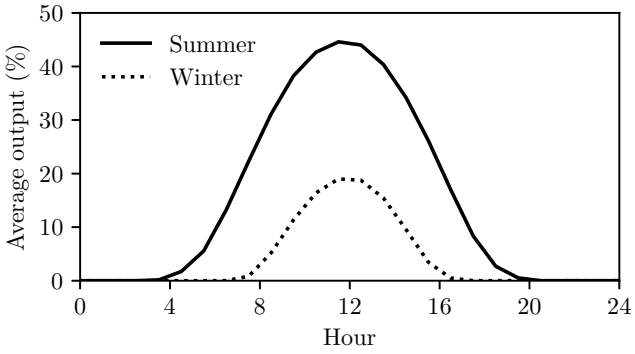
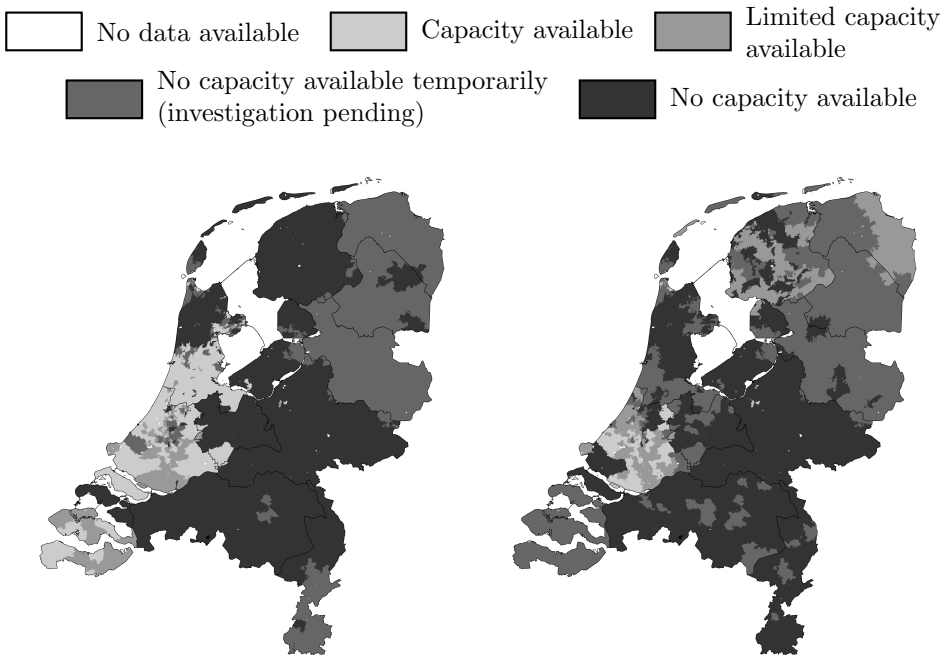


Figure 1.2: Predicted average hourly production of photovoltaic (PV) installations in the Netherlands in 2025 as a percentage of their peak power. Data from ENTSO-E (2018).



(a) Capacity for feeding into the grid **(b)** Capacity for extracting from the grid

Figure 1.3: Availability of capacity on the electricity grid managed by distribution system operators (DSOs) in the Netherlands for connections requiring at least 3x80A. Data from Netbeheer Nederland (2024).

grid that they can accommodate. See Figure 1.3 for an overview of these limitations in the Netherlands, in January 2024. Large investments in the expansion and adaptation of the electricity grid are unavoidable. This is an expensive operation, that takes a long time to complete.

Again, other efforts can and have to be made as well to align electricity supply and demand better, both on the (inter)national and the local level. One way to bridge the gap between supply and demand is storing electrical energy. However, the capacity for this is currently very limited. Storing electrical energy will be part of the solution, as storage capacity will increase in the future, but for reasons of efficiency, it remains preferable to use the electricity when it is generated, without the need for storage in between. Therefore, electricity supply and demand should be roughly equal over time. This should be the case at all times, at all points in the grid.

Traditionally, electricity demand is treated as a given, which has to be met by controlling the generation of electricity. Recently, approaches to control (part of) electricity demand have gained increased attention. *Demand response* aims to shift demand in time to better match the supply. Through the development of new technologies, it is also becoming more realistic to implement such approaches, for example by using smart appliances and energy management systems. As the supply becomes less controllable, due to the increasing share of intermittent renewable energy sources, more and more attention is directed towards controlling the demand with demand response.

The balancing of supply and demand not only guarantees stable operation of the grid on a global level, but may also reduce the need for upgrading cables on a local level.

1.2. Scheduling

Scheduling problems have been a prominent research topic for decades. The handbook on scheduling by Pinedo (2016) defines scheduling as follows:

> *"Scheduling is a decision-making process that is used on a regular basis in many manufacturing and services industries. It deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives."*

From this definition, we can extract a number of properties that characterize scheduling problems:

- Scheduling problems are *optimization problems*. Meaning that, for a given objective function, the goal is to find the solution that minimizes the value of that function.
- Scheduling problems concern the planning (scheduling) of tasks over time.
- Scheduling problems deal with the allocation of *resources* to these tasks.

From this general description, it should be clear that a wide range of scheduling problems exists. Examples range from the scheduling of processors in computing to the scheduling of shifts for personnel in hospitals.

1.2.1. Scheduling problems

Many classical scheduling problems, that are among the most widely studied, are based on a machine scheduling model. In these models, we need to find a schedule for the processing of a number of tasks (n) on a number of machines (m) that optimizes a given objective, while respecting any additional constraints imposed by the particular problem. To classify these problems, the three-field notation was introduced by Graham et al. (1979): $\alpha|\beta|\gamma$, where

- α represents the machine environment, e.g. 1 for a single machine.
- β represents the job characteristics present in the problem. E.g. r_j means that each job has a release date r_j, which is the earliest time it can be processed.
- γ represents the objective function that should be minimized. For example, $C_{\max}$ indicates that the maximum completion time (known as the *makespan*) should be minimized.

In the traditional machine scheduling problems, tasks are usually assigned to a single machine, and a machine can process a single task at a time. The concept of machines can be generalized to that of (cumulative) *resources*, where a task requires a given number of units of one or more resources to be completed. Multiple tasks can consume these resources simultaneously, as long as the total availability of that resource is not exceeded. This is the setting of the Resource Constrained Project Scheduling Problem (RCPSP). The RCPSP considers the scheduling of projects that each consist of a number of tasks that require resources to be completed. The three field notation has been extended for this type of problem as well, and the standard RCPSP is denoted by $PS|prec|C_{\max}$. This problem and its variants have been extensively studied in recent years. The surveys by Hartmann and Briskorn (2010, 2022) provide a good overview.

Taking one step further in the direction of problems encountered in the context of balancing electricity supply and demand, we can consider these resources as continuously-divisible, rather than divided in discrete units. The Cumulative Scheduling Problem (CuSP) introduced by Baptiste, Pape, and Nuijten (1999), which can be denoted by $PS, Co|r_j, d_j|-$, is an example which serves as the basis for the problems we will be studying in Chapter 2 and 3.

1.2.2. Solution approaches

For some classical scheduling problems, polynomial time algorithms exist. These algorithms are tailored to the specific problem and exploit problem-specific information to find the optimal solution in the fewest steps possible. A well-known example is Smith's rule for $1||\sum w_j C_j$, which puts job starts in order of decreasing ratio of weight to processing time (Smith 1956). Many scheduling problems, however, are NP-hard. Therefore, other solution methods have been developed.

There is a large range of possible solution methods for solving these problems. Broadly, they can be divided into three groups: exact methods, heuristic methods, and metaheuristics. The first one includes techniques such as dynamic programming, mixed-integer linear programming and branch-and-bound. All of these, given enough time, guarantee that the optimal solution is found. Heuristic methods do not, but aim to find good solutions more quickly. A heuristic usually 'cuts some corners', using problem-specific information, to find a solution that is 'good enough'. Metaheuristics, such as local search or evolutionary algorithms, are higher level procedures that aim to quickly find good solutions, without exploring the entire search space. Metaheuristics may exploit problem-specific information when evaluating potential solutions or identifying promising areas of the search space, but the general procedure is applicable to a wide range of problems. Although heuristics and metaheuristics typically do not guarantee the global optimality of a solution, they give good solutions in practice. Certain metaheuristics are even proven to converge to the optimal solution, under the right conditions.

Approaches that are often applied to scheduling problems in the literature, in addition to polynomial time algorithms, include:

- Exact approaches: mathematical programming (e.g. mixed-inter linear programming, see Chapter 2 and 3), dynamic programming and branch-and-bound;
- Heuristic methods: schedule generation schemes (see Chapter 5) and other priority rule based approaches;
- Metaheuristics: simulated annealing (see Chapter 2 and 3), tabu search and genetic algorithms.

Exact approaches may be preferred, as they guarantee the global optimality of a solution. If a problem is known to be NP-hard, however, exact methods often fail to produce good results for large instances. In practice, well-developed (meta)heuristics often provide the best results for difficult scheduling problems.

1.3. Scheduling approaches for energy-related problems

Some scheduling problems are explicitly defined as energy-related problems. For example, the unit commitment problem, which can be considered a classical operations research problem, explicitly deals with the scheduling and dispatch of electric power generation resources. Many approaches to solve variants of this problem have been proposed, most of which use mathematical programming. Several of these approaches have been applied to real-world problems. The case study we discuss in Chapter 4 is a good example of this.

However, scheduling problems that are not originally defined from an energy-related application might be applicable to such problems as well. From the scheduling problems discussed near the end of Section 1.2.1, the step to certain energy-related planning problems is not large. In particular, when considering demand

response, we can treat the amount of available electricity (either limited by the capacity of a cable, or by the availability of supply) as a renewable resource. Although this inevitably simplifies some of the characteristics of electricity networks, it is an adequate model for planning electricity use on a medium timescale.

As mentioned previously, electricity demand is traditionally considered to be entirely *uncontrollable*. This means that the demand at any given time is a given quantity that has to be matched by controlling the generation of electricity. In recent times, however, an increasingly large share of the load may be considered to be *controllable*, meaning that some measure of control can be exercised to change the load profile. With the right technology, many types of load, such as the charging of electric vehicles, the heating of an office building or the cooling of a storage space, can be shifted in time. Demand response aims to use this flexibility to bring the demand better in alignment with the supply. Often this is used to take advantage of low electricity prices when they occur during certain parts of the day, but it can also be used for other purposes. Let us consider an example inspired by the current limitations on new grid connections in the Netherlands.

Suppose that the development of an industrial area causes its electricity use to grow beyond the limits of its current connection to the grid, but it cannot be upgraded due to the unavailability of additional capacity in the grid. The companies that make use of this shared connection could consider coordinating their electricity usage in such a way that it is spread more evenly across the week, bringing the peak usage down to a level below the capacity of the current connection. For example, one company might choose to charge their electric delivery vans at a different time, and another could shift the times its cooling systems are switched on to bring down the temperature in their cold storage. While there might be plenty of room for electricity demand to be shifted across time, the coordination is a difficult task. This is where scheduling comes into play.

Part of the electricity demand consists of tasks that have a window during which they can be completed, that can be defined by a release time and a deadline, such as the charging of electric vehicles. Some tasks may be able to scale their electricity consumption up or down to match the availability better. The range of feasible consumption rates is usually limited, however. This can be captured, to a large extent, by using a lower and an upper bound to define this range. Taking all these constraints and opportunities into account requires careful consideration. The problem we study in Chapters 2 and 3 focuses on exactly this type of problem. Managing the charging of large numbers of electric vehicles on a parking lot near an office or apartment building, for example, would also be an excellent application of this approach, as we discuss in Chapter 5.

The literature dealing with controlling electricity demand is scattered across a wide range of fields, ranging from electrical engineering to economics. However, scheduling approaches have not been commonly used to solve this type of problem in the past. Typically, scheduling problems are associated with centralized decision-making, while distributed approaches have often been given priority due to legal constraints and privacy considerations. Most of the approaches in the literature,

therefore, seem to focus on distributed decision-making, where operators use price signals to incentivize agents in the network to make certain decisions.

However, the use of scheduling approaches will be no more likely to threaten the privacy or autonomy of a consumer than the implementation of any other automated approach. At the same time, solutions using scheduling techniques can provide a transparent mechanism to make automated decisions. Scheduling approaches make explicit how the desires of an end-user are considered in the decision-making process. Especially when dealing with the energy needs of households, clear up-front agreements on the implementation of such approaches are necessary, that consider privacy concerns and establish boundaries. This is true for any form of automated decision-making. Having said that, our view is that scheduling approaches can significantly contribute to the future of energy management.

We hope to provide a useful framework for thinking about applications of the scheduling problems discussed in this thesis and for dealing with this type of issues.

1.4. Thesis outline

The core of this thesis consists of two parts, each containing two chapters. The first part explores a general class of resource-constrained scheduling problems, and presents a framework for finding good solutions to these problems. The second part deals with two specific examples of electricity-related scheduling problems, for which we developed and tested an approach using scheduling techniques.

In Chapter 2, we investigate a problem inspired by the scheduling of flexible demand in electricity networks. We consider a variant of the Continuous Energy-Constrained Scheduling Problem (CECSP) introduced by Nattaf, Artigues, and Lopez (2014). A set of jobs has to be processed on a continuous, shared resource, such that the total weighted completion time of all jobs is minimized. Each job needs to be assigned a start time, completion time and a resource consumption profile. A job can only start after its release time, and has to be completed before its deadline. In the meantime, it needs to consume an amount of resource equal to its resource requirement, while its consumption rate has to stay within given lower and upper bounds. We use an event-based model. Each job has two associated events, a start and a completion event, that must be scheduled. The resource consumption profile is decided for the intervals defined by these events. A mixed-integer linear programming (MILP) formulation is presented, as well as a hybrid local search approach. We show that the hybrid local search approach matches the exact MILP formulation in solution quality for small instances, and is able to find a feasible solution for larger instances in reasonable time.

We continue by considering a generalization of the CECSP in Chapter 3, which we call the General Continuous Energy-Constrained Scheduling Problem (GCECSP).

We extend our approach from Chapter 2 and present a solution framework for problems in this class. We apply a combination of local search and mathematical programming. At the core of this framework is the decomposition in two parts, allowed by the event-based model: 1) determining the order of events and 2) finding the event times and resource consumption profiles. We use the CECSP with step-wise cost functions as an example to illustrate the effectiveness of our framework. We exploit that, for this specific problem, the cost of a solution can be computed on the basis of the order of events alone, and we present bounds to assess the feasibility of an event order more efficiently, where possible. We show the effectiveness of our approach in comparison with a MILP formulation and argue the broad applicability of the framework.

In Chapter 4 the operational management of microgrids is studied. This chapter considers the balancing of electricity supply and demand under uncertainty by managing the production and storage of electricity. Microgrids are clusters of components that each produce, store or consume energy. The combination of these components allow microgrids to operate as more or less independent of the main grid. While some microgrids have a (limited) connection to the main electricity grid, in this chapter we consider islanded microgrids, with no connection to the main grid at all. We develop two optimization models for planning the operation of diesel generators and battery storage units, while dealing with uncertainty in forecasts of load and solar power. The first one is a deterministic mixed-inter linear programming model that reserves part of the available battery capacity as a safety margin. The second one is a multi-stage stochastic optimization model that models the uncertainty using a scenario tree based on the load and solar power forecasts. These models are applied to a case study in the Netherlands, using a number of re-planning strategies. We identify features of microgrids that are important for determining the success of the developed approaches and show the effectiveness of using re-planning strategies to reduce the frequency of schedule generation.

In Chapter 5, the attention is shifted back towards controlling the demand for electrical energy, rather than the supply. A network of parking lots is considered, each with a given number of charging stations for electric vehicles (EVs). Some of the parking lots are covered by a roof with solar panels on it. The demand that can be served at each parking lot is limited by the capacity of the cables connecting them to the grid. EVs arrive at the parking lots according to a known distribution. As soon as a vehicle arrives, we learn its desired departure time, the amount of electrical energy it needs to charge its battery before that time, and the range of rates that it can be charged at. The actual departure time of an EV can be delayed if it has not finished charging in time for its desired departure. We use data collected in the city of Utrecht for the distribution of arrival times, connection times and charging volumes. The aim is to minimize the total delay, i.e. the summed difference between the desired departure time and the actual departure time, across all vehicles. We present a novel approach, based on an online variant of well-known schedule generation schemes. A number of variants of this approach are evaluated using a discrete event simulation. With this, we

show that applying scheduling approaches increases the number of EVs that can be charged at the site and reduces the average delay. Furthermore, we argue the importance of considering aspects of the grid layout in electricity networks and show the benefits of using flexible charging rates.

Finally, Chapter 6 summarizes the findings of the previous chapters and discusses some interesting avenues for future research.

References

Baptiste, P., C. L. Pape, and W. Nuijten (1999). "Satisfiability tests and time-bound adjustments for cumulative scheduling problems". *Annals of Operations Research* 92, pp. 305–333. DOI: 10.1023/A:1018995000688.

Brandenburg, K., S. Brummelkamp, H. S. Chan, L. Geurts, M. J. Linders, G. Muller, and R. Segers (Oct. 2023). *Hernieuwbare energie in Nederland 2022*. Centraal Bureau voor de Statistiek (CBS). URL: https://www.cbs.nl/nl-nl/longread/rapportages/2023/hernieuwbare-energie-in-nederland-2022.

CBS (July 2023). *Laagste energieverbruik in Nederland sinds 1990*. Centraal Bureau voor de Statistiek (CBS). URL: https://www.cbs.nl/nl-nl/nieuws/2023/27/laagste-energieverbruik-in-nederland-sinds-1990.

CBS, PBL, RIVM, and WUR (2024). *Aanbod en verbruik van elektriciteit, 1990-2022 (indicator 0020, versie 27, 24 July 2023)*. Compendium voor de Leefomgeving (CLO). URL: https://www.clo.nl/indicatoren/nl002027-aanbod-en-verbruik-van-elektriciteit-1990-2022.

Enexis (2024). *Investeringsplan 2024 Enexis netbeheer*. URL: https://www.enexis.nl/over-ons/ons-investeringsplan.

ENTSO-E (2018). *Mid-term adequacy forecast 2018*. URL: https://www.entsoe.eu/outlooks/midterm/.

ENTSO-E (2024). *Monthly hourly load values. 2023 data*. URL: https://www.entsoe.eu/data/power-stats/.

Graham, R. L., E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan (1979). "Optimization and approximation in deterministic sequencing and scheduling: a survey". In: *Discrete Optimization II*. Ed. by P. L. Hammer, E. L. Johnson, and B. H. Korte. Vol. 5. Annals of Discrete Mathematics. Elsevier, pp. 287–326. DOI: 10.1016/S0167-5060(08)70356-X.

Hartmann, S. and D. Briskorn (2010). "A survey of variants and extensions of the resource-constrained project scheduling problem". *European Journal of Operational Research* 207 (1), pp. 1–14. DOI: 10.1016/j.ejor.2009.11.005.

Hartmann, S. and D. Briskorn (2022). "An updated survey of variants and extensions of the resource-constrained project scheduling problem". *European Journal of Operational Research* 297 (1), pp. 1–14. DOI: 10.1016/j.ejor.2021.05.004.

Liander (2023). *Ontwerp investeringsplan 2024 elektriciteit en gas*. URL: https://www.liander.nl/over-ons/financiele-publicaties/investeringsplannen.

Nattaf, M., C. Artigues, and P. Lopez (2014). "A polynomial satisfiability test using energetic reasoning for energy-constraint scheduling". *Proceedings of the International Conference on Project Management and Scheduling (PMS 2014)*, pp. 169–172. URL: https://hal.archives-ouvertes.fr/hal-00978706.

Netbeheer Nederland (2024). *Capaciteitskaart elektriciteitsnet*. Version 19-01-2024. URL: https://capaciteitskaart.netbeheernederland.nl/.

PBL, TNO, CBS, and RIVM (2022). *Klimaat- en energieverkenning 2022*. Planbureau voor de Leefomgeving (PBL).

Pinedo, M. L. (2016). *Scheduling: Theory, Algorithms, and Systems (5th ed.)* Springer. DOI: 10.1007/978-3-319-26580-3.

Smith, W. E. (1956). "Various optimizers for single-stage production". *Naval Research Logistics Quarterly* 3, pp. 59–66. DOI: 10.1002/nav.3800030106.

Stedin (2024). *Investeringsplan Stedin 2024*. URL: https://www.stedin.net/over-stedin/jaarverslagen-en-publicaties/investeringsplan.

TenneT (2024a). *Ontwerpinvesteringsplan net op land 2024-2033*. URL: https://www.tennet.eu/nl/over-tennet/publicaties/investeringsplannen.

TenneT (2024b). *Ontwerpinvesteringsplan net op zee 2024-2033*. URL: https://www.tennet.eu/nl/over-tennet/publicaties/investeringsplannen.

World Resources Institute (2022). *Climate Watch data: GHG emissions*. Climate Watch. URL: https://www.climatewatchdata.org/ghg-emissions.

Part II

Continuous Energy-Constrained Scheduling Problems

Contents

Notation

Indices	
j	job
i	event, interval
e	position in event order
k	jump point index
$I(e)$	converts position to index
$E(i)$	converts index to position

Sets	
J	set of jobs
$\mathcal{E}$	list of events
$\mathcal{F}$	set of fixed-time events
$\mathcal{P}$	set of plannable events

Parameters and constants	
n	number of jobs
k	number of cost intervals
P	available amount of resource R
E_j	resource requirement of job J_j
$r_j, \bar{d}_j$	release time and deadline of job J_j
P_j^-, P_j^+	lower and upper bound on resource consumption for job J_j
w_j, B_j	weight and constant term for the cost function of job J_j
$\bar{f}_j$	step-wise increasing cost function of job J_j
K_j^k	jump point k of job J_j
M	large constant term ('big M')
L^B, L^R	penalty for violating bounds (B) or resource availability (R)

Variables	
t_i	time of occurrence of the event with index i
$p_{j,i}$	amount of resource consumed by job J_j in the interval that starts when the event with index i occurs
$a_{i,i'}$	$\begin{cases} 1 & \text{if } E(i) < E(i') \\ 0 & \text{if } E(i) \geq E(i') \end{cases}$ relative order of events i and i'
$b_{i,i'}$	$\begin{cases} 1 & \text{if } i' = \arg\min_{i' \in \mathcal{P}}(E(i') - E(i)) \\ 0 & \text{otherwise} \end{cases}$ successor relation of plannable events i and i'
$s_{j,i}^-, s_{j,i}^+$	amount of resource job J_j consumes less (more) than its lower (upper) bound in interval i
s_i^t	amount of resource consumed in interval i above the available amount P
S_j, C_j	start and completion time of job J_j
$p_j(t)$	resource consumption profile of job J_j

2

A hybrid local search algorithm for the Continuous Energy-Constrained Scheduling Problem

The contents of this chapter have been accepted for publication in the Journal of Scheduling. A preprint is available on arXiv:

R. J. J. Brouwer, J. M. van den Akker, and J. A. Hoogeveen (2023). "A hybrid local search algorithm for the continuous energy-constrained scheduling problem". Preprint at: https://arxiv.org/abs/2311.16177. DOI: 10.48550/arXiv.2311.16177

Abstract

We consider the Continuous Energy-Constrained Scheduling Problem (CECSP). A set of jobs has to be processed on a continuous, shared resource. A schedule for a job consists of a start time, completion time, and a resource consumption profile. The goal is to find a schedule such that each job does not start before its release time, is completed before its deadline, satisfies its full resource requirement, and respects its lower and upper bounds on resource consumption during processing. The objective is to minimize the total weighted completion time. We present a hybrid local search approach, using simulated annealing and linear programming, and compare it to a mixed-integer linear programming (MILP) formulation. We show that the hybrid local search approach matches the MILP formulation in solution quality for small instances, and is able to find a feasible solution for larger instances in reasonable time.

Keywords: *continuous scheduling, resource-constrained scheduling, mixed-integer linear programming, simulated annealing.*

2.1. Introduction

When considering scheduling problems with resource constraints, jobs are often assumed to have a fixed duration or are not allowed to change the amount of resource they consume over time. In certain applications, however, these assumptions are too limiting. Typically, such cases can be found in areas where a (cumulative) amount of work (or resource) is required to complete a job, but the rate of consumption is not necessarily fixed, such as energy related applications. Examples of such applications are demand-response in electricity consumption and electric vehicle charging. To find schedules that can accommodate flexible charging profiles, we consider an extension of the traditional scheduling problem, previously introduced as the Continuous Energy-Constrained Scheduling Problem (CECSP) by Nattaf, Artigues, and Lopez (2014).

This problem is described as follows. A set $\{J_1, \ldots, J_n\}$ of jobs has to be processed on a continuous resource R. This means that, at any time, multiple jobs can be processed simultaneously and at different rates, as long as their total consumption does not exceed P. Each job J_j requires a total amount of resource equal to E_j. The goal is to find a schedule such that each job J_j does not start before its release time r_j, is completed before its deadline $\bar{d}_j$, and respects its lower and upper bounds (P_j^-, P_j^+) on the resource consumption rate during processing. Preemption is not allowed. From its start until its completion, each job must consume resources at a rate of at least P_j^- units. The objective is to minimize the total weighted completion time. We look at the case where both the resource and time are continuous. We assume that there is no efficiency function influencing resource consumption, and no explicit precedence relations exist between jobs. It should be noted that the presented approach can easily be extended to deal with (piece-wise) linear efficiency functions and precedence relations. The present work, however, does not include these extensions.

The CECSP was originally introduced as a generalization of the Cumulative Scheduling Problem (CuSP) by Nattaf, Artigues, and Lopez (2014). In the case of the CuSP, we have a single resource with a given capacity, and we need to schedule a number of activities without exceeding this capacity. Each activity has a release time, deadline, (fixed) processing time and (constant) resource capacity requirement. The CuSP was formulated by Baptiste, Pape, and Nuijten (1999) as a subproblem of the Resource Constrained Project Scheduling Problem (RCPSP), relaxing precedence constraints and considering only a single resource. The RCPSP is a very general problem concerning the scheduling of activities subject to precedence, time and resource constraints. The surveys by Hartmann and Briskorn (2010, 2022) provide a good overview of the RCPSP and its extensions. Notably, continuous and event-based formulations of the RCPSP have been studied and evaluated by Koné et al. (2011) and Kopanos, Kyriakidis, and Georgiadis (2014) as well as more general models with flexible resource profiles in continuous time (FRCPSP), for example by Naber (2017).

For the CECSP, Nattaf, Artigues, and Lopez (2017) provided a hybrid branch-and-

bound algorithm and mixed-integer linear program (Nattaf, Horváth, et al. 2019) to find exact solutions for small instances. The CECSP generalizes the CuSP as the resource capacity requirement is considered to be a range with a lower and an upper bound, rather than a fixed value, and the consumption rate can vary during the execution of the job. As a result, the processing time is no longer fixed, but depends on the consumption rate during its execution. Through its relation to CuSP, Nattaf, Artigues, Lopez, and Rivreau (2016) proved that the feasibility problem for CECSP is NP-complete.

The CECSP is also closely related to the scheduling of malleable jobs (as introduced by Turek, Wolf, and Yu (1992)) on parallel machines, which involves the scheduling of jobs on P machines, while the number of machines assigned to a job can change during its execution.

Comparing the problem studied in this work to the version originally introduced by Nattaf, Artigues, and Lopez (2014), we notice two differences:

1. The version presented here does not consider efficiency functions (i.e., the amount of work done is equal to the amount of resource consumed), while in the work of Nattaf, Artigues, and Lopez, (piece-wise) linear efficiency functions are considered (i.e., the amount of work done is a linear function of the amount of resource consumed).

2. The objective function minimizes the total weighted completion time, while the objective in the work of Nattaf, Artigues, and Lopez is to minimize the amount of resource consumed.

Our contribution. We present a hybrid local search approach (Section 2.3), using simulated annealing and linear programming, to solve instances of the CECSP. In addition, we provide a mixed-integer linear programming (MILP) formulation (Section 2.2.2) and compare the performance of our hybrid approach to it in terms of quality and runtime. Finally, we found that the feasibility problem is solvable in polynomial time if we drop the lower bounds on resource consumption from the problem. We formulate a flow-based solution algorithm for that case (Section 2.4.2), that we use to select problem instances for a new benchmark set of problem instances that we use to evaluate our approach.

As far as we are aware, we are the first to propose a decomposition and try a combination of local search and mathematical programming techniques for CECSP. We show that this approach can compete with exact approaches on small instances, and has the potential to find good solutions for larger ones. Furthermore, the approach can be adapted to be applied to related (energy-constrained) scheduling problems.

The rest of this work is structured as follows. Section 2.2 contains a detailed problem description, followed by an explanation of the event-based model in Section 2.2.1. From there, we build an MILP formulation in Section 2.2.2. The hybrid local search approach is introduced in Section 2.3. We discuss how our test instances are generated in Section 2.4, which includes a description of our flow-based algo-

j	E_j	r_j	$\bar{d}_j$	P_j^-	P_j^+	w_j	B_j
1	70.0	0.0	3.0	10.0	30.0	1.0	0.0
2	20.0	1.0	3.0	10.0	40.0	3.5	0.0
3	45.0	2.5	4.0	10.0	50.0	5.0	0.0

Table 2.1: Example instance of the CECSP with three jobs

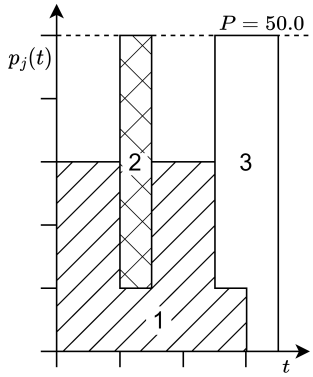


Figure 2.1: Optimal schedule for the example instance with three jobs

rithm for the feasibility problem without lower bounds. In Section 2.5 we discuss the results, followed by the conclusions and suggestions for future work in Section 2.6.

2.2. An event-based MILP model for the CECSP

In this section, we will first provide a detailed problem description. We will then present an event-based model, and an MILP formulation for the CECSP. A summary of notation is provided at the start of this part, on page 17.

An instance of the CECSP in the context of this work is defined by the following properties:

- A resource R with a constant availability of P;
- For each job J_j, $j \in \{1, ..., n\}$:
  - Resource requirement E_j;
  - Release time r_j;
  - Deadline $\bar{d}_j$;
  - Lower bound P_j^-;

- Upper bound P_j^+;
- Linear cost function $w_j \cdot C_j + B_j$.

For each job, we need to determine a start time S_j, a completion time C_j and a resource consumption profile $p_j(t)$ such that $\sum_j w_j C_j + B_j$ is minimal, while the following constraints are respected:

C1 The total amount of resource a job j consumes is exactly E_j;

C2 A job j does not start before its release time r_j, i.e. $S_j \geq r_j$;

C3 A job j completes no later than its deadline $\bar{d}_j$, i.e. $C_j \leq \bar{d}_j$;

C4 A job j only consumes resources between its start and completion time, i.e. $p_j(t) = 0$ for $t < S_j$ or $t \geq C_j$;

C5 While active, the resource consumption of a job j never drops below P_j^- or rises above P_j^+, i.e. $P_j^- \leq p_j(t) \leq P_j^+$ for all possible values of t, with $S_j \leq t \leq C_j$;

C6 The total amount of resource consumed by all jobs together at any given time can never exceed P, i.e. $\sum_j p_j(t) \leq P$ for all possible values of t.

An example instance with three jobs and $P = 50.00$ is provided in Table 2.1. Figure 2.1 gives a visual representation of the optimal schedule for this instance.

Note that the constant B_j in the cost function of a job J_j can generally be ignored, as $\sum_j B_j$ is a constant term in the objective function. It has been added to the cost function to model a generalized linear function, but the problem does not fundamentally change if we ignore the constant part of these functions.

2.2.1. Event-based approach

We model instances of the CECSP using an event-based approach. This means that we define a schedule by determining the timing of *events*, and the activity of jobs during the *intervals* that are bounded by these events.

We consider two types of events, associated with a job J_j: its start S_j and completion C_j. Now let us consider a list $\mathcal{E}$, that lists all events in the order that they occur over time.

For a given order of events, we can divide the schedule into $2n - 1$ intervals, where each interval is bounded by two consecutive events. In the following, when we refer to an interval, we always mean the span of time in between two consecutive events. So, no event ever occurs during an interval. Also note that an interval can be empty, if it is bounded by two events that happen simultaneously.

During these intervals, the set of jobs that are being processed does not change. We can show that there always exists an optimal solution in which $p_j(t)$ remains constant during each interval. We can therefore limit ourselves to such solutions.

Theorem 2.1. *For any feasible schedule $\mathcal{S}$ (with start times S_j, completion times C_j and resource consumption profiles $p_j(t)$ for all jobs $J_j, j \in \{1, ..., n\}$) that follows a given event order $\mathcal{E}$, a feasible schedule $\mathcal{S}'$ exists with the same objective value where the resource consumption $p'_j(t)$ of all jobs remains constant during each interval.*

Proof. Consider any interval between two events, i and i'. The interval starts at t_i and ends at $t_{i'}$.

A job J_j is inactive during this interval if $t_i \geq C_j$ or $t_{i'} \leq S_j$ in the schedule, and active otherwise.

The resource consumption profile of any active job J_j in $\mathcal{S}$ within this interval does not violate the lower and upper bounds of this job: $P_j^- \leq p_j(t) \leq P_j^+ \; \forall t, t_i \leq t \leq t_{i'}$.

Therefore, the average consumption of job J_j within this interval, $\bar{p}_j = \frac{\int_{t_i}^{t_{i'}} p_j(t) \; dt}{t_{i'} - t_i}$, does not exceed the bounds on consumption for job J_j: $P_j^- \leq \bar{p}_j \leq P_j^+$.

In this interval, all active jobs in $\mathcal{S}$ together consume at most $(t_{i'} - t_i)P$ of the resource, i.e.:

$$\sum_j \bar{p}_j (t_{i'} - t_i) \leq (t_{i'} - t_i) P$$

From this, it follows that $\sum_j \bar{p}_j \leq P$, i.e., the sum of the average consumption of all jobs does not exceed the available amount of resource.

We obtain $\mathcal{S}'$, from $\mathcal{S}$ by replacing all resource consumption profiles $p_j(t)$ with a step-wise constant function $p'_j(t)$ that remains constant during any interval (in between two consecutive events), and takes the value $\bar{p}_j$, i.e. the average resource consumption of job J_j in $\mathcal{S}$, during that interval.

We have shown that the schedule $\mathcal{S}'$ is indeed feasible. For each job, the resulting consumption rate is within the bounds allowed for that job and the total amount of available resource is not exceeded either. No other constraints are affected.

It remains to show that $\mathcal{S}'$ has the same objective value. The objective depends only on the value of C_j. As the timing of all events in $\mathcal{S}'$ is identical to that in $\mathcal{S}$, the objective is not affected by the modifications to the schedule. □

Corollary 2.2. *For any optimal schedule $\mathcal{S}^*$, there exists an optimal schedule $\mathcal{S}^{*\prime}$ where the resource consumption $p_j^{*\prime}(t)$ of all jobs remains constant during each interval.*

This means that we can ignore solutions with a non-constant resource consumption profile for any job during any interval. Therefore, we can fully represent a resource consumption profile of job J_j with at most $2n - 1$ values $p_{j,i}$, one for each interval

in which J_j is being processed, indicating the total amount of resource consumed during that interval.

In the following, we will maintain a link between events and jobs by their index. Every job has two associated events. For a job with index j, we will say that its start event has index $i = 2j - 1$, and its completion event has index $i + 1 = 2j$. This allows us to uniquely identify the start and completion events of any job easily in notation. *The index of an event i does not indicate its position in the event order.* We will maintain this indexing scheme throughout this work.

Furthermore, intervals are defined by the event at the start of that interval. For example, let i and i' be the indices of the first two events. The first interval starts when the first event happens at t_i; and ends when the next event starts at $t_{i'}$. Then, $p_{j,i}$ represents the amount of resource that job J_j consumes in the interval associated with event i, spanning the time between t_i and $t_{i'}$.

2.2.2. Mixed-integer linear programming formulation

Using the event-based approach, we developed an MILP formulation that will find an optimal schedule. In this section, we gradually build this formulation. The full formulation can be found in Appendix 2.A.

Although we arrived at it independently, our formulation is similar to the event based formulations presented by Nattaf, Artigues, Lopez, and Rivreau (2016). The models share their focus on intervals, delimited by events, but differ in how the order of these events is expressed. Nattaf, Artigues, Lopez, and Rivreau (2016) use assignment variables, binary variables that indicate whether events occur at a given position in the order, or whether a job is active during a given interval. Our model uses binary ordering variables, to indicate the relative order between pairs of events.

First, we define the decision variables of our model. For each of the $2n$ events i, $t_i \geq 0$ will denote the time at which it takes place. Recall that all odd event indices i belong to start times, and all even event indices belong to completion times. For each job J_j and event i, $p_{j,i} \geq 0$ is the amount of resource that J_j consumes during the interval defined by event i. We know the consumption of a job J_j during the interval starting at its completion time C_j to be 0 and therefore set $p_{j,2j} = 0$. Note that we do not know beforehand what event will be the last in the event order. The interval that is defined by that event is, theoretically, open-ended. Note, however, that in the formulations below any $p_{j,i}$ associated with this interval is forced to 0, as all jobs complete in or before the interval.

Additionally, we define binary variables $a_{i,i'}$ and $b_{i,i'}$ to determine the order of events. These can be viewed as helper variables. For two events i and i', $a_{i,i'}$ represents the order they occur in. If $a_{i,i'} = 1$, then i occurs before i' in the event order. On top of that, $b_{i,i'}$ indicates whether an event i is an immediate successor of another event i': $b_{i,i'} = 1$ if i' occurs immediately after i. In our formulation, $b_{i,i'}$ is used exclusively for the enforcement of lower bounds.

The objective is to minimize the combined cost functions of the jobs:

$$\min \sum_{j=1}^{n} (w_j t_{2j} + B_j) \tag{2.1}$$

Now, let us go through the constraints we listed at the beginning of this section.

First, we ensure that the total amount of resource consumed by each job J_j equals E_j (C1), $\forall j \in \{1, ..., n\}$:

$$\sum_{i=1}^{2n} p_{j,i} = E_j \tag{2.2}$$

Next, we enforce the release time (C2) and deadline (C3) for each job J_j. Since we can identify the start and completion event of a job by its index i, we obtain that $\forall j \in \{1, ..., n\}$:

$$t_{2j-1} \geq r_j \tag{2.3}$$

$$t_{2j} \leq \bar{d}_j \tag{2.4}$$

We ensure that $p_{j,i} = 0$ for any interval after the completion event of J_j (with index $2j$) has happened or before its start event (with index $2j - 1$) has taken place (C4). We use a big M constraint with the order variables $a_{i,i'}$ to enforce this exactly for those intervals. So $\forall j \in \{1, ..., n\}, i \in \{1, ..., 2n\}$, and sufficiently large M ($M = E_j$ will suffice) we have:

$$p_{j,i} \leq a_{i,2j} M \tag{2.5}$$

$$p_{j,i} \leq a_{2j-1,i} M \tag{2.6}$$

In between the start and completion events of a job J_j, the values for $p_{j,i}$ must respect the lower (P_j^-) and upper (P_j^+) bounds on resource consumption (C5). The upper bound can be enforced in between any two events. We use a big M constraint with order variable $a_{i,i'}$ to activate the constraint only for events that happen in the right chronological order, hence $\forall j \in \{1, ..., n\}, i, i' \in \{1, ..., 2n\}, i \neq i'$, and sufficiently large M:

$$p_{j,i} \leq P_j^+ (t_{i'} - t_i) + M a_{i',i} \tag{2.7}$$

The lower bound only applies for consecutive events, which both fall within the execution window of job J_j. We use a big M term with the $b_{i,i'}$ helper variable to activate the constraint only for events that are immediate successors. Besides that, we include the variables $a_{i',2j-1}$ and $a_{2j,i}$ to make sure the constraint is only active if i and i' are between the start event of job J_j (index $2j - 1$) and its completion

(index $2j$). Hence $\forall j \in \{1, ..., n\}, i, i' \in \{1, ..., 2n\}, i \neq i'$, and sufficiently large M:

$$p_{j,i} \geq P_j^-(t_{i'} - t_i) - (1 - b_{i,i'} + a_{i,2j-1} + a_{2j,i'})M \tag{2.8}$$

During an interval, the total resource consumption of all active jobs cannot exceed P at any time. This has to be enforced for all intervals, $\forall i, i' \in \{1, ..., 2n\}, i \neq i'$, and sufficiently large M:

$$\sum_{j=1}^{n} p_{j,i} \leq P(t_{i'} - t_i) + Ma_{i',i} \tag{2.9}$$

We need to make sure that the event times are consistent with the variables $a_{i,i'}$ that control the order of events. Therefore, $\forall i, i' \in \{1, ..., 2n\}, i \neq i'$, and sufficiently large M, we get:

$$t_i \leq t_{i'} + Ma_{i',i} \tag{2.10}$$

We ensure that either $a_{i,i'} = 1$ or $a_{i',i} = 1$, but not both, for any pair of distinct events. We could, of course, get rid of half of our binary variables by expressing $a_{i',i}$ in terms of $a_{i,i'}$ for all $i' > i$. We choose not to do so for ease of notation, therefore we need to ensure that $\forall i, i' \in \{1, ..., 2n\}, i < i'$:

$$a_{i,i'} + a_{i',i} = 1 \tag{2.11}$$

We can note here that we also already know: $\forall j \in \{1, ..., n\}$, $a_{2j-1,2j} = 1$.

Finally, we need some constraints to make sure the $b_{i,i'}$ variables indicate events that are immediate successors. First making sure that $b_{i,i'}$ can only be 1 if i' occurs immediately after i, $\forall i, i' \in \{1, ..., 2n\}, i \neq i'$, and sufficiently large M:

$$\sum_{i''=1}^{2n} a_{i,i''} - \sum_{i''=1}^{2n} a_{i',i''} \leq 1 + (1 - b_{i,i'})M \tag{2.12}$$

$$\sum_{i''=1}^{2n} a_{i,i''} - \sum_{i''=1}^{2n} a_{i',i''} \geq 1 - (1 - b_{i,i'})M \tag{2.13}$$

And second, by requiring that exactly (or equivalently, at least) $2n - 1$ variables $b_{i,i'}$ actually take the value of 1:

$$\sum_{i=1}^{2n} \sum_{i' \neq i} b_{i,i'} = 2n - 1 \tag{2.14}$$

```
prec ← ImplicitPrecedences(J)
E, best_score ← InitialSolution(J, P, prec)
T ← T_init
for iter ∈ {1, ..., max_iter} do
    E, score ← GetNeighbor(E, T, J, P, prec)
    if score < best_score then
        E_best, best_score ← E, score
    if iter%α_period = 0 then
        T ← T · α
return E_best, best_score
```

Algorithm 2.1: Hybrid local search approach

This completes the initial MILP formulation. However, we have added two types of valid inequalities to speed up the process of solving the model. We know that the range of feasible processing times for a job is limited by the lower and upper bound on its resource consumption, $\forall j \in \{1, ..., n\}$:

$$t_{2j} - t_{2j-1} \leq E_j / P_j^- \tag{2.15}$$

$$t_{2j} - t_{2j-1} \geq E_j / P+_j \tag{2.16}$$

Indeed, the addition of these inequalities has a significant impact on the time needed to find an optimal solution. The MILP formulation that our hybrid local search approach will be tested against consists of all equations (2.1) - (2.16) and is repeated in Appendix 2.A.

2.3. Hybrid local search

In the MILP formulation described above, we can identify two separable parts to the problem. The binary variables $a_{i,i'}$ (and $b_{i,i'}$) fully determine the order of events $\mathcal{E}$. Then, the continuous variables t_i and $p_{j,i}$ determine the exact schedule for that order.

We can decompose the problem into these two parts. Once we have an order of events $\mathcal{E}$, determining an optimal schedule boils down to solving an LP, as all binary variables disappear from the formulation once the event order is fixed.

A global overview of our approach is given in Algorithm 2.1. The elements in this overview will be discussed in more detail in the following sections. How solutions are evaluated is discussed in Section 2.3.1. Section 2.3.2 explains the process of finding an accepted neighbor, returned by GetNeighbor in Algorithm 2.1. Finally, the algorithms behind ImplicitPrecedences and InitialSolution are discussed in Section 2.3.3.

2.3.1. Finding an optimal schedule for a given order using linear programming

We will adapt the MILP formulation described above to solve only the (sub)problem of finding an optimal schedule for a given event order.

The LP will consist of Equations (2.1) - (2.4) and (2.7) - (2.10), eliminating any part of the inequalities that contain the variables $a_{i,i'}$ and $b_{i,i'}$.

Instead of enforcing $p_{j,i}$ to be 0 outside of J_j's processing window (as in Equations (2.5) and (2.6)), we only take into account $p_{j,i}$ for i within the processing window, which can easily be done as we already know the order of events. Similarly, upper and lower bounds are only enforced on intervals within the processing window.

For ease of notation, let us define two functions:

- $I(e)$ gives the event index i of the event in position e of the event order;
- $E(i)$ gives the position in the event order e of the event with index i.

Note that the actual values can already be substituted for these functions before we attempt to solve the LP, as they only depend on the event order.

So, for example the inequality enforcing the order of events (Equation (2.10) in the MILP formulation) now becomes, $\forall e \in \{1, ..., 2n-1\}$:

$$t_{I(e)} \leq t_{I(e+1)} \tag{2.17}$$

And the inequality controlling the total work of a job (Equation (2.2) in the MILP formulation) now becomes, $\forall j \in \{1, ..., n\}$:

$$\sum_{e \in \{E(2j-1), ..., E(2j)-1\}} p_{j,I(e)} = E_j \tag{2.18}$$

Note that $\{E(2j-1), ..., E(2j)-1\}$ represents the set of positions in the event order between the start and completion event of a job J_j, including the former and excluding the latter.

Finally, to evaluate the relative quality of infeasible event orders, we allow for the violation of lower bounds, upper bounds, and/or resource availability constraints, with appropriate cost. For this purpose, we introduce three types of slack variables:

- $s^-_{j,i}$ allowing the resource assignment to job J_j in interval i to go below its lower bound P^-_j;
- $s^+_{j,i}$ allowing the resource assignment to job J_j in interval i to go over its upper bound P^+_j;
- s^t_i allowing for increasing the amount of available resource during an interval i.

The slack variables are non-negative, but do not have an enforced upper bound. The introduction of the slack variables affects the inequalities enforcing these bounds in the LP (Equations (2.7) - (2.9) in the MILP formulation), $\forall j \in \{1, ..., n\}$, $e \in \{E(2j-1), ..., E(2j)-1\}$:

$$p_{j,I(e)} \geq P_j^-(t_{I(e+1)} - t_{I(e)}) - s^-_{j,I(e)} \tag{2.19}$$

$$p_{j,I(e)} \leq P_j^+(t_{I(e+1)} - t_{I(e)}) + s^+_{j,I(e)} \tag{2.20}$$

And, $\forall e \in \{1, ..., 2n-1\}$:

$$\sum_{j=1}^{n} p_{j,I(e)} \leq P(t_{I(e+1)} - t_{I(e)}) + s^t_{I(e)} \tag{2.21}$$

These slack variables are penalized in the objective function, using L^B for violation of upper and lower bounds and L^R for excessive resource usage. This adds the following term to the objective:

$$\sum_{i=1}^{2n-1} \left(L^R s^t_i + \sum_{j=1}^{n} L^B (s^-_{j,i} + s^+_{j,i}) \right) \tag{2.22}$$

The total value of the penalty term gives an indication of the quality of the infeasible event orders that can be used to guide the local search towards solutions that are closer to feasibility. For many event orders, no feasible schedule exists. Scoring the 'degree of infeasibility', helps the search to move on from an infeasible order towards a feasible one.

2.3.2. Improving the order of events using local search

We use a local search algorithm, simulated annealing, to explore the search space of all possible event orders. The LP is used to evaluate the quality of each event order. Below, we will describe the core of the local search, and the neighborhood operators that are used.

We start the local search with the event order that results from the greedy algorithm (Section 2.3.3). Every iteration, we modify the current solution using one of the neighborhood operators. Improvements are always accepted, while solutions that are worse than the current one are accepted with a probability determined by the current value of the temperature T. The precise settings for the simulated annealing will be discussed in Section 2.5.

If a candidate solution is rejected, we keep trying with the same neighborhood operator. We do this by generating a random permutation of the event list each time we start with a neighborhood operator, and apply the operator to the events in that order, until a candidate solution is accepted. If this is unsuccessful we move on to the next neighborhood operator. If no solution is accepted after looping

through all three neighborhood operators, we terminate the search and report the best seen solution up to that point.

We have three neighborhood operators that we use to modify the event order and obtain new candidate solutions:

- **Swap two adjacent events**. We take the next event from the random permutation, at position $e \in \{1, ..., 2n-1\}$ and its successor at position $e+1$ and reverse their order. This means that the event originally at position e now is at position $e+1$ and vice versa.
- **Move a single event**. We take the next event from the random permutation, at position $e \in \{1, ..., 2n\}$. We then determine its range of movement. On both sides, it is limited by the first event we encounter with which the event has a precedence relation. How we determine these precedence relations is discussed in Section 2.3.3. From this range, we then select a new position, where the probability of a position being selected is inversely proportional to its distance to the original position of the event. That is, smaller displacements are more likely to be selected than larger ones. The event is then moved to its new position, with the other events in between the current and former position of the event shifted accordingly.
- **Move both events associated with a single job**. We take the next job from the random permutation, job J_j $j \in \{1, ..., n\}$. We then look at both of its associated events, and determine their combined range of movement. As before, we determine the number of positions both can move in either direction without crossing over an event with which they have a precedence relation (see Section 2.3.3). We then take the minimum of these values to get a combined range. From this range, we select a new position with uniform probability. The events are then moved to their new position, both offset from their original position by the same amount, with the other events in between the current and former position of the events shifted accordingly.

Each of these neighborhood operators results in a small modification of the event order, necessitating an update of the LP for evaluation. We modify only those parts of the LP that are affected, and re-solve the model to obtain the score of the new candidate solution, starting from the current solution.

2.3.3. A greedy algorithm for initial solution generation and preprocessing

We obtain an initial event order using a greedy algorithm. The algorithm assigns resource to jobs that have been released and have not been fully served, and extracts the order of events from the resulting schedule. In the construction of the initial order, we ignore lower bounds on resource consumption and do not consider the cost function at all. Keep in mind that we are only interested in the order of events at this point, not in the schedule itself.

We take the release times r_j and deadlines $\bar{d}_j$ of all jobs, and sort them in ascending order. In case of a tie, we sort by job index, as the order in this case is immaterial

for our construction. In this way, we obtain a sequence of time periods during which the set of jobs available for processing remains the same.

We loop over these time periods, keeping track of the list of available jobs during the period. Then, we decide the assignment of resources during each time period in the following way:

1. We assign the minimal amount of resource required to all available jobs. If this combined minimum exceeds the total amount of resource available, we stop after this step and continue to the next time period;

2. In order of increasing deadline, we assign as much resource to each job as it can consume. This may be bounded by the upper bound on resource consumption of the job, or by the amount of resource that is left to be assigned at this point. We continue until the resource in the time period is fully used or no other jobs are available.

The minimal amount of resource a job has to consume is given by $\max\{0, \tilde{E}_j - (\bar{d}_j - t_{end})P_j^+\}$, while its maximum is given by $\min\{\tilde{E}_j, P_j^+(t_{end} - t_{start})\}$. Here t_{start}, t_{end} are the start and end of the current time period and $\tilde{E}_j$ is the *remaining* resource requirement of the job.

At any time during this procedure:

- If a job is assigned resources for the first time, we append the start event for that job to our current list;

- If a job's full resource requirement is satisfied, we append the completion event for that job to our current list.

At the end of the procedure we have a complete order of events, which will be the starting solution for our local search. We determine the quality of the optimal schedule for the initial event order by solving the LP as described above. This schedule is likely to assign resources differently from the greedy algorithm we used to find the event order in the first place. Note that the schedule may have positive penalty terms, i.e. it does not have to meet the available resource bound P or the job lower or upper bounds P_j^- and P_j^+.

To complete the description of our approach, we will discuss the preprocessing steps that we use in our algorithm. We deduce a number of implicit precedence constraints that we use in the neighborhood operators:

1. A start event S_j of a job J_j always has to occur before its corresponding completion event C_j;

2. Let $u_j = E_j/P_j^+$ be the minimal processing time required to complete a job. Then the possible time ranges of a start and completion event are $[r_j, \bar{d}_j - u_j]$ and $[r_j + u_j, \bar{d}_j]$, respectively. For any two events, if the last possible time within the range of event i lies before the first possible time within the range of event i', we know that i has to come before i' in any event order.

n	5, 10, 15, 20, 30, 50
P	25.00, 50.00, 100.00, 200.00
a	True, False

Table 2.2: Overview of values of n, P and a used for generating instances

2.4. Test instances

We generated a number of instances to evaluate our approach. We consider three important characteristics to group these instances: the number of jobs (n), the amount of resource (P) and whether an instance is *adversarial* (a).

An instance is said to be adversarial, when the ordering of weights is such that the weight of jobs monotonically increases with increasing deadline. This is meant as a counter to the greedy algorithm that determines a starting solution, as this assigns resources to jobs in order of increasing deadline.

By varying P, we also affect the ratios of P/P_j^-, P/P_j^+ and P/E_j, as the sampling of values for E_j, P_j^+ and P_j^- is (largely) unaffected by the chosen values of n, P and a. Each of these ratios are more interesting properties of an instance than P in itself.

Table 2.2 lists the values of n, P and a for which instances were generated. Each combination of n, P and a was used to generate four unique instances.

Below, we describe the procedure for generating instances.

2.4.1. Instance generation method

The instance generation is based on the three parameters discussed above (n, P and a) as well as four scaling parameters: a_{maxlow}, a_{minupp}, a_{rshift} and a_{pws}. The first three parameters (a_{maxlow}, a_{minupp} and a_{rshift}) are fractions, and should be between 0 and 1. The last parameter (a_{pws}) is a multiplication factor, and should have a value larger than 1. For each job we sample seven values, that together fully determine the six properties of a job as described in Section 2.2.

- A resource requirement E_j from $U(10, 100)$.
- A lower bound P_j^- from $U(0, \min\{a_{\text{maxlow}}P, a_{\text{minupp}}E_j\})$. The parameters a_{maxlow} and a_{minupp} can be used to control the largest possible lower bound. Here a_{maxlow} makes sure that the lower bound on the resource consumption per time unit never exceeds a fraction (a_{maxlow}) of what is available during a unit of time, leading to instances where the lower bounds do not restrict the schedule to only processing a single job at a time. Then, a_{minupp} has a similar purpose, but as a fraction of the total resource requirement of a job, ensuring that the lower bound does not restrict the possible duration of a job too much.

- An upper bound P_j^+ from $U(a_{\text{minupp}}E_j, E_j)$. The a_{minupp} parameter can be used to make the upper bound more likely to be restrictive, while also ensuring it does not drop below the lower bound. This distribution also ensures that no job can be completed in less than a single time unit.
- A release time r_j from $U\left(-a_{\text{rshift}}\frac{\sum_j E_j}{P}, (1-a_{\text{rshift}})\frac{\sum_j E_j}{P}\right)$. Any negative values are treated as zero. The parameter a_{rshift} shifts the window for generating release times, such that a fraction of the generated release times equal to a_{rshift} is expected to be a negative number. In this way, one can control the expected number of jobs that are available for processing immediately at time 0. The largest possible release time is limited by the expected total processing time (at full utilization).
- A deadline $\bar{d}_j$: $r_j + U\left(\frac{E_j}{\min\{P, P_j^+\}}, a_{\text{pws}}\frac{\sum_j E_j}{P}\right)$. The parameter a_{pws} sets the latest possible deadline in terms of the expected total processing time (at full utilization). The earliest possible deadline is the release time plus the minimum required processing time (its requirement divided by the minimum of its upper bound or the amount of resource available per unit of time), avoiding trivially infeasible instances.
- A weight w_j from $U(0,5)$ and an optional objective constant B_j from $U(0,10)$. Together they determine the cost function of job J_j: $w_j \cdot C_j + B_j$.

All numbers are rounded to two decimal places.

By design, this instance generation method does not entirely prevent a shortage of available resource during a certain period of time, depending on the sampled release times and deadlines. The odds of this happening can be controlled by increasing the range of release times and/or processing windows, using a_{pws} for example. We describe a method that gives a strong indication of the feasibility of an instance in the following section. In Section 2.4.3 we describe our choice of parameters that lead to mostly feasible instances.

2.4.2. Checking feasibility

To get an indication of the (in)feasibility of generated instances, we use a flow-inspired LP that solves the feasibility problem for the instance, ignoring lower bounds (P_j^-).

The lower bounds might still render an instance infeasible, even if the LP indicates that a feasible flow exists. However, it provides a good and quick indication.

We will formulate the problem as a max flow problem. We construct a bipartite graph, with one node for each job on one side, and one node for each time interval on the other. The intervals are obtained by sorting the set of all release times and deadlines and constructing an interval for every two consecutive distinct values. This results in at most $2n-1$ intervals.

There is an arc between a job node (j) and a time interval node (t), if the job

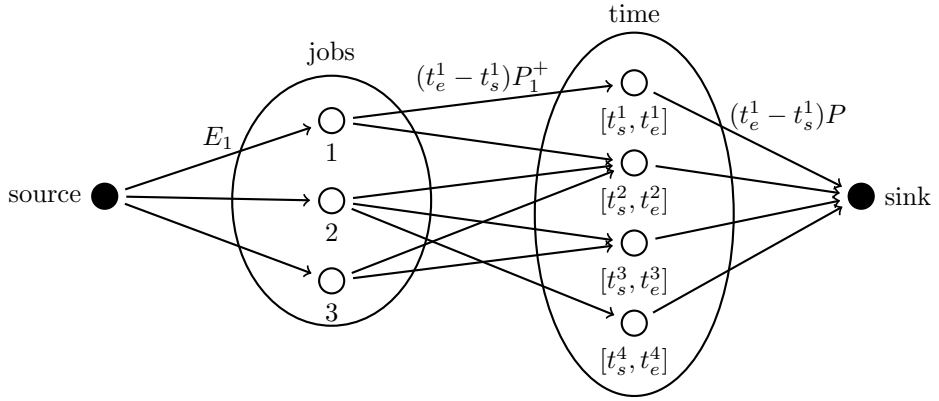


Figure 2.2: Example with three jobs and four intervals

can be processed during the interval $t = [t_s, t_e]$ (i.e. $r_j \leq t_s$ and $t_e \leq \bar{d}_j$). The capacity of that arc is equal to $(t_e - t_s)P_j^+$, the maximum amount of resource that job j can receive during that interval.

We add a source that has an arc going to every job node, with E_j (the total amount of work for that job) as its capacity, and a sink with an incoming arc from each time interval node with a capacity of $(t_e - t_s)P$ (the available resource in that interval).

A flow through this graph now will represent a resource assignment to the jobs. We use an LP analogue to solve this flow problem. If a max flow of $\sum_j E_j$ is found, we know the problem without lower bounds is feasible. If the max flow has a smaller value, we know that this instance has no feasible solution.

An example with three jobs and four time intervals is illustrated in Figure 2.2. Note that the end time t_e of an interval is identical to the start time t_s of the next interval, e.g. $t_e^1 = t_s^2$.

2.4.3. Choice of parameters for instance generation

We performed a grid search on a number of possible combinations of parameter settings, generating 100 random instances for each setting and using the flow-inspired LP described above to check the feasibility.

This analysis was performed for different values of n, while P was fixed at 50.00. A parameter setting was selected for each value of n such that for 99% of the generated instances a feasible flow exists in the relaxed problem (without lower bounds). The aim was to end up with mostly feasible instances, with the occasional infeasible instance mixed in.

The infeasible instances in the resulting test set will provide insight on the behavior of our solution approach on instances that we know to be infeasible. At the same time, using this parameter setting leads to the generation of instances with high

	a_{maxlow}	a_{minupp}	a_{rshift}	a_{pws}
$n = 5, 10$	0.25	0.25	0.125	2
$n = 15, 20, 30, 50$	0.25	0.25	0.125	1.5

Table 2.3: Parameter settings used for generating instances of different sizes

competition for resources during at least some time periods. Finding a feasible solution may already be a challenge for these instances, and this is exactly the type of challenging instance that our approach should be tested on.

Note, however, that this analysis was performed for $P = 50.00$ only. While the same parameter settings were used for other values of P, the instances generated are not necessarily equally likely to be feasible. We observe in practice, however, that the likelihood of feasibility is very similar.

The parameter settings used for each value of n are listed in Table 2.3.

2.5. Computational results

We ran computational experiments, measuring both solution time and the objective value of the best found solution for the MILP formulation and the hybrid local search algorithm. The instances we generated and used to run the tests can be found online (Brouwer 2022a), as well as the code used to run the experiments (Brouwer 2022b).

2.5.1. Parameter settings

To run our simulated annealing algorithm, we need to determine the value for the following parameters:

- Penalty for using slack variables L^B, L^R;
- Initial temperature T_{init};
- Number of iterations between temperature updates α_{period};
- Multiplication factor for updating the temperature α;
- Probability of selecting each neighborhood operator:
  - Swap p_s;
  - Single move p_m;
  - Paired move p_p.

Below, we will discuss the effect of each of these parameters and explain the choice of parameters for the hybrid approach, as presented in Table 2.4.

The values for the penalty terms L^B and L^R have to be chosen carefully. A value that is too high will discourage exploration by making traversing infeasible parts

T_{init}	α_{period}	α	L^B	L^R	p_s	p_m	p_p
n	$(2n-1)\cdot 4$	0.95	5	5	0.75	0.15	0.1

Table 2.4: Parameter settings for simulated annealing used for computational experiments

of the search space unlikely, once a feasible solution has been found. A value that is too low, may result in some infeasible solutions having a lower score than the optimal solution. The value of 5 is chosen to match the largest weight w_j a job can have.

Higher values for the initial temperature T_{init} lead to a search that is more likely to continue with solutions that are not an improvement over the current solution. The multiplication factor α and the number of iterations between updates α_{period} of the temperature control how quickly the temperature, and thereby the chance of selecting a worse solution than the current one, is lowered during the search process. Common values for α are 0.95 and 0.99, while α_{period} is generally set to be a multiple of the neighborhood size. As our dominant neighborhood is defined by the 'swap' operator, we chose its size of $2n-1$ as a starting point.

For a selection of instances of different sizes, we ran a (limited) grid search for a number of combinations of parameter values. The settings that were tested are as follows: $T_{init} \in \{5, 15, 30, 45, 60, 75\}$, $\alpha \in \{0.95, 0.99\}$ and $\alpha_{period} \in \{2, 4, 6, 8, 10\} \cdot (2n-1)$. We extrapolated the results and chose a combination of values that performed well across all instance sizes. When considering individual instances or small groups of instances, the selected set of parameters is not always the best performing set that we have tested. However, the aim is to find a single set of parameters for evaluating the hybrid approach, not to over-fit the set of parameters to every imaginable subset of instances.

The same line of reasoning is followed for the neighborhood selection probabilities p_s, p_m and p_p. For instances with a short event list $\mathcal{E}$ ($n \leq 15$), the best results are achieved when only the 'swap' operator is used. More disruptive operators, such as the '(paired) move', have an added value in instances with larger values of n. Therefore, to cover the larger ($n \geq 20$) instances with the same set of parameters, non-zero values for p_m and p_p have been chosen. For this purpose, we have evaluated the following combinations of $(p_s, p_m, p_p) \in \{(1.00, 0.00, 0.00), (0.90, 0.10, 0.00), (0.90, 0.05, 0.05), (0.75, 0.25, 0.00), (0.75, 0.15, 0.10)\}$.

In addition, since we consider the hybrid local search to be most relevant for the instances with $n = 50$, we allow restarts under certain conditions for these larger instances. If a run completes before 1800 seconds have passed, we perform 100 random swaps on the current event order, and restart the simulated annealing with the obtained order as the initial solution.

For solving both the MILP formulation and the LP subproblem of the simulated annealing algorithm we used CPLEX Studio 22.1, with a time limit of 3600 seconds,

	MILP		SA	
n	Avg. time	Avg. diff	Avg. time	Avg. diff
5	2.23	0.00%	85.63	0.00%
10	2453.67	0.15%	281.63	0.32%
15	3521.22	2.73%	532.74	0.12%

Table 2.5: Comparison of MILP formulation and simulated annealing algorithm on small instances

and limited to the use of a single thread. Everything else has been implemented in the Python programming language. The processor of the system used to run the tests on was an Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz.

2.5.2. Results

The full results of these runs can be found in Appendix 2.B.

In Table 2.5 we compare the performance of the MILP formulation and the simulated annealing on small instances ($n = 5, 10, 15$) in terms of run time and solution quality. This analysis is based on the results in Tables 2.8 - 2.10. Infeasible instances were left out of the analysis. The average runtime and difference are computed only over those instances where both approaches come up with a feasible solution. Note that the average runtime of the MILP for $n = 10, 15$ is somewhat misleading, as many runs have been cut off at 3600 seconds, resulting in a runtime equal to the cut-off time for these runs. The difference is defined to be the relative difference between the score of the best known solution and the reported solution.

It is relevant to point out that for both $n = 10$ and $n = 15$ there are two instances where the MILP is able to find a feasible solution, and the simulated annealing is not, while for $n = 15$ the simulated annealing finds a feasible solution for five instances where the MILP fails.

Figure 2.3 shows the runtime of the MILP and the hybrid approach ('SA'). For the MILP, only instances where a feasible solution was found are included in the average, and runs that were cut off after reaching the time limit are included as taking 3600 seconds. For the hybrid approach, the results for $n = 50$ have been excluded, as these runtimes include restarts. We can see that the runtime of the hybrid local search approach scales better than that of the MILP.

Our approach is competitive with the MILP on instances of size $n = 10$ and smaller. It finds equally good solutions, only taking more time for the smallest instances ($n = 5$). On instances of size $n = 15$ and above, the hybrid local search finds better solutions than the MILP in less time, if the MILP is able to find any feasible solution at all. The difference is likely to become even clearer if we would perform a number of reruns of the local search and only keep the best result, as we already did for the largest instances ($n = 50$). For $n = 15$, for example, about 6-7

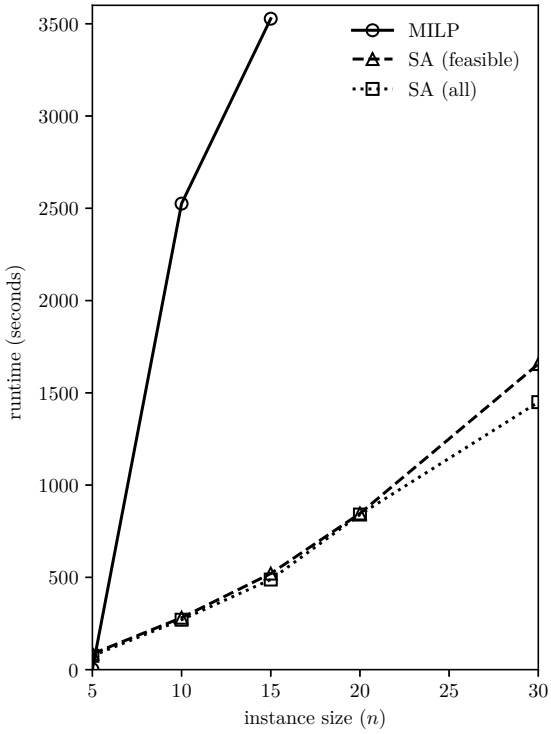


Figure 2.3: Average runtime of the compared approaches on increasing instance sizes

reruns can be performed without exceeding the time limit of 3600 that the MILP also has to respect.

We do see, however, that the performance of the hybrid approach drops with growing instance size as well. Table 2.6 summarizes the number of instances for which the (flow) feasibility test predicted a feasible solution exists, and presents the number of times a feasible solution and the best known (feasible) solution were found by the hybrid approach and the MILP formulation. Here, the value in brackets includes the instances where the hybrid approach found the best known *in*feasible solution, for instances where no feasible solution is known. This analysis is based on the results in Tables 2.8 - 2.13. By allowing restarts, we are able to find feasible solutions in the majority of cases, even for $n = 50$, but even larger instance sizes remain challenging.

Note that the best found solution for $n = 30$ is always the same as the reported solution for the hybrid local search algorithm. We added the instances of this size at a later stage to gain better insight, as the sizes of $n = 20$ and $n = 50$ differ strongly in their results. The reported solutions result from the only runs for these instances, and are therefore by default always the best known. For the instances of size $n = 20$ and $n = 50$, we observe that the number of times that our approach

	Flow	SA		MIP	
n	Feas.	Feas.	Best	Feas.	Best
5	28	28	28 (28)	28	28
10	32	30	26 (26)	32	23
15	32	30	15 (15)	27	4
20	31	30	8 (8)	-	-
30	32	28	28 (32)	-	-
50	32	21	7 (9)	-	-

Table 2.6: Number of feasible solutions found for each instance size

finds the best known solution strongly decreases. To put this in perspective, we note that the best known solution for these larger instances was often found during a trail run that took many hours. These do not offer a realistic alternative to the presented approach.

Analyzing the behavior of the hybrid approach on the four instances of size $n = 5$ that we know to be infeasible (see Table 2.8), a difference can be noted when compared to its behavior when dealing with feasible instances. It will still report the best found infeasible solution, but the time it spends before terminating the search is significantly shorter. From this, it cannot be concluded that an instance is infeasible whenever the search terminates quickly, but it does indicate that the algorithm tends to stop searching when there is no hope of further improvement.

Table 2.7 shows the relative difference between the best known solution and both the final and initial solutions of the hybrid local search. The results only include the instances for which a feasible solution was found, and are grouped by instance size. The groups are further split in the adverse and non-adverse instances (see Section 2.4). The last column shows the number of instances in each group. The intuition that sorting the weights of jobs in increasing order (the 'adverse' instances) would lead to worse initial solutions is proven wrong. In general, however, the initial solution that the greedy algorithm from Section 2.3.3 finds, provides a good starting point for the local search. For $n = 15$ it occasionally even comes up with a better solution than the MILP finds in an hour (see Table 2.10). In addition to that, the local search strongly improves the initial solution for all instance sizes.

2.6. Conclusion and future work

We apply a decomposition of the problem in two parts, where a local search algorithm is used to find event orders and an LP is used to find an optimal schedule for a given event order. Our hybrid local search approach matches the MILP formulation in solution quality for small instances ($n \leq 10$), and is able to find a feasible solution for larger instances ($15 \leq n \leq 50$) in reason-

n	adv.	Avg. diff. to best Reported	Initial	#
5	0	0.00%	34.57%	14
	1	0.00%	14.36%	14
10	0	0.20%	11.93%	15
	1	0.47%	12.73%	15
15	0	0.06%	15.00%	16
	1	0.07%	12.86%	14
20	0	0.00%	17.60%	15
	1	0.07%	18.93%	15
30	0	0.00%	27.42%	12
	1	0.00%	27.19%	16
50	0	7.00%	22.00%	11
	1	2.20%	25.70%	10

Table 2.7: Average relative distance for the reported solution and initial solution of the hybrid approach to the best known solution by group of instances

able time. This approach opens the door for finding solutions to larger instances. With further tuning and refinement of the current approach it is promising for finding solutions to instances of even larger sizes.

One of the avenues for future work would be exactly this, further tuning the approach to deal with larger instances. The addition of more preprocessing steps (e.g. identifying more implicit precedence relations) and the experimentation with different search strategies (e.g. adding some form of random restart) are promising directions for further investigation. The MILP formulation can be further strengthened by adding valid inequalities.

Finally, one could look at further extensions of the problem. Among those of interest are the addition of precedence relations and changing P from a constant into a variable resource availability function $P(t)$.

2.A. MILP formulation

The MILP formulation for the CECSP is as follows:

$$\min \sum_{j=1}^{n} (w_j t_{2j} + B_j) \qquad \text{s.t.}$$

$$\sum_{i=1}^{2n} p_{j,i} = E_j \qquad \forall j \in \{1, ..., n\}$$

$$t_{2j-1} \geq r_j \qquad \forall j \in \{1, ..., n\}$$

$$t_{2j} \leq \bar{d}_j \qquad \forall j \in \{1, ..., n\}$$

$$p_{j,i} \leq a_{i,2j} M \qquad \forall j \in \{1, ..., n\}, i \in \{1, ..., 2n\}$$

$$p_{j,i} \leq a_{2j-1,i} M \qquad \forall j \in \{1, ..., n\}, i \in \{1, ..., 2n\}$$

$$p_{j,i} \leq P_j^+ (t_{i'} - t_i) + M a_{i',i} \qquad \forall j \in \{1, ..., n\}, i, i' \in \{1, ..., 2n\}, i \neq i'$$

$$p_{j,i} \geq P_j^- (t_{i'} - t_i) - (1 - b_{i,i'} + a_{i,2j-1} + a_{2j,i'}) M \qquad \forall j \in \{1, ..., n\}, i, i' \in \{1, ..., 2n\}, i \neq i'$$

$$\sum_{j \in \{1,...,n\}} p_{j,i} \leq P(t_{i'} - t_i) + M a_{i',i} \qquad \forall i, i' \in \{1, ..., 2n\}, i \neq i'$$

$$t_i \leq t_{i'} + M a_{i',i} \qquad \forall i, i' \in \{1, ..., 2n\}, i \neq i'$$

$$a_{i,i'} + a_{i',i} = 1 \qquad \forall i, i' \in \{1, ..., 2n\}, i < i'$$

$$\sum_{i''=1}^{2n} a_{i,i''} - \sum_{i''=1}^{2n} a_{i',i''} \leq 1 + (1 - b_{i,i'}) M \qquad \forall i, i' \in \{1, ..., 2n\}, i \neq i'$$

$$\sum_{i''=1}^{2n} a_{i,i''} - \sum_{i''=1}^{2n} a_{i',i''} \geq 1 - (1 - b_{i,i'}) M \qquad \forall i, i' \in \{1, ..., 2n\}, i \neq i'$$

$$\sum_{i=1}^{2n} \sum_{i' \neq i} b_{i,i'} = 2n - 1$$

$$t_{2j} - t_{2j-1} \leq E_j / P_j^- \qquad \forall j \in \{1, ..., n\}$$

$$t_{2j} - t_{2j-1} \geq E_j / P_j^+ \qquad \forall j \in \{1, ..., n\}$$

$$p_{j,i} \geq 0 \qquad \forall j \in \{1, ..., n\}, i \in \{1, ..., 2n\}$$

$$t_i \geq 0 \qquad \forall i \in \{1, ..., 2n\}$$

$$a_{i,i'} \in \{0, 1\} \qquad \forall i, i' \in \{1, ..., 2n\}, i \neq i'$$

$$a_{i,i} = 0 \qquad \forall i \in \{1, ..., 2n\}$$

$$b_{i,i'} \in \{0, 1\} \qquad \forall i, i' \in \{1, ..., 2n\}, i \neq i'$$

2.B. Full Results

In the result tables provided in this appendix, the first four columns uniquely describe the instance. Columns 1-3 show the number of jobs n, resource availability P, and whether the weights are sorted in ascending order (adversarial, indicated as 'a'). As four instances of each type were generated, the fourth column identifies each instance with a number (0-3). We then list whether our feasibility test was passed by the instance or not (as described in Section 2.4.2).

This is followed by a column indicating the best objective value that we have encountered in any run of any of our algorithms. Note that this includes some extremely long runs done in the trail phase. Next we list the results of the MILP and hybrid local search algorithm (SA). For each we present the run time and best found objective value, while we also add the objective value of the initial solution (found by the greedy algorithm described in Section 2.3.3) for our hybrid local search approach.

Any values in *italics* indicate objective values corresponding to infeasible solutions. In the case of a solution found by our hybrid local search, this means that some slack variables have a non-zero value. An objective value of a solution found by the MILP or hybrid local search is put in **boldface** if it is equal to the best known objective value. Finally, we put `LIMIT` in the 'time' column under 'MILP' if the solver did not terminate within 3600 seconds, meaning it was unable to prove optimality of the current best found solution. We report an objective of *-1.00* if no feasible solution was found by the solver.

Note that the results for the hybrid local search in these tables are of a single run of each of the algorithms, they are not averages or best-of results.

Table 2.8: Computational results for $n = 5$

				Flow	Best	MILP		SA		
n	P	a	#	Feas.	known	time	obj	time	obj	init
5	25	0	0	yes	163.58	2.04	**163.58**	24.18	**163.58**	177.77
5	25	0	1	yes	165.72	1.58	**165.72**	189.19	**165.72**	212.33
5	25	0	2	yes	99.42	18.87	**99.42**	59.68	**99.42**	104.79
5	25	0	3	yes	78.70	1.92	**78.70**	80.97	**78.70**	97.96
5	25	1	0	yes	113.21	1.96	**113.21**	104.72	**113.21**	132.90
5	25	1	1	yes	61.94	6.30	**61.94**	91.03	**61.94**	68.71
5	25	1	2	yes	73.06	1.83	**73.06**	111.09	**73.06**	88.02
5	25	1	3	yes	96.81	6.36	**96.81**	95.73	**96.81**	98.34
5	50	0	0	yes	72.39	1.56	**72.39**	75.05	**72.39**	76.19
5	50	0	1	yes	88.61	0.87	**88.61**	103.08	**88.61**	94.87
5	50	0	2	yes	95.38	1.13	**95.38**	100.95	**95.38**	101.10
5	50	0	3	yes	80.81	1.38	**80.81**	130.12	**80.81**	92.27
5	50	1	0	yes	98.25	4.23	**98.25**	121.34	**98.25**	112.77
5	50	1	1	yes	74.93	2.24	**74.93**	101.55	**74.93**	82.42
5	50	1	2	yes	83.88	0.94	**83.88**	92.08	**83.88**	99.66
5	50	1	3	yes	102.10	2.08	**102.10**	91.60	**102.10**	102.95
5	100	0	0	yes	75.25	0.66	**75.25**	91.27	**75.25**	91.20
5	100	0	1	yes	77.71	0.51	**77.71**	36.22	**77.71**	175.64
5	100	0	2	yes	49.32	0.95	**49.32**	62.31	**49.32**	56.07
5	100	0	3	yes	51.97	0.54	**51.97**	64.66	**51.97**	111.72
5	100	1	0	yes	53.80	0.45	**53.80**	123.77	**53.80**	54.48
5	100	1	1	yes	69.92	0.71	**69.92**	43.58	**69.92**	76.33
5	100	1	2	yes	93.13	0.92	**93.13**	42.37	**93.13**	100.24
5	100	1	3	yes	53.79	0.64	**53.79**	39.44	**53.79**	55.46
5	200	0	0	yes	67.13	0.38	**67.13**	89.15	**67.13**	137.92
5	200	0	1	*no*	*120.94*	0.43	*-1.00*	0.24	*169.91*	169.91
5	200	0	2	*no*	*74.41*	0.45	*-1.00*	1.78	*74.44*	146.13
5	200	0	3	yes	57.02	0.48	**57.02**	86.05	**57.02**	59.86
5	200	1	0	yes	56.35	0.39	**56.35**	67.70	**56.35**	57.73
5	200	1	1	*no*	*81.35*	0.45	*-1.00*	1.62	*81.61*	148.08
5	200	1	2	yes	67.19	0.61	**67.19**	78.81	**67.19**	122.79
5	200	1	3	*no*	*229.58*	0.41	*-1.00*	1.16	*231.03*	330.74

Table 2.9: Computational results for $n = 10$

				Flow	Best	MILP		SA		
n	P	a	#	Feas.	known	time	obj	time	obj	init
10	25	0	0	yes	359.47	LIMIT	360.49	193.25	**359.47**	424.51
10	25	0	1	yes	*334.63*	LIMIT	337.72	206.95	*334.63*	424.35
10	25	0	2	yes	225.99	LIMIT	226.76	204.47	232.05	250.28
10	25	0	3	yes	300.08	LIMIT	**300.08**	243.97	**300.08**	392.84
10	25	1	0	yes	345.71	525.62	**345.71**	277.95	**345.71**	401.68
10	25	1	1	yes	394.69	3228.46	**394.69**	218.89	**394.69**	446.65
10	25	1	2	yes	399.06	LIMIT	**399.06**	245.50	**399.06**	493.58
10	25	1	3	yes	378.15	LIMIT	**378.15**	280.72	**378.15**	400.00
10	50	0	0	yes	193.57	LIMIT	**193.57**	293.98	**193.57**	207.17
10	50	0	1	yes	220.46	LIMIT	221.18	283.44	220.87	239.74
10	50	0	2	yes	254.38	1131.58	**254.38**	332.78	**254.38**	289.27
10	50	0	3	yes	191.05	LIMIT	**191.05**	248.19	**191.05**	213.67
10	50	1	0	yes	162.90	LIMIT	**162.90**	281.60	**162.90**	179.64
10	50	1	1	yes	171.90	LIMIT	172.04	298.19	**171.90**	174.68
10	50	1	2	yes	194.43	LIMIT	194.56	1.17	*552.53*	595.79
10	50	1	3	yes	289.80	LIMIT	**289.80**	256.29	303.90	388.97
10	100	0	0	yes	155.15	29.48	**155.15**	312.78	**155.15**	162.38
10	100	0	1	yes	139.62	LIMIT	**139.62**	308.31	**139.62**	148.31
10	100	0	2	yes	203.37	LIMIT	**203.37**	284.11	**203.37**	211.90
10	100	0	3	yes	168.99	LIMIT	171.55	306.39	**168.99**	194.37
10	100	1	0	yes	205.44	LIMIT	**205.44**	348.42	**205.44**	219.86
10	100	1	1	yes	171.26	LIMIT	171.42	393.40	**171.26**	225.93
10	100	1	2	yes	143.83	LIMIT	**143.83**	253.98	**143.83**	163.53
10	100	1	3	yes	145.66	LIMIT	148.32	282.35	148.32	162.14
10	200	0	0	yes	125.38	181.10	**125.38**	221.56	**125.38**	130.17
10	200	0	1	yes	138.32	4.58	**138.32**	309.86	**138.32**	185.09
10	200	0	2	yes	107.74	33.39	**107.74**	276.53	**107.74**	113.15
10	200	0	3	yes	100.16	4.42	**100.16**	314.35	**100.16**	104.39
10	200	1	0	yes	158.68	18.55	**158.68**	307.19	**158.68**	182.99
10	200	1	1	yes	135.62	13.38	**135.62**	443.79	**135.62**	136.47
10	200	1	2	yes	108.72	7.00	**108.72**	193.36	**108.72**	109.42
10	200	1	3	yes	101.19	LIMIT	**101.19**	233.21	**101.19**	106.54

Table 2.10: Computational results for $n = 15$

				Flow	Best	MILP		SA		
n	P	a	#	Feas.	known	time	obj	time	obj	init
15	25	0	0	yes	831.86	LIMIT	897.55	538.08	**831.86**	871.10
15	25	0	1	yes	587.90	LIMIT	*-1.00*	390.93	587.96	752.17
15	25	0	2	yes	779.20	LIMIT	790.29	399.92	**779.20**	985.93
15	25	0	3	yes	800.74	LIMIT	825.81	417.82	**800.74**	997.23
15	25	1	0	yes	733.29	LIMIT	770.22	326.79	**733.29**	882.03
15	25	1	1	yes	805.43	LIMIT	898.95	536.96	811.87	953.44
15	25	1	2	yes	649.83	LIMIT	*-1.00*	563.08	**649.83**	847.58
15	25	1	3	yes	561.05	LIMIT	572.22	402.05	561.98	662.60
15	50	0	0	yes	542.53	LIMIT	*-1.00*	448.71	542.57	634.80
15	50	0	1	yes	309.19	LIMIT	311.46	544.40	**309.19**	363.38
15	50	0	2	yes	353.07	LIMIT	358.26	411.76	353.08	382.82
15	50	0	3	yes	391.34	LIMIT	*-1.00*	462.41	391.48	611.09
15	50	1	0	yes	322.26	LIMIT	325.22	520.40	**322.26**	349.78
15	50	1	1	yes	475.77	LIMIT	492.32	2.18	*793.14*	793.14
15	50	1	2	yes	589.86	LIMIT	*-1.00*	436.84	**589.86**	626.66
15	50	1	3	yes	347.26	LIMIT	349.15	392.01	**347.26**	399.45
15	100	0	0	yes	235.75	LIMIT	246.03	427.83	239.26	250.06
15	100	0	1	yes	330.64	LIMIT	337.54	507.91	331.21	338.12
15	100	0	2	yes	314.33	LIMIT	321.52	643.93	**314.33**	348.71
15	100	0	3	yes	288.97	LIMIT	289.10	479.21	**288.97**	342.44
15	100	1	0	yes	279.48	LIMIT	291.05	542.24	279.58	299.08
15	100	1	1	yes	341.46	LIMIT	345.56	671.86	**341.46**	408.68
15	100	1	2	yes	215.79	LIMIT	230.71	2.30	*520.72*	520.72
15	100	1	3	yes	259.86	LIMIT	265.08	546.11	**259.86**	296.06
15	200	0	0	yes	168.23	LIMIT	190.76	579.06	168.24	181.16
15	200	0	1	yes	253.88	LIMIT	261.80	881.63	**253.88**	259.09
15	200	0	2	yes	143.38	LIMIT	144.71	702.94	143.51	151.02
15	200	0	3	yes	237.71	1496.19	**237.71**	549.86	**237.71**	243.81
15	200	1	0	yes	179.79	LIMIT	179.86	646.97	179.85	199.15
15	200	1	1	yes	198.08	LIMIT	**198.08**	526.01	198.10	207.30
15	200	1	2	yes	160.21	LIMIT	**160.21**	565.79	160.34	165.41
15	200	1	3	yes	244.38	LIMIT	**244.38**	556.86	244.41	254.76

Table 2.11: Computational results for $n = 20$

				Flow	Best	SA		
n	P	a	#	Feas.	known	time	obj	init
20	25	0	0	yes	1122.08	862.19	1125.55	1643.66
20	25	0	1	yes	1343.89	685.22	**1343.89**	1771.98
20	25	0	2	yes	1012.18	877.71	**1012.18**	1197.75
20	25	0	3	*no*	*1445.11*	732.78	*1518.36*	1784.42
20	25	1	0	yes	991.01	776.51	*995.64*	1195.65
20	25	1	1	yes	1316.27	797.72	**1316.27**	1768.37
20	25	1	2	yes	1094.52	549.41	**1094.52**	1399.41
20	25	1	3	yes	991.20	702.07	**991.20**	1123.76
20	50	0	0	yes	657.76	752.77	**657.76**	827.27
20	50	0	1	yes	765.34	712.16	766.59	951.32
20	50	0	2	yes	590.19	809.64	**590.19**	620.57
20	50	0	3	yes	729.51	710.34	**729.51**	1049.81
20	50	1	0	yes	571.33	678.59	571.48	635.55
20	50	1	1	yes	625.52	674.42	628.49	755.16
20	50	1	2	yes	548.63	565.51	548.74	737.98
20	50	1	3	yes	551.01	603.31	554.73	610.62
20	100	0	0	yes	301.95	989.00	302.12	314.61
20	100	0	1	yes	476.50	810.92	476.71	527.38
20	100	0	2	yes	363.01	902.16	363.12	443.13
20	100	0	3	yes	488.63	1461.01	488.68	506.45
20	100	1	0	yes	365.24	980.08	365.31	501.94
20	100	1	1	yes	493.23	1072.18	493.28	584.07
20	100	1	2	yes	415.11	1185.18	415.24	454.11
20	100	1	3	yes	359.19	664.77	359.25	482.25
20	200	0	0	yes	303.41	818.06	303.78	309.97
20	200	0	1	yes	310.99	883.86	311.31	364.42
20	200	0	2	yes	345.24	1064.08	345.28	367.84
20	200	0	3	yes	285.33	796.83	285.70	290.13
20	200	1	0	yes	305.00	1095.02	305.06	334.49
20	200	1	1	yes	337.89	753.72	337.97	346.15
20	200	1	2	yes	313.49	828.79	313.72	332.67
20	200	1	3	yes	318.78	1098.68	318.96	365.87

Table 2.12: Computational results for $n = 30$

n	P	a	#	Flow Feas.	Best known	SA time	SA obj	SA init
30	25	0	0	yes	2088.94	1502.01	**2088.94**	2599.09
30	25	0	1	yes	*3134.71*	5.58	***3134.71***	3211.39
30	25	0	2	yes	2291.27	1406.69	**2291.27**	2744.97
30	25	0	3	yes	2082.82	1424.99	**2082.82**	2860.31
30	25	1	0	yes	2920.08	1142.54	**2920.08**	3380.82
30	25	1	1	yes	2963.77	1483.27	**2963.77**	3870.67
30	25	1	2	yes	3436.78	29.96	**3436.78**	3622.45
30	25	1	3	yes	3237.29	1760.39	**3237.29**	4677.28
30	50	0	0	yes	*2426.46*	5.62	***2426.46***	2476.53
30	50	0	1	yes	*2538.60*	5.75	***2538.60***	2589.17
30	50	0	2	yes	1052.26	1728.24	**1052.26**	1254.98
30	50	0	3	yes	*2019.35*	5.69	***2019.35***	2020.03
30	50	1	0	yes	1660.95	1601.76	**1660.95**	2041.24
30	50	1	1	yes	1732.74	1614.14	**1732.74**	2391.34
30	50	1	2	yes	1778.19	1381.26	**1778.19**	2190.11
30	50	1	3	yes	1703.09	308.90	**1703.09**	2080.97
30	100	0	0	yes	747.01	1591.51	**747.01**	1482.49
30	100	0	1	yes	904.57	1628.79	**904.57**	1262.40
30	100	0	2	yes	779.62	2140.44	**779.62**	929.26
30	100	0	3	yes	865.12	1481.99	**865.12**	1160.84
30	100	1	0	yes	836.81	2110.24	**836.81**	1118.03
30	100	1	1	yes	976.40	1890.41	**976.40**	1419.84
30	100	1	2	yes	754.56	1861.07	**754.56**	1085.14
30	100	1	3	yes	695.52	1690.72	**695.52**	902.68
30	200	0	0	yes	497.55	2059.61	**497.55**	549.87
30	200	0	1	yes	486.30	1826.58	**486.30**	513.94
30	200	0	2	yes	509.32	1685.78	**509.32**	552.84
30	200	0	3	yes	538.68	2378.16	**538.68**	604.61
30	200	1	0	yes	468.25	2631.23	**468.25**	623.84
30	200	1	1	yes	586.33	1823.30	**586.33**	715.84
30	200	1	2	yes	619.83	1826.30	**619.83**	645.95
30	200	1	3	yes	568.19	2356.85	**568.19**	687.19

Table 2.13: Computational results for $n = 50$

				Flow	Best	SA		
n	P	a	#	Feas.	known	time	obj	init
50	25	0	0	yes	6002.37	4293.20	6777.69	7958.75
50	25	0	1	yes	6573.68	4665.56	7245.46	7509.22
50	25	0	2	yes	*6423.54*	5003.16	***6423.54***	7689.15
50	25	0	3	yes	7027.25	5916.94	**7027.25**	8145.16
50	25	1	0	yes	6562.65	3602.98	*7196.96*	7217.22
50	25	1	1	yes	6257.97	5594.27	*7278.21*	7520.76
50	25	1	2	yes	*7239.58*	6254.32	***7239.58***	8007.63
50	25	1	3	yes	7911.54	5082.28	7925.13	8873.06
50	50	0	0	yes	*4176.25*	5303.19	4469.83	5420.81
50	50	0	1	yes	*5837.52*	3731.27	*5857.73*	5871.31
50	50	0	2	yes	3167.27	2318.80	*3756.45*	4459.79
50	50	0	3	yes	3429.76	3484.34	4080.28	4731.48
50	50	1	0	yes	3503.36	6897.78	3516.58	4041.55
50	50	1	1	yes	3081.82	6070.92	**3081.82**	4950.52
50	50	1	2	yes	*4326.41*	2764.69	*4499.39*	4509.67
50	50	1	3	yes	3548.99	3541.65	*4359.87*	4750.75
50	100	0	0	yes	1609.52	8184.16	**1609.52**	1928.63
50	100	0	1	yes	2205.73	8174.82	**2205.73**	2588.95
50	100	0	2	yes	1658.84	3838.16	*2074.60*	2127.92
50	100	0	3	yes	1947.59	2484.76	2282.36	2622.42
50	100	1	0	yes	1628.10	4110.64	1933.85	2240.80
50	100	1	1	yes	1716.27	9738.11	**1716.27**	2096.80
50	100	1	2	yes	1637.18	6344.71	**1637.18**	2402.85
50	100	1	3	yes	1880.46	6946.31	**1880.46**	2358.75
50	200	0	0	yes	1139.72	2123.86	*1384.30*	1415.43
50	200	0	1	yes	1237.01	2861.69	1352.59	1352.59
50	200	0	2	yes	1298.67	12194.00	1306.81	1439.92
50	200	0	3	yes	1211.63	9607.54	1219.67	1442.92
50	200	1	0	yes	1052.00	13363.99	1060.47	1169.24
50	200	1	1	yes	1265.67	2003.09	*1458.85*	1459.74
50	200	1	2	yes	1060.45	9351.82	1070.15	1223.95
50	200	1	3	yes	1095.45	8013.34	1103.34	1216.94

References

Baptiste, P., C. L. Pape, and W. Nuijten (1999). "Satisfiability tests and time-bound adjustments for cumulative scheduling problems". *Annals of Operations Research* 92, pp. 305–333. DOI: 10.1023/A:1018995000688.

Brouwer, R. J. J. (2022a). *CECSP data*. Zenodo. DOI: 10.5281/zenodo.10051616. URL: https://github.com/RoelBrouwer/2022-cecsp-data/tree/jos.

Brouwer, R. J. J. (July 9, 2022b). *GCECSP: models and algorithms*. Version 0.1.0. DOI: 10.5281/zenodo.10037572. URL: https://github.com/RoelBrouwer/continuousresource/tree/jos.

Brouwer, R. J. J., J. M. van den Akker, and J. A. Hoogeveen (2023). "A hybrid local search algorithm for the continuous energy-constrained scheduling problem". Preprint at: https://arxiv.org/abs/2311.16177. DOI: 10.48550/arXiv.2311.16177.

Hartmann, S. and D. Briskorn (2010). "A survey of variants and extensions of the resource-constrained project scheduling problem". *European Journal of Operational Research* 207 (1), pp. 1–14. DOI: 10.1016/j.ejor.2009.11.005.

Hartmann, S. and D. Briskorn (2022). "An updated survey of variants and extensions of the resource-constrained project scheduling problem". *European Journal of Operational Research* 297 (1), pp. 1–14. DOI: 10.1016/j.ejor.2021.05.004.

Koné, O., C. Artigues, P. Lopez, and M. Mongeau (2011). "Event-based MILP models for resource-constrained project scheduling problems". *Computers & Operations Research* 38, pp. 3–13. DOI: 10.1016/j.cor.2009.12.011.

Kopanos, G. M., T. S. Kyriakidis, and M. C. Georgiadis (2014). "New continuous-time and discrete-time mathematical formulations for resource-constrained project scheduling problems". *Computers and Chemical Engineering* 68, pp. 96–106. DOI: 10.1016/j.compchemeng.2014.05.009.

Naber, A. (2017). "Resource-constrained project scheduling with flexible resource profiles in continuous time". *Computers and Operations Research* 84, pp. 33–45. DOI: 10.1016/j.cor.2017.02.018.

Nattaf, M., C. Artigues, and P. Lopez (2014). "A polynomial satisfiability test using energetic reasoning for energy-constraint scheduling". *Proceedings of the International Conference on Project Management and Scheduling (PMS 2014)*, pp. 169–172. URL: https://hal.archives-ouvertes.fr/hal-00978706.

Nattaf, M., C. Artigues, and P. Lopez (Oct. 2017). "Cumulative scheduling with variable task profiles and concave piecewise linear processing rate functions". *Constraints* 22 (4), pp. 530–547. DOI: 10.1007/S10601-017-9271-4.

Nattaf, M., C. Artigues, P. Lopez, and D. Rivreau (Mar. 2016). "Energetic reasoning and mixed-integer linear programming for scheduling with a continuous resource and linear efficiency functions". *OR Spectrum* 38 (2), pp. 459–492. DOI: 10.1007/S00291-015-0423-X.

Nattaf, M., M. Horváth, T. Kis, C. Artigues, and P. Lopez (2019). "Polyhedral results and valid inequalities for the continuous energy-constrained scheduling problem". *Discrete Applied Mathematics* 258, pp. 188–203. DOI: 10.1016/j.dam.2018.11.008.

Turek, J., J. L. Wolf, and P. S. Yu (1992). "Approximate algorithms for scheduling parallelizable tasks". *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 323–332. DOI: 10.1145/140901.141909.

3

A hybrid optimization framework for the General Continuous Energy-Constrained Scheduling Problem

The contents of this chapter have been submitted to the Annals of Operations Research. A preprint is available on arXiv:

R. J. J. Brouwer, J. M. van den Akker, and J. A. Hoogeveen (2024). "A hybrid optimization framework for the general continuous energy-constrained scheduling problem". Preprint at: https://arxiv.org/abs/2403.03039. DOI: 10.48550/arXiv.2403.03039

Abstract

We present a hybrid optimization framework for a class of problems, formalized as a generalization of the Continuous Energy-Constrained Scheduling Problem (CECSP), introduced by Nattaf, Artigues, and Lopez (2014). This class is obtained from challenges concerning demand response in energy networks. Our framework extends a previously developed approach (Brouwer, van den Akker, and Hoogeveen 2023; Chapter 2). A set of jobs has to be processed on a continuous, shared resource. Consequently, a schedule for a job does not only contain a start and completion time, but also a resource consumption profile, where we have to respect lower and upper bounds on resource consumption during processing. In this work, we develop a hybrid approach for the case where the objective is a step-wise increasing function of completion time, using local search, linear programming and $O(n)$ lower bounds. We exploit that the costs are known in the local search and use bounds to assess feasibility more efficiently than by LP. We compare its performance to a mixed-integer linear program. After that, we extend this to a hybrid optimization framework for the General CECSP. This uses an event-based model, and applies a decomposition in two parts: 1) determining the order of events and 2) finding the event times, and hence the start and completion times of jobs, together with the resource consumption profiles. We argue the broad applicability of this framework.

Keywords: *continuous scheduling, resource-constrained scheduling, mathematical programming, local search, decomposition.*

3.1. Introduction

In this chapter, we study scheduling problems related to demand response in energy networks. For example, when charging the battery of an e-vehicle, it must be provided with enough energy to fill it up to capacity, but the exact charging time can be freely determined as long as charging takes place between the arrival and departure time of the vehicle. Moreover, we are not necessarily limited to a fixed rate for charging the battery, although we want to take into account that preemption of charging may be undesirable, due to the response time of batteries.

This leads us to consider scheduling problems where we are primarily interested in the (cumulative) amount of work (or resource) required to complete a job, while the rate of consumption may vary within a given range. Such problems require a different approach than the more traditional ones, as both resources and time are continuously-divisible, and jobs are flexible (to a certain extent) along both axes. This type of problem, that we will refer to as the General Continuous Energy-Constrained Scheduling Problem (GCECSP), has not yet been studied extensively.

The GCECSP is described as follows. A set $\{J_1, \ldots, J_n\}$ of jobs has to be processed on a continuous, shared resource R. This means that, at any time, multiple jobs can be processed simultaneously and at different rates, as long as their total consumption does not exceed the available resource capacity P. A schedule for a job J_j does not only contain a start and completion time (respecting the release time r_j and deadline $\bar{d}_j$), but also a resource consumption profile $p_j(t)$, where we have to respect lower and upper bounds (P_j^-, P_j^+) on resource consumption during processing. The total consumption of job J_j should equal its cumulative requirement E_j. Preemption is not allowed: from its start until its completion, each job must consume resources at a rate of at least $P_j^- > 0$ units.

The GCECSP is a generalization of the Continuous Energy-Constrained Scheduling Problem (CECSP) that we studied in our previous work (Brouwer, van den Akker, and Hoogeveen 2023; Chapter 2) and was originally introduced by Nattaf, Artigues, and Lopez (2014). The CECSP is a generalization of the Cumulative Scheduling Problem (CuSP) presented by Baptiste, Pape, and Nuijten (1999). In the case of the CuSP, a number of activities have to be scheduled on a single shared resource with a given capacity. Each activity has a release time, deadline, (fixed) processing time and (constant) resource capacity requirement. The CuSP was formulated as a subproblem of the Resource Constrained Project Scheduling Problem (RCPSP), relaxing precedence constraints and considering only a single resource. The RCPSP is a very general problem that concerns the scheduling of activities subject to precedence, time and resource constraints. The surveys by Hartmann and Briskorn (2010, 2022) provide a good overview of the RCPSP and its extensions. Most closely related to the present work, continuous and event-based formulations of the RCPSP have been studied and evaluated by Koné et al. (2011) and Kopanos, Kyriakidis, and Georgiadis (2014) as well as more general

models with flexible resource profiles in continuous time (FRCPSP), for example by Naber (2017).

For the CECSP including efficiency functions that apply to the conversion of the amount of consumed resource to the contribution towards the resource requirement of a job, with minimizing resource consumption as the objective, Nattaf, Artigues, and Lopez (2017) and Nattaf, Horváth, et al. (2019) provided several algorithms aiming to find exact solutions for small instances. The CECSP generalizes the CuSP as the resource capacity requirement is considered to be a range with a lower and an upper bound, rather than a fixed value, and the consumption rate can vary during the execution of the job. As a result, the processing time can vary, depending on the consumption rate during execution. Through its relation to CuSP, Nattaf, Artigues, Lopez, and Rivreau (2016) proved that the problem of finding a feasible solution for CECSP is NP-complete.

The GCECSP is also closely related to the scheduling of malleable jobs (as introduced by Turek, Wolf, and Yu (1992)) on parallel machines, which involves the scheduling of jobs on P machines, while the number of machines assigned to a job can change during its execution. The machines can be viewed as a discretized resource.

A wide range of objectives can be considered in variants of the GCECSP. In this work, we explore the GCECSP with step-wise cost function in detail. Problems with a generalized step-wise cost function have not been widely studied. In scheduling literature, Detienne, Dauzère-Pérès, and Yugma (2011, 2012) study single $(1||\bar{f}(C_j))$ and parallel $(R|r_j|\bar{f}_j(C_j))$ machine scheduling problems with a regular step-wise cost function. The single machine problem is proven to be NP-hard in the strong sense. For both the single machine and the parallel unrelated machine problem, Detienne, Dauzère-Pérès, and Yugma introduce a dominance relation, proving that if there exists at least one feasible schedule in which each job completes before its deadline, then the schedules where jobs are sequenced in non-decreasing order of their deadlines are feasible. Tseng, Chou, and Chou (2010) study a similar problem on a single machine, but refer to it as *stepwise tardiness*.

Our contribution is twofold. (1) We present an optimization framework for the GCECSP and related problems, based on our previously developed hybrid local search approach Brouwer, van den Akker, and Hoogeveen 2023, using simulated annealing and linear programming. (2) As an example, we detail how our approach can be applied to a variant with a step-wise constant objective function, where we exploit the properties of the objective to improve the efficiency of the approach.

The rest of this work is structured as follows. First, we will give a detailed problem description in Section 3.2. Then, we will introduce our framework in Section 3.3, using an interesting variant of the problem with step-wise cost functions as an example throughout. In this section, we will discuss the event-based model, a proposed decomposition of the problem, and the implementation of our solution approach. Our approach uses local search, linear programming and a number

of algorithms for approximating the penalty terms, to avoid solving the LP in every iteration. We continue with an overview of variants of the GCECSP, and a discussion on the applicability of our framework to these variants in Section 3.4. In Section 3.5, we will describe the generation of test instances for the problem introduced in Section 3.3 and present the computational results of applying our approach to them. Finally, we will draw conclusions in Section 3.6. An overview of the notation used in this chapter can be found at the start of this part, on page 17.

3.2. Problem description

We consider a wide range of scheduling problems. The common basis of these problems, which we will call the General Continuous Energy-Constrained Scheduling Problem (GCECSP) from here on, is defined by the following properties:

- A resource R with a constant availability of P;
- For each job J_j, $j \in \{1, ..., n\}$:
  - Resource requirement E_j;
  - Release time r_j;
  - Deadline $\bar{d}_j$;
  - Lower bound P_j^-;
  - Upper bound P_j^+.

For each job, we need to determine a start time S_j, a completion time C_j and a resource consumption profile $p_j(t)$ (a function that describes the amount of resource that job J_j consumes over time) such that *some objective* (most commonly a function of C_j) is minimized, while the following constraints are respected:

C1 The total amount of resource job J_j consumes is exactly E_j, i.e. $\int_t p_j(t)\, dt = E_j$;

C2 Job J_j does not start before its release time r_j, i.e. $S_j \geq r_j$;

C3 Job J_j completes no later than its deadline $\bar{d}_j$, i.e. $C_j \leq \bar{d}_j$;

C4 Job J_j only consumes resources between its start and completion time, i.e. $p_j(t) = 0$ for $t < S_j$ or $t \geq C_j$;

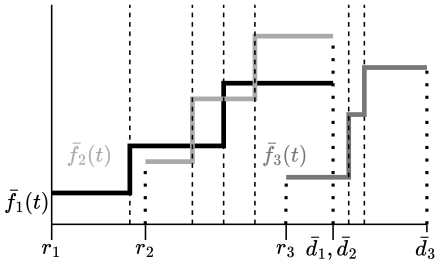
C5 While active, the resource consumption of job J_j never drops below P_j^- or rises above P_j^+, i.e. $P_j^- \leq p_j(t) \leq P_j^+$ for $S_j \leq t \leq C_j$;

C6 The total amount of resource consumed by all jobs together at any given time can never exceed P, i.e. $\sum_j p_j(t) \leq P$ for all possible values of t.

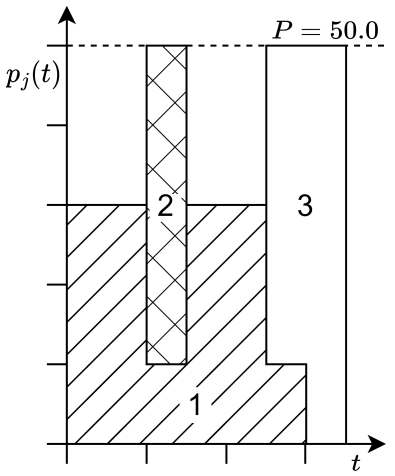
An example instance with three jobs and $P = 50.00$ is provided in Figure 3.1. Figure 3.1a lists the properties of all jobs and Figure 3.1c visualizes a feasible

j	E_j	r_j	$\bar{d}_j$	P_j^-	P_j^+
1	70.0	0.0	3.0	10.0	30.0
2	20.0	1.0	3.0	10.0	40.0
3	45.0	2.5	4.0	10.0	50.0

(a) Job properties



(b) Cost functions



(c) Visualization of a feasible schedule

Figure 3.1: Example instance with three jobs

schedule for this instance. The cost functions in Figure 3.1b are step-wise cost functions, which will be discussed in detail in Section 3.3.

3.3. Hybrid optimization framework

We will introduce our optimization framework by applying it to one particularly interesting variant. In this variant, we want to minimize an increasing step-wise cost function $\bar{f}_j(C_j)$, that determines the cost of completing a job J_j at a given time. Such a function consists of k pieces, each spanning an interval where the cost of completing a job does not change. Essentially, a job now has $k-1$ due dates, or *jump points*, after each of which the cost of completing the job increases, and a single deadline. The jump points are chosen independently for each job. Figure 3.1b shows example cost functions for the example instance, with two jump points each ($k = 3$).

3.3.1. Event-based model

Recall that a schedule consists of, for every job J_j: a start time S_j, a completion time C_j and a resource consumption profile $p_j(t)$. So, for every job, the timing of two *events* has to be determined, which leads to a total of $2n$ events. A sequence of these events $\mathcal{E}$ is the basis of any schedule. We will call these events *plannable*, as we need to decide their exact timing. We denote the (ordered) set of plannable events as $\mathcal{P}$.

In most cases, we only need these $2n$ events in our model. For the step-wise cost functions, however, we introduce another event type: *fixed-time* events. These

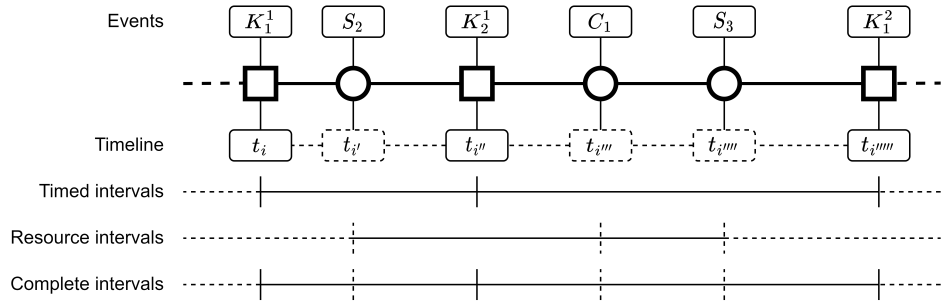


Figure 3.2: Graphical representation of part of an event order and associated intervals

are events of which the exact timing is already known. In this case, the $k-1$ jump points $\{K_j^1, ..., K_j^{k-1}\}$ of each job are added to the event list $\mathcal{E}$ as fixed-time events, bringing the total to $n(k+1)$ events. We denote the (ordered) set of fixed-time events as $\mathcal{F}$. Including them in the order of events has some benefits for determining the objective value of a potential solution during our algorithm, which will be explained in Section 3.3.3. From a given order of events $\mathcal{E}$, we can identify $2n-1$ *resource intervals*, each defined by two consecutive plannable events in our sequence and $n(k-1)-1$ *timed intervals*, each defined by two consecutive fixed-time events. Part of an example event order and associated intervals is given in Figure 3.2. Note that, whenever intervals are mentioned without further qualification in the following, these are *resource intervals*.

The following theorem is a reformulation of the theorem proven by Brouwer, van den Akker, and Hoogeveen (2023; Chapter 2) for the case where the objective is to minimize weighted completion time, but the proof can be modified to fit any objective that is a function of (plannable) event time.

Theorem 3.1. *For any feasible schedule $\mathcal{S}$ with resource consumption profiles $p_j(t)$ for all jobs $J_j, j \in \{1, ..., n\}$ and objective value W that follows a given event order $\mathcal{E}$, a feasible schedule $\mathcal{S}'$ exists with the same order of events $\mathcal{E}$ and objective value W where the resource consumption $p'_j(t)$ of all jobs remains constant during each interval.* □

Theorem 3.1 shows that we can simplify the resource consumption profile function $p_j(t)$ to a discrete number of values $p_{j,i}$, indicating the amount of resource each job J_j consumes during interval $i, i \in \{1, ..., 2n-1\}$. These values $p_{j,i}$ together with event times t_i fully describe the schedule.

The GCECSP can be solved using a mixed-integer linear program, where, besides t_i and $p_{j,i}$, additional binary variables are used to model the order of events. We adapt the formulation presented by Brouwer, van den Akker, and Hoogeveen (2023; Chapter 2) for the variant with step-wise cost functions.

The full MILP formulation is presented in Appendix 3.A. Note that the indexing scheme of events is similar to that in Chapter 2. The first $2n$ indices are reserved

for plannable events, where $i = 2j - 1$ is the start event of job J_j, and $i = 2j$ its completion. The fixed-time events follow, where the jth $k - 1$ indices (i.e., $2n+(k-1)(j-1)+1, ..., 2n+j(k-1)$) belong to the jump points of job J_j. Recall that the indices of intervals match that of the plannable event that occurs at the start of that interval.

3.3.2. Decomposition and local search

We observe that, after fixing the binary variables in the MILP, we are left with an LP that quickly finds an optimal schedule for a given order of events. This inspires the decomposition of the problem in two parts: (1) finding a good event order $\mathcal{E}$, and (2) determining an optimal schedule for a given event order. This idea forms the basis of the approach that we developed in our previous work (Brouwer, van den Akker, and Hoogeveen 2023; Chapter 2), where we use simulated annealing to explore permutations of the order of events $\mathcal{E}$, with neighborhood operators that swap two adjacent events and occasionally move an event multiple positions forwards or backwards in the event order. In every iteration of the simulated annealing, we evaluate the objective by solving an LP to find the optimal schedule for the given event order, defining event times t_i and resource consumption profiles $p_{j,i}$.

This approach generalizes well to the GCECSP. It is a shared property of the GCECSP and its variants that it is difficult to find a good event order $\mathcal{E}$, while finding a schedule for a given event order is relatively easy. The general approach can be applied to the GCECSP and its variants, while some elements, such as the neighborhood operators or the means of assessing the quality of event orders, may need to be tailored in each case. This tailoring, however, may also open up opportunities for increasing the performance of the approach. This is the case for the variant with step-wise cost functions, as will become clear in the remainder of this chapter.

In the remainder of this section, we will discuss how to tailor the general approach for the variant with step-wise cost functions. We need to make a number of changes: (1) we adapt the greedy algorithm for finding an initial solution; (2) we extend the set of implicit precedence constraints that we consider in the local search; (3) because of the increased number of events for an instance of the same size, when compared to the general case, we adapt the neighborhood operator to generate neighbors by moving an event multiple positions more often; (4) to avoid spending a lot of time on a plateau, we terminate the search if the number of iterations since the last improvement of the current solution exceeds one cooling period (see Section 3.5.1); (5) we change the way the quality of event orders is assessed, which will be explained in Section 3.3.3. Because we use fixed-time events to model jump points in the cost function, the value of the objective function is fully determined by the event order. However, we still need to determine whether a feasible solution exists for that particular event order.

3.3.3. Assessing the quality of event orders

In the general approach, an LP is used to solve the subproblem, i.e. for determining an optimal schedule for the given event order (if one exists). In order to give the simulated annealing search enough flexibility, we allow it to search through the space of infeasible solutions. To evaluate infeasible candidate solutions, we include three types of so-called *violation variables* ($s^+_{j,i}, s^-_{j,i}, s^t_i \geq 0$) that allow for the violation of the constraints enforcing the (1) upper and (2) lower bound on the resource consumption for a job, and (3) the upper bound on the amount of available resource during an interval at the cost of a penalty in the objective. If the sum of the violation variables is larger than 0, no feasible solution exists for that order, but we can still view a decrease in the penalty term as a move in the right direction.

While the search is ongoing, however, it is sufficient if we know the objective value, including the penalty term. We do not need the schedule, i.e. the exact values of all event times t_i and resource consumption during intervals $p_{j,i}$. For the case with step-wise cost functions, this means that we may not have to solve the LP every time. The value of the objective (excluding the penalty term) is fully determined by the order of events. For each completion event C_j in the event order, let $K^{\rightarrow}_j$ denote the next jump point of job J_j in the event order (where $K^{\rightarrow}_j = \bar{d}_j$ if no further jump point occurs). Then, any *feasible* schedule with the given event order will have a cost of exactly $\sum_j \bar{f}_j(K^{\rightarrow}_j)$.

At this point, we exploit that this base score may already be sufficient to reject a candidate solution. We precompute the maximum objective value that we will accept during the evaluation. If the base score already exceeds this, we will not compute the penalty term at all.

For estimating the penalty term, we propose three alternatives:

(1) solve the LP, which will always yield the true value of the penalty term;

(2) compute a maximum flow on a bipartite graph, modeling a relaxed version of the problem;

(3) compute lower bounds on the value of the violation variables using only the event order.

The results of extensive tests of using each of the three methods for computing the penalty term are presented in Section 3.5.1.

Solving the LP

The LP is, as before, what remains if we fix the order variables in the MILP and add violation variables to the relevant constraints. The objective is then to minimize the weighted sum of the violation variables.

We show the full LP formulation below. For ease of readability, we introduce some additional notation:

- $X_{j,i} = \begin{cases} 1 & \text{if } I(t_{2j-1}) \leq I(t_i) \text{ and } I(t_{2j}) \geq I(t_i) \\ 0 & \text{otherwise} \end{cases}$ defines if a job J_j is active during interval i, which we already know, based on the order of events;
- $\text{succ}(i) = E(I(i)+1)$, the index of the successor of event i in the order of events $\mathcal{E}$;
- $t_{\text{succ}(i)} = \infty$ for the last event in $\mathcal{E}$;
- $\text{succ}_{\mathcal{P}}(i)$ is the same as $\text{succ}(i)$, but skips over fixed-time events;
- $p_{j,i} = 0, \forall j$ for the last event in the order of events $\mathcal{E}$.

$$\min \sum_{i=1}^{2n} \left(L^R s_i^t + \sum_{j=1}^{n} L^B (s_{j,i}^- + s_{j,i}^+) \right) \quad \text{s.t.}$$

$$\begin{aligned}
t_i &\leq t_{\text{succ}(i)} && \forall i \in \{1, ..., (k+1)n\} \\
t_{2j} &\leq \bar{d}_j && \forall j \in \{1, ..., n\} \\
t_{2j-1} &\geq r_j && \forall j \in \{1, ..., n\} \\
\sum_{i=1}^{2n} p_{j,i} &= E_j && \forall j \in \{1, ..., n\} \\
p_{j,i} &\geq P_j^- X_{j,i} (t_{\text{succ}_{\mathcal{P}}(i)} - t_i) - s_{j,i}^- && \forall j \in \{1, ..., n\}, i \in \{1, ..., 2n\} \\
p_{j,i} &\leq P_j^+ X_{j,i} (t_{\text{succ}_{\mathcal{P}}(i)} - t_i) + s_{j,i}^+ && \forall j \in \{1, ..., n\}, i \in \{1, ..., 2n\} \\
\sum_{j=1}^{n} p_{j,i} &\leq P (t_{\text{succ}_{\mathcal{P}}(i)} - t_i) + s_i^t && \forall i \in \{1, ..., 2n\} \\
t_i &\geq 0 && \forall i \in \{1, ..., 2n\} \\
t_i &= c_i && \forall i \in \{2n+1, ..., (k+1)n\} \\
p_{j,i} &\geq 0 && \forall j \in \{1, ..., n\}, i \in \{1, ..., 2n\} \\
s_{j,i}^- &\geq 0 && \forall j \in \{1, ..., n\}, i \in \{1, ..., 2n\} \\
s_{j,i}^+ &\geq 0 && \forall j \in \{1, ..., n\}, i \in \{1, ..., 2n\} \\
s_i^t &\geq 0 && \forall i \in \{1, ..., 2n\}
\end{aligned}$$

Max flow on a bipartite graph

We can model the assignment of resources to jobs and intervals as a flow problem on a bipartite graph, if we relax some constraints. The bipartite graph models the given event order by two layers of nodes. On one side, every node represents a job, and on the other, every node represents the interval between two consecutive fixed-time events. An arc between the two exists if the job is active during that interval, i.e. if the start event of the job happens in or before that interval and its completion event happens in or after it. We set the arc capacity such that upper and lower bounds are enforced wherever possible. This will be explained in more detail below.

Arcs from the source to a job node representing job J_j initially have capacity equal to the corresponding resource requirement E_j, arcs from a job node for job J_j to an interval node $[\mathcal{F}_i, \mathcal{F}_{i'}]$ have capacity $(\mathcal{F}_{i'} - \mathcal{F}_i)P_j^+$, enforcing the upper bound, and arcs from an interval node $[\mathcal{F}_i, \mathcal{F}_{i'}]$ to the sink have capacity $(\mathcal{F}_{i'} - \mathcal{F}_i)P$, modeling the available resource in that interval. Lower bounds can be enforced on intervals that do not contain the start or completion event of a job, resulting in the adjustment of arc capacities on the arcs on the path from the source to the sink going through the specific job and interval node. The capacity of each of the arcs on this path is reduced by the amount of resource required to guarantee the lower bound for the job in that interval (i.e. $(\mathcal{F}_{i'} - \mathcal{F}_i)P_j^-$).

An example graph is shown in Figure 3.3. The capacities of a few arcs have been added in the figure. In this example, job 1 is active during the first three intervals. Therefore, we know it is active during the entire second interval, and we can already assign the lower bound. This results in the adjustment of the capacity on the arcs that are on the (unique) path from source to sink that goes through the nodes representing job 1 and interval 2 (a, c and e).

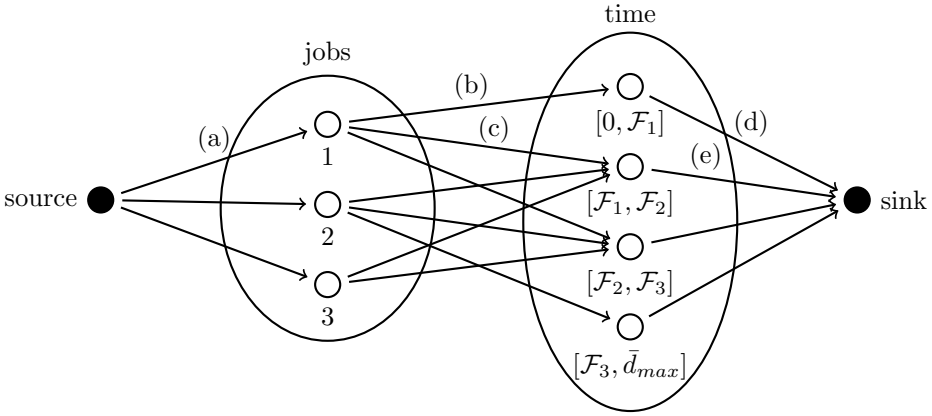
The assignment of lower bounds may already lead to the conclusion that the event order is infeasible. If it results in negative capacities in the arcs from source to job nodes, we know that a job violates its lower bounds by at least that amount. We keep track of this, and put the capacity to 0. This will contribute to the estimation of the penalty function, analogous to the $s_{j,i}^-$ variable in the LP.

Similarly, if an arc from a time node to the sink has a negative capacity, we know that there is insufficient resource availability to let all jobs consume at least their lower bound. Again, we keep track of this as a minimum violation of the resource availability, analogous to the s_i^t variable in the LP.

We compute the max flow through the resulting network. The difference between the value of the max flow and the sum of the (adjusted) resource requirements gives an indication of the total resource shortage, and therefore of the value of the resource violation ($\sum_i s_i^t$). We add to this the penalties that we already encountered while computing the arc capacities.

If the difference between these two values is zero, it is a strong indication that a feasible solution exists, or an infeasible solution (due to the order of start and completion events in a fixed-time interval, in combination with restrictive lower or upper bounds) that is fairly close to being feasible. The smaller the expected number of plannable events within a fixed-time interval, the less often such a flow will result from an infeasible event order.

A feasible flow does not guarantee a feasible solution. It does not enforce the order of plannable events within an interval in between two fixed-time events. Therefore, the resulting flow may not be valid for the exact event order provided. Ensuring one plannable event to respect its bounds may, due to the order of events, push another to a time within the interval where it cannot respect its bounds anymore. The estimation that results from this flow is almost always a close underestimation. However, it is not a true lower bound on the penalty term, as it does not (properly)



(a) $E_1 - (\mathcal{F}_2 - \mathcal{F}_1)P_1^-$ (b) $\mathcal{F}_1 \cdot P_1^+$ (d) $\mathcal{F}_1 \cdot P$
(c) $(\mathcal{F}_2 - \mathcal{F}_1) \cdot (P_1^+ - P_1^-)$ (e) $(\mathcal{F}_2 - \mathcal{F}_1) \cdot (P - P_1^-)$

Figure 3.3: Example with three jobs and four intervals

account for the possible violation of lower and upper bounds ($s^-_{j,i}$ and $s^+_{j,i}$ in the LP).

Simple lower bounds

Lower bounds can be computed on each of the three types of violation variable. For the purpose of these lower bounds, we will introduce $\mathcal{F}^{\leftarrow}(i)$ and $\mathcal{F}^{\rightarrow}(i)$, indicating the closest fixed-time event preceding or following an event i, respectively. These functions can be constructed by going over the order of events $\mathcal{E}$ once in both directions, and efficiently updated every time an event is moved.

We can determine a lower bound on the violation of the lower bounds in the following way: A job J_j can start no later than the first fixed-time event that follows its start event in the event order: $\mathcal{F}^{\rightarrow}(S_j)$, and complete no sooner than the last fixed-time event that preceded its completion event: $\mathcal{F}^{\leftarrow}(C_j)$. The time between these two time points is the minimum processing time. We can use this to determine the minimum amount of resource that the job consumes if it respects its lower bound during this time. If this amount is larger than its total resource requirement, the difference between the two gives us a lower bound on the value of $\sum_i s^-_{j,i}$:

$$\sum_{j=1}^{n} \max\left(0, \left(t_{\mathcal{F}^{\leftarrow}(C_j)} - t_{\mathcal{F}^{\rightarrow}(S_j)}\right) P_j^- - E_j\right)$$

Similarly, a lower bound on the violation of the upper bound for a job J_j ($s^+_{j,i}$) can be computed using the maximum processing time, based on the last fixed time event preceding the start event $\mathcal{F}^{\leftarrow}(S_j)$ and the first fixed time event following

the completion event $\mathcal{F}^{\rightarrow}(C_j)$:

$$\sum_{j=1}^{n} \max\left(0, E_j - \left(t_{\mathcal{F}^{\rightarrow}(C_j)} - t_{\mathcal{F}^{\leftarrow}(S_j)}\right) P_j^+\right)$$

Finally, it remains to estimate the value of the violation variable controlling the amount of available resource during an interval. We go over the event list once, adding the resource requirement E_j to the total of consumed resource up to that point if we encounter a completion event C_j. With every fixed-time event, we compute the total amount of resource available up to that point ($P \cdot t_i$, where i is the index of the fixed-time event). Any time the newly computed amount of available resource is lower than the total amount of consumed resource, we add the difference to our estimation of the violation. At the end of the event list, this results in a lower bound on the resource shortage during intervals.

$E_{total} \leftarrow 0$
$E_{shortage} \leftarrow 0$
forall $e \in \mathcal{E}$ **do**
  if IsCompletionEvent(e) **then**
    $E_{total} \leftarrow E_{total} + E_{e/2}$
  if IsFixedTimeEvent(e) **then**
    $E_{shortage} \leftarrow E_{shortage} + \max(0, E_{total} - \mathrm{t}_{I(e)} \cdot P)$
    $E_{total} = \min E_{total}, \mathrm{t}_{I(e)} \cdot P$
return $E_{shortage} \cdot L^R$

The weighted combination of these three lower bounds provides a lower bound on the total value of the penalty term. All three can be computed in $O(n)$ time. Storing intermediate results allows for efficient updates in an amount of time linear in the number of positions that an event is moved.

3.4. Applicability of the framework for the general problem

In this section, we aim to give an overview of the applicability of our approach: we will discuss the adaptations needed to apply our framework to an elaborate set of variants of the GCECSP. Our main aim is to show the flexibility and adaptability of our framework, where we do not claim that our approach is the best choice in all cases where it can be applied.

We will go through elements of the problem in three sections, to discuss possible variants and extensions, first discussing the range of possible objective functions (Section 3.4.1), then properties of jobs (Section 3.4.2) and finally the resource (Section 3.4.3).

3.4.1. Objectives

In general, the approach can easily be adapted to work for any objective that is a linear function of variables that are present in the LP. For objectives that are a function of completion time C_j (or more general, any plannable event time t_i) we distinguish between three types that we can model with increasing effort. We will discuss these three types, with a number of examples, followed by some examples that do not belong to these three types. Then, we will briefly discuss objectives that are not a function of event times.

Type I: Linear functions

For an objective that is a linear function of event times, only the replacement of the objective in the LP is required. Examples of this type of objective include:

- Minimize (weighted) completion time $\sum_j w_j C_j$ (see our previous work (2023; Chapter 2));
- Minimize total processing time $\sum_j (C_j - S_j)$.

Type II: Minmax criteria (and extensions)

These objectives require a minimal amount of additional adaptations to the LP compared to objectives of Type I. In addition to the change of the objective, for example, a helper variable and some additional constraints may have to be added to compute the objective. Examples of these are objectives that contain a min or max function, such as minimizing the makespan. They can be modeled using a helper variable for the objective ($\min w$) and a number of linear equations ($\forall j : w \geq C_j$), but not in a single linear expression. Examples of this type of objective include:

- Minimize makespan $\max_j \{C_j\}$;
- Minimize maximum lateness $\max_j \{C_j - d_j\}$;

Type III: Piece-wise linear functions

Objectives that are piece-wise linear can be modeled by splitting them in their constituent pieces using additional fixed-time events, as discussed in Section 3.3.1. For this type of objectives it is not sufficient to adapt the LP (as it is for Types I and II), but changes to the event model, or the introduction of binary variables in our LP are required. Examples of this type of objective include:

- Minimize (weighted) number of tardy jobs $\sum_j w_j U_j$, $U_j = \begin{cases} 0 & C_j \leq d_j \\ 1 & \text{otherwise} \end{cases}$;
- Minimize a step-wise function of completion time $\sum \bar{f}_j(C_j)$ (see Section 3.3).

Non-linear functions

If the objective is a non-linear function of event times, the LP-solver can be replaced by an NLP approach.

Functions of resource variables

Objectives that are a function of resource consumption $p_{j,i}$, can be accommodated along the same lines as functions of event times, as long as Theorem 3.1 holds for the particular objective. This is the case if a solution with constant resource consumption during intervals dominates solutions with other consumption profiles. Some examples of such functions are:

- Type I: minimize total resource consumption: $\sum_{j,i} p_{j,i}$, where jobs have *efficiency functions* (see Section 3.4.2);
- Type III: minimize resource cost $\sum_{j,i} c_{i,j} p_{j,i}$, where cost fluctuates across predetermined intervals. Note that the complete set of intervals have to be used in the model, where an interval is defined as the time between two consecutive events of *any* type, instead of just resource intervals (see Section 3.3.1);
- Non-linear: any objective using the consumption rate $p_{j,i}/(t_{i'} - t_i)$.

Order-determined objectives

For any objective that can already be computed based on the order of events alone, the LP is only used to determine the feasibility of the given order of events. Any (efficient) computation of the objective can be performed in the evaluation of a candidate solution before solving the LP. An example of such an objective is the minimization of the maximum number of jobs that are being processed in parallel.

3.4.2. Job properties

In the description of the GCECSP in Section 3.2 we listed a number of properties for each job. We will discuss the impact on our approach of the absence of some of these properties, and the addition of others. The removal of **release times**, **deadlines** and **upper bounds** on the resource consumption rate are fairly straight-forward, as is the introduction of jobs with a **fixed processing rate**. We will not discuss these adaptations in detail.

The **lower bound** on the resource consumption rate is an essential property of this type of problem. Maintaining the flexibility of the jobs while removing the lower bound introduces a form of preemption. The problem becomes significantly easier. While the presented approach can still be used, using a flow-based or greedy algorithm will find the optimal solution faster.

In general, **preemption** can only be modeled to a limited extent. Removing the lower bound allows preemption, but if a lower bound needs to be maintained, allowing preemption can only be done by modeling a single task as multiple jobs with precedence constraints. Then, preemption is allowed a fixed number of times. General preemption can be approximated by making this number large, but this hurts the efficiency of the approach.

Efficiency functions that apply to the conversion of the amount of consumed resource to the contribution towards the resource requirement of a job can be

easily integrated in the LP, as long as these functions are linear.

The current approach already considers implicit **precedence** relations, that follow from the release times and deadlines of jobs. It can easily be extended to include explicit precedence relations between any two events. The local search will not explore any candidate solution that does not respect such a relation.

Synchronization constraints can be included by changing the interpretation of events. In the event order, we treat each pair of events that has to happen simultaneously as a single event. Moreover, we add an equality constraint for their event times. In this way, it is ensured that any considered solution will synchronize these two events.

3.4.3. Resource

In the GCECSP, we have a single constant renewable resource. Variations on this aspect of the problem can also be modeled within our framework. We will discuss two example extensions.

Any piece-wise linear **resource availability function** can be modeled using fixed-time events similar to piece-wise linear objective functions (see Section 3.4.1).

Multiple resources can be modeled using additional resource variables $p'_{j,i}$. The bounds, resource requirement, and events can be unique to each resource, or shared among multiple resources.

A **discrete resource** can be approximated by using multiple (synchronized) jobs with identical lower and upper bounds to model a single task. This will, however, limit the number of times the resource consumption profile can change. In addition, it will hurt the efficiency of the approach.

3.5. Computational results

We will show results for the GCECSP with step-wise cost functions (see Section 3.3), using three variants of our approach. For small instances, we will compare these approaches to the MILP (solved with a time limit of 3600 seconds) as well. The data and code used are available online (Brouwer 2023a,b). In Section 3.5.1, we discuss the instances and parameter settings used in our evaluation. Following that, we present the variants of our approach used in the tests and evaluate their performance in Section 3.5.2. The full result tables are provided in Appendix 3.B.

3.5.1. Test instances and parameter settings

For generating instances, we will use the procedure developed for instances for the CECSP in our previous work (Brouwer, van den Akker, and Hoogeveen 2023; Section 2.4). Only the generation of weights will be different, and the generation of times for the jump points has to be added. Recall that k indicates the number of cost intervals in the cost function for each job.

Instead of a single weight w_j from $U(0,5)$ we generate an array of k weights, in the following way: we first generate an initial weight $w_{j,1}$ from $U(0,5)$ and then generate $k-1$ weights from $U(w_{j,1},5)$. The resulting array of k weights is then sorted to be increasing. Finally, the values are modified to be incremental, such that the cost of completing job J_j right after the first jump point becomes $w_{j,1}+w_{j,2}$, the cost of completing the job right after the second jump point becomes $w_{j,1}+w_{j,2}+w_{j,3}$, and so on.

Then, we generate $k-1$ jump points for each job. The release time r_j and deadline $\bar{d}_j$ are generated as before. We require $k-1$ additional time points in between where the weight changes. We sample these $k-1$ values from $U(r_j+\frac{E_j}{P_j^+},\bar{d}_j)$ and put them in increasing order.

This completes the modified instance generation approach. Using this, we generate a set of 72 instances: 4 unique instances for each combination of n and k for $n \in \{5,10,15,20,30,50\}, k \in \{2,3,4\}$.

We evaluated the speed and quality of the estimation methods described in Section 3.3.3. For all instance sizes, we ran a test where we computed the estimation methods on a sequence of event orders. The sequence was obtained by performing a random walk of length 1000, randomly shifting an event by one position each time. Averages of the run times are displayed in Figure 3.4.

The lower bound estimation is much faster than either of the other methods and displays linear scaling behavior. In terms of quality, we note that the computed bounds are not tight. The flow typically results in a better estimation of the penalty term. However, for guiding the search, a strong correlation with the actual penalty term is more important than a tight estimate. To evaluate this, we computed Spearman's rank correlation of the lower bounds and the penalty term, for the instance sizes where $n \geq 10$. These all have values between 0.67 and 0.86 (0.77 on average), showing a strong correlation. The flow estimation shows an even stronger correlation (0.82 on average). However, we decided to drop it from our approach, given the limited improvement on the quality of the estimation and the increased runtime compared to the 'simple' bounds.

Each of these estimates can be updated efficiently from a previous event order. We have implemented this for both the LP (1) and the 'simple' lower bound estimation (3). These are used in the approaches that will be evaluated in Section 3.5.2.

To make sure that we get the best performance out of the presented approaches, we carefully select values for a number of parameters. We chose a commonly used value for the multiplication factor for updating the temperature $\alpha = 0.95$. We performed a limited grid search using different values for the initial temperature $T_{\text{init}} \in \{0.1, 0.2, 0.5, 1, 2, 5\}n$, the number of iterations between temperature updates $\alpha_{\text{period}} \in \{2,4,6,8\}2n$ and the multipliers used to penalize violation variables $L^R, L^B \in \{0.5,1,2,5\}$. We performed runs for all combinations on twelve instances with ten jobs. We observe that finding a good initial solution is important, and exploring too much into a part of the search space consisting of infeasible solutions

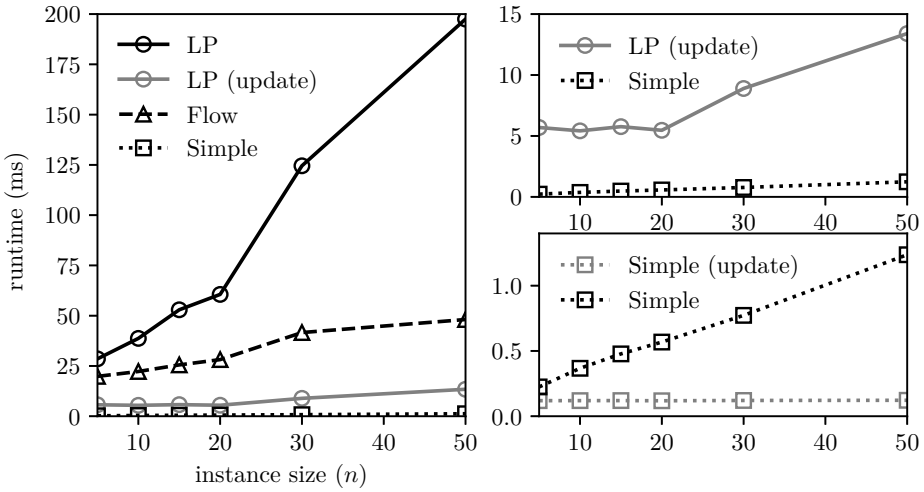


Figure 3.4: Average runtimes of evaluation methods for event orders

is detrimental to the eventual success. Our results indicate that it is beneficial to select an initial temperature that is not too high, while cooling down more slowly, and a large penalty for the violation variables, directing the search towards feasible solutions. The addition of a tabu list of length 1, forbidding the movement of the same event two times in a row, yields a slight improvement. The final parameter settings are as follows: $T_{\text{init}} = 0.2n$; $\alpha = 0.95$; $\alpha_{\text{period}} = 16n$; violation penalty: 5.0.

3.5.2. Results and discussion

We present results for the following three variants of our hybrid approach:

SA-LP uses only the (iteratively updated) LP for the subproblem;

SA-MIX uses the 'simple' bound estimations to assess the quality of an event order, but also computes the LP if the result of the estimation is 0;

SA-2PHASE uses the 'simple' bound estimations until it finds a solution for which the estimation is 0, and switches to the LP afterward.

As discussed in Section 3.5.1, we will use the 'simple' lower bound estimation and the LP to assess the quality of candidate solutions. Preliminary tests showed that using only the 'simple' lower bound estimation does not result in good solutions. Therefore, we evaluate approaches combining the use of the 'simple' lower bound estimation with the LP.

In Table 3.1, we summarize the results for each instance size. For every approach, we list three measures of its performance: the number of instances for which it found a feasible solution, the number of instances for which it found the best solution among the tested approaches, and the average distance (in %) to the best

Approach	feas.	best	dist.
$n = 5$			
MILP	12/12	12/12	0.00
SA-LP	12/12	5/12	3.48
SA-MIX	11/12	8/12	1.05
SA-2PH.	12/12	7/12	4.45
$n = 10$			
MILP	12/12	9/12	1.06
SA-LP	12/12	5/12	1.88
SA-MIX	10/12	1/12	7.23
SA-2PH.	11/12	3/12	2.76
$n = 15$			
MILP	7/12	1/12	4.98
SA-LP	12/12	8/12	0.38
SA-MIX	11/12	0/12	19.94
SA-2PH.	12/12	6/12	2.18

Approach	feas.	best	dist.
$n = 20$			
SA-LP	12/12	4/12	0.98
SA-MIX	4/12	0/12	11.02
SA-2PH.	12/12	8/12	0.21
$n = 30$			
SA-LP	12/12	5/12	0.89
SA-MIX	10/12	0/12	14.06
SA-2PH.	12/12	7/12	1.33
$n = 50$			
SA-LP	12/12	7/12	0.69
SA-MIX	2/12	0/12	10.41
SA-2PH.	12/12	5/12	1.04

Table 3.1: Summary of results for the presented approaches

solution among the tested approaches (infeasible solutions are excluded).

Figure 3.5 shows the runtime of the approaches. Each point is an average over all instances of a certain size (n). For the correct interpretation of the values for the MILP for $n = 10, 15$, note that the average includes runs that were cut off at 3600 seconds. The MILP was not included for $n \geq 20$ because of very poor performance. Our approach scales better than the MILP, but reports high run times for larger instances as well. However, a reasonable (feasible) solution is often already found relatively quickly, and much time is spent on searching for further improvements. Whenever time is a constraint, cutting off the search after a certain amount of time has elapsed will still yield good results.

The general approach (SA-LP) performs very well on all instance sizes. We are able to reliably find feasible solutions for much larger instances than the MILP. Exchanging the LP in the first part of the search for the 'simple' bound estimations (SA-2PHASE) starts to pay off with larger instance sizes. Mixing estimation methods (SA-MIX) does not perform as well as expected. As the iterative update of the LP is much faster than solving the LP from scratch, much of the time gained when using other estimation methods is lost once the LP does have to be solved again. In addition, the quality of the 'simple' bounds does seem to be insufficient to guide the search reliably towards good solutions for larger instance sizes. The SA-MIX approach uses only the 'simple' lower bound estimation for evaluation whenever these report a positive penalty term. These 'simple' bounds do not detect all causes of infeasibility, and are therefore unreliable when the search comes close

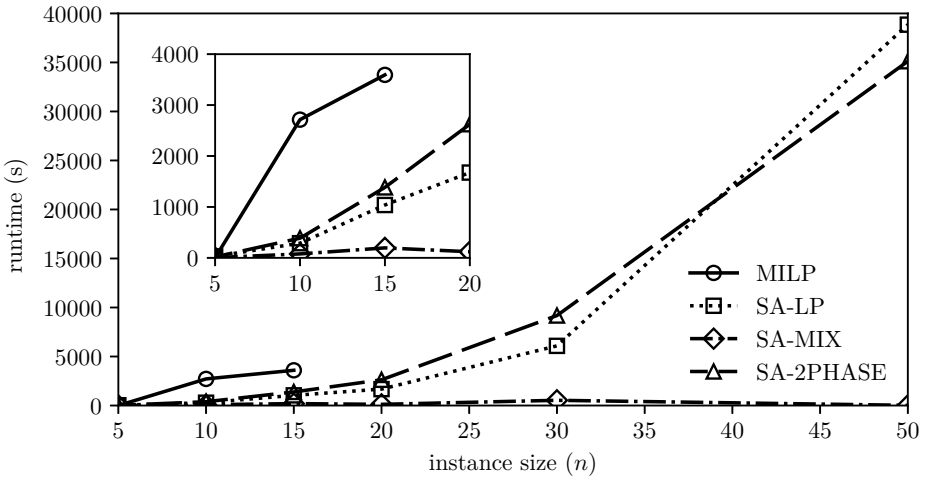


Figure 3.5: Average runtimes of discussed approaches

to a feasible solution. The LP, in contrast, computes the exact penalty term. This may result in moving to candidate solutions with a lower penalty term, but positive 'simple' lower bounds, at which point these lower bounds will again be used to evaluate the candidate solutions. We conclude that multiple iterations with the LP are needed to converge on a good solution. It should be noted, that the general approach using only the LP for the evaluation of candidate solutions (SA-LP) also benefits from the fact that the objective value (excluding the penalty term) can be computed before solving the LP. If this is sufficient to reject a candidate solution, the search will move on without solving the LP. To illustrate this point, we ran the general approach (SA-LP) as well as a modified implementation that does not make use of this fact (SA-LP naive) on all instances, seeding the random engine to ensure both approaches follow the exact same path through the search space. Figure 3.6 shows the average runtime of the general approach (shaded area) as a percentage of the naive approach (full bar). From this, we can see that this simple modification saves around 20% of computation time.

As we mentioned above, cutting of the search early on is a viable approach if time is a constraint. To back up this claim, we reran the SA-LP approach for all instances of size $n = 30$ and $n = 50$. In Figure 3.7, we have plotted the progression of the best found solution over the total runtime of the approach. The horizontal axis shows the normalized runtime, and the vertical axis the normalized distance to the best found solution. With the latter, we mean the difference between the current best objective value and the final best objective value, expressed as a fraction of the final best objective value. Each run is displayed as a gray line. Note that many lines start far outside the plot, but drop to within the range displayed after a limited number of successful iterations. The inset zooms in on the first part of the search, and shows the first found feasible solution for each run with a black 'x'.

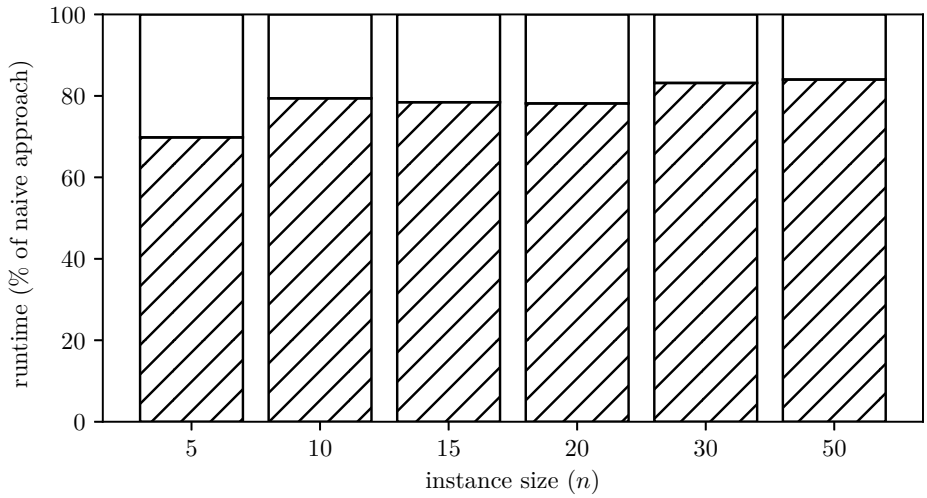


Figure 3.6: Runtime of the general approach (SA-LP) as a percentage of its naive implementation

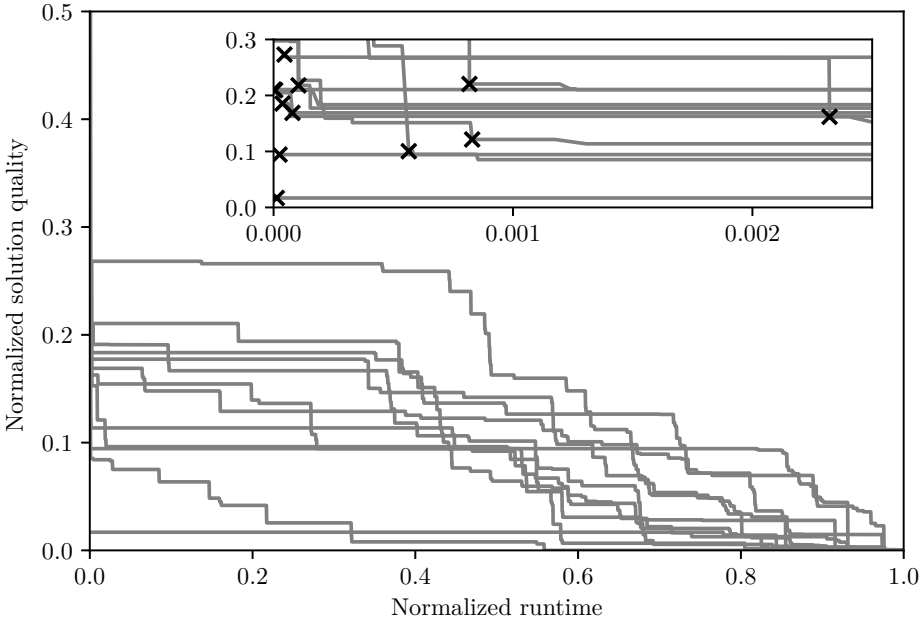
From this, we conclude that the first feasible solution is found very quickly, before 0.25% (for $n = 30$) or 2% (for $n = 50$) of the total runtime has passed. We also observe that, after a quick dramatic improvement at the start, the improvement of the best found solution is gradual. Cutting off the search early will provide a good, feasible solution, if time is a constraint.

We already noted in the previous section that finding a good initial solution is important. Even still, the local search will consistently improve the initial solution significantly. Given the sensitivity of the approach to the initial solution, it would be an interesting approach to introduce restarts for instances where the best found solution is not satisfactory. This can be done by disturbing the best found solution using the neighborhood operator a number of times, regardless of the quality of the result, and restarting the search from there.

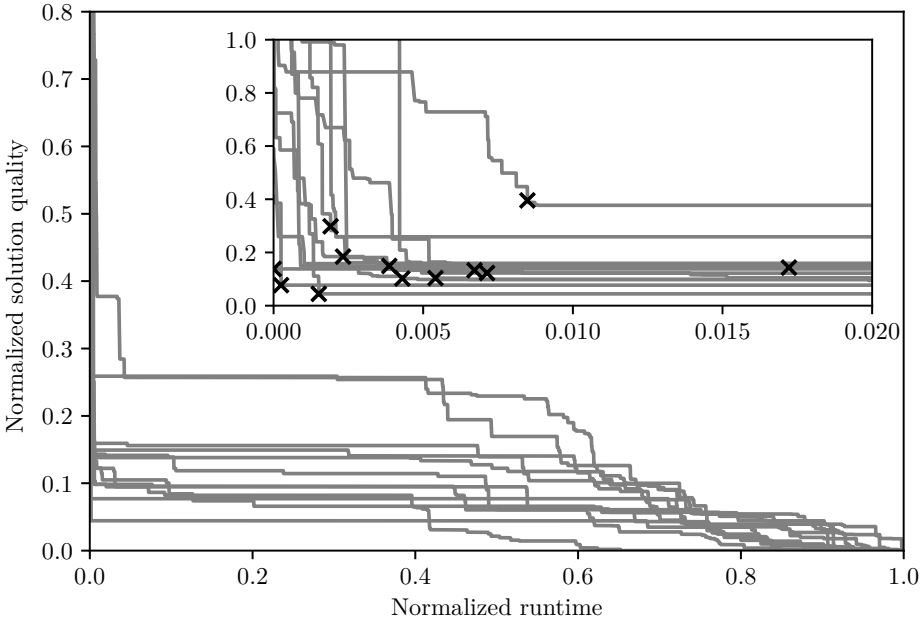
3.6. Conclusions and future work

In this work, we have introduced a hybrid optimization framework for a class of problems that are variants of the GCECSP, which we introduced as a generalization of the CECSP. We have shown the general applicability of our approach, and have studied its performance when applied to the GCECSP with step-wise cost functions. With this, we give an impression of how to tailor the approach to specific variants. Furthermore, we have shown the range of variants that our approach can be applied to, and discussed the required adaptations for a number of cases.

In the application to the variant with step-wise cost functions, we show that we are



(a) Instances with $n = 30$



(b) Instances with $n = 50$

Figure 3.7: Progression of the best found solution over the total runtime

able to find good solutions for much larger instances, when compared to the MILP. We also observe that it can be used to find reasonable solutions relatively quickly, whereas the MILP starts struggling to find even a feasible solution for instances of size $n = 15$ and larger. The fact that we know the part of the objective excluding the penalty term before solving the subproblem, allows us to improve the efficiency of the approach. Exploiting this fact further by replacing the LP for solving the subproblem by alternative approaches has not been as successful.

Furthermore, we open up the possibility of applying the approach to a broader range of problems. Further study of such variants could provide more insight of the quality of the results for specific variants. Further exploration of estimation methods for objectives of Type III (see Section 3.4.1) is of particular interest.

3.A. Full MILP formulation

The MILP formulation for the GCECSP with step-wise costs is as follows:

$$\min \sum_{j\in\{1,\dots,n\}} w_{j,1} + \left(\sum_{l=2}^{k} w_{j,l} a_{2n+(k-1)(j-1)+(l-1),2j}\right) \quad \text{s.t.}$$

$$t_i \le t_{i'} + M a_{i',i} \qquad \forall i,i' \in \{1,\dots,(k+1)n\}, i \ne i'$$

$$t_{2j} \le \bar{d}_j \qquad \forall j \in \{1,\dots,n\}$$

$$t_{2j-1} \ge r_j \qquad \forall j \in \{1,\dots,n\}$$

$$\sum_{i\in\{1,\dots,2n\}} p_{j,i} = E_j \qquad \forall j \in \{1,\dots,n\}$$

$$p_{j,i} \ge P_j^-(t_{i'} - t_i) - (1 - b_{i,i'})M \qquad \forall j \in \{1,\dots,n\},$$

$$-(1 - a_{2j-1,i'})M - (1 - a_{i,2j})M \qquad i,i' \in \{1,\dots,2n\}, i \ne i'$$

$$p_{j,i} \le P_j^+(t_{i'} - t_i) + M a_{i',i} \qquad \forall j \in \{1,\dots,n\}, \; i,i' \in \{1,\dots,2n\}, i \ne i'$$

$$p_{j,i} \le a_{i,2j} M \qquad \forall j \in \{1,\dots,n\}, i \in \{1,\dots,2n\}$$

$$p_{j,i} \le (1 - a_{i,2j-1})M \qquad \forall j \in \{1,\dots,n\}, i \in \{1,\dots,2n\}$$

$$\sum_{j\in\{1,\dots,n\}} p_{j,i} \le P(t_{i'} - t_i) + M a_{i',i} \qquad \forall i,i' \in \{1,\dots,2n\}, i \ne i'$$

$$a_{i,i'} + a_{i',i} = 1 \qquad \forall i,i' \in \{1,\dots,(k+1)n\}, i < i'$$

$$\sum_{i''\in\{1,\dots,2n\}} (a_{i,i''} - a_{i',i''}) \le 1 + (1 - b_{i,i'})M \qquad \forall i,i' \in \{1,\dots,2n\}, i \ne i'$$

$$\sum_{i''\in\{1,\dots,2n\}} (a_{i,i''} - a_{i',i''}) \ge 1 - (1 - b_{i,i'})M \qquad \forall i,i' \in \{1,\dots,2n\}, i \ne i'$$

$$\sum_{i,i'\in\{1,\dots,2n\}, i\ne i'} b_{i,i'} = 2n - 1$$

$$t_{2j} - t_{2j-1} \le E_j / P_j^- \qquad \forall j \in \{1,\dots,n\}$$

$$t_{2j} - t_{2j-1} \ge E_j / P_j^+ \qquad \forall j \in \{1,\dots,n\}$$

$$p_{j,i} \ge 0 \qquad \forall j \in \{1,\dots,n\}, i \in \{1,\dots,2n\}$$

$$t_i \ge 0 \qquad \forall i \in \{1,\dots,2n\}$$

$$t_i = c_i \qquad \forall i \in \{2n+1,\dots,(k+1)n\}$$

$$a_{i,i'} \in \{0,1\} \qquad \forall i,i' \in \{1,\dots,(k+1)n\}, i \ne i'$$

$$a_{i,i} = 0 \qquad \forall i \in \{1,\dots,(k+1)n\}$$

$$b_{i,i'} \in \{0,1\} \qquad \forall i,i' \in \{1,\dots,2n\}, i \ne i'$$

3.B. Full Results

This appendix contains the full results of the tests described in Section 3.5, in particular those on which the data in Table 3.1 and Figure 3.5 is based.

Each table shows the results for all four instances of a particular combination of n and k (displayed in the upper left corner). The first row indicates whether our feasibility test was passed by the instance or not (as described in Section 2.4.2). This is followed by a row indicating the best objective value that we have encountered in any run of the three ($n \geq 20$) or four ($n \leq 15$) listed approaches.

For each of the approaches, we then list the runtime (in seconds), the objective value of the solution found by that particular approach and, in the case of the approaches using simulated annealing, the objective value of the initial solution. In cases where the MILP is cut off because of the time limit set (3600 seconds), this is indicated with the text `LIMIT`. If no feasible solution is found by the MILP within an hour, there is no objective to report, which is indicated with a '-'. In the row that contains the objective, a value is displayed in **boldface** if it is the best results among the compared approaches. It is displayed in *italics* if it represents an infeasible solution (i.e., the sum of the violation variables is larger than 0). Values in italics only occur if the approach uses simulated annealing, as the MILP does not include violation variables and therefore does not report infeasible solutions.

Note that the results in these tables are of a single run of each of the approaches, they are not averages or best-of results.

$n = 5, k = 2$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		12.42	9.74	13.08	6.37
MILP	time	0.43	0.44	1.03	0.67
	objective	**12.42**	**9.74**	**13.08**	**6.37**
SA-LP	time	11.41	2.59	29.03	6.44
	objective	**12.42**	11.66	**13.08**	**6.37**
	init. sol.	17.24	14.58	15.50	6.37
SA-MIX	time	76.10	14.43	134.66	3.25
	objective	**12.42**	**9.74**	**13.08**	**6.37**
	init. sol.	17.24	14.58	15.50	6.37
SA-2PHASE	time	2.51	3.48	13.59	1.20
	objective	**12.42**	11.83	**13.08**	**6.37**
	init. sol.	17.24	14.58	15.50	6.37

$n = 5, k = 3$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		15.27	12.49	17.08	19.52
MILP	time	0.39	0.32	0.39	0.85
	objective	**15.27**	**12.49**	**17.08**	**19.52**
SA-LP	time	36.37	60.17	37.33	38.98
	objective	15.29	**12.49**	**17.08**	19.53
	init. sol.	18.02	13.77	17.40	21.09
SA-MIX	time	133.74	20.97	150.01	104.37
	objective	**15.27**	**12.49**	**17.08**	19.59
	init. sol.	18.02	13.77	17.40	21.09
SA-2PHASE	time	52.18	23.53	28.24	32.14
	objective	**15.27**	**12.49**	**17.08**	19.53
	init. sol.	18.02	13.77	17.40	21.09

$n = 5, k = 4$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		14.63	16.69	9.19	10.77
MILP	time	0.92	12.63	0.51	0.54
	objective	**14.63**	**16.69**	**9.19**	**10.77**
SA-LP	time	14.20	51.16	35.30	25.74
	objective	16.75	16.75	9.66	10.98
	init. sol.	167.39	19.32	10.38	17.07
SA-MIX	time	0.16	106.18	208.08	28.42
	objective	*145.89*	16.75	**9.19**	11.94
	init. sol.	167.39	19.32	10.38	17.07
SA-2PHASE	time	14.16	7.00	36.17	11.43
	objective	15.22	18.20	**9.19**	12.80
	init. sol.	167.39	19.32	10.38	17.07

$n = 10, k = 2$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		19.10	17.69	15.56	33.03
MILP	time	3403.32	LIMIT	3301.44	209.87
	objective	**19.10**	**17.69**	**15.56**	**33.03**
SA-LP	time	177.35	124.76	245.82	184.96
	objective	19.43	18.20	**15.56**	35.39
	init. sol.	243.96	308.84	18.37	38.77
SA-MIX	time	171.70	0.59	199.41	123.25
	objective	22.15	*204.66*	**15.56**	35.47
	init. sol.	243.96	308.84	18.37	38.77
SA-2PHASE	time	219.83	0.26	136.65	296.47
	objective	21.66	*85.19*	**15.56**	34.45
	init. sol.	243.96	308.84	18.37	38.77

$n = 10, k = 3$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		21.01	28.00	26.41	22.44
MILP	time	LIMIT	LIMIT	402.73	LIMIT
	objective	**21.01**	**28.00**	**26.41**	**22.44**
SA-LP	time	162.92	409.86	420.81	229.25
	objective	22.31	**28.00**	**26.41**	22.80
	init. sol.	28.53	28.09	31.05	33.31
SA-MIX	time	563.43	1270.58	1215.83	71.37
	objective	22.31	28.09	26.49	24.27
	init. sol.	28.53	28.09	31.05	33.31
SA-2PHASE	time	137.71	589.88	519.66	425.66
	objective	22.31	28.09	**26.41**	22.80
	init. sol.	28.53	28.09	31.05	33.31

$n = 10, k = 4$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		27.28	25.75	29.48	24.72
MILP	time	41.82	LIMIT	LIMIT	LIMIT
	objective	**27.28**	26.61	30.60	26.11
SA-LP	time	295.05	276.10	551.23	375.96
	objective	27.32	26.49	**29.48**	**24.72**
	init. sol.	30.70	31.10	114.09	203.20
SA-MIX	time	13.84	29.44	209.57	15.33
	objective	30.70	28.47	32.70	*31.82*
	init. sol.	30.70	31.10	114.09	203.20
SA-2PHASE	time	545.23	549.70	597.15	619.47
	objective	27.32	**25.75**	29.94	25.41
	init. sol.	30.70	31.10	114.09	203.20

$n = 15, k = 2$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		45.32	29.64	32.52	42.50
MILP	time	3472.54	LIMIT	LIMIT	LIMIT
	objective	**45.32**	29.68	-	43.18
SA-LP	time	1166.54	826.58	694.15	756.64
	objective	**45.32**	**29.64**	**32.52**	**42.50**
	init. sol.	89.63	37.38	147.25	51.33
SA-MIX	time	141.57	94.75	0.29	200.75
	objective	45.43	36.14	48.77	50.56
	init. sol.	89.63	37.38	147.25	51.33
SA-2PHASE	time	1954.29	1333.77	432.17	486.36
	objective	**45.32**	**29.64**	33.02	45.61
	init. sol.	89.63	37.38	147.25	51.33

$n = 15, k = 3$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		37.18	40.41	45.08	32.17
MILP	time	LIMIT	LIMIT	LIMIT	LIMIT
	objective	39.03	-	-	38.56
SA-LP	time	837.38	994.47	1295.04	1293.10
	objective	37.56	**40.41**	45.31	32.83
	init. sol.	50.79	50.00	105.65	40.50
SA-MIX	time	942.28	63.18	23.69	41.43
	objective	40.95	45.98	49.04	40.50
	init. sol.	50.79	50.00	105.65	40.50
SA-2PHASE	time	817.44	969.81	1671.45	2317.23
	objective	**37.18**	42.93	**45.08**	**32.17**
	init. sol.	50.79	50.00	105.65	40.50

$n = 15, k = 4$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		42.38	41.76	29.34	33.96
MILP	time	LIMIT	LIMIT	LIMIT	LIMIT
	objective	42.68	-	31.57	-
SA-LP	time	1280.55	1504.55	1017.33	787.37
	objective	**42.38**	**41.76**	29.61	**33.96**
	init. sol.	51.96	54.36	82.57	42.93
SA-MIX	time	17.26	129.05	0.99	46.33
	objective	51.96	50.36	*68.28*	42.93
	init. sol.	51.96	54.36	82.57	42.93
SA-2PHASE	time	1419.34	1909.92	2076.76	1216.30
	objective	43.84	44.17	**29.34**	34.58
	init. sol.	51.96	54.36	82.57	42.93

$n = 20, k = 2$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		52.47	49.03	49.14	44.29
SA-LP	time	1502.61	1708.22	1331.91	1156.45
	objective	54.18	49.11	50.00	**44.29**
	init. sol.	195.25	319.42	59.13	278.03
SA-MIX	time	13.42	2.71	601.63	155.35
	objective	*62.79*	*318.19*	*55.74*	46.54
	init. sol.	195.25	319.42	59.13	278.03
SA-2PHASE	time	3198.27	3191.63	1373.58	1332.58
	objective	**52.47**	**49.03**	**49.14**	44.70
	init. sol.	195.25	319.42	59.13	278.03

$n = 20, k = 3$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		65.33	57.83	52.91	62.42
SA-LP	time	1542.54	1463.81	1797.87	1969.42
	objective	65.92	58.01	**52.91**	**62.42**
	init. sol.	192.59	170.00	161.81	178.35
SA-MIX	time	80.95	4.26	131.92	7.42
	objective	68.99	*149.90*	63.37	*113.21*
	init. sol.	192.59	170.00	161.81	178.35
SA-2PHASE	time	1726.78	2661.88	2985.21	3246.13
	objective	**65.33**	**57.83**	53.54	62.51
	init. sol.	192.59	170.00	161.81	178.35

$n = 20, k = 4$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		62.09	45.71	53.69	58.47
SA-LP	time	2162.59	1909.21	1930.20	1621.52
	objective	63.22	**45.71**	55.54	58.54
	init. sol.	165.04	105.35	314.93	70.27
SA-MIX	time	2.84	1.34	1.89	36.27
	objective	*146.67*	*77.84*	*250.40*	66.43
	init. sol.	165.04	105.35	314.93	70.27
SA-2PHASE	time	2542.63	2794.88	3813.35	2615.45
	objective	**62.09**	45.85	**53.69**	**58.47**
	init. sol.	165.04	105.35	314.93	70.27

$n = 30, k = 2$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		79.77	78.99	75.59	69.29
SA-LP	time	5987.90	5293.30	7501.06	4438.27
	objective	**79.77**	81.79	**75.59**	70.03
	init. sol.	201.16	101.66	79.57	106.25
SA-MIX	time	79.23	492.44	282.40	81.74
	objective	89.98	82.82	79.57	77.21
	init. sol.	201.16	101.66	79.57	106.25
SA-2PHASE	time	9443.22	8515.09	7840.55	8913.61
	objective	83.26	**78.99**	78.20	**69.29**
	init. sol.	201.16	101.66	79.57	106.25

$n = 30, k = 3$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		81.40	85.20	74.27	85.25
SA-LP	time	6495.80	5113.45	6655.20	6148.46
	objective	**81.40**	86.49	**74.27**	85.64
	init. sol.	360.08	100.75	91.16	96.37
SA-MIX	time	4.52	155.68	103.69	26.29
	objective	*130.42*	94.67	91.16	92.82
	init. sol.	360.08	100.75	91.16	96.37
SA-2PHASE	time	6833.45	9203.74	9654.89	8082.87
	objective	83.55	**85.20**	77.64	**85.25**
	init. sol.	360.08	100.75	91.16	96.37

$n = 30, k = 4$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		75.34	75.56	83.54	73.18
SA-LP	time	5793.06	8548.83	4606.60	6463.71
	objective	75.64	**75.56**	86.28	73.46
	init. sol.	94.59	92.29	143.81	83.83
SA-MIX	time	123.32	264.12	114.96	104.98
	objective	92.28	*89.89*	105.69	83.83
	init. sol.	94.59	92.29	143.81	83.83
SA-2PHASE	time	8152.17	14057.77	7972.45	11481.82
	objective	**75.34**	76.30	**83.54**	**73.18**
	init. sol.	94.59	92.29	143.81	83.83

$n = 50, k = 2$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		128.78	118.32	121.08	127.03
SA-LP	time	41080.91	38014.12	47866.84	30884.50
	objective	**128.78**	**118.32**	**121.08**	**127.03**
	init. sol.	383.00	255.24	2634.19	180.49
SA-MIX	time	38.09	668.13	6.05	288.01
	objective	*362.09*	*130.84*	*2307.93*	135.33
	init. sol.	383.00	255.24	2634.19	180.49
SA-2PHASE	time	39614.37	35312.87	34989.71	34824.88
	objective	131.76	120.82	122.73	128.02
	init. sol.	383.00	255.24	2634.19	180.49

$n = 50, k = 3$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		134.60	141.69	139.51	152.90
SA-LP	time	44735.46	42392.18	34161.03	25714.16
	objective	135.03	**141.69**	140.43	154.21
	init. sol.	677.57	161.94	661.56	247.66
SA-MIX	time	2.87	524.66	9.37	174.86
	objective	*475.18*	161.94	*278.72*	*205.48*
	init. sol.	677.57	161.94	661.56	247.66
SA-2PHASE	time	31956.34	37477.15	34576.79	26084.72
	objective	**134.60**	145.87	**139.51**	**152.90**
	init. sol.	677.57	161.94	661.56	247.66

$n = 50, k = 4$		# 0	# 1	# 2	# 3
Flow feasible		yes	yes	yes	yes
Best known		133.09	121.30	134.82	125.22
SA-LP	time	42071.77	33440.85	38900.98	46989.68
	objective	135.06	**121.30**	**134.82**	131.50
	init. sol.	501.21	292.40	236.23	362.49
SA-MIX	time	7.28	201.64	111.17	62.61
	objective	*480.72*	*161.28*	*211.58*	*362.49*
	init. sol.	501.21	292.40	236.23	362.49
SA-2PHASE	time	41332.60	27681.98	39418.33	38166.36
	objective	**133.09**	121.89	138.18	**125.22**
	init. sol.	501.21	292.40	236.23	362.49

References

Baptiste, P., C. L. Pape, and W. Nuijten (1999). "Satisfiability tests and time-bound adjustments for cumulative scheduling problems". *Annals of Operations Research* 92, pp. 305–333. DOI: 10.1023/A:1018995000688.

Brouwer, R. J. J. (2023a). *Data for the GCECSP with step-wise cost function*. Zenodo. DOI: 10.5281/zenodo.10304834. URL: https://github.com/RoelBrouwer/2023-step-wise-cecsp-data/tree/cpaior.

Brouwer, R. J. J. (Dec. 8, 2023b). *GCECSP: models and algorithms*. Version 0.2.0. DOI: 10.5281/zenodo.10304753. URL: https://github.com/RoelBrouwer/continuousresource/tree/cpaior.

Brouwer, R. J. J., J. M. van den Akker, and J. A. Hoogeveen (2023). "A hybrid local search algorithm for the continuous energy-constrained scheduling problem". Preprint at: https://arxiv.org/abs/2311.16177. DOI: 10.48550/arXiv.2311.16177.

Brouwer, R. J. J., J. M. van den Akker, and J. A. Hoogeveen (2024). "A hybrid optimization framework for the general continuous energy-constrained scheduling problem". Preprint at: https://arxiv.org/abs/2403.03039. DOI: 10.48550/arXiv.2403.03039.

Detienne, B., S. Dauzère-Pérès, and C. Yugma (2011). "Scheduling jobs on parallel machines to minimize a regular step total cost function". *Journal of Scheduling* 14.6, pp. 523–538. DOI: 10.1007/s10951-010-0203-z.

Detienne, B., S. Dauzère-Pérès, and C. Yugma (2012). "An exact approach for scheduling jobs with regular step cost functions on a single machine". *Computers and Operations Research* 39.5, pp. 1033–1043. DOI: 10.1016/j.cor.2011.06.006.

Hartmann, S. and D. Briskorn (2010). "A survey of variants and extensions of the resource-constrained project scheduling problem". *European Journal of Operational Research* 207 (1), pp. 1–14. DOI: 10.1016/j.ejor.2009.11.005.

Hartmann, S. and D. Briskorn (2022). "An updated survey of variants and extensions of the resource-constrained project scheduling problem". *European Journal of Operational Research* 297 (1), pp. 1–14. DOI: 10.1016/j.ejor.2021.05.004.

Koné, O., C. Artigues, P. Lopez, and M. Mongeau (2011). "Event-based MILP models for resource-constrained project scheduling problems". *Computers & Operations Research* 38, pp. 3–13. DOI: 10.1016/j.cor.2009.12.011.

Kopanos, G. M., T. S. Kyriakidis, and M. C. Georgiadis (2014). "New continuous-time and discrete-time mathematical formulations for resource-constrained project scheduling problems". *Computers and Chemical Engineering* 68, pp. 96–106. DOI: 10.1016/j.compchemeng.2014.05.009.

Naber, A. (2017). "Resource-constrained project scheduling with flexible resource profiles in continuous time". *Computers and Operations Research* 84, pp. 33–45. DOI: 10.1016/j.cor.2017.02.018.

Nattaf, M., C. Artigues, and P. Lopez (2014). "A polynomial satisfiability test using energetic reasoning for energy-constraint scheduling". *Proceedings of the International Conference on Project Management and Scheduling (PMS 2014)*, pp. 169–172. URL: https://hal.archives-ouvertes.fr/hal-00978706.

Nattaf, M., C. Artigues, and P. Lopez (Oct. 2017). "Cumulative scheduling with variable task profiles and concave piecewise linear processing rate functions". *Constraints* 22 (4), pp. 530–547. DOI: 10.1007/S10601-017-9271-4.

Nattaf, M., C. Artigues, P. Lopez, and D. Rivreau (Mar. 2016). "Energetic reasoning and mixed-integer linear programming for scheduling with a continuous resource and linear efficiency functions". *OR Spectrum* 38 (2), pp. 459–492. DOI: 10.1007/S00291-015-0423-X.

Nattaf, M., M. Horváth, T. Kis, C. Artigues, and P. Lopez (2019). "Polyhedral results and valid inequalities for the continuous energy-constrained scheduling problem". *Discrete Applied Mathematics* 258, pp. 188–203. DOI: 10.1016/j.dam.2018.11.008.

Tseng, C.-T., Y.-C. Chou, and Y.-C. Chou (2010). "A variable neighborhood search for the single machine total stepwise tardiness problem". *Proceedings of the 2010 International Conference on Engineering, Project and Production Management*, pp. 101–108. DOI: 10.32738/ceppm.201010.0011.

Turek, J., J. L. Wolf, and P. S. Yu (1992). "Approximate algorithms for scheduling parallelizable tasks". *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 323–332. DOI: 10.1145/140901.141909.

Part III

Applications of scheduling in electricity networks

Contents

4

Forecast-based optimization of islanded microgrids

The contents of this chapter have been published in the proceedings of the 2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe):

R. J. J. Brouwer, J. M. van den Akker, F. P. M. Dignum, et al. (2018). "Forecast-based optimization of islanded microgrids". *Proceedings of the 2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pp. 1–6. DOI: 10.1109/ISGTEurope.2018.8571679

Abstract

In the context of the energy transition, the operational management of microgrids becomes an issue of increasing importance. In this chapter, we compare two optimization models for planning generators and storage units in an islanded microgrid, while dealing with uncertainty in forecasts of load and photovoltaic (PV) output. These models are combined with a number of re-planning strategies, and evaluated through a case study of an islanded microgrid in the Netherlands. As a result, one of our algorithms is currently implemented in this microgrid. In our analysis of the case study, we identify features of the microgrid that play an important role in determining whether these approaches are useful and we show the effectiveness of using re-planning strategies.

4.1. Introduction

Currently, we see a move from large centralized energy producers to a distributed network of small producers and consumers: an energy transition. New technologies, such as microgrids, facilitate this move. A microgrid is a small cluster of components that each produce, store or consume energy. An islanded microgrid should be able to operate autonomously, separate from the main grid, i.e. it is capable of balancing demand and supply at all times (Lopes, Moreira, and Madureira 2006). These microgrids are especially important for remote or isolated geographical areas such as islands or mountain villages for which maintaining a connection to the main grid is very costly or even impossible. However, microgrids are also applied in other cases: in energy aware home-owners or neighborhoods or in remote locations where large consumers (e.g. a hospital) and producers (e.g. photovoltaic (PV) installations) are geographically close.

Another challenging property of microgrids (and smart grids in general) is that a major part of the generation comes from renewable resources, which do not produce a guaranteed amount of energy at all times (Mohamed et al. 2015). We have to deal with these challenges in controlling a microgrid: firstly, the demand has to be fulfilled while maximizing the use of the renewable resources and minimizing the cost of additional generators or storage. Secondly, we have to deal with the uncertainty caused by the weather dependent performance of the renewable resources.

Many different modeling techniques have been used in the past to model microgrids (see (Ahmad Khan et al. 2016; Gamarra and Guerrero 2015; Meng et al. 2016)). One of the most commonly used techniques is mathematical programming, which is also applied in this chapter.

Furthermore, many strategies have been proposed to effectively deal with the uncertainty in microgrid planning problems. A popular approach is using *model predictive control* or *rolling horizon* strategies (e.g. by Palma-Behnke et al. (2013)). But considerable work has also been done on using stochastic models for similar problems (see (Dai, Zhang, and Su 2015; Huang, Pardalos, and Zheng 2017; Liang and Zhuang 2014)).

In this chapter, we compare different optimization approaches for planning the generators and storage units in the islanded microgrids. We consider an economical optimization (at the level of unit commitment and economic dispatch (Liang, Tamang, et al. 2014)) of the energy supply. We developed a deterministic optimization model and a multi-stage stochastic optimization model. The properties of the forecasts used were important drivers in the development of these models. The models are described in Section 4.2, where we also describe an algorithm for the generation of the scenario tree needed for the second model. The intuition is that the latter model in general works better, but it is not clear in advance whether the advantages of this more complex model also work for small microgrid environments.

To evaluate our approach, we performed a case-study for an islanded microgrid in the Netherlands, with the following components:

2 Olympian GEP55-1 diesel generators;

1 Kohler SDMO K44C3 diesel generator;

1 Lithium-ion battery;

3 PV-fields, with a combined maximum output of 12 kW;

5 load-groups, with a combined average load of 9.6 kW.

In the simulation experiments in this study, the grid generation and storage planning is performed by our different models in combination with different re-planning strategies and different forecasting accuracies. Our approach distinguishes itself from many similar approaches through the application of these strategies. The experiment is described in more detail in Section 4.3. In Section 4.4 we analyze the results of the simulations. Finally, some preliminary conclusions are drawn in Section 4.5. A summary of the notation used throughout this chapter is provided in Table 4.1.

4.2. Optimization models

4.2.1. Deterministic optimization model

The model presented here contains all the (controllable) components of the microgrid and enforces production and consumption to be balanced at every time step (15-minute interval) within the planning horizon ($|T|$ time steps ahead). It is similar to the model used by Palma-Behnke et al. (2013).

Uncontrollable demand and generation

The uncontrollable components of the demand (mainly consumers) and generation (mainly renewable energy resources) are modeled in D_t^+ (demand) and D_t^- (generation). The difference between these variables equals the demand that has to be met by the controllable components in the microgrid. We assume that forecasts are available for all uncontrollable components in the grid, providing an expected value and a measure of uncertainty that together define a distribution for the actual value at each time step. In this work, we assume the forecast error to be normally distributed, and thus require a mean (μ) and standard deviation (σ) to be provided.

Conventional generators

Conventional generators, like diesel generators, are represented by three variables every time step: $g_{it} \geq 0$ represents the power output of generator i at time t, $y_{it} \in \{0, 1\}$ describes whether generator i is on or off at time t and $x_{it} \in \{0, 1\}$ shows if generator i was started at time t. The behavior of a generator is modeled by the following two constraints, where P_i^{min} and P_i^{max} are the minimum and

Sets	
T	set of time steps
G	set of generators
B	set of batteries
S	set of scenarios
Parameters and constants	
D_t^+, D_t^-	uncontrollable demand and generation
P_i^{min}, P_i^{max}	minimum and maximum output of generator i
f_i, s_i, c_i^g	base operating cost per time step, start-up cost and operating cost per kWh for generator i
v_i^c, v_i^d	maximum charge and discharge speed for battery i
E_i	total capacity of battery i
η_i^b	(dis)charging efficiency of battery i
R_i^{min}, R_i^{max}	minimum and maximum thresholds for the state of charge of battery i
c_i^b	battery discharge cost per kWh
η^g	grid efficiency
μ_t, σ_t	mean and standard deviation of forecast error at time step t
ρ_l, ρ_p	correlation between time steps for the load and PV forecasts
ξ_t^i	forecasted value(s) in scenario i for time step t
c^w	penalty for demand and supply not covered in a scenario per kWh
τ	number of time steps before resolution shift
δ	deviation tolerance for 'deviate' re-planning strategy
Variables	
g_{it}	power output of generator i at time step t
y_{it}	on/off indicator for generator i at time step t
x_{it}	indicator for generator i starting up at time step t
ch_{it}	state of charge for battery i at time step t
$b_{it}^{in}, b_{it}^{out}$	input and output of battery i at time step t
r_{it}	indicator of availability of battery i for use in the schedule at time step t
w_t^+, w_t^-	demand and supply not covered in a scenario at time t

Table 4.1: Summary of notation

maximum output of generator i, respectively:

$$y_{it} - y_{it-1} \leq x_{it} \tag{4.1}$$

$$y_{it} P_i^{min} \leq g_{it} \leq y_{it} P_i^{max} \tag{4.2}$$

For each generator, three cost constants are defined: f_i (the base operating costs for a single time step), s_i (start-up costs) and c_i^g (fuel costs and operating costs per kWh).

Energy storage systems

Batteries are modeled by a variable defining their state of charge ($0 \leq ch_{it} \leq E_i$) and variables representing its input ($0 \leq b_{it}^{in} \leq v_i^c$) and output ($0 \leq b_{it}^{out} \leq v_i^d$). The constants v_i^c and v_i^d represent the maximum charging and discharging speeds, respectively, and E_i is the total capacity of the battery. For simplicity, the efficiency of charging and discharging are considered to be equal (η_i^b). Therefore, the round-trip efficiency is effectively $(\eta_i^b)^2$. This behavior is modeled in the following constraint:

$$ch_{it-1} - ch_{it} = \frac{b_{it}^{out}}{\eta_i^b} - b_{it}^{in} \eta_i^b \tag{4.3}$$

In addition, for the deterministic model, we introduce two thresholds, R_i^{min} and R_i^{max}, and a variable (r_{it}) that controls whether we can use the battery in our planning to supply energy. We enforce that $ch_{it} \geq R_i^{min}$, meaning that we have to charge the battery if its initial state is below R_i^{min}. Furthermore, the following constraints make sure that we only use the battery in our planning when its state of charge is above R_i^{max}:

$$R_i^{max} r_{it} \leq ch_{it} \tag{4.4}$$

$$b_{it}^{out} \leq r_{it} v_i^d \tag{4.5}$$

The costs associated with the use of a battery (mainly device wear) are distributed over the entire lifespan of the battery and are charged per kWh discharged from the battery (c_i^b).

Energy balance

Maintaining energy balance is crucial in a microgrid. This requirement is expressed in the following constraint:

$$\frac{D_t^+}{\eta^g} - D_t^- \eta^g = \sum_{i \in G} g_{it} + \sum_{j \in B} \left(b_{jt}^{out} - b_{jt}^{in} \right) \tag{4.6}$$

The total production of the controllable components (right-hand side) should equal the total demand of the uncontrollable components (left-hand side). In this expression, η^g is an efficiency constant, modeling grid losses.

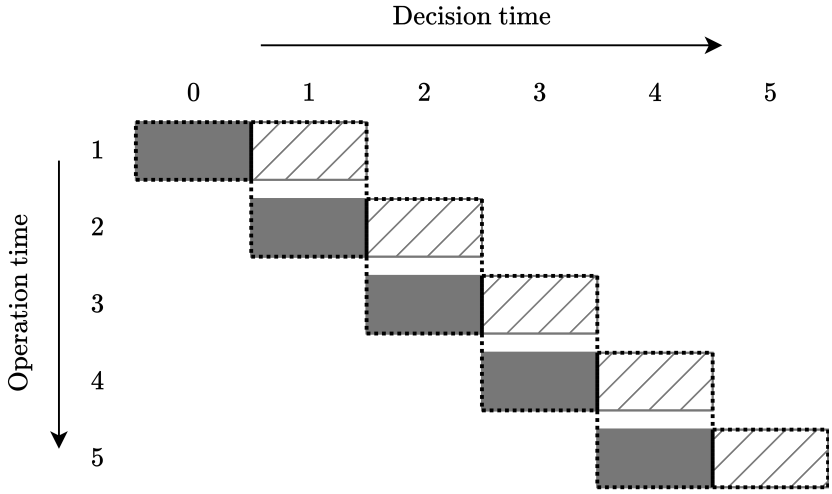


Figure 4.1: Problem structure showing when decisions are fixed (decision time) and when they are applied (operation time). The dotted outlines represent decisions that are based on the same available information.

Objective function

The objective is to manage the microgrid at minimal cost. This can be achieved by minimizing the following function, which includes the costs associated with all devices over all time steps within the planning horizon:

$$\sum_{t \in T} \left(\sum_{i \in G} (f_i y_{it} + c_i^g g_{it} + s_i x_{it}) + \sum_{j \in B} \left(c_j^b b_{jt}^{out} \right) \right) \tag{4.7}$$

Time shift

Both models include a shift in resolution after τ time steps. The first τ time steps have a fifteen minute-resolution, and the remaining time steps have a one hour-resolution. This decreases the problem size by considering the distant future in less detail.

4.2.2. Multi-stage stochastic optimization model

The deterministic optimization model is relatively simple and efficient, but relies on a single forecast for the whole period. In the multi-stage model we assume that the output, i.e. actual values of the uncertain parameters, of one period is also part of the input for the decision of the next period (as shown in Figure 4.1). Moreover, instead of safety constraints (4.4) and (4.5), this model deals with the uncertainty D_t^+ and D_t^- by defining multiple scenarios (with different values of D_t^+ and D_t^-) for each time step. The objective function becomes a recursive expression, representing the expected value by multiplying the probability of every scenario by its objective value for every time step. In addition, two (heavily weighted) penalty

Input: planning horizon $|T|$, correlation ρ, forecasts $\mu = (\mu_1, ..., \mu_{|T|})$, standard deviations $\sigma = (\sigma_1, ..., \sigma_{|T|})$

$X_1 \sim \mathcal{N}(0, 1)$
$s_1 \leftarrow \mu_1 + \sigma_1 X_1$
for $i \leftarrow 2$ **to** $|T|$ **do**
  $X_i \sim \mathcal{N}(0, 1)$
  $X_i \leftarrow \rho X_{i-1} + \sqrt{1 - \rho^2} X_i$
  $s_i \leftarrow \mu_i + \sigma_i X_i$
return $(s_1, ..., s_{|T|})$

Algorithm 4.1: Sampling a single scenario

variables (w_t^+, w_t^-) are added to constraint (4.6) in every scenario to relax the requirement that the first stage solution can satisfy the demand in *all* scenarios. For background on multi-stage stochastic optimization, refer to the book by Pflug and Pichler (2014).

The problem displays a stage-wise structure in two dimensions: on the one hand, we divide our problem in time steps. On the other, we can also divide the decisions in two groups: those that we fix beforehand, and those that we make when the scenario for that time step has been revealed. Note that the scenario-based decisions of time step t are made with the same available information as the pre-fixed decisions of time step $t + 1$. Fig. 4.1 shows this structure, where the solid black boxes represent pre-fixed decisions and the hatched boxes represent the scenario-based decisions.

For the multi-stage model, we generate a number ($|S|$) of scenarios. A scenario consists of a number of (sub)scenarios: one for each type of uncontrollable component. Each of these is sampled independently, following the algorithm presented in Fig. 4.1, taking the correlation (ρ) between time steps into account.

In our case study, we have two (sub)scenarios in every scenario: one for the uncontrollable load (with correlation-coefficient ρ_l) and one for the PV-output (with correlation-coefficient ρ_p).

The scenarios each represent a series of possible realizations of the uncertain constants in our model. By considering these scenarios individually, we would effectively be solving $|S|$ distinct deterministic problems: when the realization for the first time step is revealed, we "know" what the rest of the scenario will be like. However, we want to keep taking multiple possibilities into account. In order to achieve this, we need a scenario tree, that branches at each time step. The idea is to "merge" different scenarios that are similar in the beginning in earlier time steps.

We use the *simultaneous backward reduction* algorithm described by Gröwe-Kuska, Heitsch, and Römisch (2003) to obtain a scenario tree from the sampled scenarios.

We define the distance between two scenarios at time step t to be:

$$dist(\xi_t^i - \xi_t^j) = |(\xi_{t,load}^i - \xi_{t,pv}^i) - (\xi_{t,load}^j - \xi_{t,pv}^j)|$$

where $\xi_{t,pv}^i$ is the forecasted value of the PV-output for time step t in scenario i and $\xi_{t,load}^i$ is the forecasted value for the load at time step t in scenario i.

At every level in the tree (corresponding to a time step in the model), we "merge" the two most similar scenarios (those with the smallest distance up till time step t, using the measure given above), until the number of scenarios at that level is equal to $\frac{|S| \cdot t}{\tau + \frac{|T|-\tau}{4}}$, where t is the index of the time step (a level in the scenario tree corresponds to a time step in the optimization problem). This results in a linear increase in the number of scenarios over time. We end up with a scenario tree with a single root node and $|S|$ leaves, each corresponding to one of the original scenarios.

4.2.3. Re-planning strategies

The optimization models described above generate a planning for the next $0.25 \cdot |T|$ hours, so we have to decide how much of this planning will be used, and when to re-plan in case the forecasts deviate from the actual generation and demands of energy. The most straight-forward way to use the described planning algorithms, is by applying them every 15 minutes to generate an updated planning. This only makes sense for the deterministic model and is called "deterministic-simple" re-planning. We evaluate two other re-planning strategies:

1. *Deviate*: only generate a new planning when the reduced demand (load minus PV-output) of none of the generated scenarios is within δ kWh of the actual reduced demand in the previous time step;

2. *Recover*: only generate a new planning when the enforced decisions of the planning in the last time step had to be adjusted, meaning that the batteries were unable to compensate the difference between planning and realization.

If no new planning is generated, the simulation will use the decisions for the corresponding time-step of the most recent planning. In the deterministic case, this choice is straight-forward. In the case of the multi-stage model, we take the decisions from the scenario that is the most similar to reality observed so far.

4.3. Experiments

As a basis for our experiments, we used the data from our case study in the period of 1 – 8 June 2017. The initial state of the simulation is equal to the state of the grid on 1 June 2017 at 00:00, and the observed values of PV-output and load of this period are used as the realized values in the simulation. We performed experiments with the following model/strategy-combinations and forecasts as input for the optimization models:

Devices				Parameters	
Olympian DG		*Kohler DG*		δ	0.25
P^{min}	8 kW	P^{min}	6.4 kW	τ	24
P^{max}	40 kW	P^{max}	32 kW	$\|T\|$	96
s	€1.00	s	€1.00	$\|S\|$	200
f	€0.40	f	€0.40	ρ_p, ρ_l	0.7
c^g	€0.30	c^g	€0.30		
Battery		*Other*			
E	20 kWh	c^w	2		
v^c, v^d	12 kW	η^g	0.97		
R^{max}	3 kWh				
R^{min}	1 kWh				
η^b	0.93				
c^b	€0.10				

Table 4.2: List of constants and parameters

- Models: deterministic-simple, deterministic-deviate, deterministic-recover, multi-stage-deviate (5x) and multi-stage-recover (5x);
- Forecast-construction with a standard-deviation of: 2% (5x), 10% (5x) and 20% (5x).

  This results in 195 runs of eight simulated days for every tested week. Of each run, the first day serves as a warm-up period, which means that it is excluded from the results presented below.

An overview of the values of constants and parameters used in our experiments is provided in Table 4.2.

4.3.1. Artificial forecasts

As input for our problem, we constructed a number of artificial forecasts. We used artificial forecasts to avoid a situation where the week that is simulated happened to have very stable weather and (almost) perfect forecasts, which would be very easy to optimize for even the simplest model. Forecasts were sampled using the same algorithm that is used to sample scenarios in the multi-stage approach (Fig. 4.1), with a standard deviation of 2% (5x), 10% (5x) or 20% (5x) of the real value. Resulting in a total of 15 different artificial forecasts.

4.3.2. Simulation

Our simulation adjusts the device behavior, as described in the pre-generated planning, every time step to match the actual demand in real-time. No optimization model is used at this stage, the simulation uses a few simple rules-of-thumb to match production and consumption exactly. In general, if adjustments are neces-

Forecast st. dev.	Model	Strategy	Realized cost	# skipped
2%	deterministic	plain	€414.83	0
		deviate	€414.95	434
		recover	€414.94	293
	multi-stage	deviate	€473.85	646
		recover	€459.37	321
10%	deterministic	plain	€416.31	0
		deviate	€416.04	402
		recover	€415.68	294
	multi-stage	deviate	€472.85	536
		recover	€460.52	321
20%	deterministic	plain	€416.13	0
		deviate	€416.71	329
		recover	€417.75	314
	multi-stage	deviate	€467.52	425
		recover	€461.78	314

Table 4.3: Average metrics for different strategies

sary, generator output is adjusted first (if one is active already), then the battery output is adjusted and finally PV-output can be curtailed or an additional generator may be brought on-line. As a final measure, load-shedding might take place. Because of the large generation capacity of the generators, however, this is very unlikely. For a more detailed description of these rules, refer to the thesis by Brouwer (2017).

The costs of energy production and storage are also calculated and reported at this stage. The battery state and generator state after applying the required adjustments serve as input for the planning problem, starting from the next time step. At this point, the decision is made on whether re-planning is necessary or not, using the strategies as described in Section 4.2.3.

4.3.3. Metrics

For each group of runs, we report two metrics:

- (Average) **realized cost**. This is the total cost of operation as determined during simulation, with a correction for the difference in the state of charge of the battery between the start and the end of the simulated period.
  Let Δ_{ch} be the difference in state of charge (in kWh). The correction then equals $-c^g \cdot \Delta_{ch}$, which is equal to the minimal cost of producing the same

amount of energy using a diesel generator, times -1.

- (Average) number of **skipped** planning steps. The number of time steps that the planning stage was skipped and an old planning was re-used, because of the application of one of the strategies described in Section 4.2.3.

4.4. Results

The results are summarized in Table 4.3. In terms of the realized cost, all five approaches produce stable results in the face of growing uncertainty (Fig. 4.2). In general, the deterministic model results in lower overall costs, that only increase slightly when the uncertainty grows. The multi-stage model is more expensive. Interestingly, the deviate-strategy seems to perform better with larger uncertainty. This can be explained by the decrease in skipped planning steps (Fig. 4.3): with a larger uncertainty, the strategy is forced to re-plan more often as the probability of having considered a sufficiently similar scenario decreases. This, in turn, allows adjustment of the previous decisions to the exact current state of affairs, which increases the quality of the solution as a whole.

The deviate-strategy has a clear advantage over the other strategies in the number of planning steps that can be skipped. As would be expected, this advantage is even stronger with the multi-stage model: the multi-stage-deviate combination, only re-plans on average in less than 1 in 7 steps.

We have to conclude, however, that the multi-stage model is not capable of bringing down the cost of operation in our case study. This can be blamed, in part, on the relatively small size of the battery in the test system. We found that the range of the generated scenarios in the multi-stage problem is often larger than what can be covered by adjusting the battery output. This forces the situation where some of the scenarios are not satisfied in the second stage (i.e. penalty terms w_t^+ and w_t^- are used). These penalty terms then contribute heavily to the objective function, and the resulting planning will be one that minimizes the penalty, which is not necessarily the most cost-effective solution for an "average" scenario. Note that, when the realization *is* close to one of the unsatisfied scenarios, the planning for that scenario will still be used - even though we know for sure that it will need adjustment. This potentially compounds the issue.

The state of charge of the battery in the runs that use the multi-stage model is kept at very high levels for a majority of the time. At the same time, we see that the number of generator starts approximately doubles when compared to the runs using the deterministic model. We expect this to be the result of the issue discussed above. However, this means that the planning will be very similar for different uncertainty levels and explains part of the higher operation cost: start-ups are expensive, and a full battery might force PV-output to be curtailed in the case of overproduction, which means that "free" energy is wasted.

We expect the multi-stage model to perform better when the range of considered scenarios is (relatively) smaller: either by decreasing the considered range

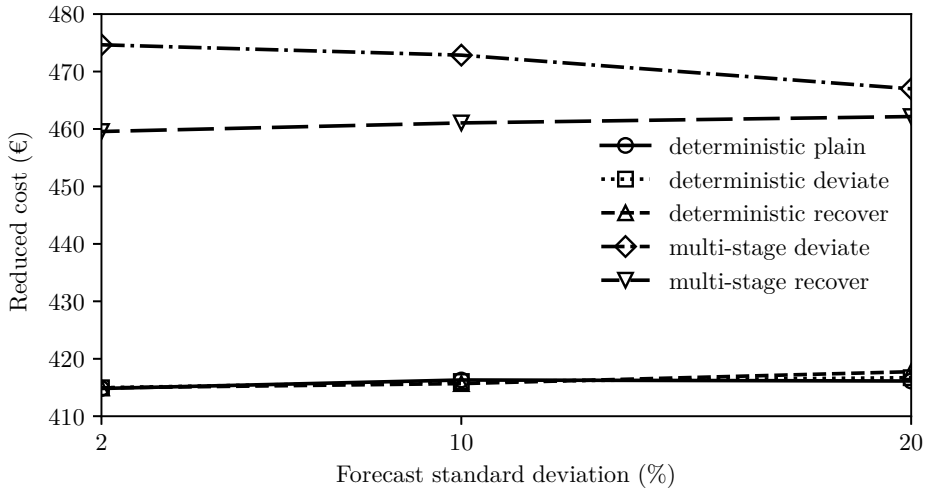


Figure 4.2: Realized cost of five approaches for different forecast accuracies

explicitly or increasing the capability of adjustment in the scenario-based decisions (either by increasing the battery size, or by allowing other decisions to be scenario-based as well).

Finally, a note on the computational performance of each of the approaches. We will not discuss exact numbers, as these may be misleading, given that a time limit of ∼10 minutes was imposed on every planning step, in order to evaluate its performance in an on-line setting. The deterministic model typically takes less than a second to be solved (using CPLEX v12.7.1), while the multi-stage model takes significantly longer. For solving a multi-stage problem, the time limit is exceeded in some time steps, which means that the solution has a larger (proven) optimality gap (on average at most 2%). The impact on the solutions is manageable, and can be reduced in multiple ways, e.g. by reducing the number of scenarios.

4.5. Conclusion

The deterministic approach that we developed has proven its worth. As a result, it is currently implemented in the microgrid that we based our case study on.

Based on the results in our case study, we cannot justify the application of the more complex model in this specific case. We expect this to be caused, at least in part, by the particular make-up of this grid (the small storage capacity, relative to the large generation capability), in combination with our modeling of battery output as the adjustable part of the planning.

The performance of the deterministic model we proposed shows, however, that

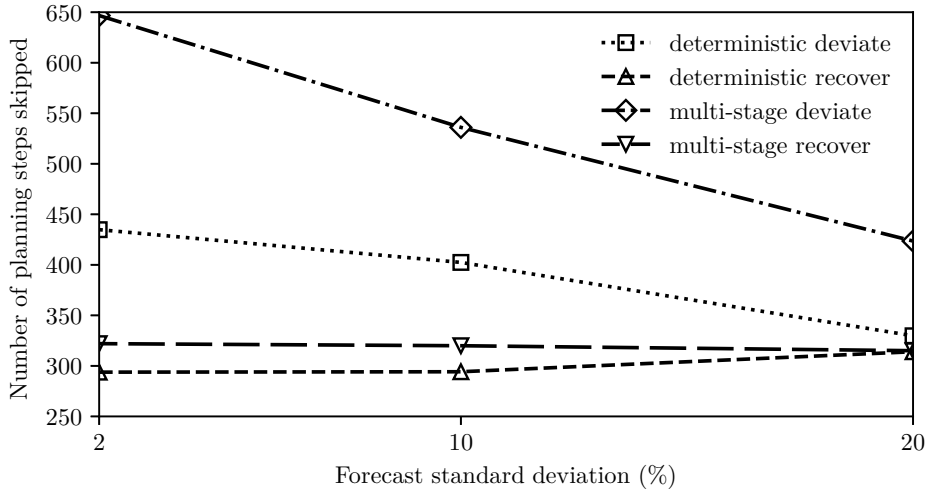


Figure 4.3: Number of planning steps skipped by four approaches for different forecast accuracies

planning the operation of a microgrid in this way has its advantages: the resulting planning is robust in the face of large uncertainty in the forecasts of load and PV output. The system displays intuitive behavior when following the planning, and the cost of operation do not increase by doing so. It should be noted that the safety constraints (constraint (4.4) and (4.5)) are crucial here. A naive model that lacks these, does not perform as well.

In addition, we have shown that applying strategies for skipping the planning phase in a number of time steps does not necessarily mean that the resulting solution will be more expensive. This means that less computing resources are needed to implement the automated planning of generators and storage units, when compared to a traditional rolling horizon strategy.

This case study is a first step in investigating the application of the approaches we proposed for the management of microgrids. Next steps include evaluating the different models for microgrids with different configurations (e.g. a relatively larger storage capacity or smaller flexible generation capability) and further investigation of the re-planning strategies.

Further fine-tuning and testing on different cases is required to deliver a final statement on the usefulness of applying complex optimization models to this problem. The success of the deterministic model and the re-planning strategies provides promising opportunities for improving typical operation strategies.

References

Ahmad Khan, A., M. Naeem, M. Iqbal, S. Qaisar, and A. Anpalagan (2016). "A compendium of optimization objectives, constraints, tools and algorithms for energy management in microgrids". *Renewable and Sustainable Energy Reviews* 58, pp. 1664–1683. DOI: 10.1016/j.rser.2015.12.259.

Brouwer, R. J. J. (2017). "Forecast-based optimal operation of islanded microgrids". Master's thesis. Utrecht University. DOI: 20.500.12932/27907.

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

Dai, H., N. Zhang, and W. Su (2015). "A literature review of stochastic programming and unit commitment". *Journal of Power and Energy Engineering* 3.4, pp. 206–214. DOI: 10.4236/jpee.2015.34029.

Gamarra, C. and J. M. Guerrero (2015). "Computational optimization techniques applied to microgrids planning: A review". *Renewable and Sustainable Energy Reviews* 48, pp. 413–424. DOI: 10.1016/j.rser.2015.04.025.

Gröwe-Kuska, N., H. Heitsch, and W. Römisch (2003). "Scenario reduction and scenario tree construction for power management problems". *Proceedings of the 2003 IEEE Bologna Power Tech Conference*. Vol. 3. DOI: 10.1109/PTC.2003.1304379.

Huang, Y., P. M. Pardalos, and Q. P. Zheng (2017). *Electrical Power Unit Commitment: Deterministic and Two-Stage Stochastic Programming Models and Algorithms*. DOI: 10.1007/978-1-4939-6768-1.

Liang, H., A. K. Tamang, W. Zhuang, and X. S. Shen (2014). "Stochastic information management in smart grid". *IEEE Communications Surveys & Tutorials* 16.3, pp. 1746–1770. DOI: 10.1109/SURV.2014.020614.00115.

Liang, H. and W. Zhuang (2014). "Stochastic modeling and optimization in a microgrid: a survey". *Energies* 7.4, pp. 2027–2050. DOI: 10.3390/en7042027.

Lopes, J. A. P., C. L. Moreira, and A. G. Madureira (2006). "Defining control strategies for microgrids islanded operation". *IEEE Transactions on Power Systems* 21.2, pp. 916–924. DOI: 10.1109/TPWRS.2006.873018.

Meng, L., E. R. Sanseverino, A. Luna, T. Dragicevic, J. C. Vasquez, and J. M. Guerrero (2016). "Microgrid supervisory controllers and energy management systems: A literature review". *Renewable and Sustainable Energy Reviews* 60, pp. 1263–1273. DOI: 10.1016/j.rser.2016.03.003.

Mohamed, M. A., A. M. Eltamaly, H. M. Farh, and A. I. Alolah (2015). "Energy management and renewable energy integration in smart grid system". *Proceedings of the 2015 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*, pp. 1–6. DOI: 10.1109/SEGE.2015.7324621.

Palma-Behnke, R., C. Benavides, F. Lanas, B. Severino, L. Reyes, J. Llanos, and D. Sáez (2013). "A microgrid energy management system based on the rolling horizon strategy". *IEEE Transactions on Smart Grid* 4.2, pp. 996–1006. DOI: 10.1109/TSG.2012.2231440.

Pflug, G. C. and A. Pichler (2014). *Multistage Stochastic Optimization*. Springer Series in Operations Research and Financial Engineering. DOI: 10.1007/978-3-319-08843-3.

5

Grid-constrained online scheduling of flexible electric vehicle charging

The contents of this chapter will be submitted for publication in the Journal of Heuristics. A preprint is available on arXiv:

E. van Huffelen, R. J. J. Brouwer, and J. M. van den Akker (2024). "Grid-constrained online scheduling of flexible electric vehicle charging". Preprint at: http://arxiv.org/abs/2403.03109. DOI: 10.48550/arXiv.2403.03109

Abstract

We study Electric Vehicle (EV) charging from a scheduling perspective, aiming to minimize delays while respecting the grid constraints. A network of parking lots is considered, each with a given number of charging stations for electric vehicles. Some of the parking lots have a roof with solar panels. The demand that can be served at each parking lot is limited by the capacity of the cables connecting them to the grid. We assume that EVs arrive at the parking lots according to a known distribution. Upon arrival, we learn the desired departure time, the amount of electrical energy it needs to charge its battery, and the range of rates that it can be charged at. Vehicle arrival patterns, connection times, and charging volume are based on data collected in the city of Utrecht. The departure time of an EV is delayed if it has not finished charging in time for its desired departure. We aim to minimize the total delay. We present a novel approach, based on an online variant of well-known schedule generation schemes. We extend these schemes and include them in a destroy-and-repair heuristic. This resulted in several scheduling strategies. We show their effectiveness using a discrete event simulation. With this, we show that applying scheduling approaches increases the amount of EVs that can be charged at a site and reduces the average delay. Furthermore, we argue the importance of considering aspects of the grid layout in electricity networks and show the benefits of using flexible charging rates.

Keywords: *smart EV charging, scheduling, simulation, OR in energy.*

5.1. Introduction

The number of electric vehicles (EVs) has rapidly increased in recent years. This trend is likely to continue, as policies aiming to reduce greenhouse gas emissions encourage their sale to replace vehicles that run on fossil fuels. The increasing number of EVs requires charging stations to be installed in many locations. At parking lots where many EVs congregate, such as one near a large office building for example, managing the demand for charging the batteries of all these EVs concurrently is a challenge. Such parking lots often have clear peaks in demand, and are relatively quiet at other times. For parking lots outside residential neighborhoods, this peak in demand often occurs during the day, which matches well with the peak in production of solar panels. Therefore, the installation of a large number of charging stations is often paired with the installation of solar panel arrays on roofs over these parking lots. Although the peaks in supply and demand do not line up as nicely for parking lots in residential areas, solar panels are commonly installed in these areas as well. The servicing of EVs at these parking lots will benefit from smart scheduling of the charging jobs to align well with the availability of power at the parking lot.

In this work, we consider a network of parking lots operated by the same provider, who offers the service of parking and charging electric vehicles (EVs). Each parking lot has a given number of charging stations for EVs. The demand that can be served at each parking lot is limited by the amount of available power. The power supply is constrained by the capacity of the cables connecting the parking lots to the grid. Some parking lots are covered by a roof with solar panels on it, supplying additional power under the right weather conditions.

We assume that EVs arrive at the parking lots according to a known distribution. As soon as a vehicle arrives, we learn its desired departure time, the amount of electrical energy it needs to charge its battery before that time, and the range of rates that it can be charged at. The charging of these EVs must be scheduled in such a way that the total delay is minimized, while the network constraints are respected. This means that the summed difference between the desired departure time and the actual departure time, which is delayed if an EV has not finished charging in time, must be minimal across all vehicles.

The charging of electric vehicles is a widely studied subject across multiple disciplines, even if we limit ourselves to the (optimal) scheduling of charging jobs. Many surveys aim to provide an overview of aspects of this problem. A recent survey on this topic was performed by Pasha et al. (2024). Earlier surveys include the work by Wang et al. (2016), who make a classification of EV charging control algorithms based on their perspective and objective. They identify three categories: *smart grid oriented*, *aggregator oriented* and *customer oriented* EV charging, describing the perspective from which the problem is considered. Although the objectives discussed in the review are mainly cost optimization objectives, we could consider our problem setting to align with the *aggregator oriented* model, as we aim to provide the best possible service within the limited capacity of the network.

The control of EV charging can be considered on several 'levels', or timescales (Arif et al. 2021). The lowest level of control is reactive, and mainly concerns stabilization of the grid. Arif et al. (2021) describe that EVs can be used at that level to provide ancillary services by controlling the charging (grid-to-vehicle, G2V) or discharging (vehicle-to-grid, V2G) at a very small timescale. Such timescales make scheduling intractable. We focus on higher levels of control: aligning demand well with forecasts of solar panel output and the demand of other EVs within the same network. The importance of the uncertainty of demand is emphasized by Al-Ogaili et al. (2019). This implies that offline scheduling methods may struggle to account for the uncertainty surrounding the arrival times and charging demands of EVs at charging locations. Therefore, we consider an online scheduling approach.

Most surveys conclude that coordinated charging strategies, where some centralized decision-making is involved, yield the best results. Hussain et al. (2021), for example, states that "it is found that centralized coordination is [the] best strategy to handle all issues effectively." Liu et al. (2015) also find that centralized approaches have better performance, but prefer decentralized approaches due to the communication overhead of centralized approaches.

We will be looking at a centralized control strategy, from the perspective of an aggregator. We aim to schedule the demand in a network of parking places as well as possible, given the limited resources available. In this sense, the problem we consider is closely related to the Resource-Constrained Project Scheduling Problem (RCPSP). The surveys by Hartmann and Briskorn (2010, 2022) give a good overview of the RCPSP and its variants. The RCPSP with flexible resource profiles (FRCPSP) such as studied by Naber (2017) and the General Continuous Energy-Constrained Scheduling Problem (GCECSP) (Brouwer, van den Akker, and Hoogeveen 2024; Chapter 3) are most closely related to the present work.

The approaches we develop in this work are inspired by priority rule based scheduling heuristics commonly employed for finding solutions to RCPSP and its variants. Kolisch (1996) provides a good overview of the application of schedule generation schemes to the classical RCPSP. Lova, Tormos, and Barber (2006) apply schedule generation schemes to the multi-mode RCPSP, where tasks can be processed in a number of different modes, which can be seen as a discretized variant of the flexible model we are considering. More recently, priority rule based approaches that are similar to schedule generation schemes have been applied to the stochastic RCPSP (Chen et al. 2018) and in the context of genetic algorithms, where compound priority rules are learned for the generation of schedules (Đumić and Jakobović 2021).

Our contribution: We present a novel approach, based on an extension of traditional schedule generation schemes to an online setting with flexible charging rates. We consider single-pass methods, where we apply a scheme once, as well as more advanced methods, where we include the schemes in a destroy-and-repair heuristic. In this way, we develop a number of variants of the approach for the generation of efficient schedules for charging EVs on a network of parking lots. As

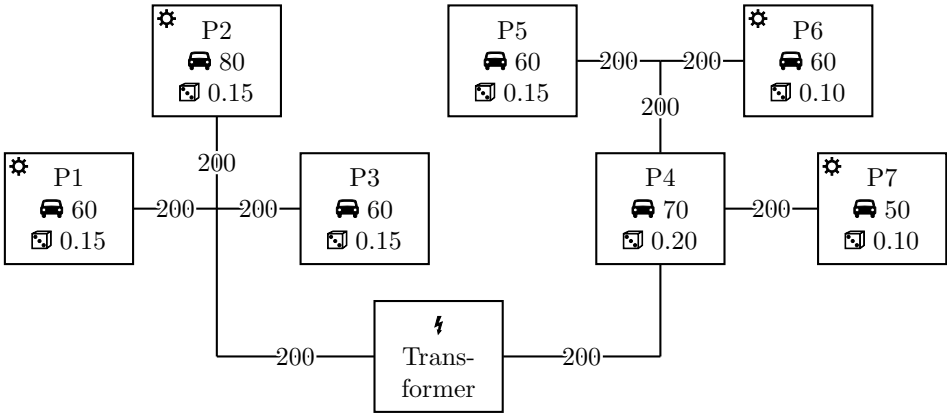


Figure 5.1: Network layout

far as we are aware, similar approaches have not been applied to this problem yet. We show the effectiveness of these approaches using a discrete event simulation. Furthermore, we argue the importance of considering aspects of the grid layout in electricity networks and show the benefits of using flexible charging rates.

The work presented here is a continuation of earlier work by van Huffelen (2023). The specific case is loosely based on an assignment used in the master's course 'Optimization for sustainability' at Utrecht University (van den Akker and Posthoorn 2021). The distribution of solar output is based on a forecast of the production of solar panels in the Netherlands in 2025 (ENTSO-E 2018). The data used for the distribution of arrival times, charging volumes and connection times is based on real-world data from the city of Utrecht, and was provided by Brinkel, AlSkaif, and van Sark (2020). The authors of this paper argue, as we do, that the usage of smart charging strategies reduces the need for grid reinforcement and is, in fact, more cost-effective (Brinkel, Schram, et al. 2020). In later work, they develop an approach for a mix of shared and privately-owned EVs using V2G techniques to mitigate grid congestion (Brinkel, AlSkaif, and van Sark 2022).

The rest of this work is structured as follows. In Section 5.2, we give a detailed problem description, followed by an explanation of the developed scheduling approaches in Section 5.3. We present the results of evaluating our approaches using that simulation in Section 5.4. Finally, we will draw some conclusions and discuss avenues for future work in Section 5.5.

5.2. Problem description

We consider an online problem, where a schedule for the charging of EVs needs to be generated with the objective to minimize the average delay. EVs arrive over time, which implies that the schedule needs to be updated regularly. We will discuss strategies for these updates in Section 5.3.

We consider a network that consists of a number of interconnected parking lots, that share a grid connection. An example network is shown in Figure 5.1. Each parking lot has:

- a number of parking spots, where EVs can be parked and charged;
- the fraction of EVs $p_{\mathrm{P}i}$ that will select parking lot Pi;
- (optionally) an array of solar panels, producing a variable amount of power.

These parking lots are connected by cables with a given maximum capacity. The cable capacity limits the amount of power that can be transported to each parking lot. We use this a simple approximation of the limits imposed by the network.

The amount of power generated by the solar panels is based on the so-called availability vector. This expresses the average production of solar panels at any given time as a fraction of their peak power. The production of the solar panels is revealed at the start of every hour, and treated as a constant production rate for the entire hour.

The EVs arrive at this location according to a Poisson process. The arrival rate is not constant, but changes every hour. Each EV j has a number of properties, which are revealed when the vehicle arrives. Here, EV j is the jth vehicle in the order of arrival:

- a connection time $d_j - r_j$, the amount of time between its arrival time r_j and its desired time of departure d_j;
- a charging volume E_j: the amount of energy required to fully charge its battery before departure (in kWh);
- charging rates $[P_j^-, P_j^+]$, a range of rates between a lower (P_j^-) and an upper (P_j^+) bound that can be used to charge it, in kW;
- a parking preference: an ordered list of three parking lots, where the EV will try to park. If all three parking lots in the parking preference of an EV are full, it will leave the area.

For each EV, we determine a start and completion time for the charging process. The start time must be after its arrival, which implies that the EV may not start charging immediately. After its start, we determine a charging profile that will be followed. It completes when its full energy demand is served, which will be earlier than or, if that is not possible, as close to its preferred departure time as possible. Note that we guarantee that the connection time is never smaller than the time it takes to charge an EV at its minimum rate, i.e. E_j/P_j^-.

In addition, note that the existence of a lower bound implies that preemption is not allowed: as soon as an EV has started charging, it cannot drop below the lower bound on its charging rate before it completes. A reason for this is that interruption of the charging process comes with some loss of efficiency caused by startup and shutdown effects.

In this setting, we want to minimize the total tardiness $\sum_j \max(0, C_j - d_j)$ over all EVs that parked at the network of parking lots.

In this work, we use a case study for the evaluation of our approaches, loosely based on an assignment used in the master's course 'Optimization for sustainability' at Utrecht University (van den Akker and Posthoorn 2021). It uses the network depicted in Figure 5.1. All cables have a capacity of 200 kW. We use the availability factor predicted for solar panels in summer in the Netherlands in 2025 (ENTSO-E 2018), which we provide in Appendix 5.B.1. We assume the average hourly power output to follow a normal distribution, with the value from the availability vector as its mean, and a standard deviation of 15%. All solar panel arrays installed have a peak power of 200 kW, unless explicitly stated otherwise. We assume an average arrival rate of 1125 EVs per day. We use hourly arrival rates, charging volumes, and connection times based on data from Brinkel, AlSkaif, and van Sark (2020). These distributions are provided in Appendix 5.B.2, 5.B.4 and 5.B.5. The distribution of charging ranges can be found in Appendix 5.B.3.

5.3. Scheduling

As EVs keep arriving continuously, we use an online scheduling approach. Only the characteristics of the EVs that have already arrived have been revealed, and the amount of power produced by the solar panels is only known for the current hour. The available information will serve as the basis for a schedule that determines when and at what rate each EV charges. In this section, we will first introduce the methods that are used to find schedules based on this information. After that, we will discuss a strategy for replacing these schedules when new information is revealed.

5.3.1. Schedule generation schemes

In this chapter, we consider priority rule based scheduling heuristics. These are made up of two components: a schedule generation scheme and a priority rule. Priority rules will be discussed in Section 5.3.2. The schedule generation scheme is a constructive heuristic that adds jobs to the schedule one by one, until a feasible schedule is obtained.

There are two main variants of schedule generation schemes that are often considered in literature: serial and parallel (Kolisch 1996). A convenient way to characterize the difference is that the serial scheme iterates over jobs, while the parallel scheme iterates over points in time. The serial scheme aims to schedule the highest priority job as early as possible, and the parallel scheme tries to fill the current time with as high-priority jobs as possible. Both variants construct a schedule by adding jobs, until all jobs have been placed. In simple schedule generation schemes, the schedule that results from this procedure is not adapted further. These schemes are considered to be *single-pass* schedule generation schemes, since they place all jobs in a single loop over jobs (serial) or time (parallel).

At any stage in the construction, the serial scheme will select the unscheduled job

with the highest priority (according to the priority rule) and schedule it as early as possible. It will then repeat this until all jobs are scheduled. The parallel scheme, in contrast, will select the highest priority job (or jobs, if enough power remains available after the scheduling of the first one) that fits in the gap at the current time in the schedule, which is not necessarily the highest priority job available overall.

We extended these schemes in the following ways to apply them to our case:

Composite resource To check whether enough resource (power) is available to process a job at a given rate, we need to check the load of all cables and parking lots along the path from its source to the lot where the EV is located. The residual capacity at a parking lot can be determined by an elementary flow computation from the load at other parking lots, including the contribution of solar panels, and the cable capacities.

Resilience against drops of solar power We need to guarantee the load on the cables does not exceed the capacity, while using as much of the available power as we can. If, for whatever reason, the solar output should fall away, we want to be able to guarantee that we can scale back charging on some EVs to remain within the bounds of the cable capacity. Therefore, we ensure that the EV's that are currently charging, can charge at their minimum rate without using solar power, i.e. solar power is only used to increase the charging rate above the lower bound.

Flexible rate The jobs have a flexible resource consumption profile. This means that we need to decide the consumption rate for each job for the duration of its execution. When a job is selected to be added, it is placed at the earliest possible time in the partial schedule, at the highest possible rate. It may start at any rate between its lower and upper bound. From that point, it will always be scheduled to charge at the highest possible rate, until it completes. This rate is equal to its upper bound, or the maximum rate allowed by the resource constraints, whichever is more restrictive. This means the rate may change over time.

Avoid preemption Applying the schemes in an online setting means that we need to take into account jobs that are already started when we reschedule. Recall that preemption is not allowed. When the construction of a new schedule starts, these jobs are initialized as charging at their lowest possible rate. As soon as they are encountered in the priority order, this rate is scaled up where possible. In this way, preemption is avoided, while respecting the priority order as much as possible.

Note that in the parallel schedule generation scheme, we do not need to generate a complete schedule, but just one that lasts until the next time a schedule will be generated in our strategy. We stop the construction as soon as it advances to a time point beyond that. This is not true for the serial scheme, as a lower priority job may still cause changes early in the schedule.

5.3.2. Priority rules

A schedule generation scheme uses a priority rule to assign a priority to each job. A priority rule essentially is a formula that assigns a numerical value to a job, that can be used to order them. Typically, jobs with a smaller value for the priority rule's formula have a higher priority. Many possible priority rules exist, but we will only list the ones that will be used in the remainder of this work:

FCFS First Come First Serve: r_j.

EDD Earliest Due Date: d_j.

ELSTu Updated Earliest Latest Starting Time: $d_j - E'_j/P_j^+$.

where E'_j is the remaining charging volume, i.e. the original charging volume E_j minus the amount charged up to the time of evaluation.

5.3.3. Destroy-and-repair

The single-pass schedule generation schemes described in Section 5.3.1 generate reasonably good schedules. We want to investigate, however, if we can improve them further by using an iterative heuristic.

Essentially, our approach boils down to a *destroy-and-repair* heuristic. We remove parts of the schedule and replace them by parts that are generated using a different priority rule. We repeat this a limited number of times to obtain an improved schedule.

A global overview of the procedure is given in Algorithm 5.1. From the initial schedule, we remove a fraction s of jobs. First, a fraction $r < s$ is removed randomly with uniform probability. For the remaining fraction $s - r$, jobs are removed with a probability proportional to their weight. The weight is defined as the number of jobs adjacent to the job in the original schedule that have already been removed. In this way, the removal procedure aims to create some large contiguous open areas in the schedule, rather than many small holes. For two jobs to be considered adjacent, two conditions must be met: (1) part of their processing windows must overlap or immediately follow each other, and (2) they must be scheduled on two parking lots that compete for resources (i.e. parking lots that share at least one cable along the path to the root node).

We start the repair phase by reinserting jobs that have a start time S_j before the start of the schedule, i.e. jobs that are already being processed, to avoid preemption. These are initially scheduled at the lowest possible rate. Then, we reinsert (or adjust) all jobs, ordering them using a different priority rule than the one used to generate the initial schedule.

The resulting schedule is compared to the best schedule we have seen so far. If it is better, we continue the procedure with the new schedule. If it is better by at least a minimum improvement $i_{\min}$, we say the procedure was successful. If the procedure was not successful a number of f consecutive times, we stop.

```
   /* Generate initial schedule with a parallel or serial
      schedule generation scheme using the first priority rule
      */
 1 S* ← InitialSchedule(J, PRIO1)
 2 fails ← 0
 3 while fails < f do
 4 |   S ← S*
 5 |   J' ← ∅
   |   /* Randomly remove r · n jobs                          */
 6 |   for iter ∈ {1, ..., r · n} do
 7 |   |   j ← GetRandomJob(S)
 8 |   |   J' ← J' ∪ {j}
 9 |   |   S ← RemoveFromSchedule(S, j)
   |   /* Further removal based on adjacency                 */
10 |   for iter ∈ {r · n, ..., s · n} do
11 |   |   j ← GetWeightedRandomJob(S)
12 |   |   J' ← J' ∪ {j}
13 |   |   S ← RemoveFromSchedule(S, j)
   |   /* Reinsert already started jobs at minimum rate      */
14 |   for j ∈ J' : S_j < t do
15 |   |   S ← ScheduleAtMinRate(S, j)
   |   /* Reinsert (or adjust) jobs in order of the second
   |      priority rule                                      */
16 |   for j ∈ SortBy(J', PRIO2) do
17 |   |   S ← AddToSchedule(S, j)
   |   /* Above some minimum improvement threshold i_min, we count
   |      a success                                          */
18 |   if Score(S) < Score(S*) + i_min then
19 |   |   fails ← fails + 1
20 |   else
21 |   |   fails ← 0
   |   /* Keep the best result                              */
22 |   if Score(S) > Score(S*) then
23 |   |   S* ← S
24 return S*
```

Algorithm 5.1: Global overview of the destroy-and-repair procedure. J is the set of available jobs (of size n), t is the current time.

5.3.4. Scheduling frequency

We consider an online problem. This means that new information regularly becomes available, either when the solar output is updated, or when a new EV arrives at a parking lot. The most naive way to deal with this is to generate a new schedule any time new information is revealed. However, since we use a rather advanced scheduling algorithm, it is reasonable to generate a new schedule less often. Therefore, we trigger the generation of a new schedule on a periodic basis, e.g. every hour. Within an interval, no new schedules are generated, unless a high-priority EV arrives. A high-priority EV is defined as an EV that has a higher priority value (for the first priority rule) than 80% of the EVs in the schedule at that time. If such an EV arrives, an additional schedule is generated at its time of arrival. It does not change the timing of the next periodic schedule generation, however.

5.4. Computational results

We have implemented a discrete-event simulation to evaluate our approach. A brief description of the simulation is presented in Appendix 5.A.

For the test runs reported below, average results over five runs are presented, each with a fixed seed used to generate the instance. Each individual run contains nine days of simulated time, the first two of which are treated as a warm-up period. This means that all reported results (except the runtime) are over days 3-9. The simulator was written in the Python programming language. The processor of the system used to run the tests on was an Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz.

The main performance metric of an approach is the resulting average delay, or the average tardiness, of an EV that is charged at one of the parking lots in the simulation. The other metrics that we will report on are maximum delay, i.e. the largest delay a single EV encounters in the simulation, the fraction of EVs that are delayed as a percentage of the total number of EVs that parked at one of the parking lots, and the number of EVs that suffered a delay of more than fifteen minutes. We present the number of EVs with a large delay rather than a percentage, as we feel that the *number* of EVs that experience a large delay is a relevant metric, even if the percentage is small.

In the remainder of this section, we first discuss the selected approaches and parameter settings in 5.4.1. We continue with an analysis of the effects of considering elements of the grid topology (Section 5.4.2) and flexible charging rates (Section 5.4.3). Then we will show the performance of the developed approaches on different scheduling intervals in Section 5.4.4. We conclude this section with a comparison of the most promising approaches in Section 5.4.5.

5.4.1. Approach selection and parameter settings

Based on their performance in initial test runs, we selected five variants that we will use in our evaluation:

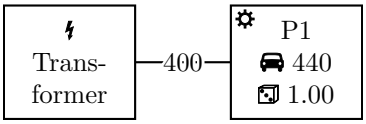


Figure 5.2: Copperplate layout

FCFS is a serial schedule generation scheme, using the *First Come First Serve* priority rule.

S generates schedules using a serial schedule generation scheme, where the planned departure of an e-vehicle is used as the priority (Earliest Due Date/EDD).

P generates schedules using a parallel schedule generation scheme, where the planned departure of an e-vehicle is used as the priority (Earliest Due Date/EDD).

SR generates initial schedules using a serial schedule generation scheme, where the planned departure of an e-vehicle is used as the priority (Earliest Due Date/EDD). Improvements are generated using the destroy-and-repair heuristic described in Section 5.3.3 with the Updated Earliest Latest Starting Time (ELSTu) priority rule.

PR generates initial schedules using a parallel schedule generation scheme, where the planned departure of an e-vehicle is used as the priority (Earliest Due Date/EDD). Improvements are generated using the destroy-and-repair heuristic described in Section 5.3.3 with the Updated Earliest Latest Starting Time (ELSTu) priority rule.

For the latter two, the values of the relevant parameters are as follows:

- Rescheduling fraction $s = 0.5$;
- Random removal fraction $r = 0.05$;
- Improvement threshold $i_{\min} = 100$;
- Unsuccessful iteration tolerance $f = 4$.

Other variants of the approaches have been tested, using different (combinations) of priority rules, parameter settings, and adjacency measures (in case of weighted removal). The presented approaches and settings (except 'FCFS') were chosen as they showed the most consistent good performance in the preliminary tests. Apart from these four main approaches, we also present results from 'FCFS' when relevant.

5.4.2. Network constraints

Where limitations exist in the transport capacity of the underlying electricity network, it is important to take these into account when planning electricity usage. To show the effects of ignoring the grid layout, we have implemented a scenario that does so (see Figure 5.2). This network consists of a single parking lot, with as many parking spots and solar panels as the seven parking lots in our original

	Metric	S	P	SR	PR
Grid	Max. delay (s)	14236.27	13882.98	5033.37	4847.46
	Avg. delay (s)	121.19	106.03	88.14	86.56
	EVs delayed (%)	1.75	1.48	1.47	1.54
	Delay $\geq$ 15 min	65.20	58.00	52.60	55.40
No Grid	Max. delay (s)	36.41	0.00	36.41	10.40
	Avg. delay (s)	0.01	0.00	0.01	0.00
	EVs delayed (%)	0.00	0.00	0.00	0.00
	Delay $\geq$ 15 min	0.00	0.00	0.00	0.00

Table 5.1: Comparison of results taking the network layout into account and those ignoring the grid

grid combined. The results of running our four main approaches on both grids are shown in Table 5.1.

We observe that delays almost disappear when we take away the grid layout. This indicates that the simplification of ignoring the grid layout gives a significant and unrealistic advantage in the scheduling model. This difference justifies the additional modeling effort required to account for the topology of the network.

5.4.3. Flexible charging rates

The assumption that using the flexibility in rates that charging of EV batteries may allow would be beneficial, is crucial for our model. To test this assumption, we compared simulations with the flexible rates as we have described them so far, to simulations with a fixed rate of 9 kW for all EVs. The value of 9 kW was chosen, as this is the average upper bound in the flexible scenario, meaning that the average potential for energy consumption is equal across both scenarios. Additionally, we excluded the solar panels from this simulation. EVs cannot scale down in the fixed rate scenario and their charging is not allowed to be preempted. Because of this, using the power from solar panels in the schedule means that the cable capacity may be violated if the amount of power they provide changes. To make the comparison as fair as possible, therefore, the solar panels are turned off. Note the absence of solar power causes markedly different results compared to e.g. Table 5.1, as less power is available for charging overall. The results of these runs are presented in Table 5.2.

Here, we present the results of the 'FCFS' alongside our four main approaches, as it performs unexpectedly well in this scenario. It seems that planning according to the 'first come first serve' principle is fairly effective in a strained network where all power flows from a single source.

We observe that the differences are not large, but the average delay is typically a few minutes shorter in the scenario with flexible charging rates. The same is true for the amount of severe delays and the percentage of EVs that are delayed.

	Metric	FCFS	S	P	SR	PR
Fixed	Max. delay (s)	68148.35	54599.73	54561.94	53366.20	56653.89
	Avg. delay (s)	15142.88	14899.12	14894.50	15055.51	15103.98
	EVs delayed (%)	61.30	85.83	85.89	85.83	85.01
	Delay $\geq$ 15 min	2106.40	2950.80	2954.40	2958.80	2911.20
Flexible	Max. delay (s)	66068.74	62563.74	59144.77	61743.67	60528.36
	Avg. delay (s)	14131.81	14600.97	14486.47	14327.40	14418.19
	EVs delayed (%)	59.02	82.34	83.25	81.61	83.86
	Delay $\geq$ 15 min	2055.60	2837.40	2870.60	2815.60	2886.60

Table 5.2: Comparison of results for flexible and fixed rate charging

Keep in mind that we excluded the solar panels from this comparison. Their output can be incorporated safely in the scenario with flexible rates, whereas it cannot be incorporated when using fixed charging rates without allowing preemption. Therefore, one could say that flexible rates are very beneficial when we want to use solar power. These results together show that the use of flexible charging rates allows for better outcomes.

5.4.4. Scheduling frequency

In an online setting, regular updates of the schedule are performed. Each time new information is revealed, the generation of a new schedule could be triggered. The results presented in the previous sections use this strategy, because it provides the most clear environment for the comparison of elements of the modeling. In practice, however, it is undesirable and unnecessary to generate new schedules that often, in particular for the more advanced scheduling methods. Generating schedules less frequently is also more realistic in terms of computation time, as new EVs arrive with high frequency during peak hours.

The more often a new schedule is generated, the better the results are expected to be. If the difference in quality is sufficiently small, however, it is reasonable to update schedules with a lower frequency. We have tested three scheduling frequencies:

1. Generate a new schedule whenever new information is revealed (i.e., when the solar forecast is updated or a new EV arrives);
2. Generate a new schedule once every 15 minutes;
3. Generate a new schedule once every hour.

In the latter two cases, we keep track of the priority of newly arriving EVs, according to the (first) priority rule used for scheduling. If an EV arrives with a priority that is higher than 80% of the EVs that are currently scheduled, we will trigger an additional schedule to be generated.

The results are presented in Table 5.3. Most approaches show a slight deterioration

	Metric	S	P	SR	PR
New information	Max. delay (s)	14236.27	13882.98	5033.37	4847.46
	Avg. delay (s)	121.19	106.03	88.14	86.56
	EVs delayed (%)	1.75	1.48	1.47	1.54
	Delay $\geq$ 15 min	65.20	58.00	52.60	55.40
	Runtime (s)	20366.03	12743.83	70506.54	40210.12
15 min.	Max. delay (s)	14706.39	13167.38	5293.81	5237.77
	Avg. delay (s)	123.94	115.11	106.39	95.23
	EVs delayed (%)	1.75	1.63	1.73	1.60
	Delay $\geq$ 15 min	65.20	58.00	52.60	55.40
	Runtime (s)	8688.63	7143.22	27071.67	15324.67
1 hour	Max. delay (s)	14889.79	14027.18	8399.59	5135.09
	Avg. delay (s)	127.69	103.22	136.22	96.95
	EVs delayed (%)	1.77	1.47	1.94	1.61
	Delay $\geq$ 15 min	65.20	58.00	52.60	55.40
	Runtime (s)	7499.83	7423.24	19654.80	14446.14

Table 5.3: Comparison of results for a number of scheduling intervals.

in the quality of the results when the scheduling frequency decreases. Only the results for the parallel scheduling scheme ('P') are somewhat inconsistent with this observation. The performance of the 'SR' approach deteriorates most of all.

We observe that the total runtime of the simulation is significantly reduced for all approaches. Note that the reported runtime includes every aspect of the simulation (not just the scheduling itself) for a total simulation time of nine days. The runtime of all approaches is such that it is realistic to implement them with a schedule generation interval of 15 minutes. In terms of the quality of the schedules, we observe that the best performing (advanced) approach ('PR') still outperforms the single-pass approaches on the main metric (average delay), even with a large scheduling interval.

5.4.5. Overall performance

From the results in the previous sections, we observe that the parallel implementation of an approach seems to dominate the serial implementation of that same approach. Based on the results of our preliminary tests, however, we must say that this is not a general result. This is consistent with earlier findings by e.g. Kolisch (1996).

In this section, we will give a final overview of the results. We present the best performing single-pass approach and the best performing approach using our destroy-and-repair heuristic. We compare them to the 'FCFS' approach, which we consider to be the most straight-forward application of scheduling using our adapted schedule generation schemes. The best performing approaches are the parallel schedule

	FCFS	P-EDD	PR-15min
Max. delay (s)	31287.96	13882.98	5237.77
Avg. delay (s)	1114.21	106.03	95.23
EVs delayed (%)	11.72	1.48	1.60
Delay $\geq$ 15 min	470.20	58.00	55.40
Runtime (s)	16168.70	12743.83	15324.67

Table 5.4: Results for the most promising approaches

generation scheme using the Earliest Due Date (EDD) priority rule, 'P-EDD', and the parallel 'improved' approach using the EDD and ELSTu (Updated Earliest Latest Starting Time) priority rules, 'PR-15min'. For the latter approach, we use a scheduling interval of 15 minutes, while the other two will continuously generate new schedules, any time new information is revealed.

The parallel 'improved' approach outperforms the other approaches on almost all metrics, while it is comparable in computational load. This seems to be the most promising approach for our case study and similar scenarios.

Note that all approaches discussed above protect the network capacity and apply scheduling techniques for charging decisions. We conclude with a note on the effectiveness of scheduling in general. Therefore, we want to make a comparison with a scenario where no control is exercised over the charging at all. We ran simulations on the same network, under the same conditions, where all EVs have a fixed charging rate of 9 kW. Rather than scheduling their demand, any EV that arrives at a parking lot immediately starts charging, until its battery is fully charged. In this scenario, we consider a 10% overload of the cable capacity (i.e. a load of 200-220 kW) to be manageable. We consider anything beyond 220 kW to be truly problematic. If we look at the two main cables connecting the two groups of parking lots to the transformer, we find that the cable on the left (connecting to P1, P2 and P3) suffers from a manageable overload 4.4% of the time, and from a problematic overload 53.7% of the time. The cable of the right (connecting to P4, P5, P6 and P7) has a manageable overload 2.6% of the time, and a problematic overload 65.8% of the time. These numbers indicate that it is absolutely necessary to manage the charging of EVs on these parking lots, and show the usefulness and effectiveness of a scheduling approach like ours.

5.5. Conclusions and future work

We have formulated an online scheduling problem dealing with the charging of EVs. For this problem, we developed a scheduling approach that extends the idea of schedule generation schemes to an online setting with flexible charging jobs. In our case study, the application of schedules generated using this approach, allows for the serving of many EVs with minimal delay, while respecting the constraints imposed by the network. If no control over the charging

was exercised at all, cables in the network would be overloaded around 60-70% of the time. Overload is completely avoided if our schedules are followed, at the cost of an average delay of just over 1.5 minutes per EV in a high-occupancy scenario. The quality of the generated schedules is such that frequent updates are not necessary. Finally, the computational load of the developed approach is reasonable for practical purposes, making it a promising candidate for implementation in real-world cases.

In addition to this main result, our case study shows that it is essential to take the network topology into account when deciding charging profiles. Furthermore, we have shown the benefits of considering a range of permissible charging rates for EVs, rather than a single fixed value.

Many avenues for future research exist. Broadly, we see two main directions for future research.

First, there are many opportunities for further development of the scheduling approach. We have proposed a novel approach for generating schedules. Other variants of this approach might be interesting. Examples include the use of more than two priority rules in the improvement process and the application of a more directed destroy-and-repair heuristic that aims to improve the schedule at the points that cause the most delay.

Second, certain extensions of the model may be interesting to investigate. For example, a more complex model could include the ability of EVs to discharge as well (V2G), and contribute to the charging of other EVs with a higher priority. A more detailed modeling of the uncertain power output of the solar panels can be considered as well. Using a more detailed model of the power network (e.g. an AC model) would be an interesting, but computationally heavy, extension. Other objectives, such as cost minimization through consideration of electricity prices, can also be investigated.

Acknowledgments

We would like to thank Philip de Bruin for providing the numbers for the simulation runs without any control or scheduling strategies, that we used in Section 5.4.5.

5.A. Simulation model

We evaluated the developed scheduling approaches using a discrete-event simulation. We will briefly explain the simulation model using the event graph displayed in Figure 5.3. Each box represents a type of event, where arcs between events indicate that an event may schedule another. A dashed arc indicates that the subsequent event happens instantaneously if it is triggered. The arcs labeled with '(a)' may or may not be present, depending on the scheduling approach used (see Section 5.3). The simulation is run by repeatedly extracting the next event from the event queue and executing its associated actions, until the *end simulation* event is encountered.

We simulate the arrival, charging and departure of electric vehicles. The charging is done according to the schedule we generate during the simulation, using one of the approaches described in Section 5.3. The shaded boxes in Figure 5.3 indicate events that are contained in the schedule. A schedule is, in essence, a series of provisional events of types *EV changes charging rate* and *EV stops charging*. The *inspect schedule* event draws these events from the schedule and enters them in the event queue. A brief description of each event is provided below:

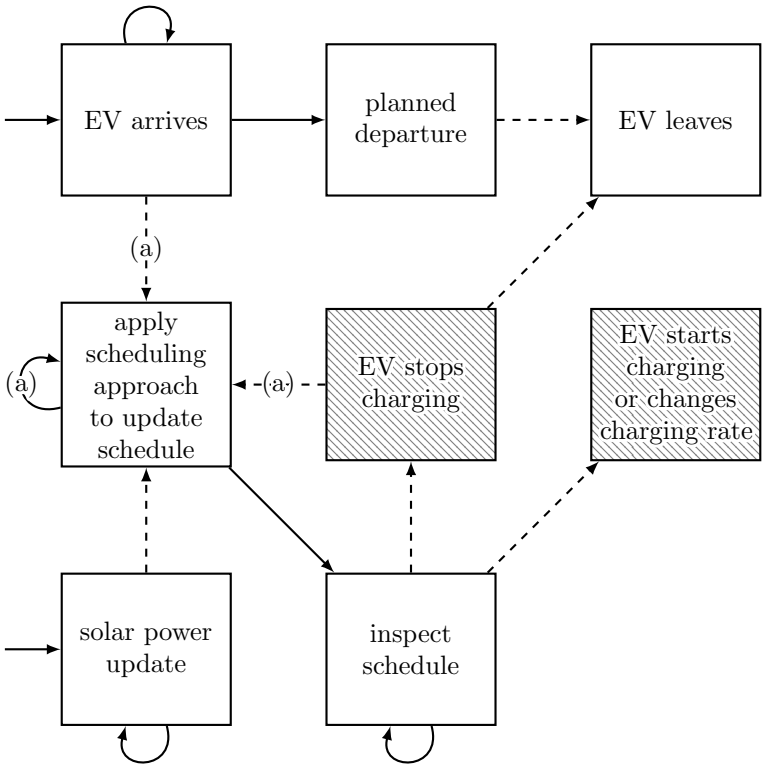


Figure 5.3: Event graph showing the relations between events in our simulation model

EV arrives happens when a new EV enters the simulation. This event handles the selection of a parking lot (or removes the EV from the simulation if all three parking lots from its preference are full). It schedules a *planned departure* event for the EV at its due date d_j and may schedule an immediate *update schedule* event, depending on the scheduling approach used. Finally, it will schedule a *EV arrives* event for the next EV to arrive.

planned departure happens when an EV is supposed to leave, at the end of its planned connection time. If the EV is fully charged, it will schedule an immediate *EV leaves* event.

EV leaves happens when an EV actually leaves. This means that it leaves its parking lot and is removed from the simulation.

EV starts charging or changes charging rate happens when an EV starts charging for the first time, or changes its charging rate later on. It updates all relevant values and rates, but schedules no additional events. This event is dictated by the schedule.

EV stops charging happens when an EV stops charging, i.e. it has received its requested charging volume E_j. It updates all relevant values and rates. If the due date d_j of the EV has already passed, it will schedule an immediate *car leaves* event. Depending on the scheduling approach used, it may also trigger an immediate *update schedule* event.

solar power update is a regularly scheduled event, that happens on a given interval: every hour of simulated time. This event retrieves the (expected) power produced by the solar panels installed for the upcoming interval, and updates the available power at each parking lot accordingly. It will trigger an immediate *update schedule* event. Finally, it will schedule another *regular forecast update* event at the end of the current forecast interval.

apply scheduling approach to update schedule happens when the schedule needs to be updated. A new schedule is generated using the relevant scheduling approach. It schedules an *inspect schedule* event at the time of occurrence of the first action in the generated schedule. Additionally, if schedules are updated on an interval different from the *regular forecast update*, it schedules another *update schedule* event at the end of the current scheduling interval.

inspect schedule will retrieve the next event from the schedule (either a *EV changes charging rate* event or a *EV stops charging* event) and, if it happens at the current time, schedules the corresponding event. It will then schedule an *inspect schedule* event at the (planned) time of occurrence of the following event in the schedule.

At the start of the simulation, three events are scheduled: (1) 'regular forecast update' at time 0; (2) 'car arrives' at the first generated arrival time; (3) 'end simulation' at the end of the period of time we intend to simulate. These correspond to the three arcs that do not originate at an event box in Figure 5.3.

5.B. Distributions

5.B.1. Solar power

The power that a solar panel produces strongly depends on the time of day. In Table 5.5, we list the average production of solar panels as a fraction of their peak power. These will be used to determine the mean of the distribution (with a standard deviation of 15%) from which the actual values are drawn during the simulation. The values are based on a forecast of the production of solar panels during the summer months in the Netherlands in 2025 (ENTSO-E 2018).

Hour	Fraction	Hour	Fraction	Hour	Fraction
0	0.000000000	8	0.311153080	16	0.169520680
1	0.000000000	9	0.382618250	17	0.083306720
2	0.000000000	10	0.426692930	18	0.026936987
3	0.001548952	11	0.446001000	19	0.005260382
4	0.017126799	12	0.440104200	20	0.000000000
5	0.055567520	13	0.403637380	21	0.000000000
6	0.132034210	14	0.342447800	22	0.000000000
7	0.222931250	15	0.261458840	23	0.000000000

Table 5.5: Average solar panel revenues as a fraction of peak power

5.B.2. Arrival of EVs

EVs arrive in the simulation according to a Poisson process. The (average) arrival rates for every hour of the day as a fraction of the total daily arrivals are listed in Table 5.6.

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

Table 5.6: Average fraction of arrivals at each hour

5.B.3. Charging rates

Each EV has a minimum and maximum charging rate (kW), drawn with uniform probability from the combinations listed in Table 5.7.

Min.	Max.	Probability	Min.	Max.	Probability
3	6	0.0625	5	8	0.0625
3	7	0.0625	5	9	0.0625
3	8	0.0625	5	10	0.0625
3	9	0.0625	5	11	0.0625
4	7	0.0625	6	9	0.0625
4	8	0.0625	6	10	0.0625
4	9	0.0625	6	11	0.0625
4	10	0.0625	6	12	0.0625

Table 5.7: Distribution of charging rates

5.B.4. Charging volumes

Each EV has a demand for a given amount of energy to be added to its battery over the duration of its connection. These are distributed in the range of 0 to 102 kWh according to the fractions given in Table 5.8. These fractions represent the probability of the charging volume to be drawn from that particular range. Within each group, values are distributed uniformly.

kWh	Weight	kWh	Weight	kWh	Weight
0-1	0.029938112	34-35	0.009603366	68-69	0.001249962
1-2	0.026188226	35-36	0.008444864	69-70	0.001128014
2-3	0.035212341	36-37	0.008292430	70-71	0.000640224
3-4	0.032681930	37-38	0.008079022	71-72	0.000853633
4-5	0.043443797	38-39	0.006585165	72-73	0.000518277
5-6	0.050333831	39-40	0.007042468	73-74	0.000731685
6-7	0.071186854	40-41	0.006920521	74-75	0.000518277
7-8	0.055973903	41-42	0.006676626	75-76	0.000762172
8-9	0.048809488	42-43	0.006219323	76-77	0.000579251
9-10	0.032163654	43-44	0.005762019	77-78	0.000396329
10-11	0.021188378	44-45	0.005822993	78-79	0.000396329
11-12	0.023383433	45-46	0.005487638	79-80	0.000182921
12-13	0.020670102	46-47	0.005640072	80-81	0.000182921
13-14	0.019725008	47-48	0.005883967	81-82	0.000091500
14-15	0.018688455	48-49	0.005121795	82-83	0.000091500
15-16	0.018261638	49-50	0.004115728	83-84	0.000091500
16-17	0.018048230	50-51	0.004115728	84-85	0.000274382
17-18	0.017347032	51-52	0.004268163	85-86	0.000243895
18-19	0.017286058	52-53	0.003871833	86-87	0.000152434
19-20	0.018718941	53-54	0.004786439	87-88	0.000182921
20-21	0.018078717	54-55	0.003810859	88-89	0.000182921
21-22	0.018901863	55-56	0.003262096	89-90	0.000182921
22-23	0.017286058	56-57	0.003384043	90-91	0.000030500
23-24	0.017316545	57-58	0.003445017	91-92	0.000000000
24-25	0.017103137	58-59	0.003201122	92-93	0.000000000
25-26	0.016615347	59-60	0.002804793	93-94	0.000030500
26-27	0.014511753	60-61	0.002987714	94-95	0.000000000
27-28	0.015487333	61-62	0.002957227	95-96	0.000000000
28-29	0.014938569	62-63	0.002317003	96-97	0.000000000
29-30	0.013200817	63-64	0.002073108	97-98	0.000000000
30-31	0.011859395	64-65	0.002256029	98-99	0.000000000
31-32	0.013322765	65-66	0.001981647	99-100	0.000000000
32-33	0.010853328	66-67	0.001585318	100-101	0.000000000
33-34	0.011127710	67-68	0.001615804	101-202	0.000030500

Table 5.8: Distribution of charging volumes of EVs

5.B.5. Connection times

Each EV has an intended time of departure, effectively a due date for the charging to be complete. This is determined by adding a connection time (in hours, distributed as in Table 5.9) to the arrival time. These fractions represent the probability of the connection time to be drawn from that particular range. Within each group, values are distributed uniformly.

Hours	Weight	Hours	Weight	Hours	Weight
0-1	0.075089174	24-25	0.005518124	48-49	0.000945093
1-2	0.089052163	25-26	0.004237676	49-50	0.000548764
2-3	0.084570592	26-27	0.002926740	50-51	0.000518277
3-4	0.069967379	27-28	0.001859699	51-52	0.000274382
4-5	0.053565440	28-29	0.001524344	52-53	0.000121948
5-6	0.036309869	29-30	0.001128014	53-54	0.000152434
6-7	0.025608975	30-31	0.000579251	54-55	0.000426816
7-8	0.028200360	31-32	0.000914606	55-56	0.000243895
8-9	0.041492637	32-33	0.001280449	56-57	0.000213408
9-10	0.037559830	33-34	0.001249962	57-58	0.000457303
10-11	0.037072040	34-35	0.001341423	58-59	0.000396329
11-12	0.040699979	35-36	0.001920673	59-60	0.000274382
12-13	0.046431511	36-37	0.002256029	60-61	0.000670711
13-14	0.055242218	37-38	0.003231609	61-62	0.000945093
14-15	0.049205817	38-39	0.003536478	62-63	0.000792659
15-16	0.034511143	39-40	0.002804793	63-64	0.000884119
16-17	0.026828450	40-41	0.002408463	64-65	0.000853633
17-18	0.022590775	41-42	0.002865766	65-66	0.000640224
18-19	0.019237218	42-43	0.001890186	66-67	0.000396329
19-20	0.016737295	43-44	0.002286516	67-68	0.000609738
20-21	0.013261791	44-45	0.001890186	68-69	0.000548764
21-22	0.010426511	45-46	0.001310936	69-70	0.000457303
22-23	0.008536325	46-47	0.001432883	70-71	0.007987561
23-24	0.006707113	47-48	0.001341423		

Table 5.9: Distribution of connection times

References

Arif, S. M., T. T. Lie, B. C. Seet, S. Ayyadi, and K. Jensen (2021). "Review of electric vehicle technologies, charging methods, standards and optimization techniques". *Electronics* 10.16. DOI: 10.3390/electronics10161910.

Brinkel, N. B. G., W. L. Schram, T. A. AlSkaif, I. Lampropoulos, and W. G. J. H. M. van Sark (2020). "Should we reinforce the grid? Cost and emission optimization of electric vehicle charging under different transformer limits". *Applied Energy* 276. DOI: 10.1016/j.apenergy.2020.115285.

Brinkel, N., T. AlSkaif, and W. van Sark (2020). "The impact of transitioning to shared electric vehicles on grid congestion and management". *Proceeding of the 2020 International Conference on Smart Energy Systems and Technologies (SEST)*, pp. 1–6. DOI: 10.1109/SEST48500.2020.9203241.

Brinkel, N., T. AlSkaif, and W. van Sark (2022). "Grid congestion mitigation in the era of shared electric vehicles". *Journal of Energy Storage* 48. DOI: 10.1016/j.est.2021.103806.

Brouwer, R. J. J., J. M. van den Akker, and J. A. Hoogeveen (2024). "A hybrid optimization framework for the general continuous energy-constrained scheduling problem". Preprint at: https://arxiv.org/abs/2403.03039. DOI: 10.48550/arXiv.2403.03039.

Chen, Z., E. Demeulemeester, S. Bai, and Y. Guo (2018). "Efficient priority rules for the stochastic resource-constrained project scheduling problem". *European Journal of Operational Research* 270.3, pp. 957–967. DOI: 10.1016/j.ejor.2018.04.025.

Đumić, M. and D. Jakobović (2021). "Ensembles of priority rules for resource constrained project scheduling problem". *Applied Soft Computing* 110. DOI: 10.1016/j.asoc.2021.107606.

ENTSO-E (2018). *Mid-term adequacy forecast 2018*. URL: https://www.entsoe.eu/outlooks/midterm/.

Hartmann, S. and D. Briskorn (2010). "A survey of variants and extensions of the resource-constrained project scheduling problem". *European Journal of Operational Research* 207 (1), pp. 1–14. DOI: 10.1016/j.ejor.2009.11.005.

Hartmann, S. and D. Briskorn (2022). "An updated survey of variants and extensions of the resource-constrained project scheduling problem". *European Journal of Operational Research* 297 (1), pp. 1–14. DOI: 10.1016/j.ejor.2021.05.004.

Hussain, M. T., N. B. Sulaiman, M. S. Hussain, and M. Jabir (2021). "Optimal management strategies to solve issues of grid having Electric Vehicles (EV): a review". *Journal of Energy Storage* 33, pp. 102–114. DOI: 10.1016/j.est.2020.102114.

Kolisch, R. (1996). "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation". *European Journal of Operational Research* 90.2, pp. 320–333. DOI: 10.1016/0377-2217(95)00357-6.

Liu, M., P. Mcnamara, R. Shorten, and S. McLoone (2015). "Residential electrical vehicle charging strategies: the good, the bad and the ugly". *Journal of Modern Power Systems and Clean Energy* 3.2, pp. 190–202. DOI: 10.1007/s40565-015-0122-2.

Lova, A., P. Tormos, and F. Barber (2006). "Multi-mode resource constrained project scheduling: scheduling schemes, priority rules and mode selection rules". *Inteligencia Artificial* 10.30, pp. 69–86. DOI: 10.4114/ia.v10i30.947.

Naber, A. (2017). "Resource-constrained project scheduling with flexible resource profiles in continuous time". *Computers and Operations Research* 84, pp. 33–45. DOI: 10.1016/j.cor.2017.02.018.

Al-Ogaili, A. S., T. J. Tengku Hashim, N. A. Rahmat, A. K. Ramasamy, M. B. Marsadek, M. Faisal, and M. A. Hannan (2019). "Review on scheduling, clustering, and forecasting strategies for controlling electric vehicle charging: challenges and recommendations". *IEEE Access* 7, pp. 128353–128371. DOI: 10.1109/ACCESS.2019.2939595.

Pasha, J., B. Li, Z. Elmi, A. M. Fathollahi-Fard, Y.-y. Lau, A. Roshani, T. Kawasaki, and M. A. Dulebenets (2024). "Electric vehicle scheduling: state of the art, critical challenges, and future research opportunities". *Journal of Industrial Information Integration* 38. DOI: 10.1016/j.jii.2024.100561.

van den Akker, J. M. and J. I. Posthoorn (2021). "Simulation assignment Optimization for Sustainability (INFOMOFS): Smart charging in Polderwijk". URL: https://osiris-student.uu.nl/onderwijscatalogus/extern/cursus?cursuscode=INFOMOFS&taal=en&collegejaar=2020.

van Huffelen, E. (2023). "Scheduling of flexible EV charging under grid constraints and uncertainty". Master's thesis. Utrecht University. DOI: 20.500.12932/45597.

van Huffelen, E., R. J. J. Brouwer, and J. M. van den Akker (2024). "Grid-constrained online scheduling of flexible electric vehicle charging". Preprint at: http://arxiv.org/abs/2403.03109. DOI: 10.48550/arXiv.2403.03109.

Wang, Q., X. Liu, J. Du, and F. Kong (2016). "Smart charging for electric vehicles: a survey from the algorithmic perspective". *IEEE Communications Surveys & Tutorials* 18.2, pp. 1500–1517. DOI: 10.1109/COMST.2016.2518628.

Part IV

Concluding Remarks

Contents

6

Conclusion

Energy consumption and production patterns are changing in a major way. To meet the challenges that come with these developments, it is necessary to rethink the way the demand for electrical energy is managed.

In this thesis, we considered the usefulness of scheduling approaches for the alignment of demand and supply in electricity networks. We first discussed a class of scheduling problems, the (General) Continuous Energy-Constrained Scheduling Problem, that is a good model for scheduling flexible demand in electricity networks. We proposed a general hybrid solution framework for this class of problem, and developed tailored approaches for two variants contained in this class of problems. In the second part of the thesis, we developed scheduling approaches for two real-world problems in the context of energy networks and investigated the performance of these approaches.

6.1. Summary and contributions

In Chapter 2, we studied the Continuous Energy-Constrained Scheduling Problem (CECSP) with the objective to minimize the weighted completion time. The problem consists of the construction of a schedule for a set of jobs that has to be processed on a continuous, shared resource. In a schedule, each job is assigned a start time, completion time and resource consumption profile. A job cannot start before its release time, and must be completed before its deadline. Once it has started, a job cannot be preempted, but must satisfy its full resource requirement in one go. The consumption rate can fluctuate during processing, as long as it stays within the range of the lower and upper bound on the consumption rate for that job. We proposed a hybrid local search approach, where we use simulated annealing to explore the search space of possible event orders and a linear program to find the optimal schedule for a given event order. An event order lists all events (start and completion times) in order of occurrence. We compared

the performance of the hybrid local search approach with a mixed-integer linear programming (MILP) formulation. We found that the hybrid approach matches the MILP formulation in solution quality for small instances. For larger instances, it outperformed the MILP approach, as it was able to find feasible solutions in reasonable time, where the MILP approach was not.

In Chapter 3, we presented a generalization of the CECSP from the previous chapter, the General Continuous Energy-Constrained Scheduling Problem. We extended the previously developed approach to a solution framework for this class of problems. A decomposition of the problem in two parts, that follows from the event-based model, lies at the core of this framework: 1) determining the order of events and 2) finding the event times and resource consumption profiles. We extended the event-based model to include fixed-time events, which may be used to model piece-wise linear objective functions, for example, and showed how this can be exploited to speed up the evaluation of candidate solutions. We argued the broad applicability of the framework, and have given an impression of the range of problems that it can be applied to. We have illustrated the effectiveness of the framework by applying it to the CECSP with step-wise cost functions. We exploited that the cost of a solution can be computed on the basis of the order of events alone, and used bounds to assess the feasibility of a given event order more efficiently. A comparison with a MILP formulation showed the effectiveness of our approach for finding reasonable solutions in limited time.

From the general class of problems, we then moved on to specific examples of scheduling problems in electricity networks. The first of these examples, discussed in Chapter 4, concerns the operational management of islanded microgrids. These are small, isolated electricity networks with a limited number of components that each produce, consume and/or store electrical energy. These assets allow the microgrid to operate autonomously, as long as it can be managed such that supply and demand are always balanced. We developed two optimization models for planning the operation of diesel generators and battery storage units. We included forecasts of the uncertain load and solar panel output. The first approach is a deterministic mixed-integer linear programming model that generates a schedule using the predicted values for the load and solar panel output. The uncertainty is managed by keeping a percentage of the battery capacity available for dealing with deviations from the predicted scenario. The second approach used a multi-stage stochastic optimization model, that explicitly models the uncertainty by constructing a scenario tree based on the forecasted distributions. Re-planning strategies were developed to minimize the amount of times new schedules have to be generated. These models were applied to a case study in the Netherlands. The re-planning strategies were shown to be very effective. The multi-stage stochastic model suffered from the (relatively) large uncertainty around the forecasts for such a small system. While the pragmatic approach implemented in the deterministic model performed well in practice.

Finally, in Chapter 5, we studied the scheduling of demand at a network of parking lots for electric vehicles (EVs). These parking lots are connected by cables in the

local energy network, and share a limited connection to the main electrical grid. Some parking lots are covered by an array of solar panels that provides additional power. EVs arrive at the parking lots according to a known distribution. When an EV arrives, its desired departure time, the amount of electrical energy it needs to charge its battery, and the range of rates it can be charged at are revealed. The departure of an EV is delayed if it has not finished charging before its desired departure time. The charging of EVs at the parking lots must be managed such that the average delay across all EVs is minimized. We presented a novel approach, based on an online variant of well-known schedules generation schemes. We evaluated a serial and a parallel implementation of a single-pass variant of this approach, and a serial and a parallel implementation of a variant that uses a destroy-and-repair heuristic to improve the schedule further. In all cases, we selected the best performing variant for further comparison. We compared the outcomes using the developed scheduling approaches to the scenario where no control over the charging is exercised at all, showing the benefits of using a scheduling approach to efficiently plan the charging jobs. Furthermore, we argued the importance of considering aspects of the grid layout in electricity networks and showed the benefits of using flexible charging rates.

Finally, we will briefly summarize the key contributions of this thesis.

1. We defined a class of scheduling problems, the General Continuous Energy-Constrained Scheduling Problem (GCECSP), that models the planning of tasks, that use a resource of which the availability is constrained, with flexible resource consumption profiles. This is particularly useful for modeling the planning of flexible electricity demand.

2. We introduced a hybrid solution framework for this class of problems and described how it can be applied and adapted to solve problems related to the GCECSP.

3. We presented and evaluated an implementation of the framework for the GCECSP with the objective of minimizing weighted completion time.

4. We presented and evaluated a more sophisticated approach for the GCECSP with step-wise constant cost functions.

5. We developed a deterministic and a multi-stage stochastic optimization approach for the operational management of microgrids under uncertainty. We defined re-planning strategies to effectively apply these approaches using a rolling horizon approach.

6. We evaluated the developed approaches for the operational management of microgrid using a real-world case study. This provides insights in the challenges of dealing with uncertainty explicitly in a small system.

7. We developed a novel scheduling approach for the planning of electric vehicle charging. We adapted ideas from classic schedule generation schemes to an online setting with flexible charging rates. We described a novel destroy-and-repair heuristic to further improve generated schedules.

8. We evaluated a number of variants of the scheduling approach for EV charging using a discrete-event simulation of a case study based on real-world data, showing the effectiveness of scheduling in general and of the approach in particular.

9. We investigated the effects of ignoring the network topology when making charging decisions and the benefits of working with flexible charging rates.

6.2. Further research

We are convinced that scheduling approaches will contribute to solutions for the challenges originating from the ongoing energy transition. We hope that we have made a convincing case across all chapters of this thesis for the application of scheduling algorithms to problems concerning the allocation of a limited power supply and the coordination of supply and demand of electrical energy.

There are many interesting variants of the General Continuous Energy-Constrained Scheduling Problem (discussed in Chapter 2 and 3) that remain to be explored. Our hybrid solution framework can be applied to many more variants. This includes, but is not limited to, variants using different objectives (particularly piecewise linear objectives). Other techniques can be developed for the evaluation of candidate solutions for specific variants to improve the performance of the approach. It would be interesting to evaluate the ease of tuning the framework to each variant, to further assess the general applicability of the approach, and to compare the performance with other solution methods. We strongly encourage the application of our approach to new variants and its use for comparisons to alternative approaches. The software is freely available (Brouwer 2024).

It is of particular interest to explore variants with an energy-related application in mind. Examples along these lines are variants with a variable resource availability, a mix of fixed-rate and flexible jobs, or an objective of minimizing electricity cost based on market prices.

It is not straightforward to apply the approach presented in the first part of this thesis to the problems studied in the second part. However, it would be interesting to try to apply it in similar settings. The main challenge is dealing with (controllable) production, varying availability of solar energy and properly considering the grid layout, as these aspects lead to a very complicated character of the resource. Moreover, it would require the modification of the framework to work in an online setting. The most difficult aspect would be the insertion of newly revealed jobs in an existing schedule, to construct a new initial solution.

An islanded microgrid with shiftable loads provides an interesting case for the inclusion of demand response. An approach that integrates an application of the framework we developed for scheduling demand with a planning for generators and battery storage could be developed.

Finally, the application of scheduling techniques on electric vehicle charging locations can be further developed by extending the model, for example with the inclusion of vehicle-to-grid capabilities. It is worthwhile to investigate if the novel schedule generation approach and the destroy-and-repair heuristic we presented can be extended with new aspects. For example, the heuristic could be adapted to use multiple priority rules during the improvement process, or to specifically target points in the schedule that cause the most delay. A further extension of the approach may include a forecast of the number of vehicles that will arrive during the planning horizon in the schedule generation process.

If this type of approach is to be implemented in practice, it is important to properly consider the legal aspects and address privacy concerns. The success of such an application depends on the cooperation of all participants, and is therefore strongly determined by the clarity of up-front agreements. With the implementation of such an approach, the structure of decision-making changes, and it must be made explicit who has control over which decision. A useful concept in this context is that of aggregators, where a group of consumers cooperates to coordinate their electricity demand. Control decisions can be made within the group based on information provided by the individual users, while presenting only an aggregated profile to the outside world.

Many opportunities for the application of scheduling approaches exist in the context of energy networks. Hopefully, this thesis has brought the exploitation of these opportunities one step closer.

Part V

Appendices

Contents

References

Ahmad Khan, A., M. Naeem, M. Iqbal, S. Qaisar, and A. Anpalagan (2016). "A compendium of optimization objectives, constraints, tools and algorithms for energy management in microgrids". *Renewable and Sustainable Energy Reviews* 58, pp. 1664–1683. DOI: 10.1016/j.rser.2015.12.259.

Arif, S. M., T. T. Lie, B. C. Seet, S. Ayyadi, and K. Jensen (2021). "Review of electric vehicle technologies, charging methods, standards and optimization techniques". *Electronics* 10.16. DOI: 10.3390/electronics10161910.

Baptiste, P., C. L. Pape, and W. Nuijten (1999). "Satisfiability tests and time-bound adjustments for cumulative scheduling problems". *Annals of Operations Research* 92, pp. 305–333. DOI: 10.1023/A:1018995000688.

Brandenburg, K., S. Brummelkamp, H. S. Chan, L. Geurts, M. J. Linders, G. Muller, and R. Segers (Oct. 2023). *Hernieuwbare energie in Nederland 2022*. Centraal Bureau voor de Statistiek (CBS). URL: https://www.cbs.nl/nl-nl/longread/rapportages/2023/hernieuwbare-energie-in-nederland-2022.

Brinkel, N. B. G., W. L. Schram, T. A. AlSkaif, I. Lampropoulos, and W. G. J. H. M. van Sark (2020). "Should we reinforce the grid? Cost and emission optimization of electric vehicle charging under different transformer limits". *Applied Energy* 276. DOI: 10.1016/j.apenergy.2020.115285.

Brinkel, N., T. AlSkaif, and W. van Sark (2020). "The impact of transitioning to shared electric vehicles on grid congestion and management". *Proceeding of the 2020 International Conference on Smart Energy Systems and Technologies (SEST)*, pp. 1–6. DOI: 10.1109/SEST48500.2020.9203241.

Brinkel, N., T. AlSkaif, and W. van Sark (2022). "Grid congestion mitigation in the era of shared electric vehicles". *Journal of Energy Storage* 48. DOI: 10.1016/j.est.2021.103806.

Brouwer, R. J. J. (2017). "Forecast-based optimal operation of islanded microgrids". Master's thesis. Utrecht University. DOI: 20.500.12932/27907.

Brouwer, R. J. J. (2022a). *CECSP data*. Zenodo. DOI: 10.5281/zenodo.10051616. URL: https://github.com/RoelBrouwer/2022-cecsp-data/tree/jos.

Brouwer, R. J. J. (July 9, 2022b). *GCECSP: models and algorithms*. Version 0.1.0. DOI: 10.5281/zenodo.10037572. URL: https://github.com/RoelBrouwer/continuousresource/tree/jos.

Brouwer, R. J. J. (2023a). *Data for the GCECSP with step-wise cost function*. Zenodo. DOI: 10.5281/zenodo.10304834. URL: https://github.com/RoelBrouwer/2023-step-wise-cecsp-data/tree/cpaior.

Brouwer, R. J. J. (Dec. 8, 2023b). *GCECSP: models and algorithms*. Version 0.2.0. DOI: 10.5281/zenodo.10304753. URL: https://github.com/RoelBrouwer/continuousresource/tree/cpaior.

Brouwer, R. J. J. (2024). *GCECSP: models and algorithms*. DOI: 10.5281/zenodo.10037570. URL: https://github.com/RoelBrouwer/continuousresource/.

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

Brouwer, R. J. J., J. M. van den Akker, and J. A. Hoogeveen (2023). "A hybrid local search algorithm for the continuous energy-constrained scheduling problem". Preprint at: https://arxiv.org/abs/2311.16177. DOI: 10.48550/arXiv.2311.16177.

Brouwer, R. J. J., J. M. van den Akker, and J. A. Hoogeveen (2024). "A hybrid optimization framework for the general continuous energy-constrained scheduling problem". Preprint at: https://arxiv.org/abs/2403.03039. DOI: 10.48550/arXiv.2403.03039.

CBS (July 2023). *Laagste energieverbruik in Nederland sinds 1990*. Centraal Bureau voor de Statistiek (CBS). URL: https://www.cbs.nl/nl-nl/nieuws/2023/27/laagste-energieverbruik-in-nederland-sinds-1990.

CBS, PBL, RIVM, and WUR (2024). *Aanbod en verbruik van elektriciteit, 1990-2022 (indicator 0020, versie 27, 24 July 2023)*. Compendium voor de Leefomgeving (CLO). URL: https://www.clo.nl/indicatoren/nl002027-aanbod-en-verbruik-van-elektriciteit-1990-2022.

Chen, Z., E. Demeulemeester, S. Bai, and Y. Guo (2018). "Efficient priority rules for the stochastic resource-constrained project scheduling problem". *European Journal of Operational Research* 270.3, pp. 957–967. DOI: 10.1016/j.ejor.2018.04.025.

Dai, H., N. Zhang, and W. Su (2015). "A literature review of stochastic programming and unit commitment". *Journal of Power and Energy Engineering* 3.4, pp. 206–214. DOI: 10.4236/jpee.2015.34029.

Detienne, B., S. Dauzère-Pérès, and C. Yugma (2011). "Scheduling jobs on parallel machines to minimize a regular step total cost function". *Journal of Scheduling* 14.6, pp. 523–538. DOI: 10.1007/s10951-010-0203-z.

Detienne, B., S. Dauzère-Pérès, and C. Yugma (2012). "An exact approach for scheduling jobs with regular step cost functions on a single machine". *Computers and Operations Research* 39.5, pp. 1033–1043. DOI: 10.1016/j.cor.2011.06.006.

Đumić, M. and D. Jakobović (2021). "Ensembles of priority rules for resource constrained project scheduling problem". *Applied Soft Computing* 110. DOI: 10.1016/j.asoc.2021.107606.

Enexis (2024). *Investeringsplan 2024 Enexis netbeheer*. URL: https://www.enexis.nl/over-ons/ons-investeringsplan.

ENTSO-E (2018). *Mid-term adequacy forecast 2018*. URL: https://www.entsoe.eu/outlooks/midterm/.

ENTSO-E (2024). *Monthly hourly load values. 2023 data*. URL: https://www.entsoe.eu/data/power-stats/.

Gamarra, C. and J. M. Guerrero (2015). "Computational optimization techniques applied to microgrids planning: A review". *Renewable and Sustainable Energy Reviews* 48, pp. 413–424. DOI: 10.1016/j.rser.2015.04.025.

Graham, R. L., E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan (1979). "Optimization and approximation in deterministic sequencing and scheduling: a survey". In: *Discrete Optimization II*. Ed. by P. L. Hammer, E. L. Johnson, and B. H. Korte. Vol. 5. Annals of Discrete Mathematics. Elsevier, pp. 287–326. DOI: 10.1016/S0167-5060(08)70356-X.

Gröwe-Kuska, N., H. Heitsch, and W. Römisch (2003). "Scenario reduction and scenario tree construction for power management problems". *Proceedings of the 2003 IEEE Bologna Power Tech Conference*. Vol. 3. DOI: 10.1109/PTC.2003.1304379.

Hartmann, S. and D. Briskorn (2010). "A survey of variants and extensions of the resource-constrained project scheduling problem". *European Journal of Operational Research* 207 (1), pp. 1–14. DOI: 10.1016/j.ejor.2009.11.005.

Hartmann, S. and D. Briskorn (2022). "An updated survey of variants and extensions of the resource-constrained project scheduling problem". *European Journal of Operational Research* 297 (1), pp. 1–14. DOI: 10.1016/j.ejor.2021.05.004.

Huang, Y., P. M. Pardalos, and Q. P. Zheng (2017). *Electrical Power Unit Commitment: Deterministic and Two-Stage Stochastic Programming Models and Algorithms*. DOI: 10.1007/978-1-4939-6768-1.

Hussain, M. T., N. B. Sulaiman, M. S. Hussain, and M. Jabir (2021). "Optimal management strategies to solve issues of grid having Electric Vehicles (EV): a review". *Journal of Energy Storage* 33, pp. 102–114. DOI: 10.1016/j.est.2020.102114.

Kolisch, R. (1996). "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation". *European Journal of Operational Research* 90.2, pp. 320–333. DOI: 10.1016/0377-2217(95)00357-6.

Koné, O., C. Artigues, P. Lopez, and M. Mongeau (2011). "Event-based MILP models for resource-constrained project scheduling problems". *Computers & Operations Research* 38, pp. 3–13. DOI: 10.1016/j.cor.2009.12.011.

Kopanos, G. M., T. S. Kyriakidis, and M. C. Georgiadis (2014). "New continuous-time and discrete-time mathematical formulations for resource-constrained project scheduling problems". *Computers and Chemical Engineering* 68, pp. 96–106. DOI: 10.1016/j.compchemeng.2014.05.009.

Liander (2023). *Ontwerp investeringsplan 2024 elektriciteit en gas*. URL: https://www.liander.nl/over-ons/financiele-publicaties/investeringsplannen.

Liang, H., A. K. Tamang, W. Zhuang, and X. S. Shen (2014). "Stochastic information management in smart grid". *IEEE Communications Surveys & Tutorials* 16.3, pp. 1746–1770. DOI: 10.1109/SURV.2014.020614.00115.

Liang, H. and W. Zhuang (2014). "Stochastic modeling and optimization in a microgrid: a survey". *Energies* 7.4, pp. 2027–2050. DOI: 10.3390/en7042027.

Liu, M., P. Mcnamara, R. Shorten, and S. McLoone (2015). "Residential electrical vehicle charging strategies: the good, the bad and the ugly". *Journal of Modern Power Systems and Clean Energy* 3.2, pp. 190–202. DOI: 10.1007/s40565-015-0122-2.

Lopes, J. A. P., C. L. Moreira, and A. G. Madureira (2006). "Defining control strategies for microgrids islanded operation". *IEEE Transactions on Power Systems* 21.2, pp. 916–924. DOI: 10.1109/TPWRS.2006.873018.

Lova, A., P. Tormos, and F. Barber (2006). "Multi-mode resource constrained project scheduling: scheduling schemes, priority rules and mode selection rules". *Inteligencia Artificial* 10.30, pp. 69–86. DOI: 10.4114/ia.v10i30.947.

Meng, L., E. R. Sanseverino, A. Luna, T. Dragicevic, J. C. Vasquez, and J. M. Guerrero (2016). "Microgrid supervisory controllers and energy management systems: A literature review". *Renewable and Sustainable Energy Reviews* 60, pp. 1263–1273. DOI: 10.1016/j.rser.2016.03.003.

Mohamed, M. A., A. M. Eltamaly, H. M. Farh, and A. I. Alolah (2015). "Energy management and renewable energy integration in smart grid system". *Proceedings of the 2015 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*, pp. 1–6. DOI: 10.1109/SEGE.2015.7324621.

Naber, A. (2017). "Resource-constrained project scheduling with flexible resource profiles in continuous time". *Computers and Operations Research* 84, pp. 33–45. DOI: 10.1016/j.cor.2017.02.018.

Nattaf, M., C. Artigues, and P. Lopez (2014). "A polynomial satisfiability test using energetic reasoning for energy-constraint scheduling". *Proceedings of the International Conference on Project Management and Scheduling (PMS 2014)*, pp. 169–172. URL: https://hal.archives-ouvertes.fr/hal-00978706.

Nattaf, M., C. Artigues, and P. Lopez (Oct. 2017). "Cumulative scheduling with variable task profiles and concave piecewise linear processing rate functions". *Constraints* 22 (4), pp. 530–547. DOI: 10.1007/S10601-017-9271-4.

Nattaf, M., C. Artigues, P. Lopez, and D. Rivreau (Mar. 2016). "Energetic reasoning and mixed-integer linear programming for scheduling with a continuous resource and linear efficiency functions". *OR Spectrum* 38 (2), pp. 459–492. DOI: 10.1007/S00291-015-0423-X.

Nattaf, M., M. Horváth, T. Kis, C. Artigues, and P. Lopez (2019). "Polyhedral results and valid inequalities for the continuous energy-constrained scheduling problem". *Discrete Applied Mathematics* 258, pp. 188–203. DOI: 10.1016/j.dam.2018.11.008.

Netbeheer Nederland (2024). *Capaciteitskaart elektriciteitsnet*. Version 19-01-2024. URL: https://capaciteitskaart.netbeheernederland.nl/.

Al-Ogaili, A. S., T. J. Tengku Hashim, N. A. Rahmat, A. K. Ramasamy, M. B. Marsadek, M. Faisal, and M. A. Hannan (2019). "Review on scheduling, clustering, and forecasting strategies for controlling electric vehicle charging: challenges and recommendations". *IEEE Access* 7, pp. 128353–128371. DOI: 10.1109/ACCESS.2019.2939595.

Palma-Behnke, R., C. Benavides, F. Lanas, B. Severino, L. Reyes, J. Llanos, and D. Sáez (2013). "A microgrid energy management system based on the rolling horizon strategy". *IEEE Transactions on Smart Grid* 4.2, pp. 996–1006. DOI: 10.1109/TSG.2012.2231440.

Pasha, J., B. Li, Z. Elmi, A. M. Fathollahi-Fard, Y.-y. Lau, A. Roshani, T. Kawasaki, and M. A. Dulebenets (2024). "Electric vehicle scheduling: state of the art, crit-

ical challenges, and future research opportunities". *Journal of Industrial Information Integration* 38. DOI: 10.1016/j.jii.2024.100561.

PBL, TNO, CBS, and RIVM (2022). *Klimaat- en energieverkenning 2022*. Planbureau voor de Leefomgeving (PBL).

Pflug, G. C. and A. Pichler (2014). *Multistage Stochastic Optimization*. Springer Series in Operations Research and Financial Engineering. DOI: 10.1007/978-3-319-08843-3.

Pinedo, M. L. (2016). *Scheduling: Theory, Algorithms, and Systems (5th ed.)* Springer. DOI: 10.1007/978-3-319-26580-3.

Smith, W. E. (1956). "Various optimizers for single-stage production". *Naval Research Logistics Quarterly* 3, pp. 59–66. DOI: 10.1002/nav.3800030106.

Stedin (2024). *Investeringsplan Stedin 2024*. URL: https://www.stedin.net/over-stedin/jaarverslagen-en-publicaties/investeringsplan.

TenneT (2024a). *Ontwerpinvesteringsplan net op land 2024-2033*. URL: https://www.tennet.eu/nl/over-tennet/publicaties/investeringsplannen.

TenneT (2024b). *Ontwerpinvesteringsplan net op zee 2024-2033*. URL: https://www.tennet.eu/nl/over-tennet/publicaties/investeringsplannen.

Tseng, C.-T., Y.-C. Chou, and Y.-C. Chou (2010). "A variable neighborhood search for the single machine total stepwise tardiness problem". *Proceedings of the 2010 International Conference on Engineering, Project and Production Management*, pp. 101–108. DOI: 10.32738/ceppm.201010.0011.

Turek, J., J. L. Wolf, and P. S. Yu (1992). "Approximate algorithms for scheduling parallelizable tasks". *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 323–332. DOI: 10.1145/140901.141909.

van den Akker, J. M. and J. I. Posthoorn (2021). "Simulation assignment Optimization for Sustainability (INFOMOFS): Smart charging in Polderwijk". URL: https://osiris-student.uu.nl/onderwijscatalogus/extern/cursus?cursuscode=INFOMOFS&taal=en&collegejaar=2020.

van Huffelen, E. (2023). "Scheduling of flexible EV charging under grid constraints and uncertainty". Master's thesis. Utrecht University. DOI: 20.500.12932/45597.

van Huffelen, E., R. J. J. Brouwer, and J. M. van den Akker (2024). "Grid-constrained online scheduling of flexible electric vehicle charging". Preprint at: http://arxiv.org/abs/2403.03109. DOI: 10.48550/arXiv.2403.03109.

Wang, Q., X. Liu, J. Du, and F. Kong (2016). "Smart charging for electric vehicles: a survey from the algorithmic perspective". *IEEE Communications Surveys & Tutorials* 18.2, pp. 1500–1517. DOI: 10.1109/COMST.2016.2518628.

World Resources Institute (2022). *Climate Watch data: GHG emissions*. Climate Watch. URL: https://www.climatewatchdata.org/ghg-emissions.

A

Nederlandse samenvatting

We zitten midden in een energietransitie. Elektriciteit voorziet in een steeds groter deel van onze energiebehoefte. Tegelijkertijd verandert de manier waarop die elektriciteit wordt opgewekt. Een steeds groter aandeel komt van hernieuwbare bronnen, waarbij de hoeveelheid opgewekte energie sterk afhangt van de weersomstandigheden. Dit levert nu al grote problemen op in het Nederlandse elektriciteitsnet. Niet alleen is het een steeds grotere uitdaging om het totaal van vraag en aanbod continue in balans te houden, maar ook het daarvoor benodigde transport van elektrische energie vormt een groeiend probleem. In dit werk passen we technieken uit de scheduling toe op problemen in deze context. We richten ons daarbij met name op het spreiden van de stuurbare vraag, zodat die beter kan worden afgestemd op de productie van energie, of goed gebruik maakt van de beschikbare capaciteit van een beperkte netaansluiting. Met scheduling technieken zoeken we naar een zo goed mogelijk schema hiervoor. De ontwikkelde methodologie zou bijvoorbeeld kunnen worden ingezet voor het coördineren van de vraag op een plek waar de transportcapaciteit van het netwerk (te) beperkt is, of voor het beter afstemmen van de pieken in vraag en aanbod. In algemene zin betogen we dat er winst te behalen is door het toepassen van scheduling technieken op dit type problemen. De kern van dit werk valt uiteen in twee delen met elk twee hoofdstukken. In het eerste deel verkennen we een algemene klasse van problemen en presenteren we een raamwerk dat kan worden ingezet voor het vinden van goede oplossingen. In het tweede deel kijken we naar twee specifieke voorbeelden van elektriciteitsplanningsvraagstukken, waarvoor we oplossingsmethoden hebben ontwikkeld.

In Hoofdstuk 2 bestuderen we het Continuous Energy-Constrained Scheduling Problem (CECSP), met het minimaliseren van de gewogen voltooiingstijd als doelstellingsfunctie. Dit probleem houdt in dat een schema gevonden moet worden voor een set activiteiten die gebruik maken van een continue, gedeelde grondstof (bijvoorbeeld: elektrisch vermogen), waarvan voortdurend een beperkte hoeveelheid

beschikbaar is. Voor elke activiteit moet een starttijd, voltooiingstijd en een consumptieprofiel gevonden worden, zodat voldaan wordt aan de volgende eisen. Een activiteit kan niet eerder starten dan de tijd waarop hij beschikbaar komt, en moet voor zijn deadline klaar zijn. Dat wil zeggen dat de volledige grondstofbehoefte van de activiteit voor de deadline voldaan is. Als een activiteit eenmaal begonnen is, mag hij niet onderbroken worden. Het consumptieprofiel, de hoeveelheid grondstof die een activiteit op elk gegeven moment consumeert, mag tussen de start- en voltooiingstijd fluctueren, zolang het maar binnen de grenzen blijft die gedefinieerd worden door de onder- en bovengrens op de grondstofconsumptie voor de activiteit. We introduceren een hybride aanpak, waarbij we simulated annealing gebruiken om de zoekruimte van mogelijke eventvolgordes te verkennen en een lineair programma om het optimale schema te vinden voor een gegeven eventvolgorde. Een eventvolgorde omvat alle events (start- en voltooiingstijden) in chronologische volgorde. We vergelijken onze hybride aanpak met een mixed-integer linear programming (MILP) formulering. Daarbij zien we dat de kwaliteit van oplossingen die we met onze hybride aanpak vinden voor kleine instanties overeenkomt met de kwaliteit van de oplossingen die uit de MILP formulering volgen. Voor grotere instanties presteert de hybride aanpak beter, aangezien het in staat is toegelaten oplossingen te vinden in relatief korte tijd. De MILP aanpak slaagt daar niet in.

In Hoofdstuk 3 presenteren we een generalisatie van het CECSP uit het vorige hoofdstuk: het General Continuous Energy-Constrained Scheduling Problem. We breiden de hybride aanpak uit tot een oplossingsraamwerk voor deze klasse problemen. De basis van dit raamwerk is de decompositie van het probleem, die volgt uit het eventgebaseerde model, in twee delen: 1) het bepalen van de eventvolgorde en 2) het vinden van de event tijden en consumptieprofielen. We breiden het eventgebaseerde model uit met events met vaste tijden. Deze events kunnen onder andere gebruikt worden voor het modelleren van stuksgewijs lineaire doelstellingsfuncties. We laten zien hoe dit gebruikt kan worden om de beoordeling van kandidaatoplossingen te versnellen. Verder beargumenteren we de brede toepasbaarheid van het raamwerk, en geven we een idee van de omvang van de klasse problemen waar het op toegepast kan worden. We illustreren de effectiviteit van het raamwerk met de toepassing op het CECSP met stapsgewijze kostenfuncties. Hierbij gebruiken we het feit dat de kosten van een oplossing alleen afhangen van de eventvolgorde en gebruiken we grenzen om op een efficiënte manier in te schatten of een oplossing toegelaten is. De vergelijking met een MILP formulering toont de effectiviteit van onze aanpak voor het vinden van oplossingen in relatief korte tijd.

Na de bespreking van deze generieke klasse van problemen, richten we ons op specifieke voorbeelden van scheduling problemen in elektriciteitsnetwerken. Het eerste voorbeeld, dat wordt besproken in Hoofdstuk 4, betreft het operationele beheer van microgrids zonder aansluiting op het landelijke netwerk. Deze grids zijn kleine, geïsoleerde netwerken met een beperkt aantal componenten. Elk van deze componenten produceert elektriciteit, consumeert elektriciteit, of slaat elektriciteit op. Een microgrid kan autonoom opereren, zolang het in staat is om de componenten zo in te zetten dat vraag en aanbod van elektrische energie conti-

nue in balans zijn. We presenteren twee optimalisatie modellen voor het plannen van de aansturing van diesel generatoren en batterijen. We gebruiken daarbij voorspellingen van het verbruik en de opwek van de zonnepanelen. De eerste is een deterministisch mixed-integer linear programming model dat een schema genereert op basis van de voorspelde waarden voor het verbruik en de opwek van de zonnepanelen. Een deel van de capaciteit van de batterij wordt niet gebruikt in de planning, en dient als buffer om afwijkingen van de voorspelling op te vangen. De tweede aanpak gebruikt een multi-stage stochastisch optimalisatie model dat de onzekerheid expliciet modelleert door middel van een boom van scenario's, gebaseerd op de voorspelde waarden en afwijkingen. We hebben herplanningsstrategieën ontwikkeld om het aantal keer dat een nieuw schema gegenereerd moet worden te minimaliseren. Beide modellen zijn toegepast op een casus in Nederland. De herplanningsstrategieën blijken daarbij zeer effectief te zijn. Het multi-stage stochastisch optimalisatie model lijdt onder de grote onzekerheid bij het voorspellen van verbruik en opwek in een klein systeem. De pragmatische aanpak van het deterministische model laat daarentegen goede resultaten zien.

Tot slot bekijken we in Hoofdstuk 5 het plannen van vraag in een netwerk van parkeerplaatsen voor elektrische voertuigen (EVs). Deze parkeerplaatsen zijn onderling verbonden door kabels met een beperkte capaciteit, en delen een (beperkte) aansluiting op het elektriciteitsnetwerk. Sommige parkeerplaatsen zijn overdekt, waarbij zonnepanelen op het dak zijn geïnstalleerd die elektriciteit terugleveren. EVs arriveren bij de parkeerplaatsen volgens een bekende verdeling. Wanneer een EV aankomt wordt bekend wat zijn gewenste vertrektijd, laadbehoefte en spreiding van toegelaten laadsnelheden zijn. Het vertrek van een EV wordt vertraagd als de laadbehoefte niet volledig voldaan is voor zijn gewenste vertrektijd. Het opladen van de EVs moet zo gebeuren dat de gemiddelde vertraging geminimaliseerd wordt. We presenteren een nieuwe aanpak, gebaseerd op een online variant van klassieke schedule generation schemes. We evalueren een seriële en een parallelle implementatie van een single-pass variant van deze aanpak, evenals een seriële en een parallelle implementatie van een variant die een destroy-and-repair heuristiek gebruikt om het schema verder te verbeteren. In alle gevallen selecteren we de best presterende combinatie van parameters voor verdere evaluatie. We vergelijken de uitkomsten van de verschillende varianten met het scenario waarbij geen enkele aansturing plaatsvindt, waarmee we de voordelen van het efficiënt plannen van het opladen laten zien. Daarnaast beargumenteren we dat het meenemen van bepaalde aspecten van het netwerk van belang is en dat het gebruik van flexibele laadsnelheden voordelen biedt.

B

Curriculum Vitæ

Roelof Jacob Jan Brouwer

1994 Born in Rotterdam, The Netherlands.

Education

2015–2017 **Master of Science** in **Computing Science** (cum laude)
Utrecht University, Utrecht
Thesis: Forecast-based optimal operation of islanded microgrids
Supervisors: J.M. van den Akker & W. Vermeiden

2013–2015 **Bachelor of Science** in **Computer Science** (cum laude)
Utrecht University, Utrecht

2012–2015 **Bachelor of Science** in **Artificial Intelligence**
Utrecht University, Utrecht
Thesis: Implementing Social Practices for Agent Systems
Supervisor: F.P.M. Dignum

2006–2012 **Voorbereidend Wetenschappelijk Onderwijs** (VWO)
Stedelijk Gymnasium Johan van Oldenbarnevelt, Amersfoort

Work

2017–present **Research Engineer**
Information and Technology Services, Utrecht University

2017–2024 **PhD Candidate**
Information and Computing Sciences, Utrecht University