Aggregation Techniques for Energy Flexibility

Ph.D. Dissertation
Emmanouil Valsomatzis

Dissertation submitted August, 2017

A thesis submitted to the Technical Faculty of IT and Design at Aalborg University (AAU) and the Universitat Politècnica de Catalunya, BarcelonaTech (UPC), in partial fulfillment of the requirements within the scope of the IT4BI-DC programme for the joint Ph.D. degree in computer science. The thesis is not submitted to any other organization at the same time.

Thesis submitted:	August, 2017
AAU Ph.D. Supervisor:	Prof. Torben Bach Pedersen Aalborg University (AAU), Denmark
UPC Ph.D. Supervisor:	Assoc. Prof. Alberto Abello Polytechnic University of Catalonia (UPC), Spain
AAU Ph.D. Co-Supervisor:	Assoc. Prof. Katja Hose Aalborg University, Denmark
Ph.D. Committee:	Assoc. Prof. Simonas Saltenis, Aalborg University, Denmark
	Assoc. Prof. Lukasz Golab University of Waterloo, Canada
	Senior Researcher Henrik W. Bindner Technical University of Denmark, Denmark
Ph.D. Series:	Technical Faculty of IT and Design, Aalborg University

ISSN (online):
ISBN:

Published by:
Aalborg University Press
Skjernvej 4A, 2nd floor
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2017

Abstract

Over the last few years, the cost of energy from renewable resources, such as sunlight and wind, has declined resulting in an increasing use of Renewable Energy Sources (RES). As a result, the energy produced by RES is fed into the power grid while their share is expected to significantly increase in the future.

However, RES are characterized by power fluctuations and their integration into the power grid might lead to power quality issues, e.g., imbalances. At the same time, new energy hungry devices such as heat-pumps and Electric Vehicles (EVs) become more and more popular. As a result, their demand in power, especially during peak-times, might lead to electrical grid overloads and congestions. In order to confront the new challenges, the power grid is transformed into the so-called Smart Grid. Major role in Smart Grid plays the Demand Response (DR) concept.

According to DR, Smart Grid better matches energy demand and supply by using energy flexibility. Energy flexibility exists in many individual prosumers (producers and/or consumers). For instance, an owner of an EV plugs-in his EV for more time than it is actually needed. Thus, the EV charging can be timely shifted. The load demanded for charging could be moved to time periods when production from wind turbines is high or away from peak-hours. Thus, RES share is increased and/or the electrical grid operation is improved.

The Ph.D. project is sponsored by the Danish TotalFlex project (http://totalflex.dk). Main goal of the TotalFlex project is to design and establish a flexibility market framework where flexibility from individual prosumers, e.g., household devices, can be traded among different market actors such as Balance Responsible Parties (BRPs) and distribution system operators. In order for that to be achieved, the TotalFlex project utilizes the flex-offer concept.

Based on the flex-offer concept, flexibility from individual prosumers is captured and represented by a generic model. However, the flexible loads from individual prosumers capture very small energy amounts and thus cannot be directly traded in the market. Therefore, aggregation becomes es-

sential. The Ph.D. project focuses on developing aggregation techniques for energy flexibilities that will provide the opportunity to individual prosumers to participate in such a flexibility market.

First, the thesis introduces several flexibility measurements in order to quantify the flexibility captured by the flex-offer model and compare flex-offers among each other, both on an individual and on an aggregated level. Flexibility is both the input and the output of the aggregation techniques. Aggregation techniques aggregate energy flexibility to achieve their goals and, at the same time, they try to retain as much flexibility as possible to be traded in the market. Thus, second, the thesis describes base-line flex-offer aggregation techniques and presents balance aggregation techniques that focus on balancing out energy supply and demand. Third, since there are cases where electrical grid congestions occur, the thesis presents two constraint-based aggregation techniques. The techniques efficiently aggregate large amounts of flex-offers taking into account physical constraints of the electrical grid. The produced aggregated flex-offers are still flexible and when scheduled, a normal grid operation is achieved. Finally, the thesis examines the financial benefits of the aggregation techniques. It introduces flex-offer aggregation techniques that take into account real market technical requirements. As a result, individual small flexible loads can be indirectly traded in the energy market through aggregation.

The proposed aggregation techniques for energy flexibilities can contribute to the use of flexibility in the Smart Grid in both current and future market frameworks. The designed techniques can improve the services offered to the prosumers and avoid the very costly upgrades of the distribution network.

Abstract in Danish / Resumé

Gennem de senere år er prisen faldet på energi fra vedvarende energikilder såsom sollys og vind, hvilket har medført et stigende forbrug af vedvarende energi. Dette har resulteret i, at energi, der produceret af vedvarende energi, sendes ud i elnettet og andelen forventes at stige markant i fremtiden.

Vedvarende energi er imidlertid karakteriseret af effektsvingninger, og integrationen i elnettet kan føre til kvalitetsproblemer med strømmen som for eksempel uligevægt. Samtidig bliver enheder, der sluger vedvarende energi såsom varmepumper og elektriske køretøjer, mere og mere populære. Dette resulterer i, at efterspørgslen på energi, især i spidsbelastede situationer, kan medføre overbelastning og trængsel på elnettet. For at konfrontere de nye udfordringer bliver elnettet ændret til et såkaldt Smart Grid. Konceptet om udbud og efterspørgsel Demand Response (DR) spiller her en meget stor rolle.

Ifølge DR, imødegår Smart Grid bedre udbud og efterspørgsel af energi ved at bruge fleksibel energi. Fleksibel energi eksisterer i mange individuelle producenter og/eller forbrugere. For eksempel tilslutter en ejer af et elektrisk køretøj sit køretøj i mere tid end det rent faktisk er nødvendigt. På denne måde kan tidspunktet for opladningen ændres rettidigt. Belastningen, der kræves for opladning, kunne flyttes til perioder, hvor produktion fra vindmøller er høj eller væk fra de spidsbelastede tidspunkter. Således øges vedvarende energi' andel og/eller elnettets drift er forbedret.

Dette Ph.D. projekt er sponsoreret af det danske TotalFlex projekt (http://totalflex.dk). TotalFlex' formål er at designe og etablere et fleksibelt elmarkedsystem, hvor fleksibilitet fra individuelle producent og/ eller forbruger f.eks. husholdningsenheder kan blive udvekslet mellem forskellige markedsaktører såsom balanceansvarlige parter og eldistributionsnettets operatører. For at opnå dette, udnytter TotalFlex flex-offer konceptet.

Baseret på konceptet om flex-offer, bliver fleksibilitet fra individuelle prosumers fanget og repræsenteret i en generisk model. Fleksible belastninger fra de individuelle prosumers fanger imidlertid kun meget små energimængder og kan ikke udveksles direkte på markedet. Derfor bliver aggregering essentielt. Ph.D. projektet fokuserer på at udvikle aggregering-

steknikker for energifleksibilitet, der kan give individuelle prosumers mulighed for at deltage i et sådant fleksibilitetsmarked.

Først vil afhandligen introducere adskillige fleksibilitetsmålinger for at kvantificere fleksibiliteten, der fanges af flex-offer modellen og sammenligne flex-offer med hinanden både på et individuelt og et aggregeret niveau. Input og output af aggregeringsteknikker er fleksibilitet. Aggregeringsteknikker samler energifleksibilitet for at opnå dets mål og forsøger på samme tid at beholde så meget fleksibilitet som muligt til at blive udvekslet på markedet. Herpå forsøger afhandligen for det andet at beskrive basis flex-offer aggregeringsteknikker og præsenterer balance-aggregeringsteknikker, der fokuserer på at afbalancere energiudbud og -efterspørgsel. Siden der er situationer, hvor overbelastninger af elnettet forekommer, præsenterer afhandlingen for det tredje, to begrænsningsbaserede aggregeringsteknikker. Teknikkerne samler effektivt store mængder af flex-offers og tager samtidig hensyn til fysiske begrænsninger i elnettet. De producerede, samlede flex-offers er stadig fleksible og efter det er planlagt, opnås et normaltfungerende net. Til slut vil afhandlingen undersøge de økonomiske fordele ved aggregeringsteknikkerne. Den introducerer flex-offer aggregeringsteknikkerne, der tager højde for de reelle, tekniske krav, der er på markedet. Resultatet kan være, at individuelle små fleksible belastninger indirekte kan udveksles på energimarkedet gennem aggregering.

De foreslåede aggregeringsteknikker til energi-fleksibilitet kan bidrage til brug af fleksibilitet i Smart Grid i både nuværende og fremtidige markedsrammer. De designede teknikker kan forbedre de tilbudte ydelser til prosumers og undgå de meget dyre opgraderinger af distributionsnetværk.

Abstract in Spanish / Resumen

Durante los últimos años, la bajada en el precio de la energía procedente de fuentes renovables, tales como luz solar y eólica, ha resultado en un aumento del uso de este tipo de recursos de Energía Renovables (ER). Como consecuencia de este aumento, la energía producida a través de ER es inyectada en la red eléctrica y se espera que la proporción de energía suministrada a la red crezca significativamente en los próximos años.

Sin embargo, las ER se caracterizan por ser muy fluctuantes y su integración en la red eléctrica podría acarrear problemas de calidad, como por ejemplo desequilibrios energéticos. Al mismo tiempo, nuevos dispositivos de alto consumo de energía, como bombas de calor y vehículos eléctricos, son cada vez mas populares y la alta demanda de estos, especialmente en horas puntas, puede crear sobrecargas y congestiones en la red. Para afrontar estos restos, la red eléctrica se transforma en la llamada Red Inteligente, dónde el concepto de respuesta a la demanda juega un papel.

Esta thesis de doctorado está patrocinada por el proyecto danés TotalFlex (http://totalflex.dk). El objetivo principal de este proyecto es diseñar y establecer el marco de flexibilidad de mercado, dónde la flexibilidad de productores/consumidores, por ejemplo los dispositivos del hogar, pueda ser comercializada entre los diferentes actores del mercado como las comercializadoras de electricidad y los operadores de sistemas de distribución. Para lograr este propósito, el proyecto TotalFlex utiliza el concepto flex-offer flexibilidad en la oferta.

Basado en el concepto flex-offer, la flexibilidad de consumidores y productores individuales es capturada y representada a través de un modelo genérico. Sin embargo, las cargas flexibles de estos individuos producen pequeñas cantidades de energía y, por lo tanto, no pueden ser directamente negociadas en el mercado. Esto significa que la agregación de esta energía es esencial. Este Ph.D está enfocado desarrollo de técnicas de agrega para que permitirán a productores y consumidores individuales participar en dicha.

En primer lugar, esta tesis introduce medidas de flexibilidad con la finalidad de cuantificar la flexibilidad calculada por el modelo flex-offer y comparar las diferentes ofertas entre ellas, tanto a nivel individual como agregado. Flexibilidad es tanto la entrada como la salida de las técnicas de agregado, las cuáles agregan flexibilidad energética para lograr sus objetivos y, al mismo tiempo, retener la máxima flexibilidad para comerciarla en el mercado. En segundo lugar, la tesis describe la base de las tecnicas de agregado flex-offer y presenta técnicas de que se enfocan en un balance entre la oferta y la demanda energética. Tercero, dado que existen casos dónde se producen congestiones en la red eléctrica, la tesis presenta tecnicas de agregado basadas en restricciones. Dichas técnicas agregan grandes cantidades de flex-offers considerando restricciones físicas de la red eléctrica. Las flex-offers agregadas que se producen son aún flexibles y, cuando se programan, se logra una operación normal de red. Por último, en la tesis se examina los beneficios económicos de las técnicas agregadas, introduciendo técnicas de agregado flex-offer que tienen en cuenta los requisitos técnicos del mercado real. Como resultado, las pequeñas cargas individuales y flexibles pueden ser indirectamente negociadas en el mercado energético a través de la agregación. Las técnicas de agregado propuestas para favorecer la flexibildad energética puede contribuir al uso de flexibilidad en la red inteligente tanto en el presente como en el futuro. Mejorar los servicios ofrecidos a consumidores y productores así como evitar las costosas actualizaciones de la red de distribución.

Acknowledgements

First of all, I would like to thank my main Ph.D. supervisor Professor Torben Bach Pedersen for being an exquisite teacher. I feel lucky and proud to have crossed his path. It has been a great pleasure to collaborate with him and I am grateful to him for giving me the opportunity to be his student.

I would like to express my sincere gratitude to Associate Professor Alberto Abello and Associate Professor Katja Hose for all their help and valuable contributions to my research.

I would also like to thank all my colleagues and staff in the Computer Science department. I really enjoyed my time among them and I feel happy to have been a member of this academic group.

I acknowledge that the TotalFlex project sponsored by the ForskEL program of Energinet.dk supported in part this research.

I would like to thank my family for their support through all these years.

Last but not least, a great thanks to Aliki who supported me and encouraged me through all these Ph.D years.

Contents

Thesis Details

Thesis Title:	Aggregation Techniques for Energy Flexibility
Ph.D. Student:	Emmanouil Valsomatzis
Supervisors:	Prof. Torben Bach Pedersen, Aalborg University (AAU Supervisor) Assoc. Prof. Alberto Abello, Polytechnic University of Catalonia (UPC Supervisor) Assoc. Prof. Katja Hose, Aalborg University (AAU Co-Supervisor)

The main body of this thesis consist of the following papers.

[1] Emmanouil Valsomatzis, Katja Hose, Torben Bach Pedersen, Laurynas Siksnys,"Measuring and Comparing Energy Flexibilities," *In Proceedings of the Workshops of the EDBT/ICDT 2015 Joint Conference (EDBT/ICDT), Belgium, Brussels*, vol. 1330, pp. 78–85, 2015.

[2] Laurynas Siksnys, Emmanouil Valsomatzis, Katja Hose, Torben Bach Pedersen,"Aggregating and Disaggregating Flexibility Objects," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 27, no. 11, pp. 2893–2906, 2015

[3] Emmanouil Valsomatzis, Katja Hose, Torben Bach Pedersen, "Balancing energy flexibilities through aggregation," *In Proceedings of the Second International Workshop on Data Analytics for Renewable for Renewable Energy Integration (DARE)*, Nancy, France, pp. 17-37, 2014

[4] Emmanouil Valsomatzis, Torben Bach Pedersen, Alberto Abelló, Katja Hose,"Aggregating energy flexibilities under constraints", *In Proceedings of the 7th IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Sydney, Australia, pp. 484-490, 2016

[5] Emmanouil Valsomatzis, Alberto Abelló, Torben Bach Pedersen, "Trading Aggregated Flex-Offers via Flexible Orders", *DBTR series*,

http://dbtr.cs.aau.dk/DBPublications/, DBTR- 38.pdf, Aalborg University, Technical Report, 2017

In addition to the main papers, the following publication has also been made. A short version of paper number 4:

[6] Emmanouil Valsomatzis, Torben Bach Pedersen, Alberto Abelló, Katja Hose, Laurynas Siksnys, "Towards constraint-based aggregation of energy flexibilities," *In Proceedings of the Seventh International Conference on Future Energy Systems Poster Sessions (e-Energy '16)*, Waterloo, Canada, 2 pages, 2016

This thesis has been submitted for assessment in partial fulfillment of the joint Ph.D. degree. The thesis is based on the submitted or published scientific papers which are listed above. Parts of the papers are used directly or indirectly in the extended summary of the thesis. As part of the assessment, co-author statements have been made available to the assessment committee and are also available at the Faculty. The permission for using the published and accepted articles in the thesis has been obtained from the corresponding publishers with the conditions that they are cited, DOI pointers and/or copyright/credits are placed prominently in the references.

Chapter 1

Introduction

1 Background and Motivation

Smart Grid uses advanced Information and Communication Technology (ICT) [5] to improve the power grid services quality and increase the use of energy from RES, such as wind and sunlight [8].

However, the Smart Grid confronts many challenges. In particular, RES are characterized by volatile generation. As a result, high energy production might be generated during times when consumption is low which might lead to power grid problems [34]. On the other hand, new technological advancements, such as heat-pumps, demand a lot of energy [7, 45] and during peak-hours, they might create congestions into the power grid [51]. Energy storage is an expensive solution to avoid such problems while feeding the surplus energy into the electrical grid might be dangerous for its infrastructure. That is the reason flexibility plays a prominent role in Smart Grid and contributes to the achievement of its goals. Flexibility can be used to shift loads away from peak-hours and balance out production with consumption. Consequently, a very expensive electrical grid upgrade is avoided, the use of renewable energy sources is increased, and new business opportunities arise in a new market [27] were flexibility can be traded [61].

The Ph.D. project is supported by the TotalFlex project. The main objective of the TotalFlex project is the design of a new flexibility market where individual prosumers (producers and/or consumers) with flexible loads can participate, e.g., loads from household devices. In such a market, flexibility can be traded in order to tackle congestion and balancing problems. However, the loads from individual devices are too small to be traded in the market and the scheduling of such small devices results in a highly complex Unit Commitment (UC) problem [79]. Therefore, it is essential to aggregate the flexible loads from the individual devices in order to trade them in the

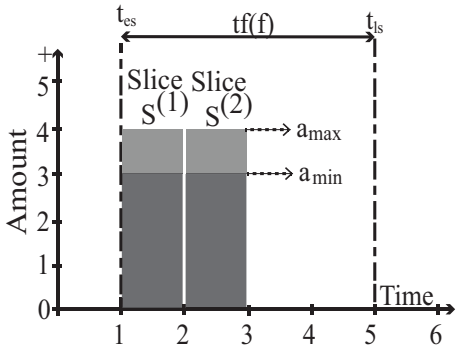


Fig. 1.1: Illustration of a flex-offer

market and reduce the complexity of the scheduling problem [80].

The goal of the Ph.D. project is to design and develop aggregation techniques that produce aggregated energy flexible loads so that the complexity of the UC problem is reduced and loads from individual prosumers are traded in the market.

2 Life-cycle of Flex-offers

In order to achieve its goal, the Ph.D. project utilizes the flex-offer concept—introduced in the MIRABEL project [56]—that captures flexibility in both the amount and time dimensions as presented in the following example.

Example 2.1
The owner of an EV plug-ins her EV at 1 a.m. She would like the car to be charged within the 80% to 100% of its full capacity at 7 a.m. The EV needs only 2 hours to be charged. Thus, its charging cans start between 1:00 and 5:00 a.m.

The flex-offer is defined accordingly. A flex-offer f is a two elements tuple. Its first element captures the range of the potential starting time (time flexibility interval). The second element of the tuple is a sequence of slices with minimum and maximum amount requirements per time unit. For instance, the EV in the above example is represented with the flex-offer $f = ((1,5)\langle[3,4],[3,4]\rangle)$, see Figure 1.1. 1 and 5 are the *earliest* and *latest starting time* of the time flexibility interval, respectively. Flex-offer f has two identical slices. Their minimum and maximum amounts are 3 and 4, respectively.

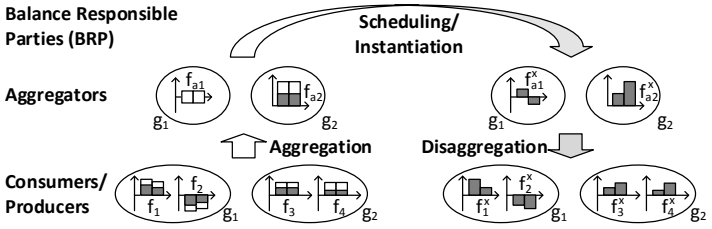


Fig. 1.2: The life-cycle of the flex-offers [75]

In the market scenario described by the TotalFlex project, a new market actor called Aggregator handles a portfolio of a large number of flex-offers. The flex-offers derive from different prosumers, see Figure 1.2. A flex-offer can be either positive, negative, or mixed. The positive flex-offers have positive values for all of their slice amounts and represent devices that consume energy, e.g., dishwashers and washing machines. The negative flex-offers represent devices that supply energy, such as solar panels, and all their slices are negative. The mixed flex-offers have both positive and negative slices and represent devices that can both consume and produce energy, e.g., an EV with vehicle to grid discharging option.

The Aggregator has a collateral agreement with the prosumers. The prosumers offer their flexibility to the Aggregator. The Aggregator gains control of their devices, but he respects the comfort settings of the prosumers. At the same time, the prosumers are awarded accordingly, e.g., with lower tariffs. The goal of the Aggregator is to aggregate the flex-offers and produce aggregated ones, taking into account different objectives, e.g., balancing energy, see g_1 and g_2 on the left side of Figure 1.2. During aggregation, the Aggregator utilizes part of the flex-offers flexibility to achieve his objective and the produced aggregated flex-offers are still flexible. Moreover, the aggregated flex-offers respect all the amount and time requirements of the individual flex-offers. As a result, disaggregation that maps to the initial flex-offers is also supported. The flexible aggregated flex-offers are then handled by the BRPs who trade them in the market. Afterwards, the aggregated flex-offers are scheduled. Thus, their flexibility is instantiated, i.e., their starting time and the amounts of all the slices are defined. The instantiated aggregated flex-offers are then disaggregated so that the individual flex-offers are transformed into assignments. Their starting times and their slice amounts are assigned according to the instantiated aggregated flex-offers .

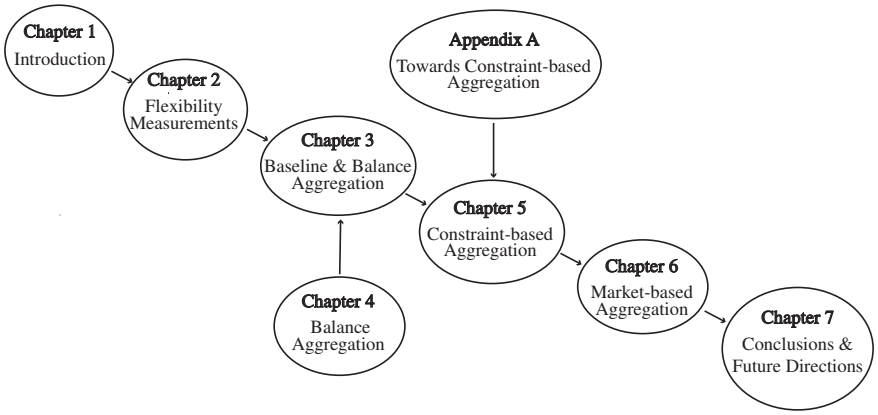


Fig. 1.3: The Ph.D. story

3 Thesis overview

This section is an overview of each chapter and the appendix in this thesis. Figure 1.3 illustrates the thesis flow. An overall description of the thesis is given in Chapter 1. Chapter 2 introduces several flexibility measurements that are used to develop and evaluate the aggregation techniques. Chapter 3 discusses baseline flex-offer aggregation and introduces several balance aggregation techniques that focus on balancing out demand and supply. Chapter 4 introduces two variations of the balance aggregation techniques described in Chapter 3 and it also presents an extensive experimental setup dedicated on evaluating balance aggregation. Chapter 5 describes the constraint-based aggregation techniques that are applied on flex-offers in order to confront electrical grid bottlenecks. Chapter 6 describes the market-based flex-offer aggregation and introduces several aggregation techniques that take into account market requirements in order to trade the aggregated flex-offers in the market. Appendix A contains the initial stage of constraint-based aggregation work. It describes the preliminary concepts and results of constraint-based aggregation. Finally, Chapter 7 summarizes the outcome of the thesis and describes the future research directions.

3.1 Chapter 2: Measuring and Comparing Energy Flexibilities

The volatile nature of the RES acts like a barrier to their share because of the potential energy shortages that might be caused to the power grid [34]. That is the reason why the Smart Grid uses energy flexibility to tackle such challenges. Subsequently, flexible loads are activated when energy genera-

tion from RES occurs and flexibility becomes valuable. The TotalFlex project exploits the flexibility of small loads, e.g., household appliances, to achieve the aforementioned goal by using the flex-offer concept [73].

However, a flexibility measure is needed to identify how much and what kind of flexibility is offered. In addition, due to the high number of the flex-offers and the small load amounts that such devices capture, aggregation of flex-offers is essential. Flex-offer aggregation reduces the complexity of the UC problem (scheduling) [62] and gives the aggregated flex-offers the opportunity to be traded in a flexibility market. Due to the fact that aggregation techniques utilize flexibility to reach their objectives, a flexibility measure is also needed to evaluate the aggregation techniques. Different flexibility definitions have been proposed, with some of them focusing on the use of flexibility [16, 65] e.g., tackling balancing issues, and others focusing on the main characteristics of the flexible power units [66]. However, a common agreed flexibility definition is still pending [65].

In comparison, Chapter 2 introduces several flexibility definitions to quantify flexibility in both supply and energy demand based on the generic flex-offer model. The flex-offer model represents models from different devices and takes into account time and energy dimensions, both individually and combined. Chapter 2 initially suggests, the *time* and the *energy flexibility* measurements that take into account time and energy dimension, respectively. However, when flex-offers have both time and energy flexibility, those two measurements do not capture their combined effect. Thus, it is proposed that, the *product flexibility* measurement can be used to capture the combined effect of time and energy flexibility. The chapter also suggests the *vector flexibility* measurement that handles time and energy flexibility as vector components. The length of the vector can be computed with relevant norms, such as the Manhattan and the Euclidean, and expresses the total flexibility of the flex-offer. As a result, the vector flexibility captures both time and energy dimension. Moreover, the chapter proposes 2 more flexibility definitions so-called, *time-series* and *assignments* flexibility that are both based on the assignments of a flex-offer. The time-series flexibility definition considers the difference between the minimum and the maximum assignment whereas the assignments flexibility definition takes into account the number of assignments that a flex-offer has. Both definitions can capture flexibility derived from energy and time and they can also be applied on all the types of flex-offers. However, they do not take into account the size of the flex-offers. Consequently, flex-offers with different load sizes might be considered to have the same flexibility. Thus, Chapter 2 also proposes a flexibility measurement that is size related. In particular, it proposes the *absolute area-based* flexibility measurement that takes into account the sum of the potential energy that a flex-offer could capture due to its time flexibility. Moreover, the chapter introduces the *relative area-based* flexibility that is a size-independent measurement. Both

absolute and relative area-based flexibility measurements consider the combined effect of time and energy. However, absolute area-based flexibility is a size-dependent measurement whereas relative are-based measurement expresses the flexibility per energy unit and can be used to compare flex-offers with different sizes.

3.2 Chapter 3: Aggregating and Disaggregating Flexibility Objects

Flexibility of flex-offers is very important for aggregation. Goal of the aggregation, apart from reducing the number of the flex-offers, is to retain flexibility. This is due to the fact that flexibility is used by the scheduling process to find the optimum scheduling. Chapter 3 discusses baseline aggregation techniques that try to reduce the number of flex-offers and retain as much flexibility as possible measured according to the introduced measurements of Chapter 2. It shall be noted that in Chapter 3, the generic term flex-object is used instead of the term flex-offer. The reason is that the baseline aggregation was initially introduced considering a broader application domain. However, the Ph.D. study focuses on energy domain and considers a market scenario where flexibility can be offered. Hence, the term flex-offer was selected thereafter as a more appropriate one and it is used in this chapter to avoid misconceptions.

Chapter 3 proposes the Start-Alignment (SA) aggregation that is applied on flex-offers. SA is applied on a set of flex-offers and produces a single aggregated flex-offer. The earliest starting time of the aggregated flex-offer is the minimum earliest starting time among the flex-offers that participate in aggregation. The time flexibility of the aggregated flex-offer is the minimum time flexibility of the flex-offers that participate in aggregation. In order to reduce the flexibility losses, a grouping phase is introduced prior to SA aggregation. The grouping of the flex-offers takes into account two parameters, the earliest starting time tolerance and the time flexibility tolerance. The flex-offers with earliest starting time and time flexibility difference less or equal to the corresponding defined grouping parameters are grouped together. Moreover, aggregation supports the bin-packing phase where a constraint is applied on the grouping result. For instance, the size of the group shall be within the range specified by the bin packing parameters. After the grouping and the bin-packing phase, SA aggregation is applied on each group and an aggregated flex-offer per group is produced. Based on an extensive experimental setup, Chapter 3 shows that grouping parameters with values close to zero minimize the flexibility losses but increase the number of the aggregated flex-offers. That is a proof of the trade-off between the output size of aggregation and flexibility. A smaller number of aggregated flex-offers leads to higher flexibility losses. The clustering techniques proposed in the liter-

ature, e.g., [23, 52, 72, 92], can only partially solve the aggregation problem. On the contrary, the proposed aggregation techniques efficiently handle both the grouping and the flex-offer aggregation problem. Moreover, aggregation techniques suggested for temporal, spatio-temporal, and multidimensional data [6,14,15,39,89] are designed for inflexible data and thus are inapplicable to aggregation of flex-offers.

Since the output of the aggregation (the aggregated flex-offers) are handled by a BRP whose main goal is the energy balancing and the avoidance of electrical grid overloads, Chapter 3 also introduces the balance aggregation. Balance aggregation takes advantage of counteracting flex-offers and produces aggregated flex-offers with slice amounts close to zero. In particular, goal of balance aggregation is the minimization of the the sum of the absolute balance of the aggregated flex-offers. The absolute balance of a flex-offer is defined as the sum of the absolute average amount flexibilities of its slices. Chapter 3 proposes five different balance aggregation techniques that can be applied on a group of flex-offers and take advantage of the different profile alignments of the flex-offers. The *exhaustive search* technique examines all the potential alignments among the flex-offers of the group to identify the aggregated flex-offer that minimizes the absolute balance. It demands a lot of time to be executed, but it identifies the optimal solution based on a single aggregated flex-offer. Chapter 3 also introduces the *zero terminated exhaustive search* which, similarly to exhaustive search, terminates when a zero absolute balance value has been identified. The *dynamic simulated annealing* technique is also introduced in Chapter 3. In order to reduce the execution time of the exhaustive search, dynamic simulated annealing examines only a random number of alignment combinations defined by a parameter.

Finally, Chapter 3 proposes two more greedy techniques, the *simple* and the *exhaustive greedy*. Both techniques are also applied on a group of flex-offers, but they might produce more than one aggregated flex-offer. Simple greedy starts by selecting the flex-offer with the most negative balance (initial flex-offer) and aggregates it with flex-offers that have balance closer to the opposite one. Thus, it tries to balance out the first selected flex-offer. It then examines all the potential alignments using the flex-offers in the group as long as the absolute balance of the aggregated flex-offer is reduced. Otherwise, it selects a new initial flex-offer and continues until all the flex-offers have been examined. Exhaustive greedy also might produce more than one aggregated flex-offer. Differently from simple greedy, it starts by selecting the flex-offer with the greatest absolute balance (initial flex-offer). The technique then examines all the potential alignment combinations between the initial flex-offer and all the remaining flex-offers in the group. It continues aggregation as long as the absolute balance of the aggregated flex-offer is reduced. Otherwise, it selects a new initial flex-offer and terminates when all the flex-offers in the group have been examined. An extensive experimental setup

focusing on the evaluation of balance aggregation is presented in Chapter 4

3.3 Chapter 4: Balancing Energy Flexibilities through Aggregation

Chapter 4 focuses on balance aggregation techniques. In particular, it evaluates both simple and exhaustive greedy (introduced in Chapter 3) under four different experimental setups. It also introduces two variations of the greedy techniques using a different starting point and it also examines the performance of start alignment aggregation introduced in Chapter 3.

Each experimental setup consists of 10 groups of 8 datasets that are created based on real historical consumption data from the MIRABEL project The datasets include flex-offers that capture both production and consumption devices. The datasets are also characterized by different time flexibility distributions and different profile length distributions. Energy scenarios that take into account different flexibility characteristics for the flex-offers and simulate cases where RES co-exist with power hungry devices are also considered.

The experimental setup shows that both exhaustive and zero exhaustive search identify the solutions with zero absolute balance, when the grouping parameters are set to zero. However, they both have very high execution times. On the contrary, start alignment aggregation achieves the worst absolute balance, yet the lowest execution time, as it examines only a single alignment combination per group. Simple and exhaustive greedy achieve to produce balanced aggregated flex-offers with low processing times as well. They also show an almost constant rate of flexibility losses, as the size of population increases. Exhaustive greedy achieves a better absolute balance when the input size is high, as it examines a larger solution space compared to simple greedy.

The problem of balancing energy production and supply has been previously handled as part of the UC problem [42, 62] either with distributed (e.g., [50]) or centralized solutions (e.g., [35]). Moreover, scheduling techniques for flex-offers are also proposed in [80]. However, the proposed balance aggregation techniques in both Chapter 3 and Chapter 4 integrate balancing into aggregation and produce aggregated flex-offers, so that their flexibility can be then used by a scheduling technique. Thus, the balance aggregation techniques reduce scheduling complexity and, at the same time, they partially handle the balancing problem.

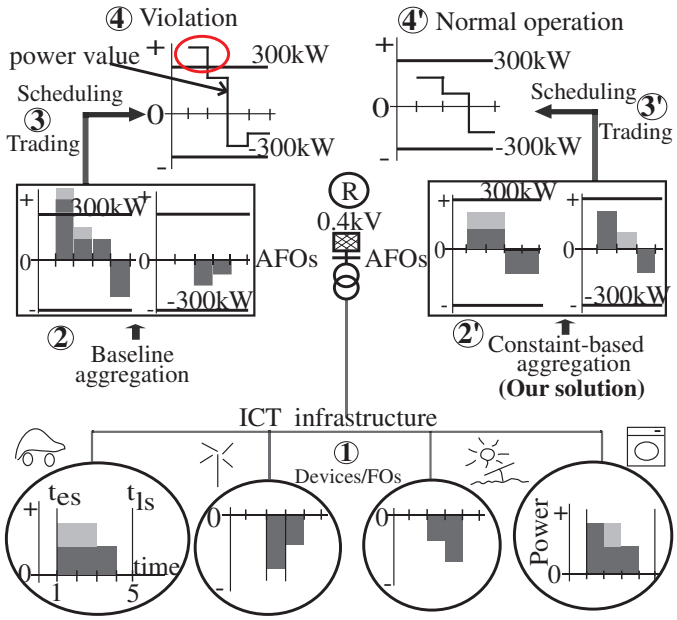


Fig. 1.4: Traditional vs Constraint-based aggregation [83].

3.4 Chapter 5: Aggregating Energy Flexibilities under Constraints

There are cases where the objectives of a market actor contradict the objectives of another actor. That might especially happen at electrical grid locations with limited power capacity (bottlenecks). For instance, there might be high demand for energy consumption and a business case of selling energy, but at the same time a bottleneck might occur, endangering the grid operation. In such a case, flexible loads become valuable and flexibility can be used to confront bottlenecks by shifting the loads away from congestions. Therefore, Chapter 5 proposes flex-offer aggregation techniques that take into account grid constraints.

The flex-offers that participate in aggregation are also part of the electrical grid and their operation depends on the flex-offers utilization. For instance, there are low voltage grid elements such as a distribution transformer with a maximum power value of few hundred of kW. Thus, a simultaneous charge of few hundred EVs might create severe problems to the electrical grid. Chapter 5 maps the structure and the constraints of the grid to a tree based on [88]. The bottleneck of the grid is the root of the tree, see Ⓡ in Figure 1.4. The flex-offers represent devices, all connected to the grid, see bottom of Figure 1.4. The aggregation of flex-offers is needed (② in Figure 1.4), in order

for the flex-offers trading to take place, while the complexity of scheduling is reduced (③ in Figure 1.4). However, when baseline aggregation techniques, introduced in Chapter 3 are applied, the grid constraint imposed by the root is not respected. The aggregated flex-offers have amounts above the grid constraint value and as a result, the scheduling applied afterwards cannot find a solution respecting the grid constraint. Thus, the grid constraint is violated and it might collapse, see ④ in Figure 1.4. Chapter 5 proposes two techniques that take into account the grid constraint during aggregation. Therefore, when the produced aggregated flex-offers are scheduled, a valid solution (schedule) that respects the grid constraint is generated. Subsequently, a normal grid operation is achieved, see ④′ in Figure 1.4.

Due to the fact that the complexity of constraint-based aggregation is extremely high, Chapter 5 introduces two heuristics, i.e., the *Simple Greedy* (SG) and the *Exhaustive Greedy* (EG). Both techniques consider a target function that represents the objective of a BRP and a constraint function that represents the power grid constraint. However, a normal grid operation is prioritized. Hence, a weighted function to compute the distance to both the constraint and the target function is used, with a higher coefficient for the target function.

Simple and exhaustive greedy are based on binary aggregations to reduce their complexity. They consider different starting points, while SG examines a smaller solution space compared to EG. As a result, SG is much faster, whereas EG is more effective when a high number of flex-offers participates in aggregation. The reason is that EG explores a larger solution space than simple greedy does. Thus, it produces aggregated flex-offers, which, when scheduled, respect the grid constraint.

Chapter 5 experimentally evaluates the proposed techniques and compares them with the aggregation techniques introduced in Chapter 3. Chapter 5 considers a use case scenario where the objective of a BRP (target function) contradicts the objective of the distribution system operator (constraint function and power grid constraint). Thus, both a medium voltage grid and a mixed portfolio of flex-offers are taken into account. Different sizes of datasets are also considered with different characteristics. The experimental setup shows that SG outperforms EG in terms of processing time, yet it produces a higher number of aggregated flex-offers. As a result, SG also has less time flexibility losses compared to EG. However, EG is the only technique that respects the constraint in all cases. When a high number of flex-offers is connected to the grid, it guarantees a normal operation, whereas baseline aggregation violates the grid in the 18% of the time horizon.

3.5 Chapter 6: Trading Aggregated Flex-Offers via Flexible Orders

The thesis previous chapters introduce flex-offer aggregation techniques that focus on the technical benefits reaped by utilizing flexibility as defined in Chapter 2. Chapter 3 focuses on the balance between demand and supply and Chapter 5 on guarantying a normal grid operation. In comparison, Chapter 6 examines the market perspective of aggregation and focuses on aggregation techniques that produce aggregated flex-offers which comply with real market requirements. In particular, it focuses on flex-offers from electrical vehicles and the day-ahead market of the Nordic region. EVs are power hungry devices characterized by time flexibility, since they are mainly plugged-in for more time than their charging demands. Hence, their flexibility can be used by market actors and be traded in the energy market, creating new business opportunities [37, 44].

In the bibliography, there are many works proposing different techniques to tackle energy trading of flexible loads. Their goal is to trade the energy required to charge (and/or discharge) a population of EVs in day-ahead markets at a minimum cost [71]. For instance, in [90] and [48] several scheduling techniques are proposed based on day-ahead prices to minimize the charging cost of EVs. The main characteristic of the proposed techniques in research is the scheduling of the flexible loads. A market actor, e.g., a BRP, controls a large population of EVs and schedules them considering different parameters such as driving activity uncertainties [90] or price signals [24]. The outcome of techniques is aggregated *non-flexible* loads. On the contrary, Chapter 6 introduces 3 market-based aggregation techniques that produce aggregated *flexible* loads (aggregated flex-offers) that can be traded in the day-ahead market. The market schedules them and defines when their utilization occurs.

Chapter 6 considers a recently introduced product of the Elspot day-ahead market called *flexible order*. According to the flexible order product, a BRP can bid to purchase energy from Elspot. The BRP defines the energy amount needed, and the time interval when the purchased energy can be utilized. For instance, the BRP requests to purchase 2MW for one hour between 17:00 and 22:00. It places the flexible order in the market and the market defines the activation hour of the order based on the optimization of the social welfare, e.g, at 20:00. However, flexible orders have some strict requirements. In particular, only 5 flexible orders per trading day and per BRP are permitted using hourly granularity and 0.1MW volume granularity. The requested power volume must be the same for the whole requested duration. Moreover, the time interval that defines the potential activation of the order shall exceed the duration of the order for at least one hour. Therefore, market-based flex-offer aggregation is essential to produce aggregated flex-offers that fulfill the flexible order requirements.

Due to the fact that the complexity of market-based aggregation is extremely high, Chapter 6 introduces 3 heuristic aggregation techniques, namely, Dynamic Time Flexibility (DTF), Largest Profile (LP), and Dynamic Profile (DP). All the proposed techniques produce aggregated flex-offers that fulfill the flexible order requirements. Thus, the aggregated flex-offers can be transformed to flexible orders and traded in Elspot market. LP is the fastest technique among the proposed ones, while it also achieves the highest participation percentage. DTF produces the most flexible but long ones aggregated flex-offers. Hence, they are suitable for low prices occurring later than the plug-in times of the EVs. DP produces less flexible and shorter aggregated flex-offers which makes them suitable for low prices occurring close to the plug-in times of the EVs and for a short period of time. Based on real market prices, Chapter 6 financially evaluates the 3 heuristics and compares them to both the aggregation techniques proposed in Chapter 3 and a *non-realizable in practice* optimal solution where each flex-offer is directly traded in the market. Chapter 6 shows that all proposed techniques achieve more than 19% cost reduction on energy purchase compared to the charging cost according to plug-in times of the EVs. In particular, DP achieves 96,6% of the optimal cost reduction for the flex-offers that participate in aggregation.

3.6 Appendix A

Appendix A does not include any new content. It is included for completeness because the work was published individually. However, it is redundant with Chapter 5. Appendix A describes the motivation of constraint-based aggregation and a preliminary experimental setup with a small dataset of flex-offers used for the evaluation of the aggregation techniques.

4 Structure of the Thesis

The thesis is organized as a collection of papers with each chapter (Chapter 2 - Chapter 6) and appendix corresponding to an individual research work. Thus, each chapter/appendix is self-contained and can be read autonomously. Chapter 3 introduces several balance aggregation techniques. Additionally, Chapter 4 focuses on two of the balance aggregation techniques and presents an extensive experimental setup. Therefore, there can be an overlap of concepts between the two chapters regarding the balance aggregation techniques discussion. Moreover, all the research papers are based on the flex-offer concept and a scenario where energy flexibility is valuable and can be traded in the market. Hence, there can be some overlaps of concepts and texts in the introduction and preliminary sections of different chapters as they are formulated in relatively similar kind of settings. Additionally,

the layout of the papers in each chapter is slightly modified during the integration. For instance, every reference to "paper" is replaced with "chapter". The bibliography of the chapters is also integrated into one and presented after Chapter 7.

In particular, the reader could omit Appendix A (Paper A) as it is is a short version of Chapter 5.

The papers included in this thesis are listed in the following. Chapter 2 is based on Paper 1, Chapter 3 is based on Paper 2, Chapter 4 is based on Paper 3, Chapter 5 is based on Paper 4, Chapter 6 is based on Paper 5, Appendix A is based on Paper 6.

1. Emmanouil Valsomatzis, Katja Hose, Torben Bach Pedersen, Laurynas Siksnys,"Measuring and Comparing Energy Flexibilities," *Proceedings of the Workshops of the EDBT/ICDT 2015 Joint Conference (EDBT/ICDT), Belgium, Brussels*, vol. 1330, pp. 78–85, 2015

2. Laurynas Siksnys, Emmanouil Valsomatzis, Katja Hose, Torben Bach Pedersen,"Aggregating and Disaggregating Flexibility Objects," *IEEE Transactions on Knowledge and Data Engineering)*, vol. 27, no. 11, pp. 2893–2906, 2015

3. Emmanouil Valsomatzis, Katja Hose, Torben Bach Pedersen, "Balancing energy flexibilities through aggregation," *In Proceedings of the Second International Workshop on Data Analytics for Renewable for Renewable Energy Integration (DARE)*, Nancy, France, pp. 17-37, 2014

4. Emmanouil Valsomatzis, Torben Bach Pedersen, Alberto Abelló, Katja Hose,"Aggregating energy flexibilities under constraints", *Proceedings of the 7th IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Sydney, Australia, pp. 484-490, 2016

5. Emmanouil Valsomatzis, Alberto Abelló, Torben Bach Pedersen, "Trading Aggregated Flex-Offers via Flexible Orders", *DBTR series*, http://dbtr.cs.aau.dk/DBPublications/, DBTR- 38.pdf, Aalborg University, Technical Report, 2017

6. Emmanouil Valsomatzis, Torben Bach Pedersen, Alberto Abelló, Katja Hose, Laurynas Siksnys, "Towards constraint-based aggregation of energy flexibilities," *In Proceedings of the Seventh International Conference on Future Energy Systems Poster Sessions (e-Energy '16)*, Waterloo, Canada, 2 pages, 2016

Chapter 2

Measuring and Comparing Energy Flexibilities

The paper has been published in the
Proceedings of the Workshops of the EDBT/ICDT 2015 Joint Conference (EDBT/ICDT), Belgium, Brussels , vol. 1330, pp. 78–85, 2015.
The layout of the paper has been revised.

Abstract

Flexibility in energy supply and demand becomes more and more important with increasing Renewable Energy Sources (RES) production and the emergence of the Smart Grid. So-called prosumers*, i.e., entities that produce and/or consume energy, can offer their inherent flexibilities through so-called demand response and thus help stabilize the energy markets. Thus, prosumer flexibility becomes valuable and the ongoing Danish project TotalFlex [1] explores the use of prosumer flexibility in the energy market using the concept of a flex-offer [13], which captures energy flexibilities in time and/or amount explicitly. However, in order to manage and price the flexibilities of flex-offers effectively, we must first be able to measure these flexibilities and compare them to each other. In this chapter, we propose a number of possible flexibility definitions for flex-offers. We consider flexibility induced by time and amount individually, and by their combination. To this end, we introduce several flexibility*

measures that take into account the combined effect of time and energy on flex-offer flexibility and discuss their respective pros and cons through a number of realistic examples.

1 Introduction

A common challenging goal is to increase the use of energy produced by renewable energy sources (RES), such as wind and solar and at the same time reduce the CO_2 emissions. However, RES are characterized by fluctuating energy production and increased use of RES can lead to peaks (and valleys) in energy production and thus create congestion problems (or shortages) in the electric grid [34]. On the other hand, new devices such as heat pumps, increase the demand of energy and will lead to undesirable consumption peaks and a need for load shedding.

In this new energy scenery, the forthcoming Smart Grid [26] uses advanced information and communication infrastructures to activate the concept of demand side management (DSM) [45, 59]. According to DSM, the individual energy prosumers (producers and consumers) have a prominent role in the energy market due to their inherent flexibility. Flexibility can be used to mainly let the energy demand follow the energy supply and adjust the energy requirement according to energy production. The TotalFlex project explores the effect of prosumer flexibility on the energy market by introducing a new commodity using the *flex-offer* [13] concept that captures flexibilities in operating times and energy amounts of devices, as presented in the following use case.

Flex-offer use-case example. An electrical vehicle (EV) is plugged in and ready for charging at 23:00. Its battery is totally empty and it needs 3 hours to be charged. Moreover, its owner is satisfied with a minimum charging of 60% because this is sufficient enough for his needs tomorrow, e.g., going to work. Thus, we can see a flexibility regarding the energy demand of the EV due to the energy range satisfaction (60%−100%). Furthermore, the owner wants the car to be charged by 6:00 the latest, where he/she leaves home. As the battery requires 3 hours of charging, it should start being charged at 3:00 the latest. Therefore, we can also see a flexibility regarding the starting time range (23:00-3:00) of recharging the EV. The energy supplier is notified about the EV owner's energy requirement as well as the associated flexibilities in time and amount in the form of a flex-offer. Utilizing the flex-offer, the charging of the battery is scheduled (the starting time and energy demand for operating are assigned) at 1:00 because wind production will increase at that time. Furthermore, in order to ensure the owner's participation and to take advantage of the EV flexibility, the owner is offered lower energy tariff prices.

Flexibility, harnessed from many prosumers (using flex-offers) and handled according to the use-case example above, brings many advantages to society as well as to the actors participating in the energy market. Specifically, the utilization of RES is substantially increased and CO_2 emissions are reduced. Individual energy demands from prosumers are met and lower energy tariffs are offered. Marginal costs are reduced for Balanced Responsible Parties (BRPs) who trade energy. Congestion problems of Distributed System Operators (DSOs) can be handled without costly upgrades of physical grid infrastructures.

However, in order to take flexibility into consideration, we need to be able to measure *how much flexibility* is offered and identify the *kind of flexibility* offered. Only with a proper flexibility measure, different flexibility offerings can be compared together. Focusing on the use-case of flex-offers and flexibility represented by these, we now present two scenarios where measuring flexibility is particularly useful.

Scenario Nr. 1 Flex-offers must be scheduled at some point in time to be able to satisfy the prosumers' energy needs, as described in the use case example above. Flex-offer scheduling problem [80], being similar to the unit commitment problem [62], is highly complex [79], when considering a large number of flex-offers, issued for a variety of appliances such as EVs, heat-pumps, dish washers, and smart refrigerators. To reduce the complexity of scheduling, flex-offer aggregation [73] plays a crucial role by trying to reduce the number of flex-offers while retaining as much as possible of their flexibility. In addition, the TotalFlex project is further utilizing the aggregation not only to reduce the number of the flex-offers, but also to partially handle the balancing task as well [82]. For all the aggregation techniques, it is essential to quantify and then to minimize flexibility losses, and therefore a flexibility measure is needed.

Scenario Nr. 2 Consider an energy market where flex-offers are traded. It is infeasible to trade flex-offers from individual prosumers directly in the market due to their small energy amounts. It is desirable for a BRP or for any other participating actor (e.g., an Aggregator) to first aggregate flex-offers from individual prosumers (e.g., household appliances) into "larger"' *aggregated flex-offers* (e.g., at the district level) before entering the market. Consequently, only large aggregated flex-offers are allowed to be traded in the market, and, when traded, used, e.g., by a BRP to ensure balance between the physically dispatched energy and energy traded in the energy spot-market, thus avoiding imbalance penalties. In this scenario, it is preferable for aggregated flex-offers to retain as much flexibility as possible in order to obtain a better value in the energy market when they are traded. Thus a flexibility measure to quantify flexibility of various flex-offers traded as commodities is needed.

In this chapter, we employ the existing flex-offer definition [73] capturing

flexibilities regarding time and energy amount. We assume that a flex-offer is already generated and it captures the energy and associated flexibility of a single prosumer unit (e.g., an EV). Our goal, is to express the flexibility, in time, amount, and both time and amount, with a single flexibility measure that can be applied on a single flex-offer or on a set of flex-offers. Therefore, we introduce 8 possible flexibility measures that can be used to quantify flexibilities of flex-offers and to compare flex-offers together in terms of their flexibilities. These include so-called *time*, *energy*, *product*, *vector*, *time-series*, *assignments*, *absolute area-based*, and *relative area-based* flexibility measures, which treat time and energy amount either as independent or dependent flex-offer dimensions. We discuss their advantages and disadvantages using illustrative real-world based examples. Our proposed flexibility definitions can be used not only for the valuing of flex-offers, but also for evaluation of flex-offer aggregation techniques and their algorithmic implementation. In fact, depending on the application needs, the flexibility of a flex-offer can be measured using one or more of the proposed measures, each with their advantage.

The remainder of the chapter is structured as follows. In Section 3, we introduce and propose different flexibility definitions. We discuss in Section 4 about the use-case of the introduced definitions mentioning their pros and cons. We refer to related work in Section 5, and we conclude and mention our future work in Section 6.

2 Preliminaries

In this chapter, we consider the dimensions of *time* and *energy*, where time has the domain of natural numbers including zero ($\mathbb{N}_0$) and energy has the domain of integers ($\mathbb{Z}$). These assumptions are without loss of generality as we can achieve any desired finer granularity/precision of time and energy by simply multiplying their values with the desirable coefficient. Based on [73], we define a flex-offer according to Definition 1.

Definition 1. *A* flex-offer f *is a 2-tuple* $f=([t_{es}, t_{ls}], \langle s^{(1)}, \ldots, s^{(s)} \rangle)$. *The first element of the tuple denotes the start time flexibility interval where* $t_{es} \in \mathbb{N}_0$ *and* $t_{ls} \in \mathbb{N}_0$ *are the* earliest start time *and* latest start time, *respectively. The second element is a sequence of* s *consecutive slices that represents the energy profile. Each* slice $s^{(i)}$ *is an energy range* $[a_{min}, a_{max}]$, *where* $a_{min} \in \mathbb{Z}$ *and* $a_{max} \in \mathbb{Z}$. *The duration of slices is 1 time unit.*

A flex-offer also has a total minimum c_{min} and a maximum c_{max} energy constraint. The minimum constraint is smaller than or equal to the maximum one and they are lower and upper bounded by the sum of all the minimums and the sum of all the maximums of energy of the slices, respectively. If all the

energy values of a flex-offer are positive then the flex-offer represents energy consumption (positive flex-offer), e.g., a dishwasher. If all the energy values of a flex-offer are negative then the flex-offer represents energy production (negative flex-offer), e.g. a solar panel. If the energy values of a flex-offer are both positive and negative then the flex-offer represents both energy consumption and production (mixed flex-offer), e.g., a "vehicle-to-grid".

A flex-offer f can be instantiated into a so-called *assignment* of f, f_a, is a time series defining the starting time and the exact energy amounts satisfying all flex-offer constraints.

Definition 2. *An* assignment f_a *of a flex-offer* $f = ([t_{es}, t_{ls}], \langle s^{(1)}, \ldots, s^{(s)} \rangle)$ *is a time series* $\{f_a\}_{t=t_{start}}^{t_{start}+s} = \langle v^{(1)}, \ldots, v^{(s)} \rangle$ *such that:*

- $t_{es} \leq t_{start} \leq t_{ls}$
- $\forall i = 1..s : s^{(i)}.a_{min} \leq v^{(i)} \leq s^{(i)}.a_{max}$
- $c_{min} \leq \sum_{i=1}^{s} v^{(i)} \leq c_{max}$

A (valid) flex-offer assignment satisfies the constraints of a flex-offer. Specifically, for each slice of the flex-offer, the assignment has a corresponding energy value which must be within the corresponding slice energy range of the flex-offer. In addition, the sum of the energy values of a flex-offer assignment must be within the total minimum and the total maximum energy constraints of the flex-offer. Furthermore, the first non-zero energy value of the assignment that defines the actual starting time of the flex-offer must be within the start time flexibility interval of the flex-offer. A single flex-offer (typically) has several flex-offer assignments. We use the set $L(f)$ to define all (valid) flex-offer assignments. For instance, Figure 2.1 illustrates a flex-offer with four slices $f = ([1,6], \langle [1,3], [2,4], [0,5], [0,3] \rangle)$. One valid assignment of f is $f_{a1} \in L(f)$ such that $\{f_{a1}\}_{t=2}^{5} = \langle 2,3,1,2 \rangle$, shown as bold lines in Figure 2.1.

3 Flexibility Definitions and Measures

We now introduce different flexibility definitions and measures associated with a flex-offer.

3.1 Time and energy flexibility

There are two different types of flexibilities associated with a flex-offer, either derived by the starting time interval or by the energy ranges of the slices.

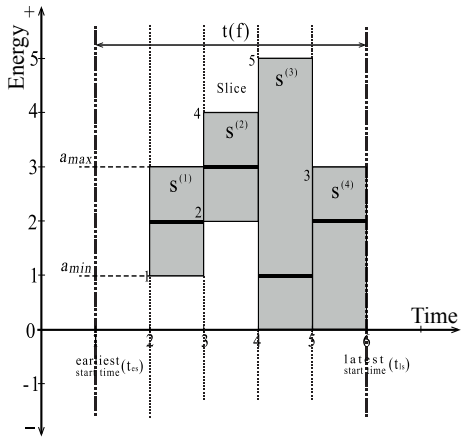


Fig. 2.1: Illustration of a flex-offer f

Based on the flexibility definitions introduced in [73], we consider the *time flexibility* $tf(f)$ of a flex-offer f to be the difference between the latest and the earliest start time of f, measured in time units, i.e., $tf(f) = f.t_{ls} - f.t_{es}$.

Example 3.1
The flex-offer f in Figure 2.1 has t_{ls}=6 and t_{es}=1, thus time flexibility is: $tf(f) = 6 - 1 = 5$.

Moreover, since the total maximum and the total minimum energy constraints impose the allowed energy range of a flex-offer, we also define *energy flexibility* of a flex-offer f to be the difference between the total maximum and the total minimum energy constraints, i.e., $ef(f) = c\_max(f) - c\_min(f)$

Example 3.2
The flex-offer f in Figure 2.1 has the sum of maximum slice values equal to 15 and the sum of minimum slice values equal to 3. Given that, $c\_max(f)$=15, $c\_min(f)$=3, and the energy flexibility of f is $ef(f)$=15−3=12.

3.2 Combined flexibility measures

As seen above, quantifying either time or energy flexibilities on their own is rather straightforward. It is more tricky to consider them in combination. Therefore, we now define and discuss several alternative measures for this.

Product flexibility. The existing definition of *total flexibility* [73] originally specified the total (joint) flexibility of a flex-offer f as the product of the time flexibility and the sum of the energy flexibilities of all the slices. However, as we have additionally introduced the total energy constraints of a flex-offer, we define the *product flexibility* of a flex-offer as follows:

Definition 3. *The* product flexibility *product_flexibility(f) of a flex-offer f is the product of the time flexibility and the energy flexibility of f, i.e., product_flexibility(f) = $tf(f) \cdot ef(f)$.*

Example 3.3
The flex-offer f in Figure 2.1 has product flexibility *product_flexibility*$(f) = 5 \cdot 12 = 60$.

Vector flexibility. Since a flex-offer is characterized by both time and energy we define the flexibility of a flex-offer to be a vector where time and energy flexibilities are the vector components.

Definition 4. *The vector flexibility vector_flexibility(f) of a flex-offer f is a vector v with 2 components. The first component of the vector is the time flexibility of f, and the second component is the energy flexibility, i.e., $v = \langle tf(f), ef(f) \rangle$.*

The total flexibility is then intuitively given by the "length" of the vector, computed using a given norm. Possible relevant norms in our two dimensions include Manhattan (L^{1-norm}) and Euclidean norm (L^{2-norm}).

Example 3.4
The flex-offer f in Figure 2.1 has vector flexibility *vector_flexibility*$(f) = \langle 5, 10 \rangle$, and we can compute its length as either $\|$*vector_flexibility*$(f)\|_1$=5+10=15 or $\|$*vector_flexibility*$(f)\|_2$=$\sqrt{(5^2 + 10^2)}$=11.180.

Time-series flexibility. A flex-offer allows multiple assignments, each expressing a possible instantiation of the flex-offer. Since every assignment of a flex-offer is a time series, the difference between two assignments is also a time series. We consider the two most dissimilar time series (assignments), *minimum* and *maximum*, defined as follows:

Definition 5. *The minimum assignment $f_a^{min}(f)$ of a flex-offer $f = ([t_{es}, t_{ls}], \langle s^{(1)}, \ldots, s^{(s)} \rangle)$ is the assignment with the first energy value positioned at the earliest starting time of f and energy values equal to the minimum slice values of f, i.e., $f_a^{min}(f) = t$, where $\{t\}_{t=t_{es}}^{t_{es}+s} = \langle f.s^{(1)}.a_{min}, \ldots, f.s^{(s)}.a_{min} \rangle$.*

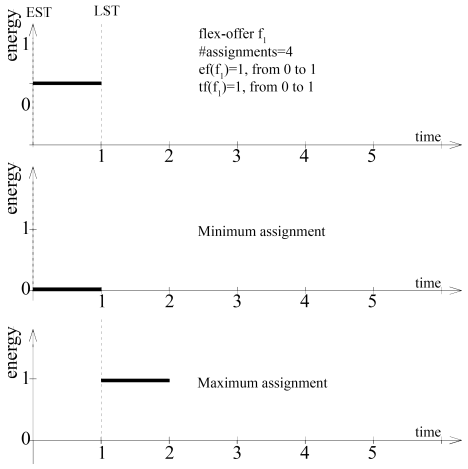


Fig. 2.2: Time series definition example with $ef(f_1) = 1$ and $tf(f_1) = 1$

Definition 6. *The maximum assignment* $f_a^{max}(f)$ *of a flex-offer* $f = ([t_{es}, t_{ls}], \langle s^{(1)}, \ldots, s^{(s)} \rangle)$ *is the assignment with the first energy value positioned at the latest starting time of* f *and energy values equal to the maximum slice values of* f*, i.e.,* $f_a^{max}(f) = t$*, where* $\{t\}_{t=t_{ls}}^{t_{ls}+s} = \langle f.s^{(1)}.a_{max}, \ldots, f.s^{(s)}.a_{max} \rangle$.

Using minimum and maximum assignments, we define *series flexibility* as follows:

Definition 7. *The time series flexibility, series_flexibility*(f)*, of a flex-offer* f *is the difference the maximum and the minimum assignments of* f *(time series), i.e., series_flexibility*(f)=$f_a^{max}(f)$-$f_a^{min}(f)$.

Since we use two dimensions, we again propose the Manhattan and Euclidean norms to quantify the difference between two assignments.

Example 3.5
Figure 2.2 illustrates a flex-offer f_1 with 1 slice, *earliest start time* $= 0$, and *latest start time* $= 1$, $f_1 = ([0,1], \langle [0,1] \rangle$, $c_{min}(f_1) = 0$, and $c_{max}(f_1) = 1$.

Flex-offer f_1 has 4 assignments, and the following minimum and maximum assignments: $\{f_{1a}^{min}(f_1)\}_{t=0}^{1} = \langle 0,0 \rangle$, $\{f_{1a}^{max}(f_1)\}_{t=0}^{1} = \langle 0,1 \rangle$. Let the difference between $f_{1a}^{max}(f_1)$ and $f_{1a}^{min}(f_1)$ be f_{d1} so that f_{d1}=$f_{1a}^{max}(f_1)$-$f_{1a}^{min}(f_1)$. In this example $\{f_{d1}\}_{t=0}^{1} = \langle 0,1 \rangle$, L^{1-norm}, $|\{f_{d1}\}_{t=1}^{1}|_1 = 1$, and L^{2-norm}, $|\{f_{d1}\}_{t=1}^{1}|_2 = 1$. According to both L^{1-norm} and L^{2-norm}, *series_flexibility*(f_1)=1.

Assignment flexibility. As mentioned in Section 2, a flex-offer allows a

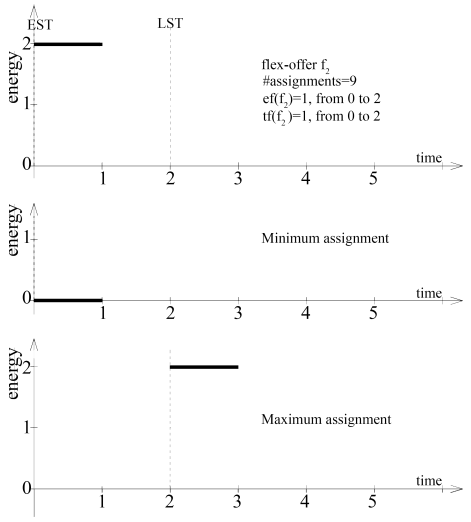


Fig. 2.3: Number of assignments example with $ef(f_2) = 2$ and $tf(f_2) = 2$

number of possible assignments. The number of possible assignments directly depends on time and energy flexibility and is the number of the combinations between all the allowed amount and time values of all its slices. Therefore, we use the number of possible assignments as a combined measure induced by both time and amount flexibility.

Definition 8. *We define* assignment flexibility*, assignment_flexibility(f), of a flex-offer $f = ([t_{es}, t_{ls}], \langle s^{(1)}, \ldots, s^{(s)} \rangle)$ to be the number of all possible assignments of f, i .e., assignment_flexibility(f)=*

$$=(t_{ls}-t_{es}+1)\cdot\prod_{i=1}^{s}(s^{(i)}.a_{max}-s^{(i)}.a_{min}+1).$$

Example 3.6
Flex-offer $f_2 = ([0,2], \langle [0,2] \rangle)$ in Figure 2.3 has $t_{ls}-t_{es}+1=3$ and since it has one slice $s^{(1)}.a_{max}-s^{(1)}.a_{min}+1=3$. Thus, f_2 has 9 assignments in total.

Absolute area-based flexibility. Absolute area-based flexibility is based on the area that all flex-offer assignments jointly cover, considering all of their possible values of start time and energy. As a basis for calculating this area, we consider a two-dimensional (time and energy) grid $G = \mathbb{N}_0 \times \mathbb{Z} = \{(t,e) : t \in time, e \in energy\}$, in which the x axis corresponds to discretized time and the y axis to discretized energy. Cells of the grid are identified by their lower left coordinates. For instance, the cell with identifier $(0,0)$ has the following corner coordinates: $(0,0)$, $(0,1)$, $(1,0)$, $(1,1)$.

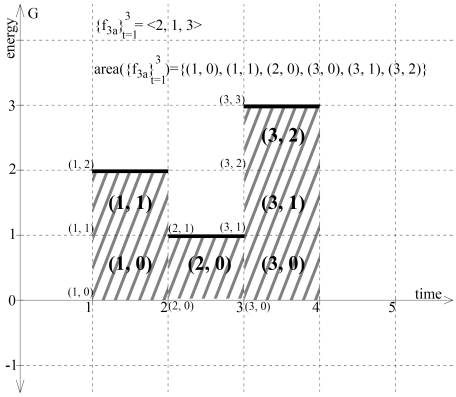


Fig. 2.4: Area of the assignment $\{f_{3a}\}_{t=1}^{3} = \langle 2, 1, 3 \rangle$

First, we define the area of a single flex-offer assignment.

Definition 9. *The area of an assignment f_a of a flex-offer f, denoted as $area(f_a)$, is the set of cells that falls between the f_a energy values and the X-axis of the grid.*

Example 3.7
Given an *assignment* of flex-offer f_3, $\{f_{3a}\}_{t=1}^{3} = \langle 2, 1, 3 \rangle$ the area is as follows: $area(\{f_{3a}\}_{t=1}^{3}) = \{(1,0), (1,1), (2,0), (3,0), (3,1), (3,2)\}$, which is represented by the hatched cells in Figure 2.4.

This area represents the total assigned amount of a single flex-offer. However, multiple assignments with different areas are possible for a flex-offer. The total coverage of all these assignment areas gives us the area of the flex-offer flexibility. This joint area expresses all the possible amounts at all the possible time instances that a flex-offer could have. Furthermore, we are interested in the size (a numerical value) of this area of flexibility. To specify this, we additionally take into account the minimum total energy constraint *c_min*, which is applicable to all assignments and is thus considered inflexible.

Definition 10. *The* absolute area-based flexibility *of a flex-offer f is the difference between the size of the total area covered by all the assignments of f and the total minimum constraint of f: absolute_area_flexibility=$|\bigcup_{as\_f \in L(f)} area(as\_f)| - c\_min(f)$*

Example 3.8
Figure 2.5 illustrates the flex-offer $f_4 = ([0,4], \langle [2,2] \rangle$, $c_{min}(f_4)$=2, and $c_{max}(f_4)$=2. Flex-offer f_4 has 5 different assignments and each

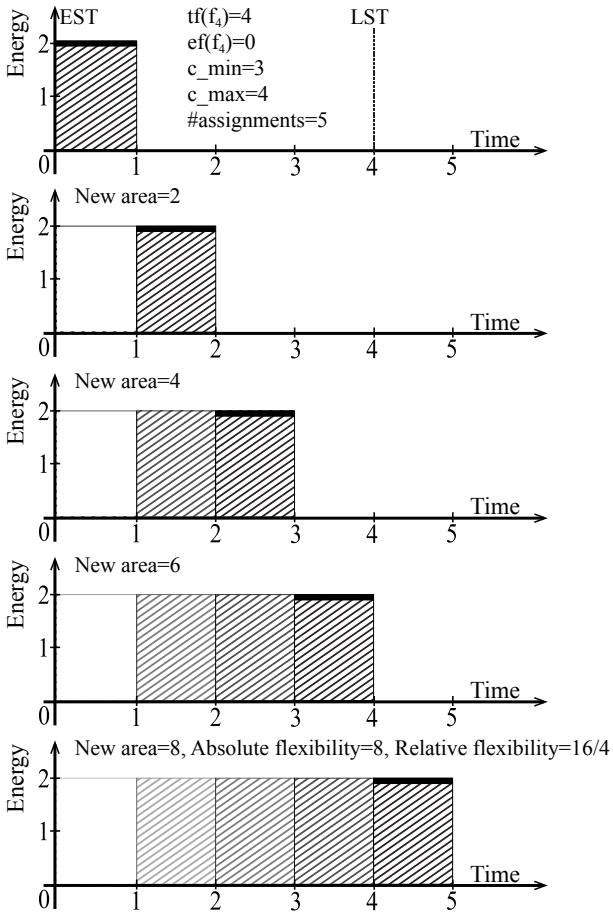


Fig. 2.5: Absolute and relative area-based flexibility of the flex-offer f_4

one covers an area of two cells, see Figure 2.5. Flex-offer f_4 has *absolute_area_flexibility*(f_4)=10−2=8.

Example 3.9
Figure 2.6 illustrates the flex-offer $f_5 = (\langle[0,4],\langle[1,1],[2,2]\rangle$, $c_{min}(f_5)$=3, and $c_{max}(f_5)$=3. Flex-offer f_5 has 5 different assignments and each one covers an area of three cells, see Figure 2.6. Flex-offer f_5 has *absolute_area_flexibility*(f_5)=10−2=8.

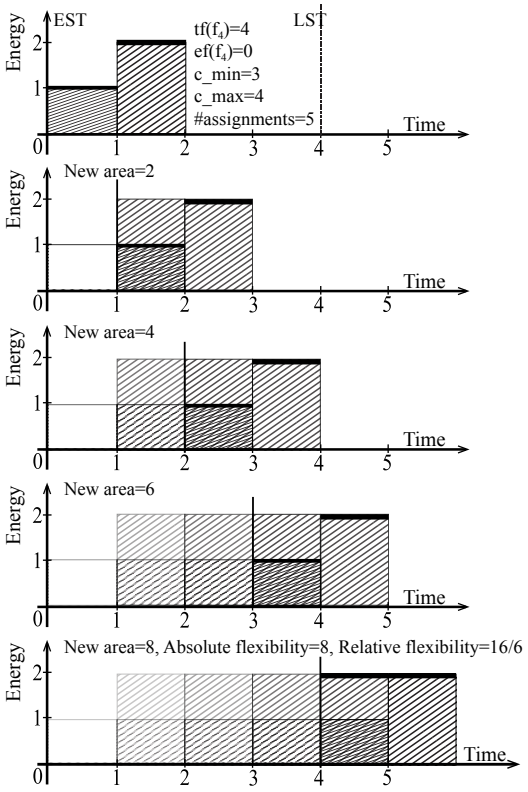


Fig. 2.6: Absolute and relative area-based flexibility of the flex-offer f_5

Relative area-based flexibility. For most of the presented flexibility measures (incl., absolute area-based flexibility), the value of the flexibility depends on the actual amounts specified in the flex-offer. However, in cases when we need to compare flex-offers of different sizes in terms of amount, we need a size-independent measure. For these cases, we propose a *relative area-based flexibility*.

Definition 11. *The relative area-based flexibility of a flex-offer f is equal to the absolute flexibility divided by the average of the energy total constraints of f:*

$relative\_area\_flexibility(f) = \frac{2*absolute\_area\_flexibility(f)}{|c\_min(f)|+|c\_max(f)|}, \quad |c\_min(f)| + |c\_max(f)| \neq 0$

Example 3.10
Flex-offer $f_4 = ([0,4], \langle [2,2] \rangle, c_{min}(f_4)=2, c_{max}(f_4)=2$, shown in Figure 2.5, has $relative\_area\_flexibility(f_4) = \frac{2*8}{|2|+|2|}=4$. Flex-offer $f_5 =$

$([0,4],\langle[1,1],[2,2]\rangle)$, $c_{min}(f_5)$=3, $c_{max}(f_5)$=3, shown in Figure 2.6, has *relative_area_flexibility*$(f_5)=\frac{2*8}{|3|+|3|}$=16/6.

4 Discussion

In this section, we discuss the pros and cons of the proposed flexibility measures, and scenarios in which we can use each of these measures.

Product flexibility. The product flexibility measure, defined in Definition 3, is only applicable in cases when a flex-offer f has positive time and energy flexibilities, i.e., $tf(f) > 0$ and $ef(f) > 0$. In cases, when either the time or the amount flexibility is equal to zero, the value of the product flexibility is also equal to zero. As the flex-offer is still flexible in the other dimension (time or energy), this measure is not particularly accurate.

Example 4.1
Flex-offer f_x=$([2,8],\langle[5,5]\rangle)$ has $tf(f)$=6, $ef(f_x)$=0, and *product_flexibility*$(f_x) = 6 \cdot 0 = 0$. Moreover, two flex-offers f_x=$([1,3],\langle[1,5]\rangle)$ and f_y=$([1,3],\langle[101,105]\rangle)$ have equal product flexibility values, i.e., *product_flexibility*(f_x)=*product_flexibility*(f_y)=8, even if the minimum energy requirement of f_y is more than 100 times greater than the minimum energy requirement of f_x.

Furthermore, product flexibility does not take into account individual slice energy requirements. It relies only on total energy requirements (c_{min} and c_{max}). Nevertheless, Definition 3 can still be applicable in scenarios where the flex-offer represents production, consumption, or both, as long as there are no mixed flex-offers. Additionally, it can be generalized for sets of flex-offers. To compare two or more sets of flex-offers, we should sum the product flexibilities of the flex-offers in each set.

Vector flexibility. Vector flexibility measure, as defined in Definition 4, can be applicable to either individual flex-offers or sets of flex-offers, like the product flexibility. However, unlike the product flexibility, it can capture the flexibility in cases where either time or energy flexibility of a flex-offer is equal to zero. Furthermore, it is independent of the sign of the energy values of the slices of a flex-offer. In particular, it can express flexibility of flex-offers that represent either energy production, consumption, or both. Like the product flexibility, it does not take into account individual slice energy requirements, solely relying on total energy requirements (c_{min} and c_{max}). Lastly, vector flexibility does not take into account the actual values of energy

("size of the flex-offer"), but, instead, captures only the difference between energy bounds.

Example 4.2
The two flex-offers f_x=($[1,3]$, $\langle[1,5]\rangle$) and f_y=($[1,3]$, $\langle[101,105]\rangle$) from Example 4.1 have the same vector flexibility irrespectively of the used norm, even if the minimum energy requirement of f_y is more than 100 times greater than the minimum energy requirement of f_x. Specifically, $\|vector\_flexibility(f_x)\|_1$=$\|vector\_flexibility(f_y)\|_1$=6 according to the Manhattan norm, and $\|vector\_flexibility(f_x)\|$=$\|vector\_flexibility(f_y)\|_2$=4.472 according to the Euclidean norm.

Time-series flexibility. Norms such as Manhattan and Euclidean, applicable with time-series flexibility (see Definition 7), do not take into account the temporal structure of the time series [46] and thus cannot capture the joint effect of time and energy flexibilities. As a result even if time-series captures both time and energy, the norms applied on a difference between time-series can capture only energy flexibility. However, the time-series definition can be applied on positive, negative, and mixed flex-offers, as well as on sets of flex-offers – by computing the sum of time-series flexibilities of the flex-offers in the set.

Example 4.3
As mentioned in Example 3.5 , flex-offer $f_1 = ([0,1], \langle[0,1]\rangle$, $c_{min}(f_1) = 0$, and $c_{max}(f_1) = 1$ results in time series $\{f_{d1}\}_{t=0}^{1} = \langle 0,1\rangle$, and its norms are as follows: L^{1-norm}, $|\{f_{d1}\}_{t=1}^{1}|_1 = 1$, and L^{2-norm}, $|\{f_{d1}\}_{t=1}^{1}|_2 = 1$. However, another flex-offer $f_1' = ([0,10], \langle[0,1]\rangle$, $c_{min}(f_1') = 0$, and $c_{max}(f_1') = 1$ with 10 times greater time flexibility than f_1 results in a similar time series $\{f_{d1}'\}_{t=0}^{1} = \langle 0,0,0,0,0,0,0,0,0,0,1\rangle$ with identical norms: L^{1-norm}, $|\{f_{d1}'\}_{t=1}^{1}|_1 = 1$, and L^{2-norm}, $|\{f_{d1}'\}_{t=1}^{1}|_2 = 1$.

Assignment flexibility. Assignment flexibility, as defined in Definition 8, considers only the number of flex-offer assignments, and this number is independent of the actual values of the time and energy bounds. The limitation of this measure is that energy flexibility has an exponential impact on the number of the assignments, i.e., the number of assignments increases exponentially when energy flexibility is increased. In comparison, the number of flex-offer assignments increases linearly when time flexibility is increased. Thus, this measure favors energy flexibility over time flexibility. Moreover, assignment flexibility, as defined in Definition 8, does not take into account

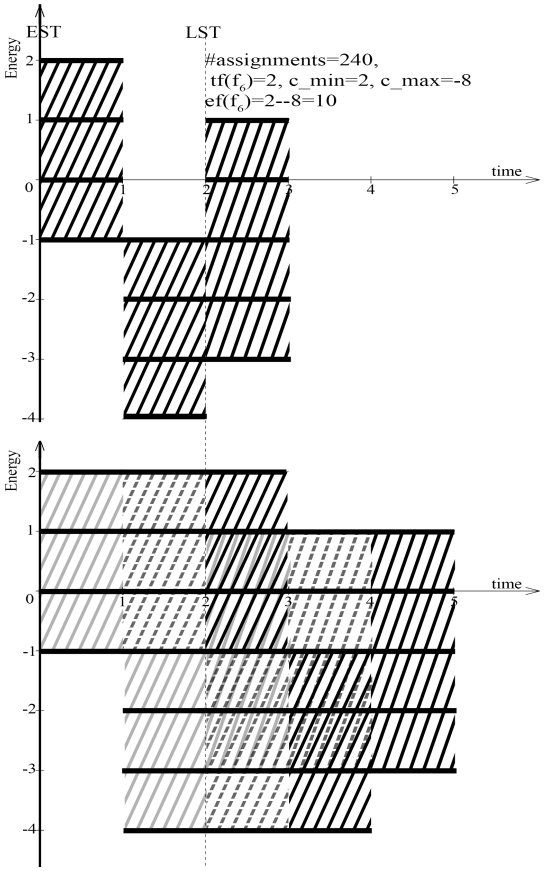


Fig. 2.7: Number of assignments example, flex-offer f_6

the total energy requirements (c_{min} and c_{max}), and gives the same values for flex-offers with the same time and amount flexibilities, but differing in energy amounts. Furthermore, it can express flexibility of flex-offers that represent either production, consumption, or both. It can be used to compare individual flex-offers and to compare sets of flex-offers by counting the number of possible assignments for the whole set.

Example 4.4
The flex-offer f_2 with $tf(f_2)$=$ef(f_2)$=2, shown in Figure 2.3, has 9 possible assignments. If $tf(f_2)$ were 0, flex-offer f_2 would have 3 possible assignments, but if $ef(f_2)$ were 0, f_2 would have 2 possible assignments. The flex-offer f_6 with $tf(f_6)$=2 and $ef(f_6)$=10, shown in Figure 2.7, has 240 assignments. If $tf(f_6)$ were 0, f_6 would have 80 assignments, but if $ef(f_6)$ were 0, f_6 would have 3 assignments.

Absolute and relative area-based flexibility. Both the absolute and relative area-based flexibility measures (Definitions 10–11) can be used to capture the joint effect of time and energy flexibilities. However, the absolute area-based flexibility measure should only be used for (pure) consumption flex-offers only, as its value is adjusted using the total minimum energy constraint (c_{min}), which is meaningful only for the consumption case where amounts are positive. For the production flex-offer case, where amounts are negative, the total maximum energy constraint (c_{max}) should be used instead. However, for cases when the flex-offer represents both production and consumption, this flexibility measure is not feasible.

Example 4.5
For instance, flex-offer f_4=([0,2],⟨[−1,2],[−1,−4],[−3,1]⟩) in Figure 2.7 has $c_{min}(f_6)$=−8 and $c_{max}(f_6)$=2, but neither of them expresses the lower or upper bounds of the area, jointly covered by the assignments of f_6. In this case, *absolute_area_flexibility*(f_6)=24−(−8)=32 and *relative_area_flexibility*(f_6)= $\frac{2*32}{|8|+|2|}$=6.4.

On the other hand, both absolute and relative area-based flexibility measures can be used to compare individual flex-offers. Only absolute area-based flexibility can be used to compare the total absolute flexibility of two or more sets of flex-offers, e.g., by summing up the individual absolute area-based flexibility values of the flex-offers in the sets. To assess the relative flexibility for a set of flex-offers, the sum of relative flexibilities is not meaningful, instead the average relative flexibility could be used.

All the flexibility measures can be applied for both individual flex-offers and sets of flex-offers to compare their underlying flexibility. However, as we see in Table 2.1, which summarizes the characteristics of all the proposed flexibility definitions, each flexibility measure has specific characteristics and should be used under specific circumstances only. For example, the product flexibility measure cannot properly capture flexibility unless both time and amount flexibility is exhibited. The time-series flexibility measure captures only flexibility induced by energy flexibility. Only the absolute and relative area-based flexibility measures take into account the amount values (size) of the flex-offers. However, the absolute and relative area-based flexibility measures have problems expressing the flexibility of mixed flex-offers.

Application Scenarios. There are 2 major scenarios (see Section 1) where the different measures can be applied. In Scenario 1, the goal of aggregation is to reduce the input complexity of scheduling and retain as much flexibility of flex-offers as possible. In this scenario, measures that capture flexibility

Characteristics	Flexibility Measures			
	Time	Energy	Product	Vector
Captures time	Yes	No	No	Yes
Captures energy	No	Yes	No	Yes
Captures time & energy	No	No	Yes	Yes
Captures size	No	No	No	No
Captures positive flex-offers	Yes	Yes	Yes	Yes
Captures negative flex-offers	Yes	Yes	Yes	Yes
Captures Mixed flex-offers	Yes	Yes	Yes	Yes
Single Value	Yes	Yes	Yes	(Yes)

Characteristics	Flexibility Measures			
	Time-series	Assignments	Abs. Area	Rel. Area
Captures time	No	Yes	Yes	Yes
Captures energy	Yes	Yes	Yes	Yes
Captures time & energy	No	Yes	Yes	Yes
Captures size	No	No	Yes	Yes
Captures positive flex-offers	Yes	Yes	Yes	Yes
Captures negative flex-offers	Yes	Yes	Yes	Yes
Captures Mixed flex-offers	Yes	Yes	No	No
Single Value	Yes	Yes	Yes	Yes

Table 2.1: Flexibility definitions characteristics.

induced by both time and energy, e.g., product flexibility and assignments flexibility, are qualified. Measures that capture only time or energy flexibility, such as time-series flexibility, are not appropriate for Scenario 1. However, in cases where aggregation handles the balancing task as well, measures that capture flexibility of mixed flex-offers are needed since the aggregated flex-offers might be mixed ones. Thus, measures that are not suitable for mixed flex-offers, i.e., absolute and relative area-based flexibility, are inappropriate to express flexibility. Instead, measures that capture flexibility of mixed flex-offers such as vector and assignments flexibility, are qualified. In Scenario 2, where an energy market actor (e.g., an Aggregator) trades flex-offers as commodities, measures capturing only time or energy can be used. The reason is because an Aggregator might handle flex-offers from specific appliances that are characterized only by time or energy flexibility. Thus, the time-series measure, the time and energy flexibility measures, and the product flexibility measure are appropriate. In cases where an Aggregator wants to explore and evaluate the potentials of achieving a local balance and handle a power capacity limitation, measures for mixed flex-offers are more appropriate. However, only absolute and relative area-based flexibilities take into account the size of a flex-offer, but they cannot be applied on mixed flex-offers. Therefore, a combination of measures that includes the absolute or the relative area-based flexibility can be used to handle these more complex cases. *Weighting* is one way of combining different flexibility measures and balancing their

influences to fulfill specific characteristics mentioned in Table 2.1.

5 Related work

Flexibility in energy supply and demand has a prominent role in the Smart Grid domain, and, among others within this domain, can be associated with distributed generation, load management and demand side management [45]. Many definitions of flexibility have been proposed, but a formal universal definition is still pending [65]. Some proposed measures of flexibility focus on operational aspects and take into account transmission constraints [16], while others are based on time shifting of loads [66]. Furthermore, there has been proposed categorizations of power units based on their characteristics, taking into consideration their qualities and capabilities to dispatch power and solve balancing issues [65].

In comparison, this chapter proposes and discusses specific measures to quantify flexibility in energy supply and demand, namely in the units connected to the Smart Grid such as electric vehicles, solar panels, wind turbines, and refrigerators. We use the existing definition of a flex-offer [73], which is a generic model for representing flexibility and adjust it for the cases of energy consumption, production, and both consumption and production. The proposed measures can be applied on individual electrical units and on sets of units as well, e.g., when solving the unit commitment problem [62] or tackling balancing or congestion problems occurring in the grid [80].

6 Conclusion and future work

In this chapter, we proposed and explored 8 measures for quantifying flexibility in demand and supply based on the generic flexibility model of a flex-offer, capturing the energy behavior of units connected to the Smart Grid. We identified the independent flexibilities of time and energy and proposed a number of combined measures – *product*, *vector*, *time-series*, *assignments*, *absolute area-based*, and *relative area-based* – which take both time and energy into account. These measures can be used to compare the flexibility of individual flex-offers as well as sets of flex-offers. We demonstrated and discussed the impact of the proposed measures using elaborate graphical examples. We concluded through a discussion that such single-value measures can be used to express the flexibility of the units connected to the Smart Grid. However, none of the measures have all the desirable characteristics. Instead, each measure has specific characteristics and can be used in specific circumstances, all discussed in the chapter.

In future work, we will examine the use of the suggested measures for flex-offer aggregation algorithms, including those that partially address the

energy balancing problem and consider electric grid constraints. The proposed flexibility measures will be added to the constraints and/or objective functions of these aggregation algorithms, performing aggregation jointly with flexibility optimization. We will also experimentally evaluate the flexibility measures and their effect on the scheduling process in different scenarios. Moreover, we will extend the current proposals to new types of measures capturing more aspects of flexible electrical loads.

Chapter 3

Aggregating and Disaggregating Flexibility Objects

The paper "Aggregating and Disaggregating Flexibility Objects" has been published in the
IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 11, pp. 2893–2906, 2015.
The layout of the paper has been revised.
DOI: 10.1109/TKDE.2015.2445755

Abstract

In many scientific and commercial domains we encounter flexibility objects, i.e., objects with explicit flexibilities in a time and an amount dimension (e.g., energy or product amount). Applications of flexibility objects require novel and efficient techniques capable of handling large amounts of such objects while preserving flexibility. Hence, this chapter formally defines the concept of flexibility objects (flex-objects) and provides a novel and efficient solution for aggregating and disaggregating flex-objects. Out of the broad range of possible applications, this chapter will focus on

smart grid energy data management and discuss strategies for aggregation and disaggregation of flex-objects while retaining flexibility. This chapter further extends these approaches beyond flex-objects originating from energy consumption by additionally considering flex-objects originating from energy production and aiming at energy balancing during aggregation. In more detail, this chapter considers the complete life cycle of flex-objects: aggregation, disaggregation, associated requirements, efficient incremental computation, and balance aggregation techniques. Extensive experiments based on real-world data from the energy domain show that the proposed solutions provide good performance while satisfying the strict requirements.

1 Introduction

Many scientific and commercial domains deal with flexibilities in terms of *time* and *amount*. In the *smart-grid* domain, for instance, the EU FP7 research project MIRABEL [13] and the ongoing Danish project TotalFlex (`www.totalflex.dk`) exploit *time* and *energy amount* flexibilities of various electricity consumers and producers to increase the share of renewable energy sources (RES) such as wind-turbines and solar-panels.

Such flexibilities can be captured by *flexibility objects* (in short, *flex-objects*), specifying (1) *how much energy* is needed, (2) *when* it is needed, and (3) what the *tolerated flexibilities* are regarding *time* (e.g., between 9 PM and 5 AM) and *energy amount* at consecutive time intervals (e.g., between 2 and 4 kWh in the first hour and 3 and 5 kWh in the second hour). By interconnecting thousands of European Electricity Market participants – *consumers*, *producers*, *aggregators*, and *Balance Responsible Parties* (BRPs) – using a large-scale Energy Data Management System [13], these flex-objects undergo the cycle of *aggregation*, *instantiation*, and *disaggregation*, depicted in Figure 3.1. Generated by individual consumers/producers (e.g., a smart-home automation system for charging the battery of an electric vehicle (EV)), flex-objects ($f_1, \ldots, f_4$) are sent to aggregators, which first group similar (time-overlapping) flex-objects (g_1 and g_2) and then aggregate them into larger "macro" flex-objects (f_{a1} and f_{a2}). Utilizing such "macro" flex-objects, the BRP schedules (globally balances) flexible loads of the "macro" flex-objects to match the forecasts of inflexible consumption and RES production. During the scheduling, the "macro" flex-objects are transformed (*instantiated*) into so-called "macro" *fix-objects* (f^x_{a1} and f^x_{a2}) having concrete values assigned for the time and amount dimensions within the flex-object flexibility intervals. Then, the aggregators disaggregate the "macro" fix-objects into "micro" fix-objects ($f^x_1, \ldots, f^x_4$), which specify the exact time and magnitude of energy that has to be consumed (or produced) by the consumers/producers. If electricity is consumed and produced according to the fix-objects, consumers/producers are ultimately rewarded based on the flexibility they offer.

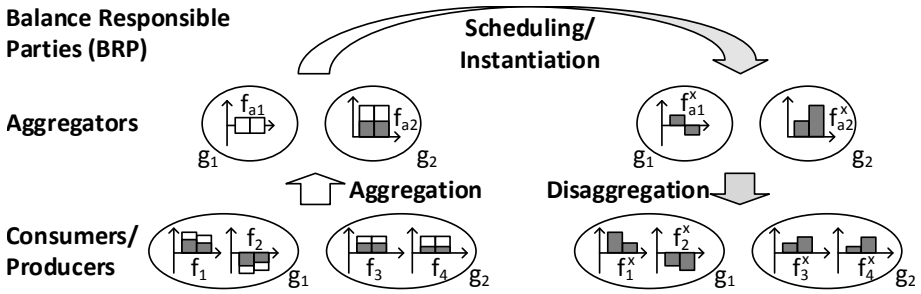


Fig. 3.1: The lifecycle of flex-objects

In this cycle, the aggregation and disaggregation operations have a number of associated requirements. First, aggregation must reduce the total number of flex-objects to lower the complexity of solving the scheduling problem. It should retain as much flexibility as possible in the aggregated flex-objects, while not introducing more flexibility than that of the non-aggregated flex-objects. Second, the aggregation must produce aggregated flex-objects conforming to individual BRP requirements, setting the limits for, for instance, the number, magnitude of amount, and covered time window, of the flex-objects. Third, aggregation must be able to support rapid and continuous flex-object additions and removals issued by consumers/producers. Fourth, to reduce the risk of congestions (caused by RES and EVs), it is very important to balance demand and supply locally by aggregators prior to scheduling (global balancing) by BRPs. Thus, the aggregation must be able to perform local balancing when producing aggregated flex-objects. Figure 3.1 demonstrates such "balance aggregation" using the flex-object f_{a1}, which represents the joint energy needs/offers from a consumer issuing the flex-object f_1 and a producer issuing the flex-object f_2. Lastly, the disaggregation of fix-objects must be feasible in all these cases. For use in the smart-grid (MIRABEL/TotalFlex) and other domains, no flex-object aggregation/disaggregation solution satisfying all these requirements exists.

Considering this need, the contributions of this chapter are as follows. First, we formally define flex-objects, fix-objects, measures to quantify flexibility, aggregation and disaggregation functions, and associated requirements. Second, we present a basic technique to aggregate many flex-objects into a single aggregated flex-object and to disaggregate a fix-object into many fix-objects. Third, we present an advanced technique that generates many aggregated flex-objects and that is able to disaggregate fix-objects. Here, the aggregation is performed incrementally. Given a sequence of flex-object deltas, our technique partitions flex-objects into disjoint groups of similar flex-objects. The partitioning is performed in two steps – grid-based *grouping* and *bin-packing* – ensuring that flex-objects in the groups are similar enough

and that the groups themselves fulfil a given (aggregate) criterion. After bin-packing, similar flex-objects from different groups are merged into aggregated flex-objects, which are finally returned as output in the form of a sequence of aggregated flex-object deltas. Fourth, we present five techniques for balance aggregation. For all techniques, we provide detailed algorithms in pseudo-code along with computational complexity estimates. Finally, we discuss the results of our extensive set of experiments with both regular and balance aggregation and show that our solution scales well and handles aggregation and disaggregation efficiently and effectively.

This chapter significantly extends our previous work [73] by providing (1) a more concise problem formulation, also including balance aggregation, (2) detailed aggregation/disaggregation algorithms with pseudo-code, (3) five concrete balance aggregation algorithms, (4) computational complexity analyses, and (5) a set of comprehensive balance aggregation experiments.

The remainder of the chapter is structured as follows. Section 2 formally defines all relevant concepts. Sections 3–4 describe basic and advanced flex-object aggregation/disaggregation techniques. Balance aggregation techniques are introduced in Section 5. Section 6 describes the experimental evaluation, while Section 7 discusses related work. Finally, Section 8 concludes the chapter and discusses future work. The Appendix A of the chapter provides additional proofs and the detailed complexity analyses of the proposed algorithms.

2 Problem Formulation

We now formalize the problem of aggregating and disaggregating flexibility objects. Our formalization includes (1) definitions of flex-object concepts, (2) measures for quantifying flexibility, and (3) functions for *aggregation* and *disaggregation* and their associated constraints.

Let e be an *entity* from a *domain* D utilizing a *resource* r; the utilization of the resource is characterized by a *continuous amount* over a *discrete time*. An example of an entity (e) is an electric vehicle (EV) that is connected to a charging station (D) and consuming specific amounts of electrical energy (r) at different hours to fully charge its battery. We define flexibility in how the resource can be utilized (i.e., flexible utilization) using a so-called *flex-object*, which is a multidimensional object capturing two aspects: (1) a *time flexibility interval* and (2) an *amount profile* with a sequence of consecutive *slices*, each defined by minimum and maximum bounds of the amount.

Definition 12. *A* flex-object f *is a tuple* $f = ([t_{es}, t_{ls}], p)$ *where* $[t_{es}, t_{ls}]$ *is the* start time flexibility interval *and* p *is the* amount profile. *The time is discretized into equal-sized units, e.g., 15 minute intervals. Thus, we use* $t_{es} \in \mathbb{N}$ *to specify the* earliest start time *and* $t_{ls} \in \mathbb{N}$ *to specify the* latest start time. *The* p *is a sequence*

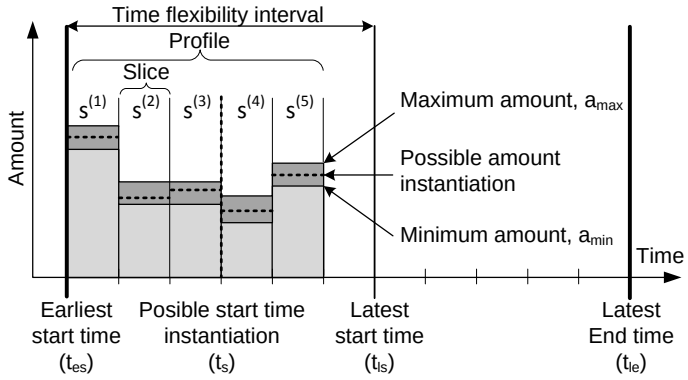


Fig. 3.2: A generic flex-object

of slices $\langle s^{(1)}, \ldots, s^{(m)} \rangle$*, where a* slice $s^{(i)}$ *is a continuous range* $[a_{min}, a_{max}]$ *defined by a* minimum amount a_{min} *and a* maximum amount a_{max}*. The extent of* $s^{(i)}$ *in the time dimension is 1 unit. Hence, a flex-object's* profile duration *is computed as* $p_{dur}(f) = |f.p|$*, its* earliest end time *as* $t_{ee}(f) = f.t_{es} + p_{dur}(f)$*, and its* latest end time *as* $t_{le}(f) = f.t_{ls} + p_{dur}(f)$*.*

Figure 3.2 depicts an example of a flex-object specifying the intended consumption of electricity of a single EV connected to a charging station, where the EV, electricity, and charging station are the above presented entity e, resource r, and domain D, respectively. The flex-object has a profile with five slices: $\langle s^{(1)}, \ldots, s^{(5)} \rangle$. Every slice is represented by a bar in the figure. The top of the light-shaded bar represents the minimum amount value (a_{min}) and the top of the dark-shaded bar represents the maximum amount value (a_{max}).

Depending on the values of the amount bounds, we distinguish the following three types of flex-objects:

Definition 13. *A flex-object* f *is called* positive *if* $\forall s \in f.p : s.a_{min} \geq 0$*. A flex-object* f *is called* negative *if* $\forall s \in f.p : s.a_{max} < 0$*. A flex-object* f *is called* mixed *if it is neither* positive *nor* negative*, i.e., if* $\exists s_n, s_p \in f.p : s_n.a_{min} < 0 \wedge s_p.a_{max} \geq 0$*.*

For example, a *positive* flex-object can be used to describe the consumption profile and associated flexibilities of a heat-pump heating a house. A *negative* flex-object can be used for various production-only power systems, e.g., solar panels, wind-turbines, or power generators. A mixed electrical behaviour is exhibited by more advanced power systems, e.g., EVs whose batteries can be charged or discharged. A *mixed* flex-object can be used to capture such specific behaviour.

We distinguish two types of flexibilities associated with f. The *time flexibility* $tf(f)$ is the difference between the latest and earliest start time, i.e.,

$tf(f) = f.t_{ls} - f.t_{es}$. Similarly, the *amount flexibility* $af(f)$ is the sum of the differences between the amount bounds of all slices in f's profile, i.e., $af(f) = \sum_{s \in f.p} s.a_{max} - s.a_{min}$.

A flex-object with time and amount flexibilities equal to zero is called a *fix-object*. In this case, the fix-object $f = ([t_{es}, t_{ls}], p)$ is such that $t_{es} = t_{ls}$ and $\forall s \in f.p : s.a_{min} = s.a_{max}$. A fix-object may or may not be a *valid instance* of a given flex-object.

Definition 14. *A valid* instance *(instantiation) of a flex-object* $f = ([t_{es}, t_{ls}], \langle s^{(1)}, \ldots, s^{(m)} \rangle)$ *is a fix-object* $f^x = ([t_s, t_s], \langle s_x^{(1)}, \ldots, s_x^{(m)} \rangle)$ *such that* $t_{es} \leq t_s \leq t_{ls}$ *and* $\forall i = 1..m : s^{(i)}.a_{min} \leq s_x^{(i)}.a_{min} = s_x^{(i)}.a_{max} \leq s^{(i)}.a_{max}$. *We use the notation* $f^x \triangleright f$ *to denote that a fix-object* f^x *is a valid instance of a flex-object* f. *We refer to* t_s *as the (assigned)* start time.

In the general case, there is an infinite number of possible instances of a flex-object ($f^x \triangleright f$). One possible instance is shown as the dotted line in Figure 3.2. To quantify the size of the possible instantiation space (flexibility), we define *total flexibility* as follows:

Definition 15. *The total flexibility of a flex-object* f *is the product of* time *and* amount flexibility, *i.e.,* $flex(f) = tf(f) \cdot af(f)$.

A flex-object with a larger *total flexibility* represents a larger variety of instantiations compared to a flex-object with lower *total flexibility*. Consider a flex-object f=$([2,7], \langle s^{(1)}, s^{(2)} \rangle)$ where $s^{(1)}$=$[10,20]$ and $s^{(2)}$=$[18,30]$. The time flexibility of f is equal to $7-2$=5. The amount flexibility $af(f)$ is equal to $(20-10)+(30-18)$=22. Hence, the total flexibility of f is equal to 110, and it is considered "more flexible" than, for example, a flex-object with a total flexibility of 100.

We now generally define the concepts of flex-object *aggregation* and *disaggregation*, and formulate associated constraints and requirements. Later, in Sections 3–5, we elaborate on how these operations are performed.

Definition 16. *Flex-object* aggregation *generalizes the joint flexible utilization of a resource within the domain. This is performed by a function* $AGG(F)$ *that takes a set of flex-objects* F *and produces a set of flex-objects* A, $|A| \leq |F|$. *Every* $f_a \in A$ *is called an* aggregated flex-object.

Definition 17. *Flex-object* disaggregation *de-generalizes (coarse) instances of aggregated flex-objects by generating (detailed) instances of non-aggregated flex-objects. This is performed by a function* $DAGG(F, A, A^X)$ *where* A^X *is a set of fix-objects that are instances of the aggregated flex-objects from* A *such that* $A = AGG(F) \wedge \forall f_a \in A : \exists f_a^x \in A^X$ *and* $f_a^x \triangleright f_a$. *The function produces a set of non-aggregated flex-object instances,* F^X, *such that* $\forall f \in F : \exists f^x \in F^X$ *where* $f^x \triangleright f$.

There exist many different pairs of aggregation and disaggregation functions (AGG, $DAGG$). However, we are primarily interested in those pairs that "correctly summarize" amounts (and their allocations in time). We formulate this as the following requirement.

Amount conservation requirement. For any given F, A^X, and $F^X = DAGG(F, AGG(F), A^X)$, the following equality must hold for every time instance $t \in \mathbb{N}$:

$$\sum_{f_a^x \in A^X} \sum_{i=1}^{|f_a^x.p|} [f_a^x.p[i].a_{min} | t = f_a^x.t_{es} + i] = \tag{3.1}$$

$$\sum_{f^x \in F^X} \sum_{i=1}^{|f^x.p|} [f^x.p[i].a_{min} | t = f^x.t_{es} + i]. \tag{3.2}$$

The conservation requirement ensures that the amounts of flex-object instances are equal before and after (dis-)aggregation at all time intervals. In addition, individual aggregation and disaggregation functions must comply with a number of requirements that are inspired by the MIRABEL/TotalFlex use-cases, but are also important for flex-object aggregation in general:

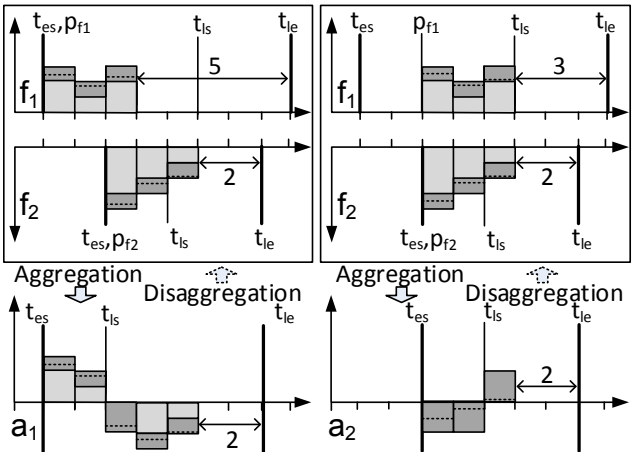
Compression/flexibility trade-off requirement. AGG must allow controlling the trade-off between the number of aggregated flex-objects and the *flexibility loss* – the difference between the *total flexibility* (see Definition 15) before and after aggregation.

Aggregate constraint requirement. Every aggregated flex-object $f_a \in AGG(F)$ must satisfy a so-called *aggregate constraint* C, which is satisfied only if the value of a certain flex-object attribute, e.g., *total maximum amount*, is within the given bounds and thus the aggregated flex-objects are "properly shaped" to meet the BRP rules.

Incremental update requirement. Flex-object updates (addition/removal) should be processed efficiently and cause minimal changes to the set of aggregated flex-objects. This is vital in scenarios, like MIRABEL, where addition/removal of flex-objects are very frequent.

Balance requirement. When local balance is required, the aggregation function AGG must generate flex-objects by specifying the joint flexible resource utilization of counter-acting entities within the domain D. Thus, AGG should minimize the sum of the *absolute balance* of the aggregated flex-objects, where the absolute balance of a flex-object f, $AbsBalance(f)$, is the sum of the absolute amount averages ($|\frac{a_{min}+a_{max}}{2}|$) of its slices.

This ensures that amount bounds of an aggregated flex-object are centred (balanced) around zero, as in f_{a1} = ($[1, 10]$,$\langle[-10, +10], [-10, +10]\rangle$) in Figure 3.1. Such an aggregated flex-object is, potentially, *mixed* and captures (1) the balanced utilization of a resource within a domain D as a nominal option, and (2) feasible deviations from the balance, describing how the utilization



(a) Start-alignment (b) Balance-alignment

Fig. 3.3: N-to-1 aggregation using different profile alignment options, and the 1-to-N disaggregation

can be unbalanced while respecting all flex-object constraints. In practice, such balance-aggregated flex-objects can be used by BRPs, for example, to balance demand and supply while charging a fleet of EVs and at the same time following price signals from the regulating power market.

Note that the *compression/flexibility trade-off* and *balance* requirements are conflicting, because it is not possible to achieve a small number of aggregated flex-objects with high flexibility and balance at the same time.

3 Aggregation and Disaggregation

In this section, we propose basic flex-object aggregation and disaggregation functions satisfying only the amount conservation requirement. As these two functions produce (consume) a single aggregated flex-object, we denote them as N-to-1 and 1-to-N, respectively. In Section 4, we generalize these functions for a larger set of aggregated flex-objects (N-to-M and M-to-N) and revisit the remaining requirements.

N-To-1 Aggregation. As defined in Section 2, the flex-object profile is not fixed in time, but it must be positioned so that the earliest start time $f.t_{es}$ and the latest start time $f.t_{ls}$ bounds are respected. Hence, the aggregation of even two flex-objects is not straightforward. Consider aggregating two flex-objects f_1 and f_2 with time flexibilities 5 and 2. Thus, we have 18 $((5+1)\cdot(2+1))$ different profile *positioning* combinations, each of them realizing a different aggregated flex-object. Figure 3.3 depicts two such combinations.

Algorithm 1 N-To-1 aggregation using *start-alignment*

Input: F - a set of flex-objects;
Output: f_a - an aggregated flex-object;

1: **function** AGG-N-TO-1(F)
2: **for** $f \in F$ **do**
3: $p_f \leftarrow f.t_{es}$; ▷ Start-aligning
4: **end for**
5: $f_a.t_{es} \leftarrow min_{f \in F}(p_f)$
6: $f_a.t_{ls} \leftarrow f_a.t_{es} + min_{f \in F}(f.t_{ls} - p_f)$
7: **for** $t \in [min_{f \in F}(p_f) + 1, max_{f \in F}(p_f + p_{dur}(f))]$ **do**
8: $s_a \leftarrow f_a.p[t - f_a.t_{es}]$
9: $s_a.a_{min} \leftarrow \sum_{f \in F} f.p[t - p_f].a_{min}$
10: $s_a.a_{max} \leftarrow \sum_{f \in F} f.p[t - p_f].a_{max}$
11: **end for**
12: **return** f_a
13: **end function**

In the simplest case, flex-object profiles can be positioned at their earliest start time (t_{es}), see Figure 3.3(a). We refer to this type of positioning as *start-alignment*. In this case, Algorithm 1 is used to aggregate flex-objects from a set F into a single flex-object f_a (N-To-1).

First, Algorithm 1 partially instantiates flex-objects in F by choosing the absolute profile positions $p_{f_1}, \ldots, p_{f_{|F|}}$ using *start-alignment* (Lines 2–8). Then, the start time flexibility bounds of the aggregated flex-object are computed conservatively so that the aligned profiles of $f \in F$ can always be shifted within the flexibility range of f_a (Lines 5–6). Finally, the profile of the aggregated flex-object is constructed by summing up the minimum and maximum amounts of the slices of the aligned profiles, ignoring the slices with out-of-range indices (Lines 7–11).

In general, there are many other ways to align profiles by choosing the profile positions $p_{f_1}, \ldots, p_{f_{|F|}}$ at Lines 2–8. Each of these alignments determines where amounts from individual flex-objects are allocated within the profile of f_a. A few basic alignment options have already been presented and discussed [73]. Additionally, we propose a novel so-called *balance-alignment*, solving the non-linear balance optimization problem to find the profile positions, so that the amount bounds of aggregated flex-object are allocated around zero, as formulated by the *balance requirement*. The effects of the *start* and *balance* alignments are illustrated in Figure 3.3(a-b). The balance alignment is further elaborated on in Section 5.

1-To-N Disaggregation. Given a set of flex-objects F, an aggregated flex-

Algorithm 2 1-To-N disaggregation

Input: F - a set of flex-objects;f_a - an aggregated flex-object; f_a^x - an instance of f_a; $P_F = \{p_{f_1}, \ldots, p_{f_{|F|}}\}$ - a set of profile alignment positions

Output: F^X - a set of flex-object assignments

1: **function** DAGG-1-TO-N(F, f_a, f_a^x, P_F)
2: **for** $i \leftarrow 1..|f_a^x.p|$ **do**
3: $s^x \leftarrow f_a^x.p[i]$
4: $s \leftarrow f_a.p[i]$
5: $s^x.a_{min} \leftarrow s^x.a_{max} \leftarrow \frac{s^x.a_{min} - s.a_{min}}{s.a_{max} - s.a_{min}}$
6: **end for**
7: **for** $f \in F$ **do**
8: $f^x.t_{es} \leftarrow f^x.t_{ls} \leftarrow f_a^x.t_{es} - f_a.t_{es} + p_f$;
9: **end for**
10: **for** $i \leftarrow 1..|f.p|$ **do**
11: $f^x.p[i].a_{min} \leftarrow f^x.p[i].a_{max} \leftarrow f.p[i].a_{min} + (f.p[i].a_{max} - f.p[i].a_{min}) \cdot (f_a^x.p[p_f - f_a.t_{es} + i].a_{min})$
12: **end for**
13: **return** $\{f_1^x, f_2^x, \ldots, f_{|F|}^x\}$
14: **end function**

object $f_a = AGG\text{-}N\text{-}to\text{-}1(F)$, its instance f_a^x ($f_a^x \triangleright f_a$), and the set of profile alignment positions $P_F = \{p_{f_1}, \ldots, p_{f_{|F|}}\}$, where P_F is $\{f_1.t_{es}, \ldots, f_{|F|}.t_{es}\}$ for *start-alignment*, the disaggregation of f_a^x is performed using Algorithm 2.

First, the algorithm normalizes the amounts of f_a^x, making them relative to the amount bounds of f_a (Lines 2–6). Then, it builds instances for every $f \in F$ so that their start-times are equally indented (Line 8) and the amounts are distributed proportionally, having the same relative allocations as the f_a^x amounts (Lines 10–12). The disaggregation of two aggregated flex-object instances (dashed lines) are shown in Figure 3.3(a-b).

The pair of aggregation and disaggregation functions (*AGG-N-to-1*, *DAGG-1-to-N*) satisfies the *amount conservation requirement* (the proof is given in Appendix A.1). Therefore, for a given instance of an aggregated flex-object it is always possible to generate valid instances of non-aggregated flex-objects such that the amounts of fix-objects before and after disaggregation match. The N-to-1 aggregation with *start-alignment* and the 1-to-N disaggregation require $\mathcal{O}(|f_a.p| \cdot |F|)$ time, where $|f_a.p|$ is the length of the aggregated flex-object profile (the full analysis is given in Appendix A.2).

To summarize, the N-to-1 aggregation and the 1-to-N disaggregation functions can be used to aggregate and disaggregate flex-objects while satisfying the *amount conservation* requirement. However, the aggregated flex-objects exhibit *time flexibility* losses, which depend on the alignment and the

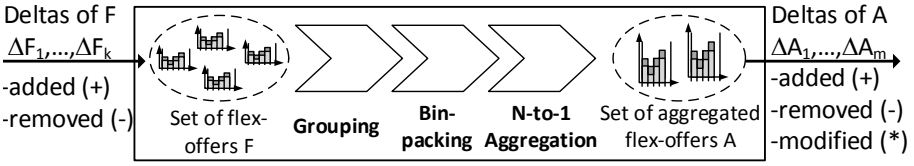


Fig. 3.4: N-to-M aggregation input, output, and phases

flex-object with the smallest time flexibility (see Lines 5–6 in Algorithm 1). To address this limitation and to meet the additional requirements from Section 2, we now propose N-to-M aggregation and M-to-N disaggregation approaches.

4 N-to-M Aggregation

As discussed in Section 3, aggregating flex-objects with different start time flexibilities may result in an unnecessary loss of time flexibility. This loss can be avoided by carefully grouping flex-objects and thus ensuring that their time flexibility intervals are similar (or equal). We now describe an incremental (N-to-M) approach that performs multiple levels of grouping to aggregate a set of flex-objects $\mathcal{F}$ into a set of aggregated flex-objects $\mathcal{A}$ while satisfying the *compression/flexibility trade-off*, *aggregate constraint*, and *incremental update* requirements.

4.1 Overview of the N-To-M aggregation

As shown in Figure 3.4, the N-to-M aggregation approach (internally) manages a set of non-aggregated flex-objects F and a set of aggregated flex-objects A. The sets F and A are maintained using sequences of deltas (updates). $\Delta F_1, \ldots, \Delta F_k$ is the input and $\Delta A_1, \ldots, \Delta A_m$ is the output of the aggregation. Each delta ΔF_i is of the form (f, c), where f is a non-aggregated flex-object and $c \in \{+, -\}$ indicates insertion (+) or deletion (-) of f in F. Similarly, a delta ΔA_i is of the form (f_a, c_a), where f_a is an aggregated flex-object and $c_a \in \{+, -, *\}$ indicates insertion (+), deletion (-), or modification (*) of f_a in A. To explain how the output $\Delta A_1, \ldots, \Delta A_m$ is generated from the input $\Delta F_1, \ldots, \Delta F_k$, we now provide a high-level and intuitive explanation of the *logical phases* of *grouping*, *bin-packing*, and *N-to-1 aggregation*.

4.2 Logical phases of the N-To-M Aggregation

The input $\Delta F_1, \ldots, \Delta F_k$ is passed through the *logical* phases of (1) *grouping*, which partitions flex-objects (from F) into disjoint groups of similar flex-objects, (2) *bin-packing*, which packs flex-objects into smaller sub-groups to

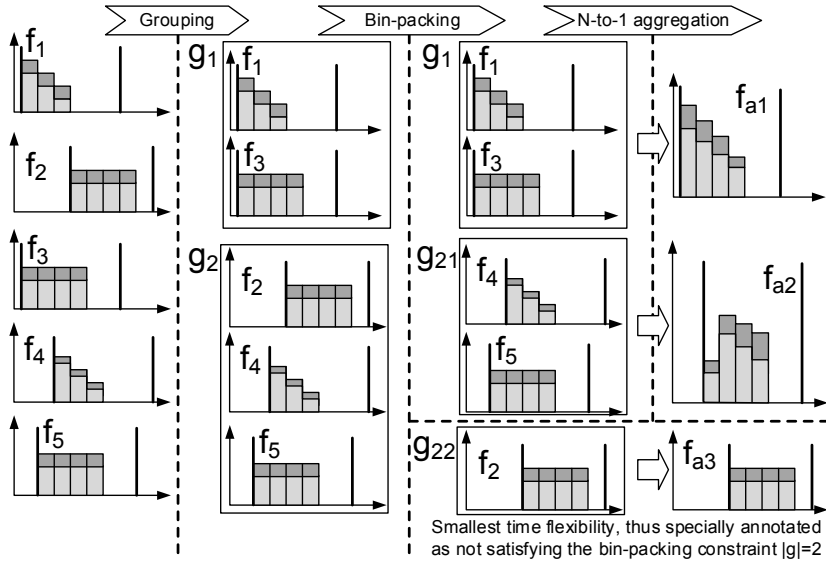


Fig. 3.5: Flex-object aggregation using the N-to-M approach

ensure an aggregation constraint (e.g., the total energy should be below 100*kWh*), and (3) *N-to-1 aggregation*, which applies the N-to-1 aggregation function for each sub-group to produce aggregated flex-objects (see Figure 3.5). Propagation of data is done incrementally, avoiding re-computation of groups, sub-groups, and aggregated flex-objects unaffected by the deltas $\Delta F_1, \ldots, \Delta F_k$.

Grouping phase. In this logical phase, the set of flex-objects F is maintained according to $\Delta F_1, \ldots, \Delta F_k$ and then incrementally partitioned into disjoint groups of similar flex-objects. For this purpose, flex-object *similarity grouping* is performed: two flex-objects are grouped together if the values of specific flex-object attributes, e.g., *earliest start time*, *latest start time*, and/or *time flexibility*, differ by no more than respective user-defined tolerance thresholds. The associated grouping attributes and user-defined tolerance thresholds are called *grouping parameters*. In the example in Figure 3.5, flex-objects $f_1, f_2, \ldots, f_5$ are partitioned into the groups g_1 and g_2 during the grouping phase.

Bin-packing phase. This logical phase incrementally enforces the aggregate constraint (see Section 2). Each affected group g produced in the *grouping* phase is either passed to the next logical phase (if g satisfies the constraint already) or further partitioned into the minimum number of bins (groups) such that the constraint $w_{min} \leq w(b) \leq w_{max}$ is satisfied and time flexibility is retained by each bin b. Here, $w(b)$ is a weight function and w_{max} and w_{min} are the upper and lower bounds. We refer to w_{min}, w_{max}, and w as *bin-*

packing parameters. By adjusting these parameters, groups with, for instance, a bounded number of flex-objects or a bounded total amount can be built. Note that it may be impossible to satisfy the constraint for certain groups. For example, consider a group with a single flex-object, while we impose a lower bound of two flex-objects in all groups. Aggregated flex-objects of the sub-groups failing to satisfy the aggregate constraint are specially annotated (see g_{22} in Figure 3.5).

N-to-1 Aggregation phase. For each of the affected bins, aggregated flex-objects are produced by applying an incremental variant of the N-to-1 aggregation (see Section 3). The alignment option (strategy) is specified as an *aggregation parameter*.

Figure 3.5 visualizes the processing of 5 flex-object insert deltas $(f_1, +), \ldots, (f_5, +)$ in all three logical phases. In these phases, $f_1, \ldots, f_5$ are aggregated into the flex-objects f_{a1}, f_{a2}, and f_{a3}, which are ultimately packed into the insert deltas $(f_{a1}, +)$, $(f_{a2}, +)$, and $(f_{a3}, +)$ returned as output. In this example, grouping parameters are set so that the maximum difference between the earliest start times (t_{es}) is at most 2. The bin-packing parameters are set so that resulting groups have exactly two flex-objects, i.e., $w_{min} = w_{max} = 2, w(g) = |g|$. In the N-to-1 aggregation phase, *start-alignment* is used.

Here, the formation of groups, sub-groups, and aggregates is configured using user-defined parameters – grouping, bin-packing, aggregation parameters. In practice, users are not required to set or fine-tune these (cumbersome) parameters. Instead, they will be offered a number of meaningful pre-defined parameter configurations corresponding to typical business requirements, e.g., *short (long) balanced profiles* or *amounts as early as possible and limited to 100*.

4.3 Algorithms for the N-To-M Aggregation

For realizing the N-to-M aggregation, we use the functions *initAgg*, *processDelta*, and *aggregateInc*. Algorithm 3 uses these functions to perform the N-to-M aggregation. The algorithm produces a set of aggregated flex-objects $F_A = \{a_1, \ldots, a_M\}$ given a set of non-aggregated flex-objects $F = \{f_1, \ldots, f_N\}$ and the discussed *grouping*, *bin-packing*, and *aggregation* parameters.

In Algorithm 3, *initAgg* initializes a number of global data structures (variables) for a specific setting of *grouping*, *bin-packing*, and *aggregation* parameters (Line 2). *processDelta* processes each provided delta from the sequence of deltas, in this case the sequence $(f_1, +), \ldots, (f_N, +)$ generated in Lines 3–4. *aggregateInc* returns all (latest) changes of aggregated flex-objects that occurred since its previous invocation. As it is invoked only once in Algorithm 3 (with an initially empty set of flex-objects), it returns a sequence of aggregated flex-object insert deltas (Line 6), which are then converted into a

Algorithm 3 N-To-M aggregation for inserts only

Input: F - a set of flex-objects; P_G - grouping parameters; P_B - bin-packing parameters; P_A - N-to-1 aggregation parameters;
Output: F_A - a set of aggregated flex-objects

```
1:  function AGG-N-TO-M(F, P_G, P_B, P_A)
2:      initAgg(P_G, P_B, P_A)
3:      for each f ∈ F do
4:          processDelta(f, '+')
5:      end for
6:      ΔF_A ← aggregateInc()
7:      for each (f, c) ∈ ΔF_A do
8:          switch c do
9:              case '+'
10:                 F_A ← F_A ∪ {f};
11:     end for
12:     return F_A;
13: end function
```

set of aggregated flex-objects F_A (Lines 7–10), returned as output (Line 12).

Additionally, the function *processDelta* can handle *delete* deltas (of the type '-') in Line 4. In this case, *aggregateInc* might return insertion (+), deletion (-), and modification (*) deltas, denoting that new flex-objects were added or existing flex-objects were deleted or changed in the set of aggregated flex-objects during the processing of deltas in Line 6. In the following, we discuss each of these functions (*initAgg*, *processDelta* and *aggregateInc*) in more detail and discuss how they together support the logical phases of *grouping*, *bin-packing*, and *N-to-1 aggregation*.

Function *initAgg*. This function initializes the global data structures, including *group hash*, *bin hash*, and *aggregate hash* holding groups, sub-groups (bins), and aggregates generated in the *grouping*, *bin-packing*, and *N-to-1 aggregation* phases. It also initializes the *group changes list* that stores (recent) group modifications.

Function *processDelta* (Algorithm 4). This function performs *pre-grouping*, i.e., it maintains flex-object groups in the group hash according to a delta (f, c) given as input. Here, f is the affected flex-object and c is the type of the delta ('+' or '-'). First, f is mapped into a d-dimensional point (Line 2). This point belongs to a cell in a d-dimensional uniform grid. The extent of a cell in each dimension is defined by the tolerance thresholds from the grouping parameters, and every cell is identified by its coordinates in the grid. Only *populated* cells with flex-objects from F are tracked, combining adjacent populated cells into a *group*. The groups are stored in the group hash, which

Algorithm 4 The pre-grouping function *processDelta*

Global data: GH - a group hash; CL - group changes list;
Input: f - a flex-object to be updated; c - the type of delta: '+' for insertion; '-' for deletion;

```
1:  function processDelta(f, c)
2:      p ← mapToPoint(f);
3:      g ← GH.findGroup(p);
4:      switch c do
5:          case '+'
6:              if g = NULL then
7:                  g ← new Group(p);
8:                  GH.insertGroup(g);
9:              end if
10:             g.insertFOs({f});
11:             CL.registerChange(g,'+',{f});
12:         case '−'
13:             if g ≠ NULL then
14:                 g.removeFOs({f});
15:                 CL.registerChange(g,'−',{f});
16:             end if
17: end function
```

is an in-memory hash table with the cell coordinates as the key and the flex-object group associated with this cell as the value. In Algorithm 4, the group hash is probed for an affected group (Line 3), and the respective changes are made to the group while registering *add* (+) or *delete* (-) modifications in the *group changes list* (Lines 4–15). In case a group is not found in the group hash, a new group with a single populated cell is created (Line 7). If the group changes list already contains a change record for a particular group, the record is updated to reflect the combination of the changes (Line 11 and Line 15).

Figure 3.6 shows the effect of adding a flex-object f_1 using *processDelta*. f_1 is mapped to a 2-dimensional point that lies in grid cell c_2 (Line 2). The coordinates of c_2 are used to locate a group in the group hash (Line 3). The found group is updated by inserting f_1 into its list of flex-objects (Line 10). Finally, a change record indicating that f_1 was *added* (+) to the group is inserted into the group changes list (Line 11).

Function *aggregateInc* (Algorithm 5). This function finalizes *grouping* and performs (the full phases of) *bin-packing* and *N-to-1 aggregation*. First, it optimizes each of the affected groups using the *optimizeGroup* sub-function while storing new group updates in the group changes list (Line 3). Then, each

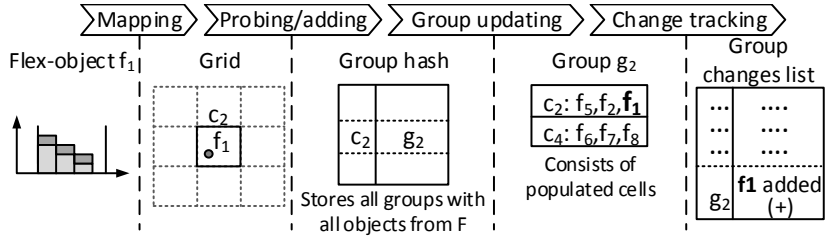


Fig. 3.6: Processing the addition of a flex-object in the grouping phase

Algorithm 5 The grouping, bin-packing, and N-to-1 aggregation function *aggregateInc*

Global data: CL - the group changes list;
Output: ΔF_A - a set of aggr. flex-object deltas;

```
1:  function AGGREGATEINC
2:      for each Δgrp ∈ CL do
3:          optimizeGroup(Δgrp.getGroup());
4:      end for
5:      for each Δgrp ∈ CL do
6:          for each Δbin ∈ updateBins(Δgrp) do
7:              for each Δf_a ∈ updateAggs(Δbin) do
8:                  ΔF_A = ΔF_A ∪ {Δf_a};
9:              end for
10:         end for
11:     end for
12:     CL.clear();
13:     return ΔF_A;
14: end function
```

group change from the updated list is processed with the sub-functions *updateBins* and *updateAggs*, which perform bin-packing and N-to-1 aggregation. Finally, the aggregated flex-object deltas are accumulated (Line 8) and returned as output (Line 13), clearing the group changes list (Line 12).

Sub-function *optimizeGroup* (Algorithm 6). This function is used to optimize affected groups, making them more compact and balanced. First, this function computes a minimum bounding rectangle (MBR) over all points containing flex-objects from a given group g (*getFoMBR* in Line 3). If the extent of the MBR exceeds the grouping parameter thresholds in at least one dimension (Line 3), g is split into a number of sub-groups satisfying the thresholds. For this purpose, g is first disassembled into a number of cells and then *bottom-up hierarchical clustering* [23] is performed on the MBRs of the flex-objects within these cells to form new sub-groups (Line 6). All relevant

Algorithm 6 The group optimization function *optimizeGroup*

Input: GH - the group hash; CL - the group changes list; P_G - grouping parameters;
Input: g - a group to be optimized;

```
1:  function optimizeGroup(g)
2:      F_a ← g.getFlexObjs();
3:      if getFoMBR(F_a) ≤ P_G.getMBR() then   ▷ Tolerances violated, split g
4:          g.removeFOs(F_a);
5:          CL.registerChange(g,' −', F_a);
6:          for each s ∈ clusterHierarch(g.getCells(), P_G) do
7:              GH.insertGroup(s);
8:              CL.registerChange(s,' +', s.getFlexObjs());
9:          end for
10:     else                                   ▷ Consider merging neighbors
11:         for each n ∈ GH.findNbrGroups(g) do
12:             F_a ← g.getFlexObjs() ∪ n.getFlexObjs();
13:             if getFoMBR(F_a) ≤ P_G.getMBR() then
14:                 F_n ← n.getFlexObjs()                          ▷ Merge
15:                 n.removeFOs(F_n);
16:                 CL.registerChange(n,' −', F_n);
17:                 g.insertFOs(F_n);
18:                 CL.registerChange(g,' +', F_n);
19:             end if
20:         end for
21:     end if
22: end function
```

group changes are registered in the group hash and the group changes list (Lines 4–8). Additionally, group merging is considered, while first probing the group hash to identify adjacent neighbouring groups (Line 11). If the extent of the MBR computed over all flex-object points from g and an adjacent group are within the grouping parameter thresholds in all dimensions (Lines 12–13), then g and the adjacent groups are merged while tracking all associated group changes (Lines 14–18). Figure 3.7 demonstrates group splitting and merging.

Sub-function *updateBins* (Algorithm 7). This function propagates a given group delta Δgrp to a number of bins stored in the *bin hash*, which is a hash table with a group ID as the key. First, added and deleted flex-objects, Δ_{added} and Δ_{delete}, are retrieved (Lines 2–3), and flex-objects from Δ_{delete} are discarded from the existing bins (Lines 4–6). Groups with a total weight less than w_{min} are deleted and flex-objects from these groups and Δ_{added} are

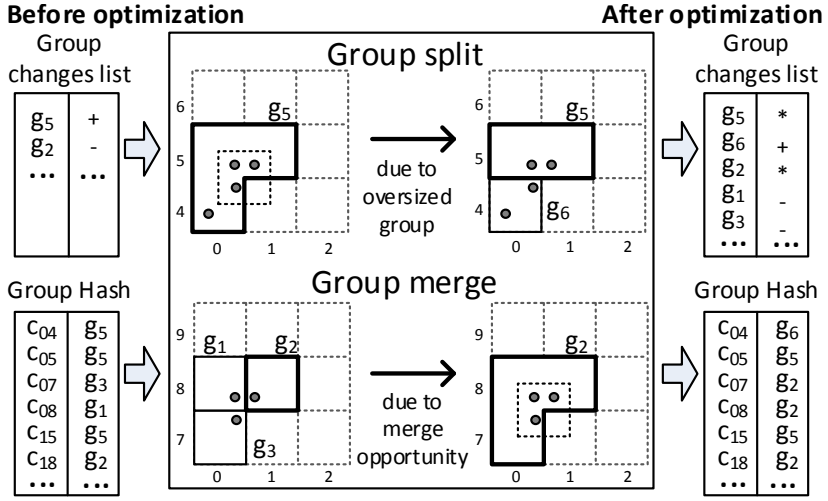


Fig. 3.7: Flow of data in the group optimization

included into other existing bins using the first fit decreasing strategy [43], creating new bins with total weight less than w_{max}, if needed (Line 8). Finally, the changes of affected bins are computed and returned as output (Lines 9–12).

Sub-function *updateAggs* (Algorithm 8). This function maintains the aggregate hash, which is a hash table, mapping the ID of an individual bin to an aggregated flex-object. First, a bin, bin ID, an aggregated flex-object f_a, and the sets of added and deleted flex-objects Δ_{add} and Δ_{delete} are retrieved for an associated bin delta Δbin (Lines 2–6). Then, f_a is incrementally maintained using the N-to-1 aggregation (Algorithm 1): if there are no deletes, f_a is re-aggregated together with flex-objects from Δ_{add} (Lines 7–10); otherwise, flex-objects from the bin are aggregated into f_a from scratch (Lines 16–18), deleting aggregated flex-objects of empty bins if needed (Lines 12–14). Finally, all aggregated flex-object deltas are provided as output (Line 21).

The *average-case* complexity of Algorithm 3 combining all these sub-functions is $\mathcal{O}(|F| \cdot P_{avg})$, where P_{avg} is the average length of aggregated flex-object profiles. The complexity analysis details and the *worst-case* complexity are given in

4.4 M-to-N Disaggregation and Discussion

To disaggregate the instances of aggregated flex-objects, the *DAGG-1-to-N* function from Section 3 is applied independently to each of the affected instances – which are generated in a single (scheduling) step by a single entity (BRP) rather than by multiple (consumer/producer) entities. Therefore, the

Algorithm 7 The bin-packing function *updateBins*

Global data: BH - the bin hash
Input: Δgrp - a delta of a group
Output: ΔB - a set of bin deltas

```
1:  function updateBins(Δgrp)
2:      Δadd ← (Δgrp).getAddedFOs()
3:      Δdelete ← (Δgrp).getDeletedFOs()
4:      bins ← BH.getBins((Δgrp).getGrp().getId())
5:      for each b ∈ bins do
6:          b.removeFOs(Δdelete);
7:      end for
8:      firstFitDecreasing(bins, Δadd)
9:      ΔB ← ∅
10:     for each b ∈ bins do
11:         if b.isAffected() then
12:             ΔB ← ΔB ∪ {b.getBinDelta()}
13:         end if
14:     end for
15:     return ΔB
16: end function
```

(M-to-N) disaggregation is straightforward in comparison to the (N-to-M) aggregation, and we therefore omit the presentation of an explicit disaggregation algorithm.

In summary, the discussed N-to-M aggregation and (M-to-N) disaggregation techniques allow efficiently aggregating flex-objects and disaggregating their instances. As the N-to-1 aggregation and the N-to-1 disaggregation functions from Section 3 are used inherently, *amount conservation* is ensured. Different grouping parameter combinations allow building flex-object groups with different levels of flex-object similarity and thus controlling the trade-off between flex-object compression and flexibility loss. The *aggregate constraint* is ensured by bin-packing. Finally, *incremental updates* are feasible, which allow efficiently processing flex-object additions and removals without the need to recompute aggregated flex-objects from scratch.

5 Balance Aggregation

In Section 3, we have proposed the use of *balance-alignment* to satisfy the *balance requirement*. In relation to *balance-alignment*, we now describe several extensions to the N-to-M aggregation and practical techniques for solving the (non-linear) minimization problem of finding the profile positions

Algorithm 8 The N-to-1 aggregation *updateAggs*

Input: Δbin - a delta of a bin;
Output: ΔF_a - a set of aggr. flex-object deltas
Data: AH - the aggregate hash

```
1:  function UPDATEAGGS(Δbin)
2:      bin ← (Δbin).getBin()
3:      binId ← bin.GetId()
4:      f_a ← AH.getAgg(binId)
5:      Δ_add ← (Δbin).getAddedFOs()
6:      Δ_delete ← (Δbin).getDeletedFOs()
7:      if Δ_delete = ∅ then                                  ▷ No deletes
8:          f_a ← AGG-N-to-1({f_a} ∪ Δ_add);
9:          AH.updateAgg(binId, f_a)
10:         ΔF_a ← ΔF_a ∪ {(f_a,' +')}
11:     else
12:         if (bin.getFlexObjs() = ∅ then                    ▷ Handle deletes
13:             AH.deleteAgg(binId)
14:             ΔF_a ← ΔF_a ∪ {(f_a,' −')}
15:         else                                              ▷ Aggregate from scratch
16:             f_a ← AGG-N-to-1(bin.getFlexObjs())
17:             AH.insertAgg(binId, f_a)
18:             ΔF_a ← ΔF_a ∪ {(f_a,' ∗')}
19:         end if
20:     end if
21:     return ΔF_a;
22: end function
```

$p_{f_1}, p_{f_2}, \ldots, p_{f_{|F|}}$ and thus building aggregated flex-objects with the amount bounds allocated around zero (see Figure 3.3b). The extensions are applicable to the N-to-1 aggregation phase (Section 4) and aggregate only flex-objects that are placed in the same bin.

Exhaustive search. This technique explores all feasible profile position combinations of all the flex-objects in a bin. After examining all possible combinations, the technique generates an aggregated flex-object with the minimum *absolute balance* (see Section 2).

Zero terminated exhaustive search. This technique is similar to the previous one. For each bin, it stops examining further combinations when the current aggregated flex-object has an absolute balance equal to zero.

Dynamic simulated annealing. This is an approximate technique based on *simulated annealing* [2]. The parameter h is used to limit the total number of random combinations to be examined. By default, h is set to half of all possible combinations, where the total number of combinations is com-

Algorithm 9 Simple greedy technique

Input: B - a bin of flex-objects;
Output: B_a - a set of aggregated flex-objects;

```
1:  function SimpleGreedy(B)
2:      f_nom ←FlexObjectWithMinBalance(B);
3:      B ← B \ f_nom; B_a ← ∅;
4:      while |B| > 0 do
5:          f_a ← f_nom
6:          v←balance(f_nom)
7:          f_tmp←ClosestTo-vBalance(B, v)
8:      end while
9:      for each profile position of f_tmp do
10:         f_x ←AGG-N-to-1(f_tmp ∪ f_nom)
11:         if AbsBalance(f_x)<AbsBalance(f_a) then
12:             f_a ← f_x
13:         end if
14:         if AbsBalance(f_a)<AbsBalance(f_nom) then
15:             B ← B \ f_tmp
16:             f_nom ← f_a
17:         else
18:             B_a ← B_a ∪ f_a
19:             B ← B \ f_nom
20:             f_nom ←FlexObjectWithMinBalance(B)
21:         end if
22:     end for
23:     return B_a
24: end function
```

puted individually for each bin. The algorithm terminates if a solution with absolute balance equal to zero is found.

Simple greedy. This technique, described in Algorithm 9, starts by selecting (Line 2) and removing from the bin the flex-object f_{nom} with the largest negative balance v (Line 3); where $balance(f)$ is the sum of the average values of all the slices of a flex-object f. The technique further aggregates f_{nom} with the flex-object f_{tmp} that has (non-absolute) balance closest to $-v$ (Line 2). It examines all the profile positions of f_{tmp} and selects the aggregated flex-object f_a that gives the smallest absolute balance (Lines 3–13). In case the absolute balance of f_a is lower than the absolute balance of f_{nom} (Line 5), the algorithm continues aggregation of the same aggregated flex-object by assigning f_a to f_{nom}. Otherwise (Line 17), it stores f_{nom} as a new aggregated flex-object (Line 19) and starts a new aggregation by selecting f_{nom}, the flex-object with the largest negative balance, from the remaining flex-objects (Line 20). The

Algorithm 10 Exhaustive greedy technique

Input: $f_{nom}, f_a, f_{tmp}, f_x$ - flex-objects; B - a bin of flex-objects;
Output: B_a - a set of aggregated flex-objects;

```
1:  function ExhaustiveGreedy(B)
2:      f_nom ←FlexObjectWithMaxAbsBalance(B)
3:      F ← F \ f_nom
4:      F_a ← ∅
5:      while |B| > 0 do
6:          f_a ← f_nom
7:          for f ∈ B do
8:              for all profile positions of f do
9:                  f_x ←AGG-N-to1(f ∪ f_nom)
10:                 if AbsBalance(f_x)<AbsBalance(f_a) then
11:                     f_a ← f_x
12:                     f_tmp ← f
13:                 end if
14:             end for
15:         end for
16:         if AbsBalance(f_a)<AbsBalance(f_nom) then
17:             B ← B \ f_tmp
18:             f_nom ← f_a
19:         else
20:             B_a ← B_a ∪ f_a
21:             B ← B \ f_nom
22:             f_nom←FlexObjectWithMaxAbsBalance(B)
23:         end if
24:     end while
25:     return B_a
26: end function
```

technique stops when there are no more flex-objects to be aggregated (Line 4, when $|B|$=0). As a result, the technique might produce more than one aggregated flex-object per bin (Line 23).

Exhaustive greedy. Similar to simple greedy, this technique (Algorithm 10) also produces more than one aggregated flex-object per bin. It starts by selecting (Line 2) and removing (Line 4) the flex-object f_{nom} from the bin with the maximum absolute balance. However, it considers all the time flexibility values of all the remaining flex-objects (Line 7 and Line 8) in the bin to find the one that, in combination with the (intermediate) aggregated flex-object f_a, reduces the absolute balance the most. Similar to the simple greedy algorithm, it continues the aggregation of the same aggregated flex-object by assigning f_a to f_{nom} if the absolute balance is reduced (Line 12).

Otherwise, it stores f_{nom} as a new aggregated flex-object (Line 21) and starts a new aggregation by selecting from the remaining flex-objects the flex-object f_{nom} having the largest absolute balance (Line 22). The technique stops when there are no more flex-objects to be aggregated.

6 Experimental Evaluation

In this section, we present the evaluation of the N-to-M aggregation and the balance aggregation techniques.

6.1 Experimental setup

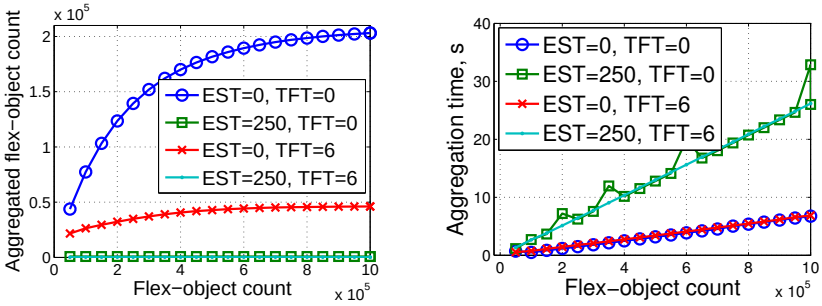
We implemented the basic N-to-M and balance aggregation techniques in Java 1.6. As there are no other flex-object aggregation and disaggregation solutions, we compared our N-to-M aggregation implementation to two rival implementations: R_{HA} and R_{SG}, using our solution for bin-packing and N-to-1 aggregation, but with different (non-incremental) implementations for the grouping phase.

- For grouping, R_{HA} uses agglomerative hierarchical clustering [23] by first assigning each flex-object to individual clusters and then repeatedly merging the two closest clusters while no grouping constraints are violated. The distance between two clusters is calculated based on the values of the grouping parameter flex-object attributes.
- R_{SG} applies the *similarity group-by operator* [77] for one grouping parameter at a time, thus partitioning the input into valid groups of similar flex-objects.

Our experiments were run on a PC with Quad Core Intel®Xeon®E5320 CPU, 16GB RAM, OpenSUSE 11.4 (x86_64).

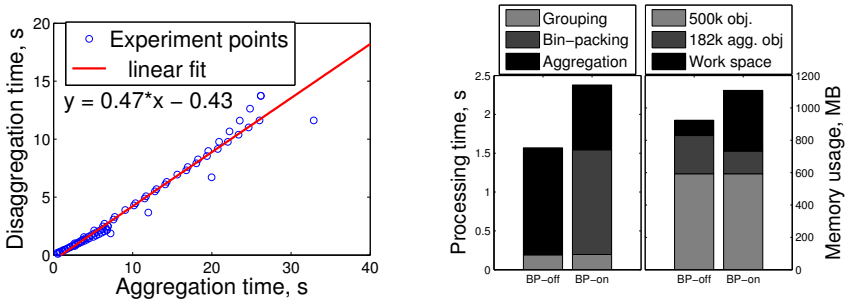
6.2 N-to-M aggregation

We evaluated the N-to-M aggregation discussed in this chapter as well as R_{HA} and R_{SG} using a synthetic flex-object dataset from the MIRABEL project, containing one million flex-objects representing consumption. The *earliest start time* (t_{es}) is distributed uniformly in the range $[0, 23228]$. The number of slices and the time flexibility values ($t_{ls} - t_{es}$) follow the normal distributions $\mathcal{N}(8, 4)$ and $\mathcal{N}(20, 10)$ in the ranges $[10, 30]$ and $[4, 12]$; the slice duration is fixed to 1 time unit (15 minutes) for all flex-objects, thus length of the profiles ranges from 2.5 to 7.5 hours. Unless otherwise stated, the default values of the experimental parameters are: (a) the number of flex-objects is $500k$; (b) $EST = 0$ (*Earliest Start Time Tolerance*) and $TFT = 0$ (*Time Flexibility Tolerance*)



(a) Compr. perf. (b) Agg. time

Fig. 3.8: Scalability of compression and aggregation time

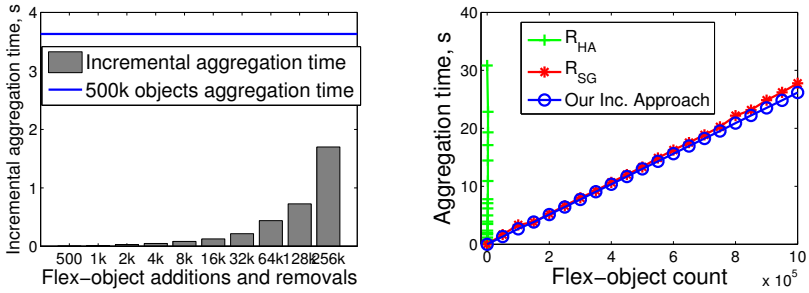


(a) Disagg./agg. time (b) Time/mem. usage

Fig. 3.9: Scalability aggregation/disaggregation and time/memory usage

are used as the grouping parameters. They are based on the *Earliest Start Time* (t_{es}) and *Time Flexibility* ($t_{ls} - t_{es}$) flex-object attributes. (c) the aggregate constraint is unset (bin-packing is disabled). We also perform experiments with bin-packing enabled (explicitly stated).

Scalability To evaluate flex-object compression in terms of performance and scalability, the number of flex-objects is gradually increased from 50k to 1000k. Aggregation is performed using two different EST and TFT parameter values: EST equal to 0 or 250, and TFT equal to 0 or 6. Disaggregation is executed with randomly generated instances of aggregated flex-objects. The results are shown in Figures 3.8 and 3.9. Figures 3.8(a-b) show that different aggregation parameter values lead to different compression factors and aggregation times. Disaggregation is approximately 2 times faster than aggregation (see Figure 3.9(a)) regardless of the flex-object count and grouping parameter values. Most of the time is spent on the bin-packing (if enabled) and N-to-1 aggregation phases (the 2 left bars in Figure 3.9(b)). Considering



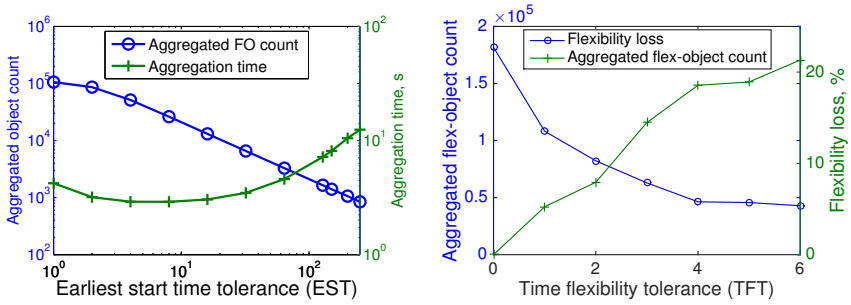
(a) Incr. agg. time (b) Diff. approaches

Fig. 3.10: Incremental behaviour

the overhead associated with incremental behaviour, the amount of memory used by the approach is relatively small compared to the footprint of the original and aggregated flex-objects. Memory usage increases when bin-packing is enabled.

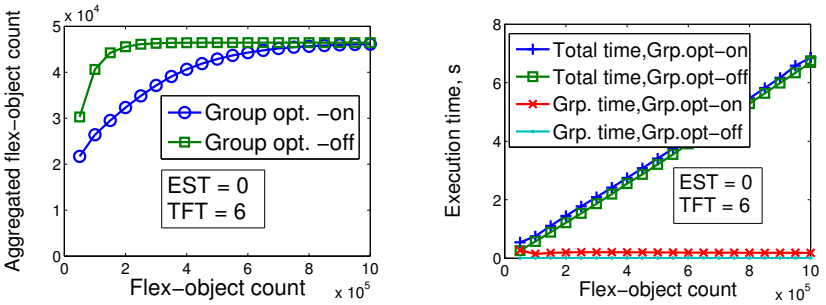
Incremental Behaviour To evaluate incremental aggregation, we first aggregate $500k$ flex-objects. Then, for different k values ranging from 500 to $256k$, we insert k new flex-objects and remove k randomly selected flex-objects. The total number of flex-objects stays at $500k$. For every value of k, we execute incremental aggregation. Figure 3.10(a) shows that updates can be processed efficiently. Hence, our approach results in substantial time savings compared to the case when all $500k$ flex-objects are aggregated from scratch (represented by the line in Figure 3.10(a)). We then compare the total time to process flex-objects with our incremental approach to the other two (inherently non-incremental) approaches (R_{HA} and R_{SG}). As Figure 3.10(b) illustrates, our approach is competitive in comparison to R_{SG} in terms of scalability. The overhead associated with the change tracking and group optimization in the incremental grouping phase is not significant in the overall aggregation time. Additionally, the hierarchical clustering-based approach (R_{HA}) results in very high execution time even for small datasets (due to a large amount of distance computations) and is thus not scalable enough for flex-object aggregation.

Effect of Grouping Parameters As shown in Figure 3.11(a), EST significantly affects the flex-object compression factor. For this dataset, increasing EST by a factor of two leads to a flex-object reduction by approximately the same factor. However, the use of high EST values results in aggregated flex-object profiles with more slices. Aggregating these requires more time (see "aggregation time" in Figure 3.11(a). The TFT parameter has a significant impact on the flexibility loss (see "flexibility loss" in Figure 3.11(b)). Higher



(a) Effect of EST (b) Effect of TFT

Fig. 3.11: Grouping, optimization, and bin-packing

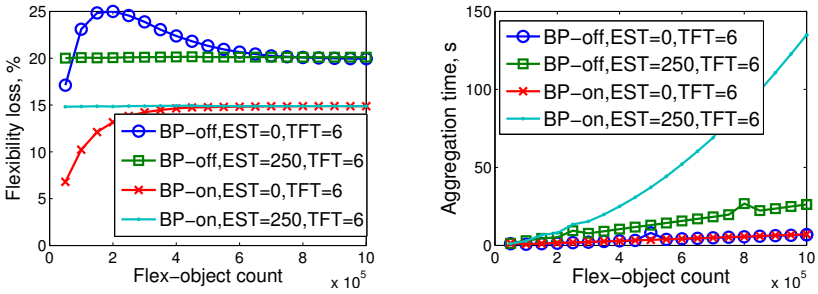


(a) Group. opt. effect (b) Group opt. cost

Fig. 3.12: Grouping, optimization, and bin-packing

values of TFT results in higher flexibility losses. When TFT is set to 0, aggregation causes no flexibility loss but results in a larger number of aggregated flex-objects. When the number of distinct time flexibility values in a flex-object dataset is low (as in our case), the best compression with no flexibility loss can be achieved when $TFT = 0$ and the other grouping parameters are unset (or set to high values).

Optimization and Bin-packing We now evaluate group optimization and bin-packing. As shown in Figure 3.12, group optimization is relatively cheap (Figure 3.12(b)) and it substantially contributes to the reduction of the number of aggregated flex-object (Figure 3.12(a)). To evaluate bin-packing, the aggregate constraint was set so that the time flexibility of an aggregate is always at least 8 ($w_{min} = 8$, equiv. to 2 hours). By enabling this constraint, we investigate the overhead associated with bin-packing and its effect on the flexibility loss. As shown in Figure 3.13(a), by restricting the time flexibility for every aggregate, the overall flexibility loss can be limited. However, bin-



(a) Bin-packing effect (b) Bin-packing cost

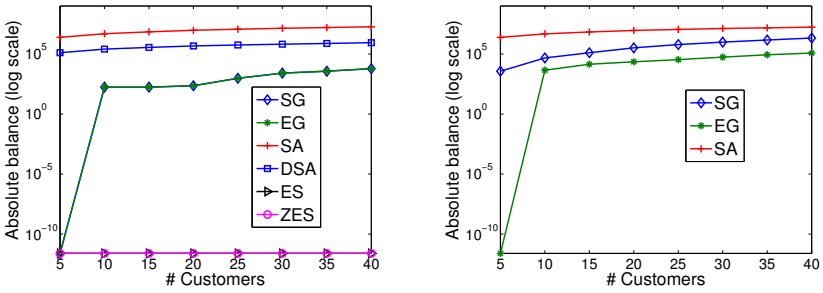
Fig. 3.13: Grouping, optimization, and bin-packing

packing introduces a substantial overhead that depends on the number of objects in flex-object groups after the group optimization (see Figure 3.13(b)). When this number is small ($EST = 0$, $TFT = 6$), the overhead of bin-packing is insignificant. However, when groups are large ($EST = 250$, $TFT = 6$), bin-packing overhead becomes very significant.

6.3 Balance aggregation

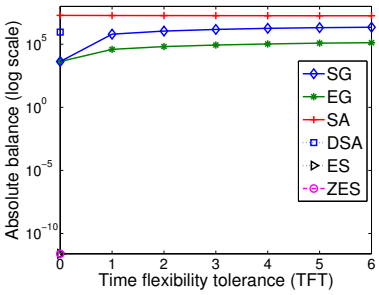
We experimentally evaluated the balance aggregation techniques, namely exhaustive search (ES), zero terminated exhaustive search (ZES), dynamic simulated annealing (DSA), simple greedy (SG), and exhaustive greedy (EG). We compared these techniques against each other, as well as to start-alignment (SA), in terms of *absolute balance*, *flexibility loss*, and *execution time* by varying the grouping tolerances EST and TFT. We also evaluate the techniques in terms of grouping parameters. We set EST to zero and test values from zero to six for the TFT parameter using datasets with 40 customers, which contain approximately 90K flex-objects (Figures 3.15 and 3.16(c, f). We used 8 distinct flex-object datasets derived from the MIRABEL dataset. The first dataset is generated from the historical measurements of 5 randomly selected customers (households), the second by adding another 5 customers, and so on, up to the eighth, which contains 40 customers in total.

We run all experiments 10 times using 10 different instances of the described 8 datasets. The average results of the experiments are illustrated in Figures 3.14–3.16(a-c) and Table 3.1. In addition, we also show the full variation of execution times in Figures 3.16(d-f) since their variations were much higher than the variations of flexibility loss and absolute balance. We often omitted the results of ES, ZES, and DSA due to the extremely high execution times when $TFT>0$. We also omitted DSA in Figure 3.16(a) since it shows the highest flexibility loss due to the limited number of iterations.



(a) TFT=0 (b) TFT=6

Fig. 3.14: Absolute balance for TFT = 0 and TFT = 6

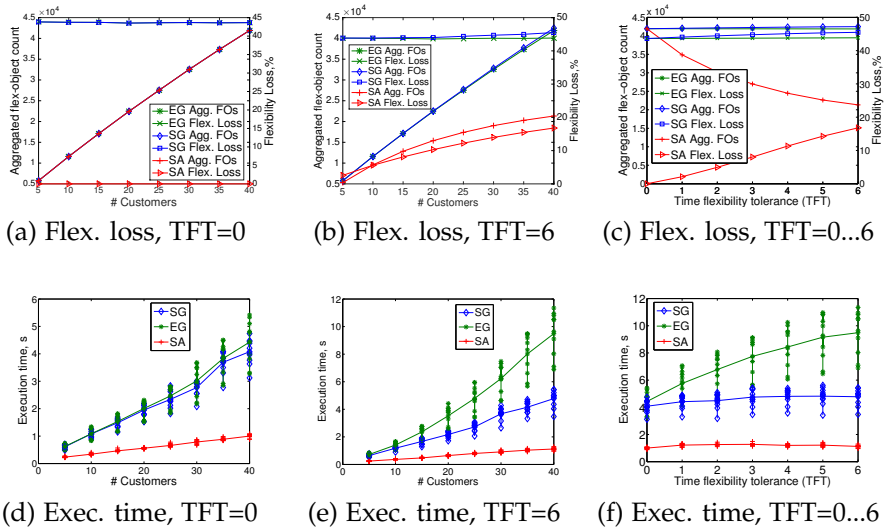


TFT=0...6

Fig. 3.15: Absolute balance for TFT=0...6

Absolute balance. Both ES and ZES achieve zero absolute balance, which is possible due to the nature of the test data, see Figure 3.14(a). In addition, EG and SG achieve a very low absolute balance compared to DSA and SA, see Figures 3.14 and 3.15. SA obtains the highest absolute balance in all examined scenarios since it does not consider balancing during aggregation.

Flexibility loss. SA has the least flexibility loss in all the scenarios followed by EG and SG (Figures 3.16(a-c)). ES and ZES have similar percentages of flexibility loss compared to EG and SG; hence we omit them in the figures. This happens because their goal is to achieve the minimum absolute balance and the flex-objects participating in the aggregation are time shifted in a similar way. In Figure 3.16(b), we see that the flexibility loss of SA increases when more flex-objects participate in the aggregation and the number of aggregated flex-objects follows the same behaviour. This happens because when $TFT > 0$, the flex-objects in each group have different latest start



(a) Flex. loss, TFT=0 (b) Flex. loss, TFT=6 (c) Flex. loss, TFT=0...6

(d) Exec. time, TFT=0 (e) Exec. time, TFT=6 (f) Exec. time, TFT=0...6

Fig. 3.16: Flexibility loss and execution time

times and therefore the aggregated flex-object will have smaller time flexibility intervals. In Figures 3.16(b-c), we see that both EG and SG create more aggregated flex-objects than SA due to their capability to generate more than one aggregated flex-object for each bin.

Execution time. SA is the fastest among all the techniques, followed by SG and EG, see Figures 3.16(d-f). SA's execution time is correlated with the number of flex-objects and, since this is constant, execution time shows a similar behaviour, see Figure 3.16(f). On the other hand, as *TFT* grows, SG is not examining a larger solution space - in contrast to EG - and generates approximately the same number of aggregated flex-objects, see Figures 3.16(c, f). Therefore, the execution time remains almost constant.

Technique	Execution time
Start alignment	0.5 sec.
Simple greedy	1.7 sec.
Exhaustive greedy	2.3 sec.
Dynamic simulated annealing	26 min.
Zero terminated exhaustive search	14.27 hours
Exhaustive search	18.76 hours

Table 3.1: Execution time (15 customers, *EST*=0, *TFT*=5)

6.4 Experiment Summary

In summary, we show that our basic N-to-M aggregation technique scales linearly with the number of flex-object inserts, and it is just as efficient as the best non-incremental grouping approach (R_{SG}). The overhead associated with the incremental behaviour is insignificant. The trade-off between the flex-object compression factor and flexibility loss can be controlled using the grouping parameters. The compression factor can be further increased efficiently by group optimization. Disaggregation is approximately 2 times faster than aggregation.

Furthermore, we show that the balance aggregation techniques EG and SG can achieve a very low absolute balance with low execution time and maintain a near-constant behaviour regarding flexibility loss. For these characteristics, EG performs better than SG and produces a lower number of aggregated flex-objects. However, all the proposed balance aggregation techniques show higher flexibility loss compared to SA, thus demonstrating the inevitable trade-off between flexibility loss and absolute balance. For larger grouping tolerance values, the difference in flexibility loss between SA and EG/SG is reduced.

7 Related work

Clustering. Many clustering algorithms have been proposed, including density-based (e.g., BIRCH [92]), centroid-based (e.g., K-Means [52]), hierarchical clustering (e.g., SLINK [23, 72]) as well as incremental algorithms, such as incremental K-means [93] and incremental BIRCH [38]. In comparison to our approach, clustering partially solves only the grouping part, which is substantially simpler than the whole problem of grouping, bin-packing, balance-aligning, and N-to-1 aggregation. For grouping alone, existing clustering algorithms provide neither effective incremental solutions nor strict guaranties of cluster object similarity, both of which are provided by our approach. The closest work is incremental grid-based clustering [36, 47, 63], where we, in comparison, improve the clusters across the grid boundaries and limit the number of items per cluster using bin-packing.

Similarity Group By. The operator SGB [76] groups objects based on the similarity between tuple values and is supported by SimDB [76]. However, SGB again solves only the grouping part of the problem and is (unlike our approach) not incremental, which is essential.

Complex objects. Complex objects with multidimensional data exist in many real-world applications [53] and can be represented with *multidimensional data models* [64]. Several research efforts (e.g., [17] and [91]) have been proposed to aggregate complex objects. However, these efforts do not consider the specific challenges related to aggregating flex-objects.

Temporal Aggregation. Several papers have addressed aggregation for temporal and spatio-temporal data including instantaneous temporal aggregation [14], cumulative temporal aggregation [6, 39, 89], histogram-based aggregation [20], and multi-dimensional temporal aggregation [15]. These techniques differ in how a time line is partitioned into time intervals and how an aggregation group is associated with each time instant. The efficient computation of these time intervals poses a great challenge and therefore various techniques that allow computing them efficiently have been proposed [28, 30, 60]. Unfortunately, these techniques only deal with simple data items without flexibilities, making them unsuitable for aggregation of flex-objects.

Balance Aggregation. Existing approaches [79] solve the energy balancing problem globally and independently from the aggregation. In comparison, our approach performs local balancing during aggregation, thus reducing distribution grid congestion risks and partially fulfilling the goal of scheduling (global balancing).

8 Conclusion and Future Work

Objects with inherent flexibilities in *time* and *amount*, so-called flexibility objects (flex-objects), occur in both scientific and commercial domains. By focusing on flex-object aggregation and disaggregation, this chapter formally defined relevant concepts and provided a novel and efficient grid-based incremental technique for flex-object aggregation and disaggregation. The paper considered the aggregation/disaggregation process, the grouping of flex-objects, alternatives for computing aggregates, and associated requirements. Additionally, it introduced the concept of energy balancing in aggregation and proposed five techniques that fulfil both balancing and other aggregation goals simultaneously. Extensive experiments using data from an energy domain project showed that the techniques provide very good performance while satisfying all entailed requirements.

Future work will address challenges of aggregating and disaggregating flex-objects with additional flexibilities in profile slice duration. Furthermore, novel balance aggregation techniques taking into account electricity grid constraints will be examined, together with advanced grouping techniques and data structures to further improve performance.

A Appendix

In this appendix, we provide (1) the proof for the amount conservation and (2) the computational complexity analysis of the basic N-to-M and balance aggregation algorithms. For the computational complexity, we first analyse

the *general* complexity of the N-to-1 aggregation functions (Algorithms 1–2). Then, we present the *general* complexities of the essential N-to-M aggregation sub-functions (Algorithms 4–8). Finally, we use the complexity estimates of all these algorithms in a subsequent analysis of the *average-* and *worst*-case run-times of Algorithm 3 that combines all the N-to-M aggregation sub-functions. We also analyse the complexity of the simple greedy balance aggregation technique.

A.1 Proof of amount conservation

We now provide a proof that the pair of N-to-1 aggregation and N-to-1 disaggregation functions from Section 3 (Algorithms 1–2) satisfy the amount conservation requirement from Section 2.

Proof. Consider the following two simple flex-objects $f_1{=}([1,1],\langle[a_1,a_1+d_1]\rangle)$ and $f_2{=}([1,1],\langle[a_2,a_2+d_2]\rangle)$ and their respective instances $f_1^x{=}([1,1],\langle[a_1^x,a_1^x]\rangle)$ and $f_2{=}([1,1],\langle[a_2^x,a_2^x]\rangle)$, where $a_1 \leq a_1^x \leq a_1+d_1$ and $a_2 \leq a_2^x \leq a_2+d_2$. Provided the set $F{=}\{f_1,f_2\}$ as input, the aggregation function *AGG-N-to-1* (Algorithm 1) generates an aggregated flex-object $f_a{=}([min(1,1),min(1,1)+min(0,0)],\langle[a_1+a_2,a_1+d_1+a_2+d_2]\rangle){=}([1,1],\langle[a_1+a_2,a_1+d_1+a_2+d_2]\rangle)$. Here, f_a represents a variety of instances $f_a^x{=}([1,1],\langle[a^x,a^x]\rangle)$ such that $a_1+a_2 \leq a^x \leq a_1+d_1+a_2+d_2$. The conservation requirement in this simplified case with a single time instance essentially states that the equality $a^x = a_1^x + a_2^x$ must hold for any valid a^x and any a_1^x and a_2^x computed by the disaggregation function *DAGG-1-to-N* (Algorithm 2). Given F, f_a, and f_a^x as input, the disaggregation function first normalizes the amounts in $f_a^x{=}([1,1],\langle[a^x,a^x]\rangle)$ (Lines 2–6), computing the normalized amount as follows $n^x = \frac{a^x-(a_1+a_2)}{d_1+d_2}$. Later, the function produces the instances $f_1^x = ([1,1],\langle[a_1^x,a_1^x]\rangle)$ and $f_2 = ([1,1],\langle[a_2^x,a_2^x]\rangle)$, where $a_1^x{=}a_1+d_1\cdot n^x$ and $a_2^x{=}a_2+d_2\cdot n^x$. By using the n^x expression, we have $a_1^x+a_2^x{=}a_1+a_2+(d_1+d_2)\cdot\frac{a^x-(a_1+a_2)}{d_1+d_2}{=}a^x$, thus proving the amount conservation for the *AGG-N-to-1* and *DAGG-1-to-N* function pair. □

A.2 Complexities of N-to-1 aggregation functions

Algorithm 1 performs the N-to-1 aggregation using the provided set F with $|F|$ flex-objects. Here, the profile positions for *start-alignment* are initialized in $\mathcal{O}(|F|)$ time (Lines 2–8). The *start-time* flexibility interval for the aggregated flex-object f_a is initialized in $\mathcal{O}(1)$ time (Lines 5–6). However, the construction of the profile of f_a requires nested looping through (1) the time range $[min_{f\in F}(f.t_{es})+1, max_{f\in F}(t_{ee}(f))]$ containing all profiles of the flex-objects in F (Line 7) and (2) all flex-objects in F (Lines 9–10). As this time range determines the total number of slices in the aggregated flex-object profile, denoted

as $|f_a.p|$, the overall complexity of Algorithm 1 is $\mathcal{O}(|f_a.p| \cdot |F|)$.

Algorithm 2 performs the 1-to-N disaggregation using F (a set of flex-objects), f_a (an aggregated flex-object), and f_a^x (an instance of f_a) provided as input. As f_a^x is an instance of f_a ($f_a^x \triangleright f_a$), both f_a and f_a^x have equal numbers of profile slices, i.e., $|f_a.p| = |f_a^x.p|$. Thus, the normalization of f_a^x amounts requires $\mathcal{O}(|f_a.p|)$ time (Lines 2–6). However, the overall run-time is dominated by the construction of non-aggregated flex-object instances, requiring $\mathcal{O}(|f_a.p| \cdot |F|)$ time (Lines 7–12). Therefore, the overall complexity of Algorithm 2 is $\mathcal{O}(|f_a.p| \cdot |F|)$.

A.3 Complexities of N-to-M aggregation sub-functions

Algorithm 4 performs the pre-grouping using the delta (f, c) provided as input. Here, the flex-object f is mapped into a d-dimensional point in $\mathcal{O}(|P_G|)$ time (Line 2), where $d = |P_G|$ is the number of similarity criteria (or grouping tolerances) used. A group is located in the group hash (Line 3), and the respective group modifications (Lines 4–15) are performed in $\mathcal{O}(1)$ time. Thus, the overall complexity of Algorithm 4 is $\mathcal{O}(1)$ because no more than 10 similarity criteria ($|P_G|$) are typically applicable to flex-objects and $|P_G|$ determining the total number of point (or grid) dimensions is just a small constant $|P_G| \leq 10$, the value of which does not grow with the input.

Algorithm 6 performs the optimization of the group g provided as input. Here, g consists of $|g|_c$ cells containing $|g|$ flex-objects in total. Both $|g|_c$ and $|g|$ depend on the nature of flex-objects in the set F and the grouping parameters P_G applied to these flex-objects, such that $0 \leq |g|_c \leq |g| \leq |F|$. Therefore, all flex-objects in g are acquired in $\mathcal{O}(|g|)$ time (Line 2). The test for group splitting is performed in $\mathcal{O}(|P_G|)$ time, where $|P_G|$ is the number of grouping tolerances used (Line 3). The complexity of the *general* bottom-up hierarchical clustering is $\mathcal{O}(n^3)$ [23], thus group splitting requires $\mathcal{O}(|g|_c^3)$ or $\mathcal{O}(|g|^3)$ time, as $|g|_c \leq |g|$ (Lines 3–8). For group merging, neighboring groups are acquired in $\mathcal{O}(min(|G|, 3^{|P_G|}))$ time when either scanning all G flex-object groups or probing the $|P_G|$-dimensional grid (group hash) for populated cells. The flex-objects are combined together in $\mathcal{O}(|g|)$ time (Line 8) and the test for group merging is performed in $\mathcal{O}(|P_G|)$ time (Line 13). Finally, the actual flex-object redistribution is performed in $\mathcal{O}(|g|)$ time (Lines 13–18). As $|P_G|$ is a small constant, the value of which does not grow with the input, the group splitting dominates the whole process and thus the group optimization is performed in $\mathcal{O}(|g|^3)$ time.

Algorithm 7 updates the number of bins according to the group delta Δgrp provided as input. Δgrp contains (carries) $|\Delta grp|$ flex-objects of a group g such that $0 \leq |\Delta grp| \leq |g|$. Therefore, the sets of added and deleted flex-objects Δ_{added} and Δ_{delete} are retrieved in $\mathcal{O}(|\Delta grp|)$ time (Lines 2–3). Associated bins are retrieved from the bin hash in $\mathcal{O}(1)$ time (Line 4).

As there might be at most $|g|$ bins (each with a single flex-object), relevant flex-objects from Δ_{delete} are removed from these bins in $\mathcal{O}(|g| \cdot |\Delta grp|)$ time (Line 6). The complexity of the *first fit decreasing* bin-packing is $\mathcal{O}(n \cdot log(n))$ [43], thus new flex-objects from Δ_{added} are packed into the existing bins in $\mathcal{O}(|\Delta grp| \cdot log|\Delta grp|)$ time (Line 8). Finally, the changes of affected bins are computed in $\mathcal{O}(|g|)$ time (Lines 9–12). Thus, as the removal of flex-objects dominates the whole process, the overall complexity of Algorithm 7 is $\mathcal{O}(|\Delta grp| \cdot |g|)$.

Algorithm 8 updates an aggregated flex-object f_a according to the bin delta Δbin provided as input. As Δbin contains (carries) at most $|\Delta grp|$ flex-objects added or removed to/from a bin having at most $|g|$ flex-objects, the sets of added and deleted flex-objects Δ_{add} and Δ_{delete} and associated data are acquired in $\mathcal{O}(|\Delta grp|)$ time (Lines 2–6). When there are no deletes ($\Delta_{delete} = \emptyset$), the aggregated flex-object f_a is updated using Algorithm 1 with *start-alignment* in $\mathcal{O}(|f_a.p| \cdot |\Delta_{add}|)$ time (Lines 7–10). When deletes are feasible, f_a is computed from scratch using Algorithm 1 with *start-alignment* in $\mathcal{O}(|f_a.p| \cdot |g|)$ time (Lines 12–18). As $|\Delta_{add}|$ and $|\Delta_{remove}|$ are bounded by $|g|$, the complexity of Algorithm 8 is $\mathcal{O}(|f_a.p| \cdot |g|)$.

Algorithm 5 performs the logical phases of *grouping*, *bin-packing*, and *N-to-1 aggregation* utilizing Algorithms 6–8. It processes $|CL|$ group deltas from the group changes list CL, covering $|CL|$ different groups in total. Utilizing Algorithm 6, the (final) *grouping* is performed in $\mathcal{O}(|CL| \cdot |g|^3)$ time (Line 3). The joint nested *bin-packing*, and *N-to-1 aggregation* are performed using Algorithms 7–8 in $\mathcal{O}(|CL| \cdot |\Delta grp| \cdot |f_a.p| \cdot |g|^2)$ time (Lines 5–8). Thus, the overall complexity of Algorithm 5 is $\mathcal{O}(|CL| \cdot |g|^2 \cdot (|g| + (|\Delta grp| \cdot |f_a.p|)))$.

A.4 Complexity of N-to-M aggregation

The complete process of N-to-M aggregation is performed using Algorithm 3, which uses insert-only deltas to transform N flex-objects from the input set F into M aggregated flex-objects. To evaluate the run-times of Algorithm 3, we first consider the less realistic *worst case* when all flex-objects from F are aggregated into a single aggregated flex-object, thus loosing most of the available flexibility. As opposed to this, we also consider the more realistic *average case* when groups with a bounded number of flex-objects in each group are created. We now use the computational complexities of Algorithms 4–5 to evaluate run-times of Algorithm 3 in the described *worst* and *average* cases.

Worst-case In the worst-case, a single flex-object group g with $|F|$ flex-objects is created, i.e., $|g| = |F|$. As only one group is affected, a single record is inserted into the group changes list, i.e., $|CL| = 1$. Furthermore, as Algorithm 3 processes all insert deltas at once (Lines 3–4), the delta of g contains (carries) all $|F|$ flex-objects, i.e., $|\Delta grp| = |F|$. Consequently, the *worst-case* complexity of Algorithm 3 is $\mathcal{O}(|F|^3 \cdot P_{max})$, where P_{max} is the length of the

longest aggregated flex-object profile.

Average-case In the average-case, the total number of flex-objects in each group is bounded by the constants G_{min} and G_{max} such that $G_{min} \leq |g| \leq G_{max}$. Therefore, the insert deltas of flex-objects from F (Lines 3–4) are processed in batches with at most G_{max} deltas, and they affect no more than $|F|/G_{min}$ groups, i.e., $|\Delta grp| = G_{max}$ and $|CL| = |F|/G_{min}$. Consequently, as G_{min} and G_{max} are just bounding constants, the values of which do not grow with the input, the *average-case* complexity of Algorithm 3 is $\mathcal{O}(|F| \cdot P_{avg})$, where P_{avg} is the average length of aggregated flex-object profiles.

A.5 Complexity of the simple greedy balance aggregation technique

Algorithm 9 performs the simple greedy technique using the provided bin B of flex-objects. Here, the computation of the balance of a flex-object f requires $\mathcal{O}(|f.p|)$ time. Moreover, the flex-object, f_{nom}, with the minimum balance is selected in $\mathcal{O}(|B|)$ time (Line 2). Thus, Line 2 requires $\mathcal{O}(|B| \cdot |f.p|)$ time in total. The assignments in Lines 6 and 6 require $\mathcal{O}(1)$ time. The "while" loop in Line 4 is executed $|B|-1$ times. The assignment in Line 2 requires $\mathcal{O}(|R|)$ time, where R, $|R|<|B|$, is the set of remaining flex-objects in the bin, and is linearly reduced by one in each iteration. Thus, it demands $\mathcal{O}(|B|^2)$ time. The aggregation in Line 4 requires $\mathcal{O}(|f.p|)$ time since 2 flex-objects are aggregated and Algorithm 1 is applied. However, the aggregation is executed $(tf(f_{tmp})+1)$ times since it is nested in the "for" loop of Line 3 multiplied by $|B|-1$ times because it is also nested in the "while" loop. The number of "for" loop iterations depends on the time flexibility of f_{tmp}. The average time flexibility, TF_{avg}, of the flex-objects in $B \setminus f_{nom}$ is: $TF_{avg} = \frac{\sum_{f \in B \setminus f_{nom}} tf(f)}{|B| - 1}$. Thus, the aggregations require $\mathcal{O}(|B| \cdot TF_{avg} \cdot |f.p|)$ time in total. The assignments in Lines 6, 6, and 19 and the checking condition Line 5 require $\mathcal{O}(1)$ time. Line 11 computes the balance of f in $\mathcal{O}(|f.p|)$ time and is repeated in Line 4 $(|B|-1) \cdot TF_{avg}$ times. Thus, it demands $\mathcal{O}(|B| \cdot TF_{avg} \cdot |f.p|)$ time in total. Line 20 scans the remaining flex-objects of the bin in $\mathcal{O}(|R|)$ time and is reduced by one in each loop iteration. Thus, it demands $\mathcal{O}(|B|^2)$ time in total. The overall complexity of Algorithm 9 is $\mathcal{O}(|B|^2 + |B| \cdot |f.p| \cdot TF_{avg})$.

Chapter 4

Balancing Energy Flexibilities through Aggregation.

The paper has been published in the
Data Analytics for Renewable Energy Integration (DARE), Nancy, France, pp. 17-37, 2014.
The layout of the paper has been revised.
DOI: 10.1007/978-3-319-13290-7_2

Abstract

One of the main goals of recent developments in the Smart Grid area is to increase the use of renewable energy sources. These sources are characterized by energy fluctuations that might lead to energy imbalances and congestions in the electricity grid. Exploiting inherent flexibilities, which exist in both energy production and consumption, is the key to solving these problems. Flexibilities can be expressed as flex-objects, which due to their high number need to be aggregated to reduce the complexity of energy scheduling. In this chapter, we discuss balance aggregation techniques that already during aggregation aim at balancing flexibilities in production and consumption to reduce the probability of congestions and reduce the complexity of scheduling.

We present results of our extensive experiments.

1 Introduction

The power grid is continuously transforming into a so called Smart Grid. A main characteristic of the Smart Grid is the use of information and communication technologies to improve the existing energy services of the power grid and simultaneously increase the use of renewable energy sources (RES) [8]. However, the energy generation by renewable sources, such as wind and solar, is characterized by random occurrence and thus by energy fluctuations [34]. Since their power is fed into the power grid and their increased use is a common goal, they may provoke overload of the power grid in the future, especially in peak demand situations [7,45]. Moreover, the use of new technologies, such as heat pumps and electrical vehicles (EV), and their high energy demand could also lead to electricity network congestions [51].

Within the Smart Grid, the EU FP7 project MIRABEL [13] and the ongoing Danish project TotalFlex [1] are using the flex-offer concept [79] to balance energy supply and demand. The concept is based on the idea that the energy consumption and production can not only take place in specific time slots, but be flexible and adjustable instead. For instance, an EV is parked during the night from 23:00 until 6:00. The EV could be charged or alternatively also act as an energy producer and feed its battery energy into the grid [68]. So the EV is automatically programmed to maximally offer 30% of its current battery energy to the grid, corresponding to 2 hours of discharging. So, in a case of an energy demand or favorable energy tariffs, the EV will be discharged, e.g. from 1:00 to 2:00, offering 15% of its battery energy.

In the MIRABEL project, an Energy Data Management System (EDMS) is designed and prototyped. The EDMS aims at a more efficient utilization of RES by the use of the flex-object concept. Such an EDMS is characterized by a large number of flex-objects that have to be scheduled (assign a specific time and energy amount) so that balancing supply and demand is feasible. Since it is infeasible to schedule a large number of flex-objects individually [79], an aggregation process is introduced, so that the number of flex-objects is decreased and consequently also scheduling complexity [80]. In the proposed EDMS architecture, the scheduling component is responsible for properly scheduling the aggregated flex-objects in order to balance out energy fluctuations. The TotalFlex project additionally considers balancing goal during aggregation so that imbalances are partially being handled by the balance aggregation.

The balance aggregation aims to aggregate flex-objects derived from consumption and production in order to create flex-objects with low energy demand and supply requirements. Thus, violations of the network's electricity

capacities could be avoided. At the same time, the aggregated flex-objects still maintain flexibility that could further be used during scheduling to avoid grid congestions, reassure a normal grid operation, and amplify RES use. For instance, in the above mentioned EV example, there could also be another EV that needs 4 hours of charging, corresponding to 70% of its battery capacity. Charging could take place during the night from 22:00 to 5:00. So, the energy of the first EV could be used to partially recharge the second one, for example, from 23:00 to 1:00. Thus, instead of the two EVs, we consider an aggregated flex-object that represents the demand of 2 hours charging, corresponding to 40% of battery capacity and the charge could take place from 23:00 to 5:00. In this work, we perform an extensive experimental investigation of the behavior of balance aggregation. We also propose alternative starting points for the techniques and evaluate the impact of aggregation parameters.

The remainder of this chapter is structured as follows. Section 2 is describing the theoretical foundations and Section 3 related work. Section 4 explains how exactly balance aggregation works. Section 5 discusses results of our extensive experiments. We conclude in Section 6 with a discussion of our future work.

2 Related work

The unit commitment problem [42, 62], where balancing energy demand and supply is taken into consideration, has been extensively investigated through either centralized (e.g., [35, 78]) or distributed approaches (e.g., [49, 50]). Moreover, in [79], the unit commitment problem has been examined by handling the units as flex-objects and by using centralized metaheuristic scheduling algorithms. In [79], the economic dispatch stage of the unit commitment problem is also elaborated by applying a cost function and thus confronting potential imbalances.

Furthermore, aggregation that takes into account flexibilities with a flex-object use case evaluation has been investigated in [73]. Scheduling aggregated flex-objects that only represent energy consumption and introducing aggregation as a pre-step of scheduling has been investigated in [80]. However, in this chapter, we examine aggregation of flex-objects that takes into account one of the goals of scheduling, i.e., balancing. We do not address imbalances by using a cost function as in [80], but instead we handle imbalances as an effort to directly balance out energy amounts derived from supply and demand. To achieve that, we integrate balancing into the aggregation process. As a result, imbalances are partially handled and flexibility still remains to be used by the scheduling procedure. In this chapter, we evaluate the techniques by taking into consideration energy flexibility representing not only from consumption as in [73, 80] but from production as well. Moreover, this

work provides an extensive experimental evaluation of balanced aggregation techniques.

3 Preliminaries

We use the following definition based on [73].

Definition 18. *A flex-object f is a tuple $f = (T(f), profile(f))$ where $T(f)$ is the start time flexibility interval and profile(f) is the data profile. Here, $T(f) = [t_{es}, t_{ls}]$ where t_{es} and t_{ls} are the* earliest start time *and* latest start time*, respectively.*

The data profile $profile(f) = s^{(1)}, \ldots, s^{(m)}$ where a slice *$s^{(i)}$ is a tuple ($[t_s, t_e]$, $[a_{min}, a_{max}]$) where $[a_{min}, a_{max}]$ is a continuous range of the amount and $[t_s, t_e]$ is a time interval defining the extent of $s^{(i)}$ in the time dimension.*

We consider three types of flex-objects: positive, negative, and mixed ones. Positive flex-objects have all their amount values of all their slices positive and correspond to energy consumption flex-object. Negative flex-objects have all their amount values of all their slices negative and correspond to energy production flex-objects. All the other flex-objects are considered as mixed and express hybrid flex-objects. Figure 4.1 illustrates a mixed flex-object f with four slices, $f = ([1,7], s^{(1)}, s^{(2)}, s^{(3)}, s^{(4)})$.

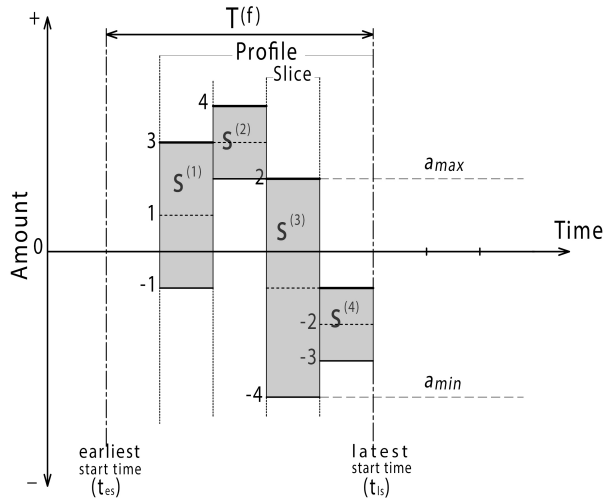


Fig. 4.1: A mixed flex-object

The time is discretized into equal size units, e.g., 15 minutes and the amount dimension represents energy. Every slice is represented by a bar in the figure. The below and above bars of each slice represent the minimum

and maximum amount value, a_{min} and a_{max}, respectively. A flex-offer also supports a lowest and a highest total amount that represents the minimum and the maximum energy required respectively [74].

Moreover, as defined in [73], the *time flexibility*, $tf(f)$, of a flex-object f, is the difference between the latest and earliest start time, the *amount flexibility*, $af(s)$, is the sum of the amount flexibilities of all slices in the profile of f, and the *total flexibility* of f is the product of the time flexibility and the amount flexibility, i.e. $flex(f) = tf(f) \cdot af(s)$. For instance, the flex-object in Figure 4.1 has $tf(f)$=7-1=6, $af(s)=(3-1)+(4-2)+(2-(-4))+(-1-(-3))$=12, and total flexibility: 6∗12=72.

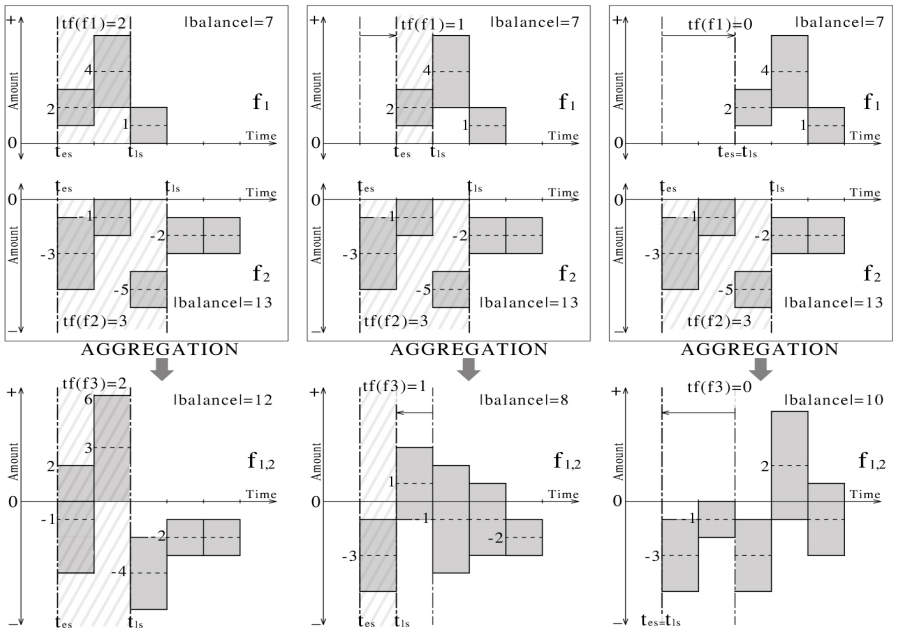
Furthermore, we consider as aggregation the process in which the input is a set of flex-objects and the output is also a set of flex-objects (aggregated) with a smaller or equal number of flex-objects. An aggregated flex-object encapsulates one or more flex-objects and is able to describe the flex-objects that created it. Furthermore, a measurement that we use to evaluate the aggregation is the flexibility loss that is defined according to [73] as the difference between the total flex-object flexibility before and after aggregation. We will further discuss the aggregation process, introduce balance aggregation, and two of its techniques in Section 4.

4 Balance aggregation

In order to describe balance aggregation we define the balance, $balance(f)$, of a flex-object f, as the sum of the average amount values of each slice, and absolute balance, $absolute\_balance(f)$ as the sum of the average absolute values of the amounts for each slice of the flex-object. For example in Figure 4.1, the flex-object $f=([1,6], s^{(1)}, s^{(2)}), s^{(3)}, s^{(4)})$ has $balance(f)=1+3-1-2=1$ and $absolute\_balance(f)=|1|+|3|+|-2| +|-1|=9$. Goal of the balance aggregation is to create aggregated flex-objects with low values of absolute balance. In this section, we sketch the aggregation process and describe approaches to achieve balance aggregation.

4.1 Flex-offer Aggregation

In Figure 4.2 we present a simple aggregation scenario where we aggregate two flex-objects, f_1 and f_2, creating the aggregated flex-object $f_{1,2}$. Both f_1 and f_2, have time and amount and so does the aggregated one. As illustrated, the time flexibility (hatched area) of f_1 is 2 and of f_2 is 3. In the first column of the figure we show the *start alignment* aggregation [73]. In that case, we align the two flex-objects so that their profiles start at the earliest start time and then we sum the minimum and maximum amounts of each aligned slice. The time flexibility of the aggregated flex-object is the minimum flexibility of



(a) Alignment1 (b) Alignment2 (c) Alignment3

Fig. 4.2: Different alignments for aggregation

the non-aggregated flex-objects. This reassures that all the possible positions of the *earliest start time* of the aggregated flex-object will not violate the time constraint that the non-aggregated flex-objects have.

However, because the time flexibility of the aggregated flex-object depends on the minimum flexibility of the flex-objects that participate in the aggregation, the flex-objects are grouped according to 2 different parameters, *EST* (*Earliest Start time Tolerance*) and *TFT* (*Time Flexibility Tolerance*) to minimize flexibility losses [73]. The value of *EST* represents the maximum difference of the earliest start times that the flex-objects could have in order to belong to the same group. The value of *TFT* represents the maximum time flexibility differences that the flex-objects could have in order to be grouped together.

As we can see in the second and the third column of Figure 4.2, there are different start time profile combinations for the flex-objects that participate in the aggregation and result in different aggregated flex-objects. Note that the absolute balance of the aggregated flex-object also depends on the start time profile combinations. For instance, in the second column where we shift the first flex-object for one time unit, we see that the absolute balance of the aggregated flex-object has reduced. On the other hand, continuing shifting

the first flex-object will increase again the absolute balance of the aggregated flex-object, see third column in Figure 4.2. However, with only two flex-objects there are few combinations and the number of combinations increases exponentially with the number of flex-objects and larger time flexibilities. The balance aggregation aims at identifying start time profile combinations and aggregate flex-object in a manner that will minimize the absolute balance of the aggregated flex-object. At the same time, the two balance aggregation techniques considered in this work do not not explore the whole solution space and thus avoid in-depth search.

4.2 Balance aggregation

We examine two different approaches of implementing balance aggregation, the exhaustive greedy and the simple greedy. The techniques are trying to find start time combinations between positive and negative flex-objects that will lead to an aggregated one with a minimum absolute balance.

Both these greedy techniques have the same start point but exhaustive greedy aims to examine a larger solution space than simple greedy. After grouping the flex-objects according to the grouping parameters, both techniques sort all the flex-objects inside each group in a descending order regarding their balance and start the aggregation choosing the one with the minimum balance. The flex-object with the minimum balance will be the most negative one representing a flex-object derived from production that is usually less flexible than the positive ones. Afterwards, exhaustive greedy considers the maximum flexibility for the selected flex-object and then examines all the possible earliest start time combinations with all the remaining flex-objects. The technique chooses the alignment of the flex-object that provides the minimum absolute balance. It continues until the absolute balance is no longer reduced. If the absolute balance is not reduced, it restarts with the remaining flex-objects.

On the other hand, simple greedy also starts the aggregation by choosing the flex-object with the minimum balance. However, aggregation continues with the flex-object that has the balance which is closest to the opposite $(+/-)$ balance of the first one. It examines all the possible start time combinations of the flex-object and chooses the aggregation that has the minimum absolute balance. It also continues the aggregation until the absolute balance of the aggregated flex-object is not further reduced. In case the absolute balance is not reduced, it considers the aggregated flex-object and continues with the remaining flex-objects. In Figure 4.3 we show how exhaustive greedy and simple greedy work in the same group of flex-objects. The example is from one of our datasets and in the figure we show the balance of each flex-object. We see that both the techniques start their aggregation by choosing the most negative flex-object, f_5, and aggregate it with flex-object

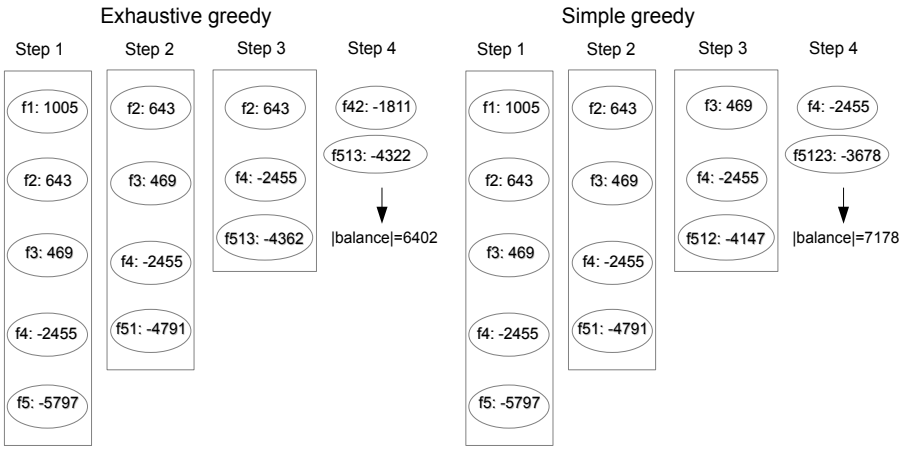


Fig. 4.3: Exhaustive and simple greedy examples

f_1. However in step 3, exhaustive greedy chooses to aggregate with f_3 because it gives a better absolute balance and simple greedy with flex-object f_2 since it is closer to the opposite of the f_{51} balance. As a result, exhaustive greedy continues the aggregation separately for f_2 and f_4 and simple greedy continues aggregation with f_3 leaving f_4 non-aggregated. Therefore, the two techniques lead to different absolute balance values.

5 Experimental Evaluation

In this section, we present an extensive experimental evaluation of the balance aggregation techniques a comparison to start alignment aggregation discussed in Section 4.

5.1 Experimental setup

For the evaluation of the balance aggregation techniques, we used 4 extensive experimental setups of 10 groups of 8 datasets, 320 datasets in total. Each setup is characterized by different time and amount probabilistic distributions corresponding to different energy scenarios.

The first experimental setup is based on the one described in [73]. It consists of 10 groups and each group has 8 datasets, 80 in total. In order to create each group of the 8 datasets, we first select flex-objects derived from the historical consumption time series of 5 random customers. Then, we incrementally add to each dataset flex-objects corresponding to the number of 5 more random customers. The last one, the eighth dataset, has flex-objects

derived from historical consumption time series of 40 random customers. Afterwards, for every dataset we apply start alignment aggregation according to [73], with EST and TFT equal to zero, resulting in an aggregated positive flex-object. In every aggregated flex-object, a random number of amount slice, between zero and its time flexibility value is added. Finally, all the positive aggregated flex-objects are converted to negative ones. As a result, there are always one or more positive flex-objects that if being aggregated have the same opposite balance as a negative one. Since we add in each dataset flex-objects derived from five more customers, the datasets have an incremental number of flex-objects that approximately corresponds to 11K additional flex-objects for every 5 customers. The way the dataset is created reassures that whenever we apply aggregation with the parameters EST and TFT set to 0, it is feasible to create an aggregated flex-object with zero absolute balance. The time flexibility values of the flex-objects follow a normal distribution N (8, 4) in the range [4, 12] and the number of the slices a normal distribution N (20, 10) in the ranges [10, 30]. Those profiles are from 2.5 to 7.5 hours long, with one to three hours time flexibility, which could represent flex-objects derived mostly from EVs. The results of the experiments of this setup are illustrated in the first row of Figures 4.4- 4.11.

Regarding the second experimental setup, we also created 10 groups of 8 datasets. The number of the flex-objects is similar to the one of the previous dataset. Furthermore, historical consumption time series of customers and the flex-object generator tool described in [41] were used to create the datasets. The flex-object generator tool was used to generate both positive and negative flex-objects. For all the datasets the number of the positive (consumption) flex-objects is twice the number of the negative (production) ones. In addition, the number of the slices of the positive flex-objects follows the normal distributions $\mathcal{N}(20, 10)$ in the ranges $[10, 30]$ and of the negative flex-objects the normal distributions $\mathcal{N}(40, 20)$ in the ranges $[20, 60]$. The time flexibility values $(t_{ls} - t_{es})$ of the positive flex-objects and of the negative flex-objects follow a discrete uniform distribution on the interval $[1, 10]$ and [1,8] respectively. This setup aims to explore a scenario in which balancing out energy and production is theoretically feasible. Thus, the negative flex-objects are half the positive ones but with double profile length. Such negative flex-objects with long profiles and less flexibility than the flex-objects representing the consumption could simulate RES. On the other hand, flex-objects characterized by more time flexibility and shorter profiles represent flex-objects derived from mostly recent technological achievements such as EVs and heat pumps. The results of the experiments of this setup are illustrated in the second row of Figures 4.4- 4.11.

Our third experimental setup is created as the second one. These datasets are variations of the second one with a deviation regarding the length and the time. More specifically, the slices of the positive flex-objects follow the

same distribution as before, but the negative flex-objects follow the normal distribution $\mathcal{N}(50, 10)$ in the ranges $[40, 60]$, which makes them longer. The time flexibility values ($t_{ls} - t_{es}$) of the positive flex-objects and of the negative flex-objects follow a discrete uniform distribution on the interval $[2, 18]$ and [1,6] respectively, making the positive flex-objects much more flexible regarding time. Such kind of flex-objects could represent not only EVs and heat pumps but also electronic devices as well. The results of the experiments of this setup are illustrated in the third row of Figures 4.4- 4.11.

Our last experimental setup is similar to the third one. However, the negative flex-objects in this setup are characterized by less energy flexible profiles compared to the positive ones, reflecting a scenario in which the RES are not that flexible regarding energy. Moreover, the positive flex-objects are twice the number of the negative ones. More specifically, the number of the slices for the positive and the negative flex-objects follow the normal distributions $\mathcal{N}(5, 2)$ and $\mathcal{N}(10, 2)$ in the ranges $[1, 10]$ and $[5, 15]$, respectively. The energy flexibility values of the positive flex-objects follow the normal distributions $\mathcal{N}(30, 10)$ in the range $[0, 50]$ and of the negative flex-object $\mathcal{N}(20, 10)$ in the same range over the same amount of flexibility. The results of the experiments of this setup are illustrated in the fourth row of Figures 4.4- 4.11..

In the experiments we investigate the three aggregation techniques regarding the absolute balance, the flexibility loss, the number of the aggregated flex-objects, and the processing time. We examine all four aspects in terms of scalability and grouping parameters, *EST*, and *TFT*. In terms of scalability, we set both the grouping parameters to zero, and examined the techniques in datasets with incremental numbers of flex-objects, starting with minimum 11K (approximately) and maximum 90K (approximately) flex-objects (Figures 4.4, 4.6, 4.8, and 4.10). For each experimental setup, we created 10 groups of datasets that have the same number of flex-objects to reduce any effect of the randomness that characterizes the dataset generation. Regarding the effect of the grouping parameters, we used datasets with almost 90K flex-objects and set one of the parameters stable and set to zero, and varied the other values from zero to six, respectively (Figures 4.5, 4.7, 4.9, and 4.11). For illustrating purposes, we show the average behavior of the similar datasets that there are in each group.

We also investigate the performance of exhaustive and simple greedy after alternating their starting point referring to them in all the figures as "exhaustive greedy" and "simple greedy1". We illustrate their performance when they both start by selecting the flex-object with the maximum absolute balance instead of the one with the minimum balance. In the first row, and the first and second column of Figures 4.4, 4.6, and 4.8 there is an overlap between the illustrated lines of the techniques because the techniques showed similar behavior. The experiments were conducted on a 2.9GHz Intel core i7 processor with two cores, L2 Cache of 256 KB, L3 Cache of 4 MB and physical

memory of 8 GB (4 of 4 GB of 1600MHz DDR3).

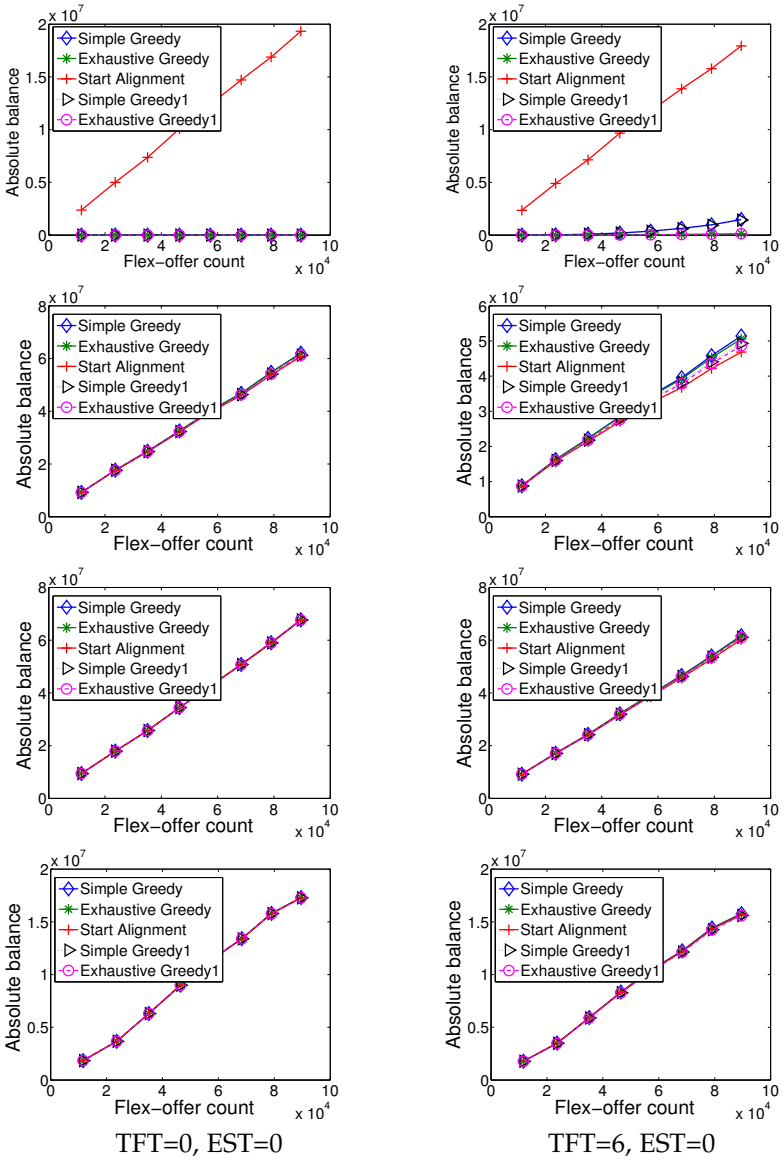


Fig. 4.4: Results of the absolute balance in terms of scalability effect

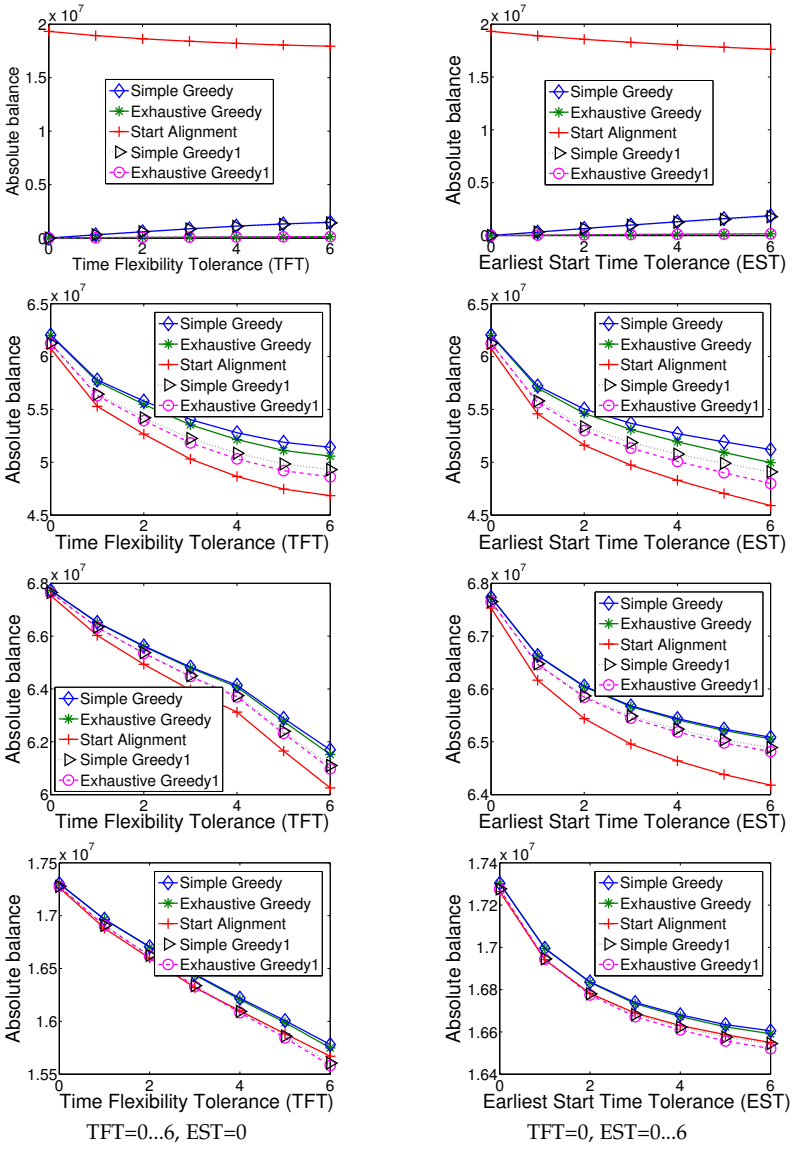


Fig. 4.5: Results of the absolute balance in terms of grouping parameters effect

5.2 Absolute balance

The results for absolute balance are shown in Figures 4.4 and 4.5. In Figure 4.4, absolute balance scales almost linearly with the number of input flex-objects. All the techniques, exhaustive greedy, simple greedy and start alignment have almost the same performance (first row of Figure 4.4) in all

the setups except the first one. In the first setup, the two greedy techniques, exhaustive and simple greedy, achieve a very low balance, first row of Figures 4.4 and 4.5.

More specifically, we see in the first column of Figure 4.4 that both techniques achieve a very close to zero balance when both EST and TFT are set to zero. When TFT is set to 6, exhaustive greedy achieves a lower balance than simple greedy (first row, second column in Figure 4.4), but close to zero for both. Due to the nature of the first experimental setup, zero absolute balance can be achieved and therefore the two greedy techniques achieve it. However, we observe that start alignment achieves the minimum absolute balance among the techniques in the last three setups when there is a large number of flex-objects, approximately 90K, see Figure 4.5. In the second to fourth line of Figure 4.5 we see that exhaustive greedy and simple greedy have absolute balance similar to start alignment, with exhaustive greedy achieving a slightly smaller value between the two greedy techniques.

In the last three experimental setups (second to fourth row in Figures 4.4 and 4.5), there is an overlap in all the slices between the positive and the negative flex-objects since the profiles of the negative flex-objects have at least double profile length in comparison to the positive ones. As a result, start alignment achieves a low absolute balance and the two greedy techniques do not take advantage of the exploration of the solution space since all the differences in absolute balance between each possible aggregation are very low. They are very low because even if there might be an earliest start time combination between a positive and a negative flex-object that reduces the total absolute balance of the mixed flex-object, the percentage will be too low because the absolute balance is mostly reflected by the long profiles of the negative flex-object. This fact combined with the large number of aggregated flex-objects that the two greedy techniques produce after aggregation in all the datasets, (Figures 4.8 and 4.9) produce a lower balance for start alignment.

The number of the flex-objects that is produced by aggregation influences the result of the absolute balance. The fewer flex-objects participate in the aggregation, the more aggregated are produced. As a result less compensations between positive and negative slices take place and that leads to a higher absolute balance. Furthermore, start alignment shows a decreasing behavior of the produced absolute balance when the values of the grouping parameters are greater than zero, first row of Figure 4.9. This occurs because when the values of the grouping parameters increase, more flex-objects participate in the aggregation and thus more positive and negative flex-objects are aggregated, achieving a lower absolute balance.

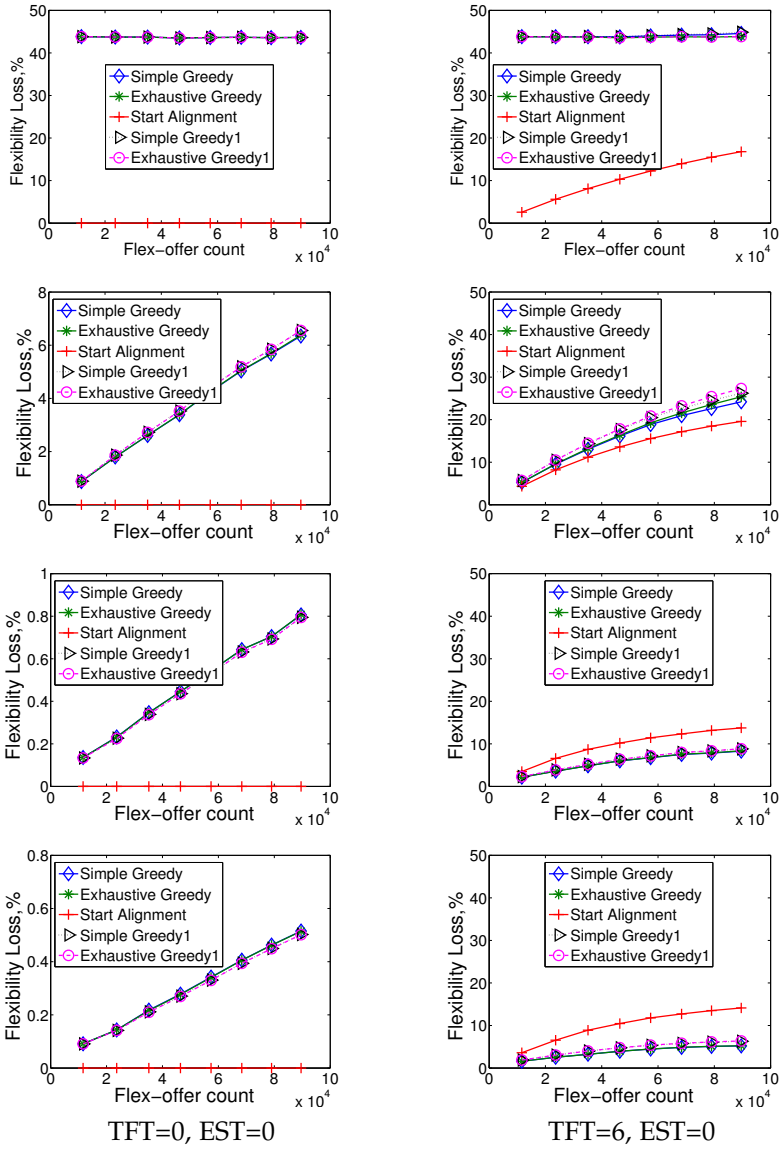


Fig. 4.6: Results of the flexibility loss in terms of scalability effect

5.3 Flexibility loss

Another aspect that we examine for the aggregation is the flexibility loss. In Figures 4.6 and 4.7 we see that the techniques show a divergent behavior in all the different setups. We see, in the first column of Figure 4.6, that start

alignment has zero absolute balance when TFT=0, followed by exhaustive greedy that has a small difference to simple greedy. The flexibility loss is

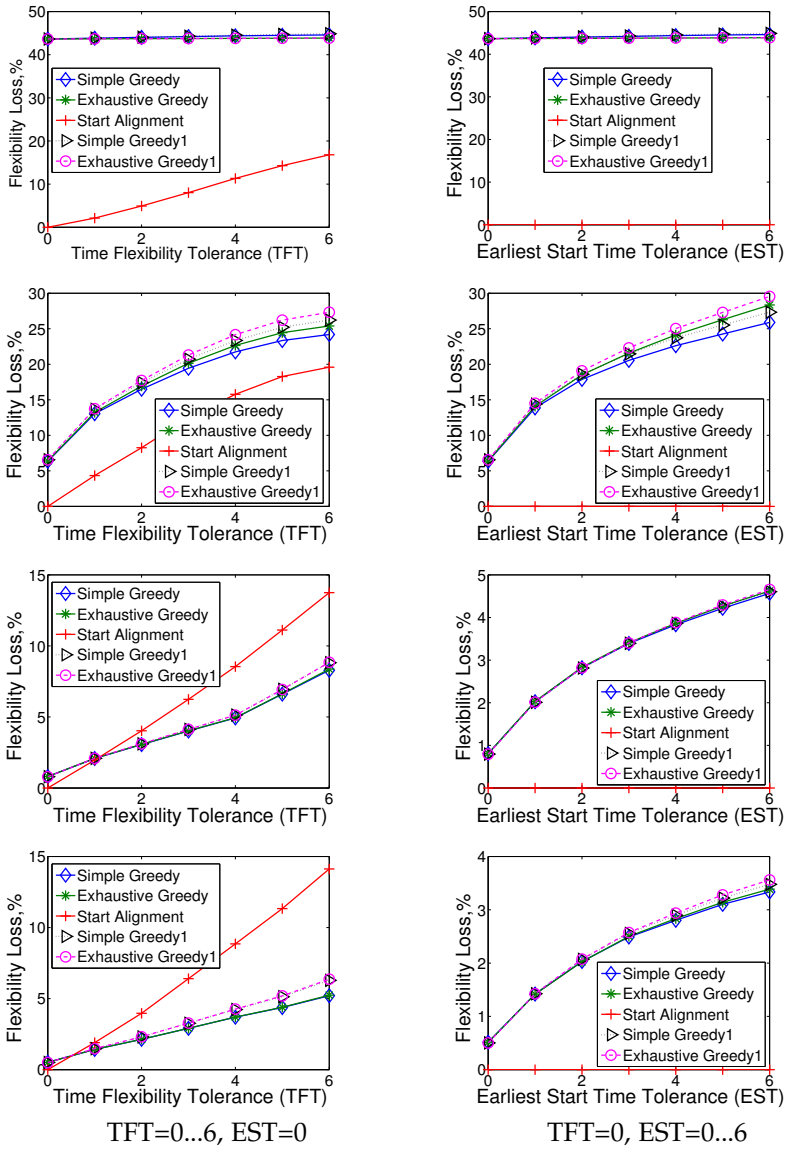


Fig. 4.7: Results of the flexibility loss in terms of grouping parameters effect

mainly affected by the time flexibility and since both EST and TFT are set to zero, it means that in each group the flex-objects have the exact same earliest start time and the same time flexibility as well. That leads to no time flexibil-

ity loss for start alignment and thus no flexibility loss at all. Furthermore, we notice a low percentage of flexibility loss for exhaustive and simple greedy in the three last setups (second column, second and third row of Figure 4.6 and first column, second and third row of Figure 4.7). The low time flexibility of the negative flex-objects of the third and the fourth setups reassures a low flexibility loss. This happens because the solution space is narrowed down, low time flexibility leads to fewer combinations, and that TFT set to 0 reassures that all the flex-objects in the group have the same low time flexibility. A higher percentage of flexibility loss for both greedy techniques is shown in the second setup (second row of Figures 4.6 and 4.7), because in this dataset the time flexibility of the flex-objects is higher.

Both exhaustive and simple greedy behave similarly to start alignment, producing almost the same number of aggregated flex-objects (first column of Figure 4.8). However, we notice a high percentage of the flexibility loss for both greedy and simple greedy in the first setup (first row of Figures 4.6 and 4.7). We see that start alignment has, as before, zero flexibility loss, and the nature of the setup favors an exploration of the solution space for both greedy techniques. Eventually, the greedy techniques identify aggregations that lead to less time flexibility and thus to flexibility loss.

Based on the second column of Figure 4.6, the first and partially the second column of Figure 4.7, we notice that for all the techniques, when the grouping parameters are increased, the flexibility loss is also increased. This happens because flex-objects with different time flexibilities are in the same group. For start alignment, this will lead to an aggregated flex-object with the lowest time flexibility and hence to flexibility loss. Regarding exhaustive and simple greedy, larger grouping parameters result in a larger solution space since more flex-objects participate in the aggregation and more earliest start time combinations exist. As a result, the techniques will most probably create an aggregated flex-object with a lowest absolute balance and a lowest time flexibility. However, no matter how much we increase the value of the EST parameter, the fact that TFT is zero will reassure the maintenance of the time flexibility for start alignment and thus no flexibility loss will occur. In almost all the datasets, start alignment shows the best behavior compared to the other two techniques. In the third and the fourth experimental setup, we see that while the number of the flex-objects increases and especially while TFT increases (third and fourth row of Figure 4.7), the two greedy techniques show a result that is competitive to start alignment result and even better, achieving a lower flexibility loss. The low flexibility losses occur due to the high value of the time flexibility that the positive flex-objects are characterized with, compared to the negative ones in the third and the fourth setup. Therefore, there are flexibility losses for start alignment, since the aggregated flex-objects have the lowest time flexibility.

The two greedy techniques achieve a lower flexibility loss because some

of the flex-objects in the group do not participate in the grouping. Hence, exhaustive and simple greedy create more aggregated flex-objects than start alignment (Figures 4.8 and 4.9), thus fewer flex-objects participate in the aggregation and less flexibility loss will occur.

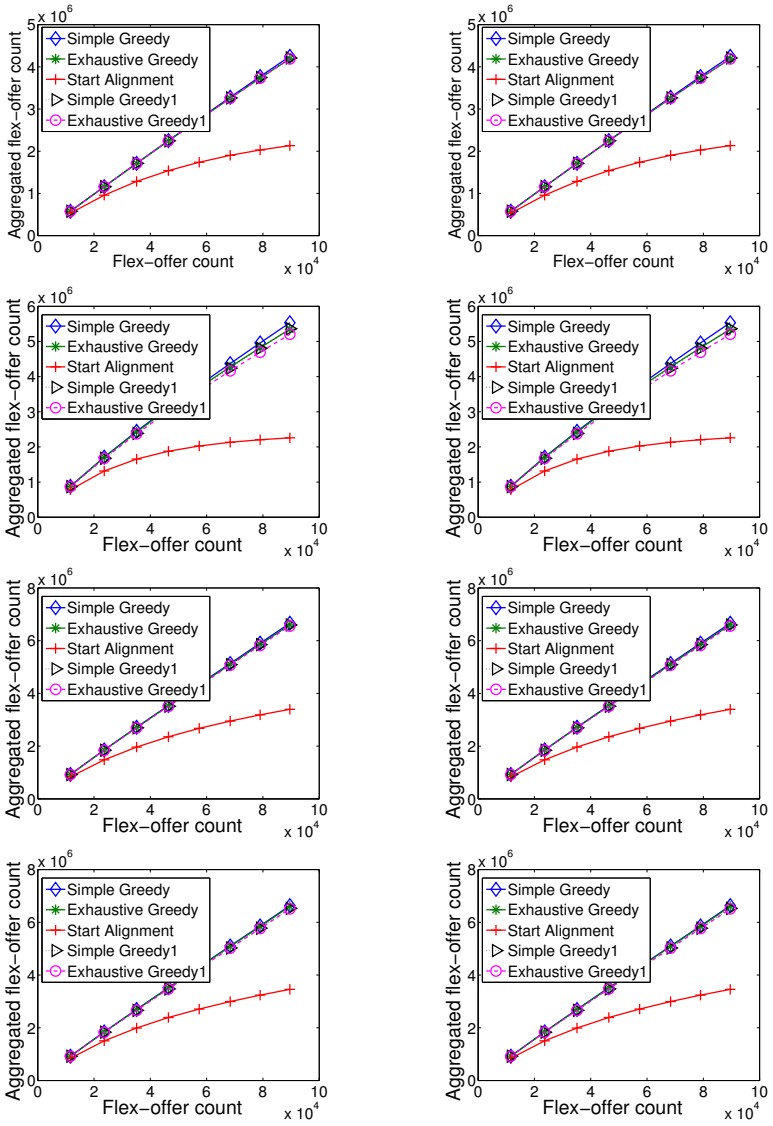


Fig. 4.8: Results of the aggregated flex-object count in terms of scalability effect

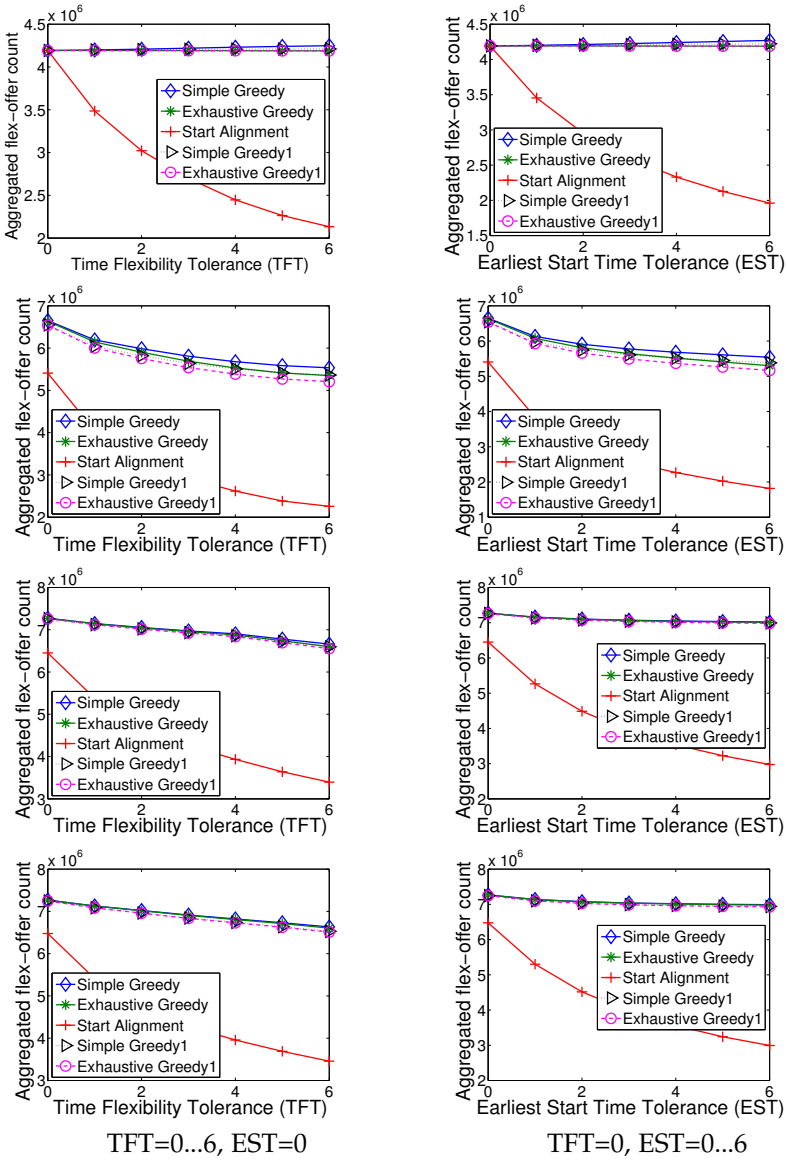


Fig. 4.9: Results of the aggregated flex-object count in terms of grouping parameters effect

5.4 Execution time and aggregated flex-objects count.

Regarding the processing time of all the techniques, we see in Figures 4.10 and 4.11 that start alignment has the best performance followed by simple greedy and exhaustive greedy. Start alignment is the fastest one since it always applies only one aggregation. It is also possible for start alignment to

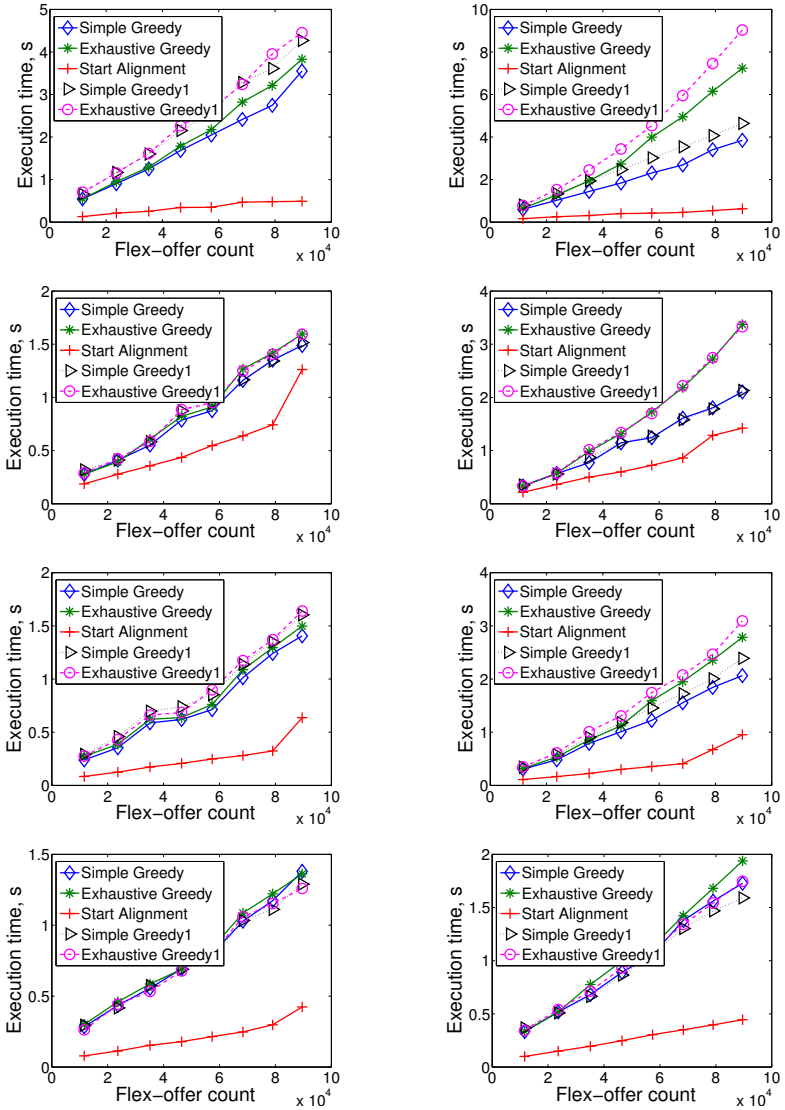


Fig. 4.10: Results of the processing time in terms of scalability effect

achieve better execution times, see third row, second column of Figure 4.11, when the grouping parameters are high and thus fewer groups are created. On the other hand, exhaustive greedy demands the most execution time because it creates more than one aggregated flex-objects as simple greedy does, but explores a larger solution space than simple greedy. This results to larger execution times for exhaustive greedy, leaving simple greedy in the

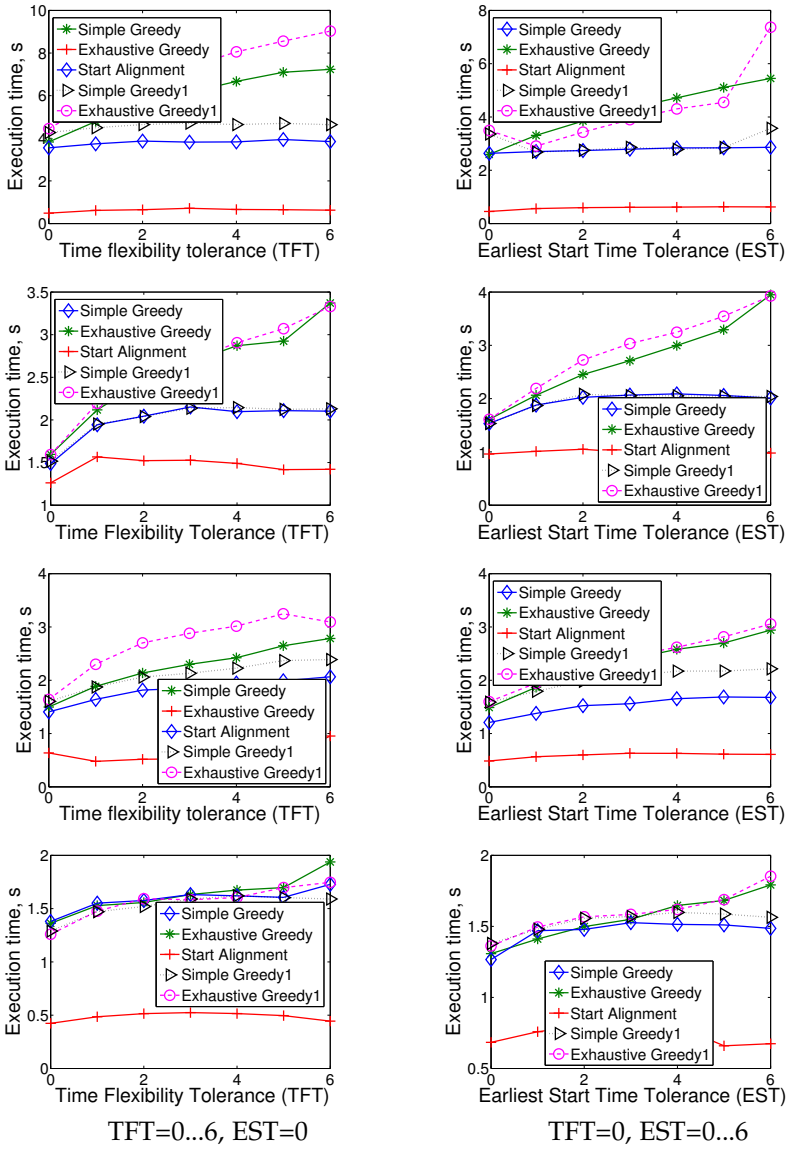


Fig. 4.11: Results of the processing time in terms of grouping parameters effect

second place. Regarding the number of the aggregated flex-objects, we see in Figures 4.8 and 4.9 that in all the experimental setups, start alignment has a lower number of aggregated flex-objects than the two greedy techniques. This is a result of the implementation of the techniques because start alignment will always create one aggregated flex-object when it is applied to a

set of a flex-objects. On the other hand, exhaustive and simple greedy will create at least one aggregated flex-object if absolute balance is not reduced during the aggregation. Regarding the alternative exhaustive and simple greedy we see no difference for the first experimental setup. However, in all the other setups, the alternate techniques achieve a better absolute balance, higher flexibility losses, and fewer aggregated flex-objects when the grouping parameters are set to values greater than zero for larger number of flex-objects (second to fourth row of Figures 4.5, 4.7, and 4.9). On the other hand, since they create fewer aggregated flex-objects, they have a bigger solution space to examine and thus they are slower than the original ones (Figures 4.10 and 4.11).

6 Conclusion and Future Work

In this chapter, we elaborated on aggregation techniques that take into account balancing issues. The techniques discussed in Section 4 reduce the number of the flex-objects that will be the input of the scheduling process and at the same time consider one of its main goals, i.e., achieving balance between energy supply and demand. We conclude through an extensive experimental evaluation that achieving the minimum balance is feasible, but there is always a trade off between balance, flexibility loss and processing time. We show that in order to achieve a good balance, we have to sacrifice time flexibility and also spend more time on processing. The comparisons of the balance techniques with start alignment aggregation showed as well that there are scenarios in which start alignment can achieve very good balance in faster processing times than the greedy techniques. However, flexibility loss between the techniques depends on the grouping parameters without providing a clear winner.

In our future work, we aim to improve the grouping phase that takes place in order to maximize the flexibility that the aggregated flex-objects will have and at the same time improve the balance. It seems also interesting to examine the balance that aggregation will achieve during hierarchical aggregation that is important for an EDMS. In such a scenario, balance aggregation seems more suitable since the input will be mixed flex-objects.

Chapter 5

Aggregating Energy Flexibilities under Constraints

The paper has been published in the
Proceedings of the 7th IEEE International Conference on Smart Grid Communications (SmartGridComm), Sydney, Australia, pp. 484-490, 2016.
The layout of the paper has been revised.
DOI: 10.1109/SmartGridComm.2016.7778808

1 Introduction

One of the main goals of the Smart Grid is the energy use increase from Renewable Energy Sources (RES). However, due to RES being characterized by volatile power production (e.g., wind power), Smart Grid takes advantage of the prosumers' inherent flexibility to better match energy demand with supply, termed Demand Response (DR), and thus enables an increased share of RES energy.

In our work, we model flexible demand/supply devices (referred to as *loads* for simplification) using the flex-offer (FO) concept [13]. An FO explicitly captures the flexibility in energy and time of a load, as presented in the following example.

Example 1.1
The owner (consumer) of an electric vehicle (EV) wants to charge his EV at 20:00 and have it charged by 7:00 the following day. The EV takes 3 hours to be charged and requires 15kWh. Thus, the EV can start its charging between 20:00 and 4:00.

The number of loads that are flexible has recently increased due to new technological achievements (e.g., EVs and heat pumps). The existence of appropriate information and communication technology (ICT) infrastructure [5] and a suitable hierarchical control architecture, offer the capability to market actors to command the DR [24]. Moreover, the establishment of a flexibility market [27] will provide flexibility with the opportunity to be traded [61]. However, the energy captured by individual FOs from small load devices cannot be directly traded in the market [12]. For instance, the power required to participate in the ancillary service market in Denmark is in the magnitude of few hundreds of kW where the consumption capacity of an EV is few kW [12]. Thus, in order to trade flexibility, it is essential to aggregate FOs and produce commodities that can be traded in the emerging energy flexibility markets. Furthermore, aggregation of FOs, applied before scheduling, is essential to reduce the highly complex Unit Commitment (UC) problem [62]. According to the UC problem, FOs are scheduled, i.e., the operational time and amount is defined, based on an objective function.

On the other hand, flexible loads and, consequently, their corresponding FOs are connected to an electrical grid. However, the grid is characterized by power capacity limitations and the high power requirements of new devices, such as EVs, might lead to grid congestions. Grid sensitive load locations (bottlenecks) are in different voltage elements. They could be in low (local distribution) and in high voltage elements (supra-regional distribution). For instance, a bottleneck might be a distribution transformer (0.4-1kV) with a maximum power value of few hundred kW. Such a transformer might serve from few (e.g., in North America) to several hundred households (e.g., in Europe) [57].

In our work, we follow the mapping applied in [88] and map a bottleneck to the root of a tree, see Ⓡ in Figure 5.1. The root is characterized by an amount constraint that defines the tolerable operational power range. For instance, the power of a distribution transformer (0.4kV) shall be in the interval [-300kW, 300kW] [24]. We also map all FOs, which belong to the bottleneck, to the leaf nodes, see ① in Figure 5.1. The leftmost circle in the figure illustrates an FO corresponding to the load of an EV. The x-axis represents time and the y-axis represents power. The energy required for charging the EV is expressed by three slices (one per time unit). The dark-shadowed parts

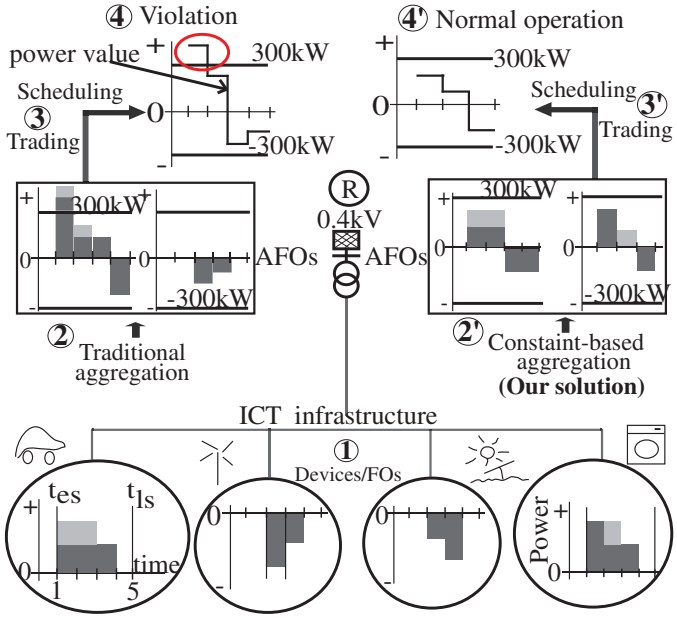


Fig. 5.1: Traditional vs Constraint-based aggregation.

represent the minimum energy requirements. The light-shadowed parts represent optional charging levels. For instance, the EV owner is satisfied when charging level is in the range [60%, 100%]. Moreover, as we see in the figure, charging of the EV can start at time 1 at the earliest (t_{es}) and at time 5 at the latest (t_{ls}). Thus, the FO profile, which consists of the three slices, can be time-shifted.

Using traditional aggregation techniques [75], the FOs are aggregated resulting in aggregated FOs (AFOs). As illustrated in Figure 5.1, the four FOs ① are aggregated into two AFOs ②. Each profile of an AFO is produced by summing up one or more profiles of the 4 FOs. Without considering constraints, loads might be placed at the same time since it may be more beneficial, e.g., from a financial point of view. However, this could lead to violations. For instance, we see that the power of the left AFO (first dark-shadowed slice in ②) exceeds the constraint imposed by the grid. After being aggregated, the AFOs are traded and scheduled, see ③. Scheduling transforms AFOs into *assignments* and forms the root power value. However, it is impossible to schedule the output of traditional aggregation and to respect the constraint. Thus, scheduling leads to a constraint violation due to inappropriate aggregation, see ④ where the power value exceeds 300kW in the first time slot (red circle). Consequently, FO aggregation techniques that take into account grid constraints are required. In this chapter, we propose

such constraint-based aggregation which produces AFOs that can be further scheduled and support a normal grid operation, see (2')-(4') in Figure 5.1.

Contributions. First, we demonstrate the problems that occur with traditional FO aggregation. Second, we introduce the objectives of constraint-based FO aggregation and propose two heuristic aggregation techniques that reduce the input by more than 90% while retaining flexibility. Third, we evaluate the proposed techniques in complex use case scenarios. We show that our techniques lead to normal grid operation where the existing state-of-the-art approaches lead to grid constraint violations at more than 15% of the examined time horizon. Finally, we show that in cases where scheduling *cannot* provide a schedule that respects grid constraints within a certain time period, our aggregation techniques efficiently narrow down the solution space and thus lead to valid scheduling results.

The remainder of the chapter is structured as follows. Section 2 introduces relevant concepts and definitions. Section 3 discusses the problems of traditional aggregation and introduces constraint-based aggregation objectives. In Section 4, the two constraint aggregation solutions are proposed. Their experimental evaluation is described in Section 5. In Section 6 related work is discussed. Finally, the chapter concludes and points to future work in Section 7.

2 Background and preliminaries

Based on [75] and using two discrete dimensions, i.e., time and amount, we define the following. We note that the granularity/precision of the discrete time and amount dimensions can be set as finely as desired, at the expense of somewhat slower computation. In our use case, 1 amount and 1 time unit correspond to 0.5kW and 1 hour, respectively. Thus, we efficiently capture loads of individual devices, e.g., EVs, and an hourly market trading.

Definition 19. *An* FO f *is a tuple* $f=(T(f),P(f))$ *where* $T(f)$ *is the start time flexibility interval and* $P(f)$ *is the amount profile.* $T(f)=[t_{es},t_{ls}]$ *where* t_{es} *and* t_{ls} *are the* earliest start time *and* latest start time, *respectively. The* amount profile *is a sequence of* ($m \in \mathbb{N}_{>0}$) *consecutive slices,* $P(f)=\langle s^{(1)},\ldots,s^{(m)}\rangle$ *where a* slice $s^{(i)}$ *is an amount range* $[a_{min},a_{max}]$. *The duration of slices is 1 time unit. For instance, Figure 5.2 illustrates FO* $f = ([1,5],\langle[3,5],[2,3]\rangle)$.

We distinguish two types of flexibilities associated with an FO that are used as individual measures taking into account time and amount separately. We consider *time flexibility* $tf(f)$ of an FO f to be the difference between its latest and earliest start time, i.e., $tf(f)=t_{ls}-t_{es}$. Moreover, we consider *amount flexibility* $af(f)$ of an FO f to be the difference between the sum of all the maximum and minimum values of all its slices, i.e.,

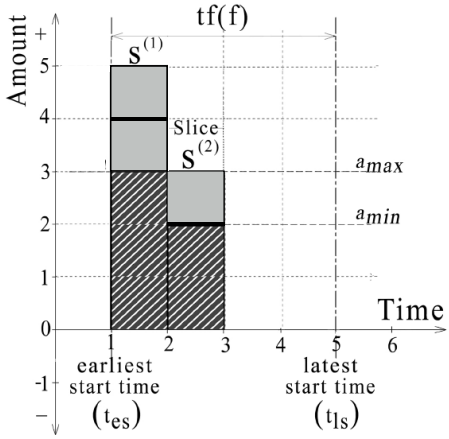


Fig. 5.2: A flex-offer f

$af(f) = \sum_{s \in P(f)} (s.a_{max} - s.a_{min})$. Time flexibility is measured in time units and amount flexibility in amount units.

An FO captures all possible amount demands and/or supplies of a device for a given time horizon. However, during the scheduling process, an FO is assigned to a specific amount at a specific time resulting in an *assignment* of the FO defined as follows:

Definition 20. *An* assignment *of an FO f is a sequence of $|P(f)| \in \mathbb{N}_{>0}$ consecutive slices, as_f=$\langle s^{(1)}, ..., s^{(|P(f)|)} \rangle$. Each slice is a* 2*-tuple, $s^{(i)}$=$(ts, am), i \in [1, |P(f)|]$. The first element, ts, indicates the actual starting time and the second one, am, the actual amount of the slice. The duration of each slice is* 1 *time unit.*

The starting time of the first slice of the *assignment* must be within the start time flexibility interval of the FO, i.e., $f.t_{es} \leq as\_f.s^{(1)}.t_s \leq f.t_{ls}$. Each slice of the *assignment* has an amount value in the range of the corresponding slice of the FO, i.e., $f.s^{(i)}.a_{min} \leq a\_f.s^{(i)}.am \leq f.s^{(i)}.a_{max}$, $\forall i = [1 \ldots |P(f)|]$. There is a finite number of assignments of an FO. We denote the set of all the assignments of an FO f by $L(f)$.

3 Problem Formulation

In this section, we discuss how aggregation is applied through traditional aggregation and introduce the concept of constraint-based aggregation.

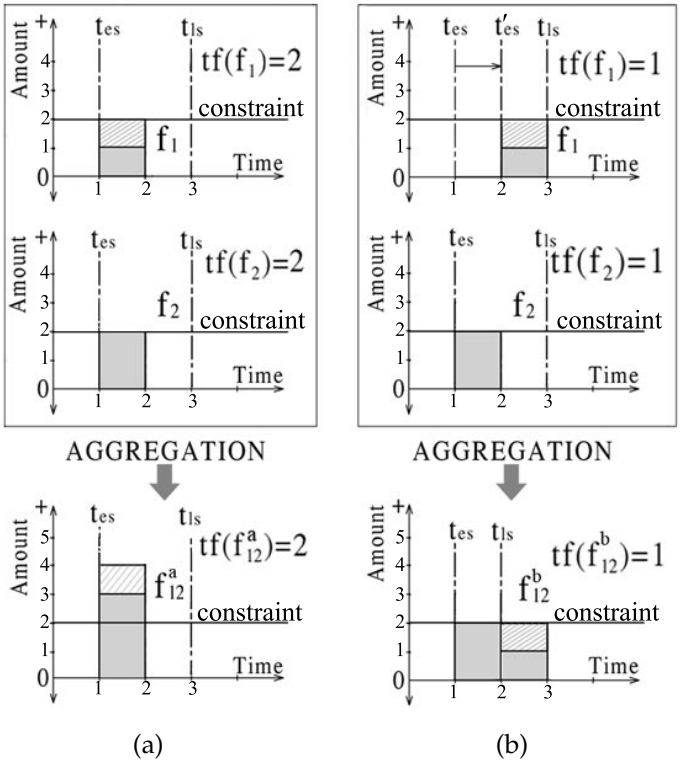


Fig. 5.3: Different alignment examples for aggregation.

3.1 Traditional FO aggregation

We consider, based on [75], traditional aggregation of FOs to be the function that given a set of FOs returns an aggregated one, taking into account the time and amount flexibilities of the FOs. Given a set of FOs, there are different alignments that lead to different AFOs due to their time flexibility.

In particular, given $|F|$ FOs with time flexibility $tf(f_1), \ldots, tf(f_{|F|})$ respectively, the number of the aggregation results (AFOs) that can be produced is: $\prod_{i=1}^{|F|} tf(f_i) + 1$. For instance, the 2 FOs, f_1 and f_2 in Figure 5.3, can be differently aligned and result in different AFOs. Thus, for f_1 and f_2 with both obtaining 3 different start times, there are $3 \cdot 3 = 9$ alignments that lead to 9 aggregation results (AFOs). We show 2 of them in Figure 5.3.

The time flexibility interval of an AFO is determined by the chosen alignments. In particular, the amount profile of an FO does not have any specified starting time until the FO is assigned. However, an FO captures all the different starting times in the start time flexibility interval, see Definition 19. As a result, when aggregation is applied, FOs that participate in aggregation are

aligned (a starting time among the interval is chosen for every FO) and the amount ranges of each aligned slice are summed, see Figure 5.3. We denote the aggregation that aligns FOs according to their earliest start time as *Start Alignment* (SA) aggregation, see Figure 5.3a. According to SA aggregation, the earliest starting time of the AFO is the minimum earliest starting time of the non-aggregated FOs. The latest starting time of the AFO is the sum of its earliest starting time and the minimum time flexibility among the FOs. As a result, the AFO respects all the starting time intervals of the non-aggregated FOs that produced it. For instance, the AFO f^a_{12} in Figure 5.3 has earliest starting time 1 ($f^a_{12}.t_{es} = min(f_1.t_{es}, f_2.t_{es})$). The latest starting time (t_{ls}) of f^a_{12} is equal to 3 ($f^a_{12}.t_{ls} = f^a_{12}.t_{es} + min(tf(f_1), tf(f_2))$).

3.2 Constraint aggregation objectives and complexity

As mentioned in Section 1, the value of a node (actual load) is given by the assignments of the FOs that belong to the node. In particular, during scheduling each FO is turned into an assignment and the result is a set of assignments. Consequently, the sum of the slice amounts with the same time forms the node value at that time. However, in order to guarantee a normal grid operation, the actual loads of the grid must be within the bounds imposed by the constraint, e.g., [-300kW, 300kW]. For instance, concurrently charging a high number of EVs can lead to transformer overload.

We assume that FOs f_1 and f_2 in Figure 5.3 belong to a node with constraint value 2. Moreover, we see that the aggregation result (f^a_{12} last row column a) of SA *does not* enable an assignment that respects the constraint. When scheduling is applied on f^a_{12}, there are several potential assignments of f^a_{12}, e.g., $as\_f^{a1}_{12} = (1, 3)$ and $as\_f^{a2}_{12} = (2, 4)$, see Figure 5.3a. However, the constraint value is 2 and the amounts of all the assignments are greater than the constraint. They should have been within the range [-2,2]. Conversely, we see that when FO aggregation takes into account the constraint, it produces AFO f^b_{12} (Figure 5.3b) that contains assignments which respect the constraint, e.g., $as\_f^b_{12} = \langle(2, 2), (3, 1)\rangle$.

In this chapter, we evaluate an aggregation result through the objectives of constraint-based aggregation.

Constraint-based FO aggregation has 3 objectives. The produced AFOs (1) shall enable scheduling results that respect the constraint of the node where the FOs belong (hard constraint). Moreover, (2) aggregation should retain as much flexibility as possible and (3) at the same time reduce the number of FOs that belong to a specific node.

1) Respect node constraints. All node constraints should be respected. A node constraint violation corresponds to a grid malfunction at the point where the node is. That results in service cutoff of FOs that belong to the violated node and thus the prosumers might not be served.

2) Minimize flexibility losses. Flexibility of FOs is important for scheduling because the more flexible FOs are, the more degrees of freedom the scheduling has to find the optimal solution. Moreover, AFOs capture larger flexibilities and can more easily be traded in the energy market. We use flexibility as a quality measure to evaluate our proposed techniques, as AFOs might lose flexibility during aggregation.

3) Minimize the number of AFOs. FOs are part of the scheduling input that takes place after aggregation. Therefore, it is important for constraint aggregation to reduce the number of FOs, because it directly reduces the complexity of the subsequent scheduling. Moreover, unless FOs are aggregated to capture large energy amounts, they cannot be traded in the energy market.

The above-mentioned objectives might be contradictory and cannot be satisfied simultaneously. In particular, as the number of FOs is reduced, time flexibility losses might increase and time flexibility might be used to respect the constraint. For instance, we see in Figure 5.3a that f_1 and f_2 have time flexibility 2. However, AFO f_{12}^b has $tf(f_{12}^b) = 1$.

Constraint aggregation complexity. Due to space limitations, we illustrate the computational complexity of constraint-based aggregation through an example. In our example, given a set of FOs, we compute the total solution space, i.e., the number of all the potential aggregation results.

Example 3.1
Given a set F of 4 FOs, f_1, f_2, f_3, f_4, with $tf(f_1)$=3, $tf(f_2)$=2, $tf(f_3)$=4, $tf(f_4)$=5, there are $B_4 = \sum_{k=1}^{4} \{{4 \atop k}\} = \frac{1}{1!}(-1)^1\binom{1}{0}0^4 + \frac{1}{2!}\sum_{j=0}^{2}\binom{2}{j}j^4 + \frac{1}{3!}\sum_{j=0}^{3}\binom{3}{j}j^4 + \frac{1}{4!}\sum_{j=0}^{4}\binom{4}{j}j^4 = 1+7+6+1=15$, partitions of F [4]. Moreover, there are $\prod_{i=1}^{4} tf(f_i)$=3·2·4·5=60 alignments. Thus, there are $15 \cdot 60$=900 possible aggregation results.

Adding a fifth FO to the set with $tf(f_5) = 5$, there are B_5=52 partitions of F and $\prod_{i=1}^{5} tf(f_i)$=$60 \cdot 5$=300 alignments. Thus, there are 52·300=15600 possible aggregation results. Therefore, we can notice a *combinatorial explosion* of the aggregation results depending on the size of the input and its average time flexibility.

4 Constraint-based FO Aggregation

Due to the high complexity of exhaustive constraint-based aggregation, we instead propose two variations of a greedy solution to tackle the problem and examine different solution spaces. In particular, the greedy approaches

process FOs that belong to a node incrementally by evaluating binary aggregations in order to reduce complexity. Evaluation is based on different metrics in order to examine whether further aggregation is favored or not. The metrics take into account both the capacity limitations of the node and the objective of the market actor who controls the FOs of the node.

4.1 Constraint and target related distances

As mentioned in Section 1, in order to guarantee a normal grid operation, the node value shall be within the bounds imposed by the constraint. In this chapter, we handle the constraint as a function.

Definition 21. *We define a (constant) positive constraint function* $c(t)=y$, $t\in\mathbb{Z}$, $y\in\mathbb{N}_0$, *where* t *is the time and* y *the amount.*

For instance, given a constraint function $c(t)=300$, the valid amount range is [-300, 300]. In cases where the node value is outside the constraint bounds, a node violation occurs, see for instance ④ in Figure 5.1. When this happens, the electrical grid is not reliable and the distribution system operator, who is responsible for the grid, needs to expand and update the power system infrastructure. Updating the grid is a very expensive and time consuming procedure. In our work, since the node value is formed by the assignments of the FOs, we correlate an assignment to the constraint function. We consider the distance of each slice of an assignment (positive or negative) to be zero when it is within the range because no grid problems occur. Otherwise, we take into account the distance to the constraint function.

Definition 22. *We define the* distance of a slice of an assignment, $D_c(as\_f.s^{(i)})$, *to a constraint function c as equal to zero if the absolute slice amount is smaller or equal to c. Otherwise,* $D_c(as\_f.s^{(i)})$ *is equal to the difference between the absolute amount value of the slice and the constraint, i.e.,* $D_c(as\_f.s^{(i)}) = max(0, |s^{(i)}.am| - c(s^{(i)}.ts))$ *where* $as\_f = \langle s^{(1)}, ..., s^{(|P(f)|)}\rangle$ *and* $i \in [1, |P(f)|]$. *Consequently, we define* distance of an assignment as_f, $D_c(as\_f)$, *to a constraint function c to be the sum of all its slice distances to c, i.e.,* $D_c(as\_f) = \sum_{i=1}^{|P(f)|} D_c(as\_f.s^{(i)})$.

The objective of the market actor controlling the FOs of a node, e.g., an aggregator, is formulated through a target function. Target expresses the optimal schedule, without considering the constraint, and can be used to represent an optimal business goal, e.g., optimal price/amount correlation. Target might contradict the constraint and it could lead to AFOs with assignments that violate the constraint, see for instance ② in Figure 5.1. We define both the target function and the assignment distance to the target function as follows:

Definition 23. *We define a (constant) signed target function $g(t) = a$, where $t \in \mathbb{Z}$ is the time and $a \in \mathbb{Z}$ the amount.*

Definition 24. *We define the* distance of an assignment *as_f to a target function g, $D_g(as\_f)$, as equal to the sum of the absolute differences between g and the amount values of all the slices of the assignment, i.e., $D_g(as\_f) = \sum_{i=1}^{m} \left(|g(s^{(i)}.ts) - s^{(i)}.am|\right)$, $as\_f = \langle s^{(1)}, ..., s^{(m)} \rangle$.*

In our work, we take into account both the capacity limitations of the grid and the market actor's objective. Thus, we consider both the distance to the constraint and the target function to evaluate our results. In particular, we take into account the sum of the distances (target and constraint) and we use weights (coefficients) to prioritize the constraint violation. Of course, when constraint is respected, only the distance to the target function is taken into account.

Definition 25. *We define the* distance of an assignment *as_f to a target function g and a constraint function c, $D_{g,c}(as\_f)$, as the weighted sum of its target and constraint distances with weights α and β respectively, i.e., $D_{g,c}(as\_f) = \alpha \cdot D_g(as\_f) + \beta \cdot D_c(as\_f)$, $\alpha, \beta \in \mathbb{R}$.*

As mentioned in Section 2, since an FO f captures a set of assignments ($L(f)$), there is at least one assignment of f that has the smallest distance.

Definition 26. *We define the* target_to_constraint distance of a FO *f to a target function g and a constraint function c, $D_{g,c}(f)$, as the minimum distance among all its assignments to g and c, i.e., $D_{g,c}(f) = \min_{as\_f \in L(f)} D_{g,c}(as\_f)$.*

Example 4.1
For instance, given $\alpha = 1$, $\beta = 10$, $c(t) = 2$, and $g(t) = 3$, an assignment of f_{12}^a in Figure 5.3 with the minimum distance is: $as\_f_{12}^a = [1, 3]$ where $D_{g,c}(as\_f_{12}^a) = 1 \cdot 0 + 10 \cdot 1 = 10 = D_{g,c}(f_{12}^a)$. On the contrary, an assignment of f_{12}^b with the minimum distance is: $as\_f_{12}^b = \langle [1, 2], [1, 2] \rangle$ where $D_{g,c}(as\_f_{12}^b) = 1 \cdot (1 + 1) + 10 \cdot 0 = 2 = D_{g,c}(f_{12}^b)$.

4.2 Aggregation techniques

We now present our 2 heuristic constraint-based FO aggregation techniques. Both the techniques are variations of the same abstract Greedy algorithm (Algorithm 11). They start by selecting (Line 2) the FO (f_{nom}) with the *maximum target_to_constraint distance* ($max_{f \in SF}(D_{g,c}(f))$. The reason is that apart from reducing the number of the AFOs, aggregation shall also produce FOs that are closer to the target in order to improve scheduling results.

Algorithm 11 Abstract Greedy

Input: SF - set of FOs; g,c - a target and a constraint function
Output: SF - set of AFOs

1: $f_{tmp} \leftarrow$ null; $f_a \leftarrow$ null;
2: $f_{nom} \leftarrow$SelectNomFO(SF); $SF \leftarrow SF \setminus f_{nom}$;
3: **while** $\exists f \in SF$ not aggregated **do**
4: $\{f_a, f_{tmp}\} \leftarrow$BestAggregation($SF, f_{nom}$)
5: **if** $D_{g,c}(f_a) < D_{g,c}(f_{nom})$ **then**
6: $SF \leftarrow SF \setminus f_{tmp}$; $f_{nom} \leftarrow f_a$
7: **else**
8: AnnotateAsAFO(f_{nom})
9: $SF \leftarrow SF \cup f_{nom}$
10: $f_{nom} \leftarrow$ SelectNomFO(SF); $SF \leftarrow SF \setminus f_{nom}$;
11: **end if**
12: **end while**
13: **return** SF

Algorithm 12 Simple Greedy extends Greedy (same input and output as Greedy)

1: **function** BestAggregation(SF, f_{nom})
2: $f_{tmp} \leftarrow$ClosestToZeroDistance(SF)
3: $f_a \leftarrow$BinaryAggregation(f_{nom}, f_{tmp})
4: **return** $\{f_a, f_{tmp}\}$
5: **end function**

Thus, starting aggregation with FOs with high "distances" (used instead of target_constraint distance for simplification) is desirable and increases the chance of reducing the overall distance. Then, the selected FO, f_{nom}, is removed from the initial set (Algorithm 11, Line 2).

Afterwards, algorithm continues until all FOs are aggregated The two variations of Greedy examine different FOs to produce an AFO, i.e., f_a (Line 4). If there is an AFO (f_a) with smaller distance than f_{nom}, the algorithm continues aggregation with the aggregated one and removes f_{tmp} from the initial set SF (Line 6). Otherwise, it annotates f_{nom} as AFO and continues by selecting another f_{nom} from the non-aggregated ones (Lines 8–10). The algorithm stops when all the FOs are annotated as AFOs (Line 4) and returns set SF with the AFOs (Line 13).

Simple Greedy (SG). Apart from f_{nom}, SG also selects a single FO f_{tmp} to examine whether it will aggregate them or not (Algorithm 12, Line 2). In particular, it selects the FO (f_{tmp}) among the set that has the closest to zero distance to increase the chances of reducing the distance of f_{nom}. Then, in

Algorithm 13 Exhaustive Greedy extends Greedy

```
1: function BestAggregation(SF, f_nom)
2:     f_a ← f_nom; f_tmp ← null;
3:     for all f ∈ SF do
4:         f_y ←BinaryAggregation(f_nom, f)
5:         if D_{g,c}(f_y)< D_{g,c}(f_a) then
6:             f_a ← f_y; f_tmp ← f;
7:         end if
8:     end for
9:     return {f_a, f_tmp}
10: end function
```

Algorithm 14 Best binary aggregation function

Input: f_{nom}, f_{tmp} - FOs
Output: f_a - an AFO

```
1: function BinaryAggregation(f_nom, f_tmp)
2:     f_a ← f_nom
3:     for all alignment al of {f_nom,, f_tmp} do
4:         f_x ←AGG-2-to-1(f_nom, f_tmp, al)
5:         if D_{g,c}(f_x)<D_{g,c}(f_a) then
6:             f_a ← f_x
7:         end if
8:     end for
9:     return f_a
10: end function
```

each step, it examines all the potential aggregations between the two FOs, i.e, f_{nom} and f_{tmp} to identify the AFO that reduces the distance of f_{nom} (Algorithm 12, Line 3).

Exhaustive Greedy (EG). EG explores a larger solution space than SG. In particular, during each step, it examines *all* the potential binary aggregations between f_{nom} and all the FOs in set SF (Algorithm 13, Line 3) compared to SG that examines only the binary aggregations among f_{nom} and one FO from SF. EG then stores the AFO with the smallest distance (Line 6). When the comparisons finish, it returns the AFO with the minimum distance (f_a) and the FO (f_{tmp}) that participated in the production of f_a (Line 9).

Constraint allocation feature. Since aggregation should lead to a valid schedule, it is desirable to examine, after each step, whether the node constraint is respected or not. However, this would require to schedule during each step the current FOs/AFOs, i.e., solve the UC problem. Due to the fact that the UC problem is an NP-complete problem [32], our aggregation algo-

Device	EST	tf	#slices	Min amounts	af
EV (day)	6	5	4	$U^*\{5,7\}$	$U\{0,2\}$
EV (night)	$N^*(18,1)$, $[17,20]$	$N(10,1)$, $[8,12]$	$U\{3,4\}$	$U\{5,7\}$	$U\{0,2\}$
CW (day)	$N(16,1)$, $[15,17]$	$U\{1,3\}$	$U\{2,4\}$	$U\{3,4\}$, $U\{1,2\}$	0
CW (night)	$N(20,1)$, $[19,21]$	$N(8,1)$, $[5,10]$	$U\{2,4\}$	$U\{3,4\}$, $U\{1,2\}$	0
HP (day)	$N(13,1)$, $[12,14]$	$N(3,1)$, $[1,5]$	$U\{4,7\}$	$U\{5,8\}$	$U\{0,2\}$
HP (night)	17	3	$U\{3,6\}$	$U\{5,8\}$	$U\{0,2\}$
WT, PV (day)	$N(14,1)$, $[13,15]$	$U\{3,4\}$	$U\{4,10\}$	$U\{8,10\}$	$U\{0,2\}$
WT, PV (night)	$N(23,1)$, $[22,24]$	$U\{1,4\}$	$U\{5,8\}$	$U\{8,10\}$	$U\{0,2\}$

Table 5.1: Flex-objects characteristics, U^*: uniform distribution, N^*: Gaussian distribution

rithms instead act preventively in terms of constraint handling. In particular, it is possible for both algorithms to consider a constraint value lower than the original one. For instance, we typically allocate the constraint to 50% of its original value. As a result, the allocation feature obstructs aggregation to violate the constraint. Consequently, in cases where more than one AFOs have slice amounts closer to the constraint, it increases the chance for scheduling to form a node value that respects the constraint.

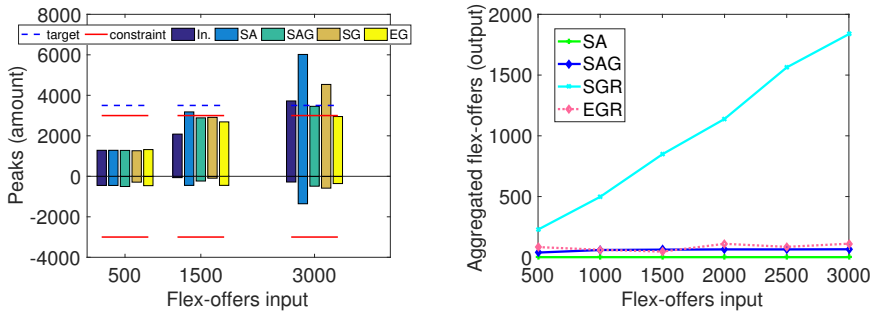
5 Experimental Evaluation

5.1 Experimental setup

We experimentally evaluate the proposed techniques in complex congestion scenarios. Our experiments are based on power characteristics from real loads (e.g., [3, 84]) that show similar use behavior and are complemented with potential flexibility, e.g., [69]. One amount unit corresponds to 0.5kW. We use a mixed portfolio of FOs that represents a variety of devices and characteristics regarding flexibility and power demand/supply. In particular, we generate 6 datasets of FOs with different sizes to be able to examine the scalability of the techniques in terms of input. The sizes of the datasets follow an arithmetic progression with both initial term and common difference equal to 500 FOs. Thus, the last dataset has 3K FOs. In order to create imbalances and congestion situations, the number of the negative FOs is 10% of every dataset.

In particular, 40% of the positive FOs represent electrical vehicles (EVs), 30% represent heat pumps (HPs), and 30% clothes washers (CWs). The negative FOs represent wind turbines (WT) and photovoltaics (PV) that are less flexible with longer profiles. We allocate FOs during day time and night time for all the devices (50%-50%). Details about the characteristics of the datasets are shown in Table 5.1.

Moreover, for comparison reasons we use two baseline aggregation techniques. We compare our techniques with Start Alignment (SA) aggregation [75] (see Section 3.1 where all the FOs are aggregated into one FO). We



(a) Peaks (b) #aggregated FOs

Fig. 5.4: $500 - 3K$ FOs, target=3.5K, constraint=3K, constraint aggregation allocation $= 1.5K$, $D_{g,c}(f) = 1 \cdot D_g + 10000 \cdot D_c$

also use a *Start Alignment* with *Grouping* (SAG) aggregation technique where a grouping phase is used in advance [75]. Consequently, FOs with the same earliest start time and the same time flexibility are grouped together and SA is applied on each group. As a result, for each group of FOs, a single AFO is produced.

In order to examine whether an aggregation result allows the constraint to be respected or not, we implemented a stochastic scheduling technique based on the Evolutionary Algorithm (EA) proposed in [80]. EA is applied on a set of FOs (aggregated or not) and forms the node value with the possible minimum distance to target and constraint function. We see in Figure 5.5 how the node value is formed when EA is applied on the results of each aggregation technique along with the constraint and the target function values.

The experiments were conducted on a 2.9 GHz Intel core i7 processor with two cores, physical memory of 8 GB, and MacOS. The techniques are implemented in Java 1.8.

5.2 Use case

We examine our techniques in a case where target is greater than the constraint so that a bottleneck appears. The grid power capacity constraint used in the experiments represents medium voltage grids, e.g., [25]. Target is 3500kW and constraint is 3000kW. We use a 0.5kW granularity (≈0.16‰ of the constraint value) and thus achieve a very fine resolution. We also use the constraint allocation feature. Thus, the constraint value used by the aggregation techniques is 1500. We set the target coefficient to 1 and we use a very high value for the constraint coefficient (10K) when the distance is computed, in order to prioritize the constraint respect.

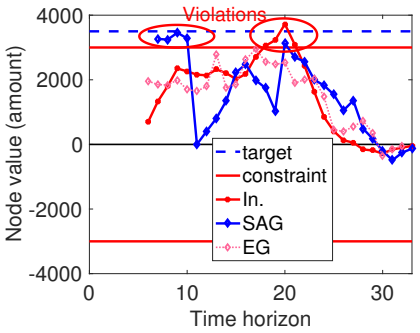
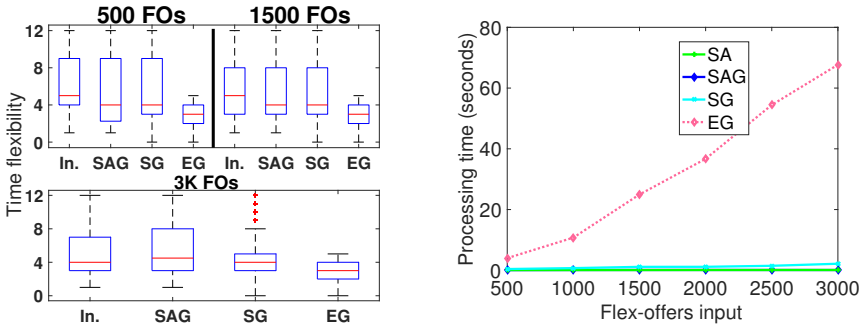


Fig. 5.5: 3K Flex-offers input

In Figure 5.4a, we see how the highest and the lowest node values (amount peaks) are formed when EA is applied on the initial ("In." label) non-aggregated set and on the aggregation result of each technique. For a better illustration we show the cases where the input is 500, 1500 and 3000 FOs. We see in Figure 5.4a that when the input size is 500, the peaks formed by EA (scheduling) are quite far away from the constraint and as the input size is increased, the peaks approach and finally exceed the constraint. In the same figure, we also observe that when the input size is small (500), all the techniques lead to scheduling that respects the constraint. When the size is increased to 1.5K FOs, SA violates the constraint, and in the last case of 3K FOs only EG respects the constraint.

Regarding the number of the AFOs, we see that SA produces only one AFO in all the input cases (Figure 5.4b) and its time flexibility is always 1, i.e., the minimum among all the FOs. We also notice that SAG, due to the grouping that applies, produces a low number of AFOs and also achieves a similar to the initial time flexibility distribution among the AFOs, see Figure 5.4c. However, the aggregation result of SAG violates the constraint when the input is increased to 2.5K and 3K (shown in Figure 5.4a). Furthermore, in Figure 5.5, we see how the node value is formed based on EA when the aggregation input is 3K. In the same figure, we notice that SAG not only violates the constraint, but there is also violation in 5 out of the 28 scheduling points (≈18% of the time horizon).

Regarding SG, the number of AFOs scales linearly with the input size and achieves a time flexibility higher than EG. However, EG is the *only algorithm* that forms a node value that *respects* the constraint **in all the input cases.** It maintains the number of AFOs low (93% input reduction on average) and uses time flexibility to lead to a schedule that respects the constraint. That is why it has the lowest average time flexibility and a distribution with low



(a) Time flexibility (b) Processing time

Fig. 5.6: $500 - 3K$ FOs, target=3.5K, constraint=3K, constraint aggregation allocation $= 1.5K$, $D_{g,c}(f) = 1 \cdot D_g + 10000 \cdot D_c$

boundaries, see Figure 5.6a.

Regarding the processing time, EG is the slowest algorithm due to the high number of comparisons it requires. It shows a similar to linear growth rate behavior, see Figure 5.6b. SG is fast since it only compares just two FOs in every step. Similarly, SA and SAG are the fastest algorithms due to the very low number of aggregations they perform.

Experimental summary. We observe that SG is fast and respects the constraint while SA does not. When size increases (>2K FOs), both SG and SAG violate the constraint. On the other hand, EG examines a larger solution space and leads to results that respect the constraint when the input size is large, see Figure 5.5, case of 3K FOs. It is indicative that even when we apply EA on the initial set of 3K FOs for 10 minutes, it still cannot provide a result that respects the constraint, see Figure 5.5 label "In.". On the contrary, EG uses approximately 67 seconds for its execution and EA applied afterwards produces the first result that respects the constraint in approximately two seconds. That means that EG is able to provide filtered inputs to scheduling so that initially unsolvable cases can be solved. However, when the input is large, EG requires high processing times. It requires 24.08 min. to process a dataset of 10K FOs.

6 Related Work

The role of an aggregator that handles flexible loads has been investigated in many previous works, e.g., [29, 33]. Such works use highly complex models and focus on controlling and scheduling methods. Their main characteristic is that the aggregator operates as an aggregated load controller that tries

to follow a power reference and eventually tackles the scheduling problem to offer DR and ancillary services, e.g., [18]. On the contrary, in our work we use a low complexity generic model to represent energy flexibilities, namely flex-offers (FOs). Moreover, the main goal of our techniques is to produce flexible and non-scheduled AFOs that can be traded as commodities in emerging energy flexibility markets. Thus, our proposed techniques, SG and EG, produce AFOs that can lead to normal grid operation and use a generic target function that can capture overall business case scenarios.

Furthermore, there is an extensive literature tackling the UC problem (scheduling), e.g., [49, 50]. In [80] the aggregation of FOs before scheduling showed an improvement of scheduling results compared to applying scheduling individually. Our work can be also applied in advance of scheduling process and not only reduces the complexity of the UC problem, but in addition, partially handles scheduling goals as it "filters" invalid results and improves their quality.

7 Conclusion and future work

This chapter introduces constraint-based aggregation over a generic data model that captures flexibilities in time and amount dimensions. It proposes two techniques that take into account the power capacity constraint limitations imposed by the grid. Moreover, the chapter evaluates the proposed techniques in complex congestion scenario. The experimental evaluation shows that the proposed techniques can efficiently aggregate FOs and at the same time enable scheduling to respect the grid constraints, unlike existing techniques.

In our future work, we will focus on enhancing our techniques by automating the setting of aggregation parameters through sampling techniques. Moreover, we will extend our proposed algorithms to investigate the financial perspective of constraint-based aggregation on the future energy market.

Chapter 6

Trading Aggregated Flex-Offers via Flexible Orders

The paper has been published in the
DBTR series, Aalborg University 2017.
The layout of the paper has been revised.

Abstract

Flexibility of small loads, in particular from Electric Vehicles (EVs), has recently attracted a lot of interest due to their possibility of participating in the energy market and the new commercial potentials. Different from existing works, the aggregation techniques proposed in this chapter produce flexible aggregated loads from EVs taking into account technical market requirements. The produced aggregated flexible loads fulfill the energy market requirements. They can be transformed into flexible orders and can be further traded in the day-ahead market by a Balance Responsible Party (BRP). As a result, the BRP achieves more than 27% cost reduction in energy purchase based on 2016 real electricity prices.

1 Introduction

The integration of EVs into the Smart Grid reveals new business opportunities by exploiting their inherent flexibility [37, 44]. A market actor that controls the charging rate and time of a portfolio of EVs could acquire financial gain from energy arbitrage [11, 19]. The energy required to charge (and/or discharge) the EVs can be traded through bids in day-ahead and/or regulation market at a minimum cost [71]. Numerous of research studies focus on trading the required energy to charge EVs taking into account different parameters.

An optimization charging approach of EVs that activates the participation in both day-ahead and regulation markets is proposed in [10]. Scheduling techniques of EV charging that aim to the maximization of the market actor's profit and take into account electricity price uncertainty are suggested in [85] and [94]. A risk-based scheduling framework for charging EVs is also proposed in [90]. The suggested algorithm is based on day-ahead prices and takes into account driving activity uncertainties in order to minimize the charging cost of the EVs. Similarly, a day-ahead optimization technique for scheduling EVs considering the impact on the day-ahead prices is suggested in [48]. Both optimization and heuristic techniques for optimal charging of EVs aiming to the maximization of the revenue by utilizing energy storage are proposed in [40].

The main characteristic of the research tackling the energy trading of flexible EV loads is the output of the proposed techniques, i.e., an aggregated scheduled load. Unlike other works, we introduce 3 aggregation techniques that produce *flexible* aggregated loads that can be traded in the market. As a result, the market, not the market actor, schedules the loads from the EVs as part of the trading process, minimizing the uncertainty of bidding. For instance, instead of placing a bid to purchase 30MW in hour 3, the market actor places a bid to purchase 30MW in any hour between hour 1 and 5. The market determines the activation time of the bid. In many cases the technical trading details of the market are omitted and the realization of the suggested techniques becomes very difficult. For instance, the proposed scheduling technique in [24] offers less than 200kW in less than an hour in the regulation power market where the minimum bid is 10MW, in full hours [11]. The bidding strategies proposed in [9] and the high power fluctuations of the scheduling outputs in [10] and in [71], would require single hour independent bids [67] that might not fulfill the energy requirements. A conservative bidding approach for the bidding strategy proposed in [86] covers less than 50% of the energy needed to charge the EVs. On the contrary, in our work we use real technical market requirements derived from a specific order (bid) type, i.e., *flexible order*, as objectives of our proposed techniques.

Contributions. First, we describe both the so-called flex-offer (flex-offer) model, which captures the flexibility of the EVs, and a realistic market framework where the flexibility is traded. Second, we investigate the market-based FO aggregation problem and its complexity. Third, we introduce 3 heuristic algorithms that take into account real market requirements and produce flexible aggregated FOs that can be further traded through flexible orders in the market. Finally, we compare our proposed techniques with 2 base-line approaches and evaluate both the technical and the financial aspect of their results based on real market prices. We show that our proposed techniques achieve more than 20% cost reduction on average in the purchased energy required to charge from 5K to 40K EVs.

The chapter is organized as follows: we introduce the preliminary definitions in Section 2 and we present the problem formulation of market-based aggregation in Section 3. In Section 4, we propose 3 heuristic market-based aggregation techniques and we experimentally evaluate them in Section 5. We conclude the chapter in Section 6.

2 Preliminaries

In this section, we describe the model of the appropriate device that can be used to trade flexibility and the market framework that is used for trading.

2.1 Electric vehicle model

We consider the energy used to charge EVs to be appropriate for flexible energy trading. The reason is that the lithium-ion batteries of EVs are ample power demand devices and their charge can be time shifted when the EVs are plugged-in for more hours than needed for charging. We take into account EVs that can be continuously charged with a power-constant voltage (CP-CV) option [81] and their charge is taking place within 20% and 90% of the state of charge (SOC) so that the battery life is preserved [54].

In our work, because we take into account time shifted loads, we use the flex-offer (FO) concept [75] to represent the charging of a flexible EV. Thus, we consider an *flex-offer* f to be a tuple $f = (T(f), P(f))$ where $T(f)$ is the start charging flexibility interval and $P(f)$ is the power profile. $T(f) = [t_{es}, t_{ls}]$ where t_{es} and t_{ls} are the *earliest start charging time* and *latest start charging time*, respectively. We consider time flexibility (tf) to be the difference between t_{ls} and t_{es}. The *power profile* is a sequence of ($m \in \mathbb{N}_{>0}$) consecutive slices, $P(f) = \langle s^{(1)}, \ldots, s^{(m)} \rangle$ where a *slice* $s^{(i)}$ has a power value p measured in kW. The duration of slices is 1 hour.

For instance, an EV is plugged-in at a house between 1 and 8 a.m. The EV continuously utilizes 3.7kW for 3.3 hours to be charged. However, energy

trading occurs hourly and we also use hourly resolution to model the EVs charging. To respect the hourly granularity, we equally distribute the sum of the energy needed during the first and the last regular charging hours and we reduce power deviations in the model. Therefore, we assume that the EV mentioned above consumes 2.4kWh both during the first and the last charging hours and 3.7kWh during the intermediate hours. The EV can be modeled by an FO $f=([1,4],\langle 2.4,3.7,3.7,2.4\rangle)$, see Figure 6.1a. Next, we describe the market framework where such FOs shall be traded.

2.2 Market framework

The Nordic/Baltic market for electrical energy named Nord Pool is considered in our work. Nord Pool is one of the most mature energy markets [87] and Europe's leading power market [67]. It consists of the day-ahead (Elspot) and intra-day markets. We focus on Elspot because it has one of the largest turnovers in the Nordic system and it also supports flexible energy trading [11]. Trading in Elspot occurs daily through orders (bids). The orders specify the energy amount a BRP desires to buy/sell and the price BRP is willing to pay/be paid for the corresponding energy. Since 2016, Elspot supports *flexible orders* [67].

When a BRP places a flexible order in Elspot, it states the *name*, the *time interval*, the *price limit*, the *volume*, and the *duration* of the order. The time unit is one hour and volume is expressed in MW. The duration expresses the number of hours during which the order can be activated. The time interval must exceed the duration by at least one hour and expresses the potential activation times of the order. A BRP can place 5 flexible orders during a trading day.

Hypothetically, a BRP could purchase the energy needed to charge the above mentioned flexible EV, represented by FO f through a flexible order. The duration of a flexible order is mapped to the number of slices of f, the volume to the power of the slices, and the time interval to the time flexibility of f. For instance, a BRP could place a flexible order named "F1", with duration 4 hours and time interval from 1 to 8. The volume of F1 is 0.0037MW (in order to satisfy all the slices) and its price limit is 35 euros/MWh. However, the energy needed to charge a single EV is too small to be traded in Elspot. In particular, the minimum contract size and the volume trade lot for a flexible order are both 100kW, while the power used by an EV is a few kW. Moreover, when the duration of a flexible order is more than one hour, the volume needed for these hours shall be constant. As a result, it is necessary to *aggregate* FOs to trade the flexible loads of the EVs through flexible orders in Elspot market.

The flexible order is activated in the time interval that optimizes social welfare provided that the price is respected [67]. Given F1 in a liquid market,

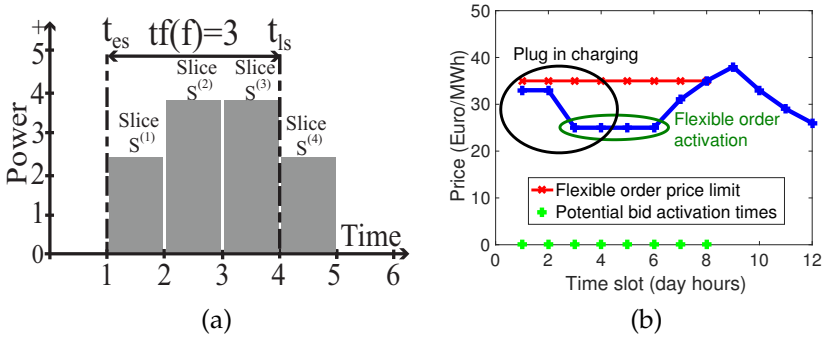


Fig. 6.1: An examle of an FO and a flexible purchase order

the order is activated when the cost of buying the required energy is minimized. For instance, we see in Figure 6.1b that F1 is activated in time slots 3, 4, 5, and 6 where the price is 25 euros/MWh. Thus, the energy needed to charge the EV costs $25 \cdot 0.0037 \cdot 4 = 0.37$ euros. On the contrary, if time flexibility of the EV is disregarded, its charging occurs based on a price independent order and its plug-in time (time slot 1-4 in Figure 6.1b). In that case and according to Figure 6.1b, the cost is $33 \cdot 0.0037 \cdot 2 + 25 \cdot 0.0037 \cdot 2 = 0.4292$ euros, 16% more than the cost achieved by flexible order F1. The absolute difference (imbalance) between the purchased energy and the energy needed is traded in the balance market and usually for a higher price than the one in Elspot. Consequently, the BRP desires to be as precise as possible regarding the purchased energy from Elspot.

3 Problem Formulation

In this section, we show how aggregation of FOs that represent flexible charging loads of EVs can fulfill the requirements of flexible purchase orders. We also introduce the problem of market-based aggregation.

3.1 FO aggregation

Based on [75], FO aggregation is the function that given a set of FOs F, produces a set of aggregated FOs AF where $|AF| \leq |F|$. Due to the time flexibility of the FOs, there are different alignment combinations that can lead to different AFOs. According to start-alignment FO aggregation, the earliest start charging time of an aggregated FO (AFO) f_a is the minimum earliest start charging time among all the FOs that produced it, i.e., $f_a.t_{es} = min_{f \in F'}(f.t_{es}), F' \subseteq F$. The latest start charging time of f_a is the sum of its t_{es} and the minimum time flexibility among all the FOs in F', i.e.,

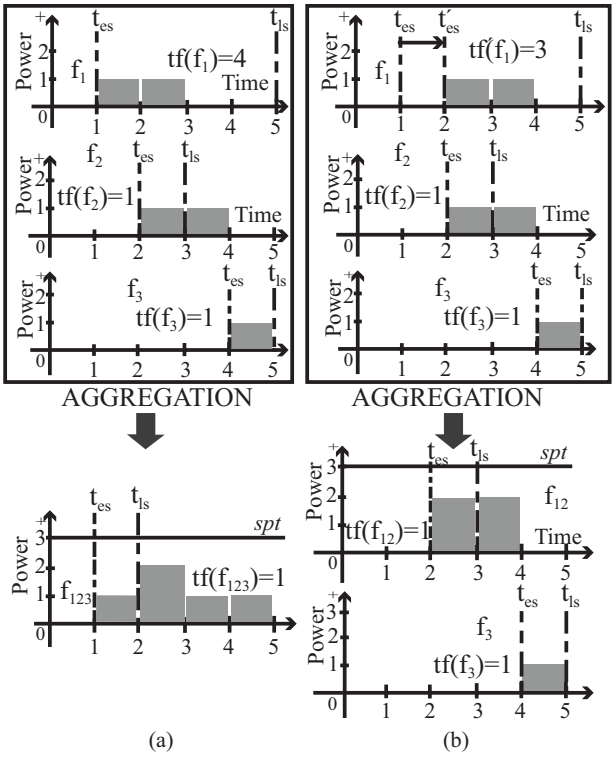


Fig. 6.2: Flex-offer aggregation according to different alignments

$f_a.t_{ls} = f_a.t_{es} + min_{f \in F}(tf(f))$. The power profile of f_a is produced by summing up the power profiles of the FOs when they are aligned according to their earliest start charging time.

For instance, we see in Figure 6.2a three FOs, $f_1 = ([1,5], \langle 1,1 \rangle)$ $f_2 = ([2,3], \langle 1,1 \rangle)$, and $f_3 = ([4,5], \langle 1 \rangle)$, that produce AFO f_{123} where $f_{123}.t_{es} = f_1.t_{es} = 1$ and $f_{123}.t_{ls}$ is the sum of $f_{123}.t_{es}$ and time flexibility of f_2 or f_3, i.e., $f_{123}.t_{ls} = 2$. The power profile of f_{123} is produced by summing up the power profiles of f_1, f_2, and f_3 based on their alignments. Thus, $f_{123}.s^{(1)}.p = f_1.s^{(1)}.p = 1$, $f_{123}.s^{(2)}.p = f_1.s^{(2)}.p + f_2.s^{(1)}.p = 2$, $f_{123}.s^{(3)}.p = f_2.s^{(2)}.p = 1$, and $f_{123}.s^{(4)}.p = f_3.s^{(1)}.p = 1$.

Due to the time flexibility of the FOs, there are different alignment combinations that can lead to different AFOs. For instance, given the 3 FOs f_1, f_2, f_3 in Figure 6.2 with time flexibility 4, 1, and 1, respectively, there are 20 $(5 \cdot 2 \cdot 2)$ alignment combinations. As a result, based on different alignments, time flexibility of the FOs can be adjusted accordingly and different power profiles for the AFOs are produced.

Moreover, a set of FOs can be partitioned and each subset can produce an

AFO. Consequently, the output size of aggregation can be greater than one. For instance, we see in Figure 6.2b that the output of aggregation is 2 AFOs, i.e., f_{12} and f_3. In particular, FO f_1 is aligned with f_2 and time flexibility of f_1 is adjusted so that its $f_1.t'_{es}$ is equal to $f_2.t_{es}$. Consequently, the power profiles of f_1 and f_2 are summed up and they produce AFO f_{12}.

3.2 Market-based Flex-Offer aggregation

Given a portfolio, the goal of a BRP is to maximize its profit by purchasing, for the minimum price, the energy that it sells to its customers. We consider flexible EVs to be part of a BRP's portfolio and, since the energy purchase takes place through orders, we examine if the energy needed to charge the flexible EVs can be purchased through flexible orders. The purchasing strategy of a BRP depends on many different factors, e.g., the content of the portfolio (factories, households, etc.) and pricing forecast. The strategy is out of scope of this work and left for Future work. However, since a flexible order has in general a higher probability to achieve a lower purchase price, we consider the goal of a BRP to be the maximization of the purchased energy through flexible orders.

In order for an FO to fulfill the flexible order requirements, the FO must have (1) time flexibility at least one and (2) between 1 and 23 slices. Moreover, since the minimum contract size and the trade lot of a flexible order are both 100kW, (3) the values of the slices of the FOs shall be multiples of 100kW. In our work, we introduce *market-based FO aggregation* to be the aggregation that given a set of FOs, outputs at least one AFO that fulfills the flexible order requirements. For illustrating purposes, we assume in our example below that *both the volume and the trade lot for a flexible order is 2kW instead of 100kW*. For instance, we see in Figure 6.2 that none of the individual FOs fulfills the power profile requirements of a flexible order (2kW). Thus, market-based FO aggregation is necessary. In that case, market-based FO aggregation produces AFO f_{12} that fulfills the flexible-order requirements since its time flexibility is 1 and the power of both the slices equals to 2, see Figure 6.2b. FO f_3 is also part of the aggregation output, but it is not a valid AFO because it does not fulfill the power profile requirement, i.e., its slice amount is lower than 2.

The flexible EVs are represented by a set of FOs. For instance, 5000 EVs that are part of a BRP's portfolio are represented by a set of FOs F. Each EV is an FO f of the set, i.e., $f \in F, f = (T(f), P(f)), T(f) = [t_{es}, t_{ls}], P(f) = \langle s^{(1)}, \ldots, s^{(m)} \rangle$. A BRP must aggregate the FOs to produce AFOs that fulfill the flexible order requirements and can be further placed in the market as flexible orders. The volume of energy is expressed through the sum of the slices of the FOs and the power of each slice must be a multiple of 100kW. However, due to technical charging characteristics (EV power demand is in the interval [3.7kW,11kW] for household charging), we take into account a

power range to define the valid power amounts. Thus, instead of considering exact multiples of 100kW for the power amount of each slice, we permit an insignificant amount deviation of ekW per slice, e.g., 5kW. When the financial evaluation of market-based aggregation occurs, the deviated amount will be considered to be traded in balance market, see Section 2.2. Hence, the problem of maximizing the bidden energy through flexible orders given a set of FOs is formulated as follows:

$$
\begin{aligned}
\text{Maximize} \quad & \sum_{f_a \in A} \sum_{s \in P(f_a)} s.p \\
\text{subject to} \quad & A = MAGG(F), 1 \leq |A| \leq 5 \\
& \forall f_a \in A, tf(f_a) \geq 1 \\
& \forall f_a \in A, 1 \leq |P(f_a)| \leq 23 \\
& \forall f_a \in A, \forall s \in P(f_a), \\
& s.p = x \cdot 100\text{kW} \pm e\text{kW}, x \in \mathbb{N}_{>0}, e \in [0, 5]
\end{aligned}
$$

Market-based FO aggregation complexity. Given a set of flex-offers F, the number of aggregation results that can be produced is: $\sum_{k=1}^{5} \{^{|F|}_{k}\} \cdot avg(al)$ where $avg(al)$ is the average number of alignments per partition [31]. For instance, given a set with 100 flex-offers and 20 alignments per partition on average, there are in total $1.3148 \cdot 10^{69}$ potential aggregation results (approximately the estimated number of atoms in the Milky Way Galaxy) that have to be examined in order to find the optimal one. Furthermore, tackling the problem as an Integer Linear Programming (ILP) problem, requires on the order of $\mathcal{O}(avg(al) \cdot |F| \cdot \sum_{k=1}^{5} \{^{|F|}_{k}\})$ decision variables to identify the partition(s) and the aggregation(s) that maximize the bidden energy. The complexity of the problem is thus too high to be solved by state-of-the-art solvers [58]. The reader can find a comprehensive version of the complexity analysis in Appendix A of the chapter.

4 Heuristic solutions

Due to the unrealistically large solution space, we instead propose 3 variations of a heuristic algorithm, i.e., *Heuristic Market-based Aggregation Main Algorithm* (HMAMA) that tackles the market-based aggregation problem.

4.1 Heuristic Market-based Aggregation Main Algorithm

The goal of HMAMA is to produce AFOs that respect the flexible order requirements while avoiding the high complexity of the problem and at the same time provide good results in terms of bidden energy amount. Thus, given a set of FOs F, HMAMA (Algorithm 15) performs incremental binary

Algorithm 15 Heuristic Market-Based Aggregation

Input: F - set of flex-offers, e - amount deviation
Output: AF - set of aggregated flex-offers
1: $continue \leftarrow true$, $AF \leftarrow \emptyset$
2: **while** $continue = true$ **do**
3: $ppt \leftarrow 23$, $spt \leftarrow 100$
4: $PF, UF, f_{ini}, tft \leftarrow$Initialize($F$)
5: $PF, AF \leftarrow$Process($PF, AF, f_{ini}, tft, ppt, spt, e$)
6: $F, continue \leftarrow$Examine($PF, UF, AF, continue$)
7: **end while**
8: **return** Top5EnergyAFOs(AF)

aggregations so that the produced AFOs increase the captured energy in each step. In addition, the algorithm maps the flexible order requirements to threshold parameters that must be respected during the performed aggregations. Consequently, it introduces 3 thresholds, namely, the slice power (*spt*), time flexibility (*tft*), and power profile (*ppt*) thresholds that correspond to flexible order requirements. It sets *spt* to 100 since flexible orders must have multiples of 100kW power. Moreover, HMAMA assigns 1 and 23 to *tft* and *ppt*, respectively, since flexible orders must have a time interval of 1 and duration at most 23 hours. Permitted amount deviation is represented by e that is assigned values from 0kW to 5kW.

The body of HMAMA consists of 3 phases (functions), i.e., *initialization*, *processing*, and *examination* (Lines 2–7). During the initialization phase (Line 4), HMAMA identifies the FO with which to start binary aggregations (f_{ini}) and the subset of the FOs (*PF*) that participates in the aggregations. Then, during the processing phase (Line 5), it produces all the potential binary aggregations between f_{ini} and the FOs in *PF* to produce AFOs that fulfill the flexible order requirements. Afterwards, during the examination phase (Line 6), HMAMA examines whether it shall restart using the remaining FOs or terminate.

4.2 Main Algorithm variants

The initialization phase is salient for the outcome of the algorithm as it mainly defines the solution space that the algorithm explores. Hence, we introduce 3 variants of HMAMA that have different initialization phases, namely, the *Largest Profile* (LP), *Dynamic Profile* (DP), and *Dynamic Time Flexibility* (DTF).

LP focuses on producing AFOs with many slices because an FO with many slices usually captures large energy amounts. On the other hand, given an FO with many slices, it is very difficult to fulfill the flexible order amount requirements and, especially, the slice amount equality required. For this

Algorithm 16 Longest Profile - Initialization phase

1: **function** INITIALIZE(F)
2: $f_{ini} \leftarrow$SelectAmongLongestTheMostFlexibleFO(F)
3: **return** $F \setminus f_{ini}, \emptyset, f_{ini}, 1$
4: **end function**

Algorithm 17 Initialization phase - Dynamic Profile algorithm

1: **function** INITIALIZE(F)
2: $uf \leftarrow$UpperFenceProfileSize(F)
3: $PF \leftarrow$FOsWithProfileAtLeastUF(F, uf)
4: $f_{ini} \leftarrow$SelectTheMostFlexibleFOAmongLongest(PF)
5: **return** $PF \setminus f_{ini}, F \setminus PF, f_{ini}, 1$
6: **end function**

reason, DP excludes from aggregation the FOs with extremely large profiles (outliers). DTF focuses on time flexibility of the FOs that has a prominent role in aggregation since it is directly correlated to the alignments. Thus, DTF takes into account the time flexibility distribution of the initial set and gradually excludes from aggregation the FOs with low time flexibility compared to the initial set.

LP - Initialization phase. LP starts by selecting the most flexible FO among the ones with the largest profile size (Algorithm 16, Line 2). An FO with large profile size and high time flexibility has high probability to time-wise overlap with profiles of other FOs. So, AFOs that fulfill the flexible order requirements through different alignments can be produced. LP uses the initial set F as the processing set PF (Line 3) and then executes the processing and examination phase.

DP - Initialization phase. During the initialization phase, DP divides the initial set F into 2 subsets. First, DP computes the upper fence (uf) [55] of the power profile size of the FOs in F (Algorithm 17 Line 2). Then, it stores in PF the FOs that have profile size of at most uf (Line 3). It selects as f_{ini} the most flexible FO in PF among the ones with the longest profile and removes it from PF (Lines 4–5). For instance, given the set F in Figure 6.3a ($\{f_1, \ldots, f_6\}$), uf is 4, see Figure 6.3b. DP excludes f_1, which has a very long profile compared to the other FOs (red circle in Figure 6.3b), from F and selects FO f_6 as f_{ini}. FOs with very long profiles have difficulties satisfying the slice equality and it is likely that they have small time flexibility due to their long profiles (e.g., many charging hours for the EVs). Thus, they have less potential alignments to further satisfy the flexible order requirements. Then, DP continues aggregation with the processing and examination phase using PF, i.e., $F \setminus \{f_{ini} \cup f_1\}$.

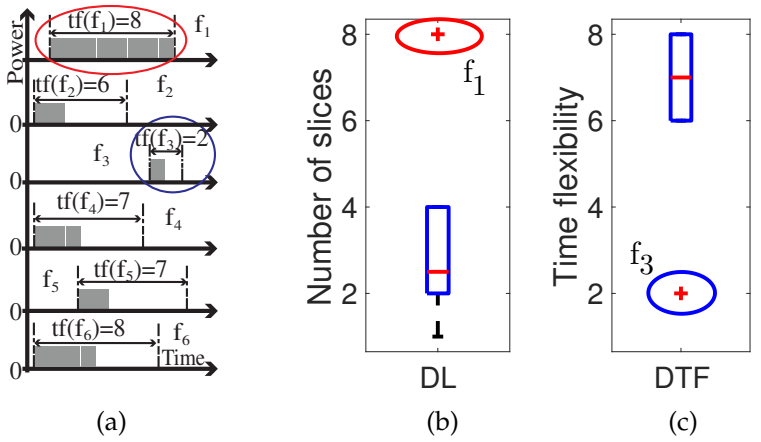


Fig. 6.3: DL and DTF example, profile size and time flexibility box plots

Algorithm 18 Initialization phase - Dynamic Time Flexibility

1: **function** INITIALIZE(F)
2: $tft \leftarrow$LowerFenceTimeFlexibility(F)
3: $PF \leftarrow$FOsWithTimeFlexibilityAtLeasttft(F, tft)
4: $f_{ini} \leftarrow$SelectTheMostFlexibleFOAmongLongest(PF)
5: **return** $PF \setminus f_{ini}, F \setminus PF, f_{ini}, tft$
6: **end function**

DTF - Initialization phase. DTF takes into account the time flexibility distribution of the initial set F and excludes FOs with low time flexibility compared to the initial set. It computes the lower fence of time flexibility distribution of F and sets the time flexibility threshold (tft) equal to the lower fence [55] (Algorithm 18 Line 2). It splits F based on the lower fence of the time flexibility distribution in the set. It stores the FOs with time flexibility at least tft in PF (Line 3). DTF then selects f_{ini} from PF (Line 4). As a result, the algorithm excludes the FOs that have very small time flexibility. For instance, given the set F in Figure 6.3a, tft equals 6, Figure 6.3c. Thus, DTF excludes f_3, which has very low time flexibility compared to the other flex-offers in the set, from F, see the blue circle in Figure 6.3c. DTF then sets tft to 6, selects FO f_1 as f_{ini}, and continues aggregation with PF, i.e., $F \setminus \{f_{ini} \cup f_3\}$. FOs with small time flexibility have lower probability to contribute in aggregation due to the low number of alignments that they have. Moreover, by setting tft equal to the lower fence, DTF reduces the number of examined alignments and consequently the complexity of the algorithm. Thus, AFOs with greater time flexibility are more likely to be produced.

Processing phase. In the processing phase, HMAMA examines all the

Algorithm 19 Processing phase

```
1: function Process(PF, AF, f_ini, tft, ppt, spt, e)
2:     PF_tmp ← ∅, f_a ← null
3:     for all f ∈ PF do
4:         f_cand ← null, bestCV ← ∞
5:         for all alignment al of {f_ini, f} do
6:             f_x ←BinaryAggregation(f_ini, f, al, tft, ppt)
7:             if RMSE(f_x, spt)<RMSE(f_ini, spt) then
8:                 if CV(f_x)<bestCV then
9:                     bestCV ←CV(f_x), f_cand ← f_x
10:                end if
11:            end if
12:        end for
13:        if f_cand ≠ null then
14:            PF_tmp ← PF_tmp ∪ f, f_ini ← f_cand
15:        end if
16:        if ∀s ∈ P(f_ini), spt − e < s.p < spt + e then
17:            f_a ← f_ini
18:            PF ← PF \ PF_tmp
19:            PF_tmp ← ∅, spt ← spt+100
20:        end if
21:    end for
22:    return PF, AF ∪ f_a
23: end function
```

potential binary aggregations between f_{ini} and the FOs in PF defined in the initialization phase. The FOs are examined in descending order according to their time flexibility. FOs with high time flexibility have more potential to participate in an aggregation that fulfills the flexible order requirements because of high number of alignments.

HMAMA examines, through the potential alignments, all the binary aggregations that fulfill the time flexibility *tft* and the power profile thresholds *ppt* (Algorithm 19, Lines 3–5). Among the AFOs that reduce the root mean square error (RMSE) between f_{ini} and the slice power threshold *spt*, it chooses the one with the minimum coefficient of variation (CV) (Lines 7–11). By promoting the reduction of RMSE, the produced AFO f_{cand} has a power profile closer to *spt*. In particular, the use of RMSE during aggregation prevents the increase of profile length of the potential AFO and contributes to the production of slices with values closer to *spt*. Consequently, alignments that lead to power profiles that time-wise overlap each other are preferred for aggregation. Moreover, because the slices of an AFO might have power deviations, the second condition of CV (Line 9) is used. A low CV of f_{cand} contributes

Algorithm 20 Examination phase

```
1: function EXAMINE(PF, UF, AF, continue)
2:     if PF ∪ UF=∅ OR (|AF|≥5 and
           totalEnergy(PF∪UF)<Energy5th AFO(AF)) then
3:         continue ← false
4:     end if
5:     return PF ∪ UF, continue
6: end function
```

to the elimination of power profile deviations and to the production of AFOs with slice power amounts closer to each other. For instance, given the FOs in Figure 6.2 and *spt* equal to 3, the RMSE between the slices of AFO f_{12} and *spt* is equal to 1 and lower than the RMSE between the longest FO f_{123} and *spt*, which is 1.8028. Similarly, f_{12} and f_{123} have CV equal to 0 and 0.4, respectively, with f_{12} having no power fluctuations. Thus, the reduction of RMSE and CV lead to AFOs that fulfill the flexible order energy requirements.

When an AFO with power amounts around *spt* is produced, an *e* kW deviation per slice is permitted (Algorithm 19 Line 16). At that point, an AFO f_a that fulfills the flexible order criteria is produced (Line 17). The FOs that participate in aggregation are temporally stored (Line 14) and when an AFO f_a is produced, they are removed from *PF* (Line 18). Then, *spt* is increased by 100 (Line 19) so that AFOs with larger energy are produced during the following aggregation. As a result, the processing phase produces an AFO that captures large amounts of energy and fulfills the time flexibility and power amount requirements of a flexible order. When all the FOs in *PF* are processed, HMAMA returns both *PF* and the output set *AF* with the aggregated FO f_a (Line 22).

Examination phase. During the examination phase, HMAMA first examines if there are any FOs in either *PF* or *UF* to further continue aggregation (Algorithm 20 Line 2). In case, the total energy of the remaining FOs is larger than the 5th in descending size energy AFO, HMAMA continues using the remaining FOs (Line 5). Otherwise, HMAMA does not continue the execution (Line 3). As a result, the algorithm ensures that the remaining FOs cannot produce an AFO with energy greater than one of the 5 produced AFOs. Since the 5 AFOs with the most energy will be transformed to flexible orders, the algorithm terminates (Algorithm 15 Line 8).

	Distr.	Mean	St. dev	Min	Max
Battery capacity (kWh)	UD∗	23	4	16	30
Arrival time	TGD∗	19:00	2h	16:00	1:00
Departure time	TGD∗	7:00	2h	5:00	12:00
Initial Battery SOE (%)	TGD∗	75	25	20	85

∗ UD: uniform distribution, TGD: truncated Gaussian distribution

Table 6.1: EV data probability distribution

5 Experimental Evaluation

5.1 Experimental setup

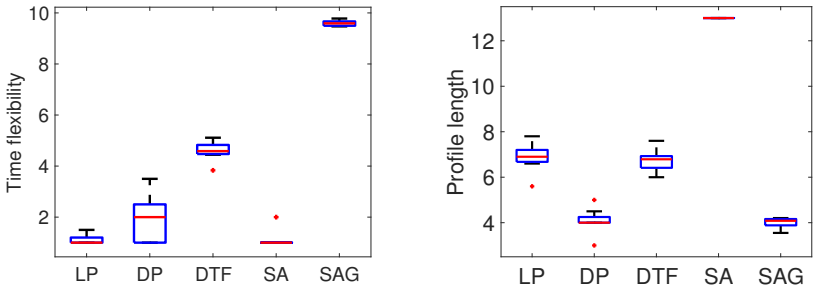
We consider a BRP managing a portfolio of EVs represented by flex-offers. The BRP utilizes our proposed aggregation algorithms to produce AFOs that respect the flexible order requirements. The BRP transforms the 5 AFOs which capture the highest amount of energy to flexible orders and trades them in Elspot. In order to examine the scalability of our proposed algorithms, we create 8 differently-sized flex-offer datasets, from 5K to 40K flex-offers (multiples of 5K), with characteristics based on the probability distributions suggested in [81]. Moreover, we consider that all EVs use the charging option described in Section 2.1 and need to be fully charged. Thus, the initial SOC of all EVs is within [20%, 85%], while they must be charged up to 90%. Details about the characteristics of the datasets are in Table 6.1.

We compare our techniques with two baseline aggregation techniques [75]. We use Start-Alignment (SA) aggregation, see Section 3.1 and Start-Alignment Grouping (SAG) aggregation. SAG groups together FOs that have both the same earliest start charging time and the same time flexibility and then applies SA on each group. As a result, it produces one AFO per group. We evaluate our techniques in terms of output size (#AFOs), participation of FOs in aggregation, percentage of energy traded in the market, running time, and both time flexibility and profile length of AFOs.

5.2 Market-based aggregation results

Output size. SA always produces one AFO whereas SAG produces more than 100 AFOs in all cases. Both LP and DP produce less than or equal to 5 AFOs in all cases. DTF produces more than 5 AFOs in 75% of the cases as the energy threshold is activated in a later step compared to the other techniques due to the division of the processed set.

Time flexibility and profile length. Regarding the baseline techniques, SA produces long AFOs with very low time flexibility as it aggregates all FOs into one. On the contrary, SAG produces short and time flexible AFOs



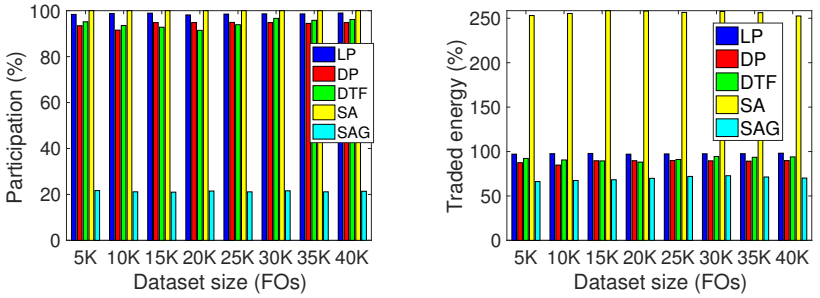
(a) Avg. time flexibility (b) Avg. profile length

Fig. 6.4: Average time flexibility and average profile length

due to the grouping phase it applies, see Figure 6.4a, b. LP uses as initial FO (f_{ini}) the longest FO of the dataset. Usually, such an FO has low time flexibility and so do the produced AFOs. Due to the long profile of f_{ini}, LP might utilize all the time flexibility of the remaining FOs to produce an AFO that reduces the distance to the power profile threshold (*ppt*). Consequently, LP produces long AFOs with very low time flexibility, see Figure 6.4a, b. The AFOs produced by DP are more flexible than the ones from LP since DP applies a dynamic profile size approach and excludes from aggregation very long FOs. As a result, FOs with similar profiles are aggregated together and less time flexibility is required to find a proper alignment that minimizes the distance to *ppt*. Consequently, AFOs with less slices compared to LP are produced, see Figure 6.4b. Finally, DTF produces the most flexible AFOs among our proposed techniques. We see in Figure 6.4b that the average time flexibility of the produced AFOs is greater than 4 in all datasets. DTF achieves it by utilizing the time flexibility threshold. However, DTF produces long AFOs, similar to LP, because it also selects as f_{ini} the longest AFO of the processed set, see Figure 6.4b.

Participation and traded energy. In order to quantify the participation of FOs in aggregation, we take into account only the FOs that participate in the aggregation of the 5 (or less) largest in energy AFOs, i.e., the AFOs that are transformed into flexible orders. Similarly, we compute the traded energy by taking into account only the energy captured by the AFOs that are transformed to flexible orders.

SA aggregates all FOs into one AFO and thus participation in aggregation is 100%, see Figure 6.5a. The slices of the AFO have very high power differences and since a flexible order requires a flat power profile, the power of the highest slice is considered for the whole profile of the AFO. As a result, on average, 2.5 times the energy captured by that AFO is traded, see Figure 6.5b.

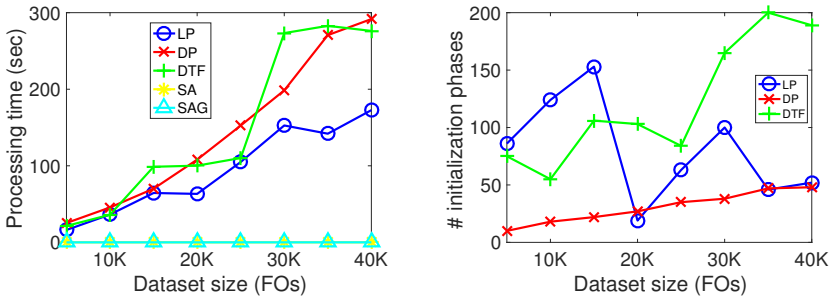


(a) Participation in aggregation (b) Energy bidden

Fig. 6.5: Participation of FOs and energy bidden

On the contrary, SAG produces too many AFOs and since only the 5 largest are traded, we see a very low participation percentage and the lowest percentage of traded energy among the techniques (69.7% on average). In general, the longest AFOs capture more energy as they have more slices and more FOs participate in their aggregation. Thus, LP, which produces the longest AFOs, obtains both the highest participation percentage (98.6%) and energy bidden percentage (97.5%) in all the cases, see in Figure 6.5a, b. DTF follows with an average participation value of 94.4% and 91.7% percentage of bidden energy. DP has the lowest percentage in both participation and energy bidden, 94.2% and 88.8% on average respectively. The reason is that DP excludes very long FOs, which usually capture large energy, from aggregation.

Processing time. Both SA and SAG are fast techniques with processing times below one second as they examine a very small solution space and do not consider the market requirements. LP is the fastest among all our proposed techniques since it efficiently activates the energy threshold, see Figure 6.6a. The processing time of DP follows a close to linear growth rate. DTF has an increasing trend for processing time, but it shows similar processing times for datasets with different sizes, e.g., for datasets with 30K and 35K FOs. The reason is that the processing time is highly driven by the number of initialization phases. The size of the dataset might increase, but the new added FOs might lead to less initialization phases and therefore to less aggregation comparisons. That is why we also notice that both processing time and number of initialization phases follow similar patterns. Whenever the number of initialization phases is increased compared to the previous dataset, processing time also increases. For instance, we see in Figure 6.6b that when the size of the dataset is increased from 15K to 20K for both LP and DTF, the number of initialization phases is reduced. As a result, the processing time is similar for both the datasets and slightly increases for the



(a) Processing time (b) # of initialization phases

Fig. 6.6: Processing time and number of initialization phases for all datasets

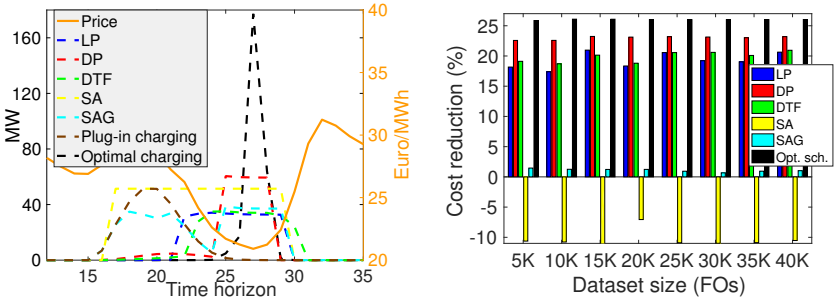
25K. Eventually, when the size of the dataset is further increased, it becomes more difficult for DTF to fulfill the market requirements and thus both the initialization phases and the processing time are highly increased.

5.3 Financial evaluation

Since the overall goal of a BRP is to trade the AFOs in the market using flexible orders, we financially evaluate our aggregation techniques. We compare the cost of buying the energy needed to charge the EVs based on plug-in time (traditional approach) with the cost of charging the EVs by utilizing flexible orders. Moreover, in order to compare our techniques with the optimal solution, we consider a *fictitious* scenario where each FO directly participates in the market without aggregation and each EV is charged when the charging cost is minimized.

Due to the fact that flexibility appears during the night [40], we consider a 48 hours trading period with a repetition of the 24h Elspot average prices of 2016 [67], see price curve in Figure 6.7a. In the same figure, we illustrate the time and the energy amount used to charge the 40K dataset based on our techniques, the two baseline techniques, the plug-in times of the EVs, and the optimal charging. We see that the charging of the EVs based on the plug-in time occurs when the prices are still high and it does not take advantage of the price drop that occurs in the night of the first 24 hours.

SA and SAG produce AFOs that do not fulfill the market requirements. As a result, more energy than it is needed has to be traded in the market. In particular, SA trades 1.52 times more energy than needed to charge the EVs. Thus, the surplus energy is traded in the regulation market and it results in losses for the BRP, see negative cost reduction in Figure 6.7b. Regarding SAG, the produced AFOs capture a low percentage of the energy needed



(a) Charging time vs pricing (b) Cost reduction

Fig. 6.7: Charging times and pricing for 40K dataset, and cost reduction for all datasets

and they also require extra energy to be traded in order to fulfill the market requirements. Consequently, the cost reduction due to the flexible orders trading is compensated by the losses from the surplus energy trading. As a result, we see only 1.1% cost reduction on average when SAG is applied. On the contrary, the optimal charging option charges all the EVs when the price has the lowest value. That is why we see a spike in the graph reaching 180MW after the 24^{th} hour.

Our proposed aggregation techniques also take advantage of the lowest prices. LP produces long AFOs which expand over many hours and have low time flexibility. That is why we see in Figure 6.7a that part of the charging occurs when the prices are high. DTF produces AFOs that are also long, but they are more flexible than the AFOs produced by LP. Therefore, EVs are charged when prices are a bit lower and DTF achieves a higher cost reduction, Figure 6.7b. Finally, DP produces short and flexible AFOs. As a result, it takes advantage of the lowest prices occurring only for a few hours, see Figure 6.7a.

When the energy for the 40K FOs dataset is bought based on the plug-in times of the EVs, it costs 7,521 euros. On the contrary, when LP, with the highest participation, is applied on the 40K dataset, 39,584 flex-offers participate in aggregation, see first bar (98.96%) in Figure 6.5a. The 39,584 flex-offers produce 5 AFOs which are further transformed to flexible orders. The cost of buying the energy needed for the 5 AFOs is computed based on the flexible orders trading and it is 5851 euros, see Figure 6.7a. The price also includes the cost (0.40 euro) of the imbalances (62kW) of the flexible orders, see Section 2.2. The energy needed for the remaining 416 $(40,000 - 39,584)$ flex-offers is bought based on their plug-in time and it is 117 euros. Thus, the overall energy bought to charge 40K EVs, when LP is used, costs $5,851 + 117 = 5,968$ euros. Therefore, LP achieves a 20.65% cost reduction in energy

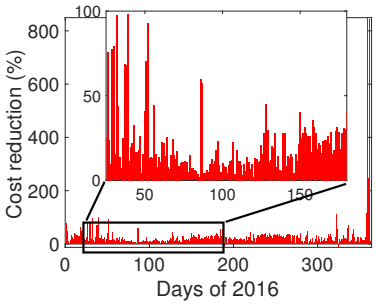


Fig. 6.8: Cost reduction based on DP

buying, see LP bar for 40K dataset in Figure 6.7b.

We see in Figure 6.7b that DP achieves on average a 23.01% cost reduction. DTF follows with 19.87% and LP with 19.29% average cost reduction respectively. The cost reduction based on the optimal solution is 26.01% on average. Thus, LP, DTF, and DP achieve 74.2%, 76.4%, and 88.5% of the optimal cost reduction, respectively. Notably, the cost reduction that DP achieves *only* for the flex-offers that participate in aggregation is on average 96.6% of the optimal one.

In Figure 6.8, we illustrate the cost reduction that DP achieves during 2016 (leap year). We consider 365 trading periods of 48 hours. The first trading period includes both the 1^{st} and the 2^{nd} day of 2016. The second trading period includes the 2^{nd} and the 3^{rd} day of 2016 and so on. The average cost reduction is 27.57% and, interestingly, we notice at the end of the year a cost reduction of more than 800%. The reason is that for several consecutive days, Elspot prices were negative early in the morning and even reached -53 euros/MWh on the 27^{th} of December at 3:00.

Summary: By applying our proposed techniques on the aforementioned 365 trading periods, DP achieves the highest cost reduction in 52% of the periods and DTF achieves the highest cost reduction in the remaining 48% of the periods. The reason is that the financial impact of the techniques is highly correlated with the pricing curve of the trading period. Thus, in cases where the price drops for only few hours close to the plug-in charging time, DP is the most suitable technique. On the other hand, when the price drops for longer periods but much later than the plug-in charging time, DTF achieves a higher reduction than DP.

6 Conclusion and Future work

This chapter investigates the market-based aggregation problem using the FO model that captures flexible charging loads of EVs. It proposes 3 market-based FO aggregation techniques that efficiently aggregate loads from thousand of EVs taking into account real market requirements. Consequently, the techniques produce aggregated FOs that can be further transformed to flexible orders and be traded in the energy market. The chapter financially evaluates the proposed techniques based on real electricity prices and shows that a 27% cost reduction on energy purchase can be achieved via flexible orders.

In our future work, we will enrich our techniques considering pricing forecast models and uncertainty in driving patterns. Moreover, we will examine a price-maker market scenario and different market strategies for the BRPs.

A Appendix

In this appendix, we describe the complexity of market-based flex-offer aggregation.

A.1 Number of solutions

Given a set of flex-offers F, there are $\{{}^{|F|}_{k}\}$ ways (Stirling numbers of the second kind [31]) to partition the $|F|$ flex-offers into k subsets. Applying aggregation on each subset produces an AFO. In market-based FO aggregation, the size of the output is between 1 and 5. Thus, k can be assigned values from 1 to 5. Therefore, there are $\{{}^{|F|}_{1}\}$ ways to partition $|F|$ flex-offers into 1 non-empty subset of flex-offers. There are $\{{}^{|F|}_{2}\}$ ways to partition the $|F|$ flex-offers into 2 non-empty subsets, where the aggregated flex-offers are 2 and so on, given $|F|$ flex-offers, there are $\{{}^{|F|}_{1}\} + \{{}^{|F|}_{2}\} + \cdots + \{{}^{|F|}_{5}\} = \sum_{k=1}^{5} \{{}^{|F|}_{k}\}$ ways to partition the flex-offers.

Moreover, the number of the different aggregated flex-offers depends on the alignments of the flex-offers that participate in aggregation and thus on their time flexibility. In particular, given a set of flex-offers SF ($SF \subseteq F$) with time flexibility $tf(f_1), \ldots, tf(f_{|SF|})$ respectively, the number of the aggregation results (aggregated flex-offers) that can be produced is: $\prod_{i=1}^{|SF|} tf(f_i)$.

Example A.1
Given a set with 100 flex-offers there are $\sum_{k=1}^{5} \{{}^{100}_{k}\} = 6.5738 \cdot 10^{67}$ potential partitions that can produce from 1 to 5 AFOs. Assuming a lower bound

of 4 alignments per partition on average, there are in total $4 \cdot 6.5738 \cdot 10^{67} = 2.62952 \cdot 10^{68}$ potential aggregation results that have to be examined in order to find the optimal one.

A.2 Integer Linear Programming problem complexity

Given a set of flex-offers F, there are $\left\{ {|F| \atop k} \right\}$ ways (Stirling numbers of the second kind [31]) to partition the $|F|$ flex-offers into k subsets. Applying aggregation on each subset produces an AFO. In market-based FO aggregation, the size of the output is between 1 and 5. Thus, k can be assigned values from 1 to 5. Therefore, there are $\left\{ {|F| \atop 1} \right\}$ ways to partition $|F|$ flex-offers into 1 non-empty subset of flex-offers. There are $\left\{ {|F| \atop 2} \right\}$ ways to partition the $|F|$ flex-offers into 2 non-empty subsets, where the aggregated flex-offers are 2 and so on... Given $|F|$ flex-offers, there are $\left\{ {|F| \atop 1} \right\} + \left\{ {|F| \atop 2} \right\} + \cdots + \left\{ {|F| \atop 5} \right\} = \sum_{k=1}^{5} \left\{ {|F| \atop k} \right\}$ ways to partition the flex-offers. As a result, tackling the market-based aggregation problem as an Integer Linear Programming (ILP) problem, requires on the order of $\mathcal{O}(|F| \times \sum_{k=1}^{5} \left\{ {|F| \atop k} \right\})$ decision variables to identify the partition(s) that maximizes the bidden energy.

Moreover, the number of the different aggregated flex-offers depends on the alignments of the flex-offers and thus on their time flexibility. Therefore, for each FO, all potential start charging times have to be examined and that requires a number of decision variables equal to the time flexibility. Hence, given an average time flexibility $\overline{tf}$ of set F, the number of the decision variables needed to identify the partitions and the aggregations that maximize the bidden energy is on the order of $\mathcal{O}(\overline{tf} \times |F| \times \sum_{k=1}^{5} \left\{ {|F| \atop k} \right\})$.

For instance, given a set with 100 flex-offers there are $\sum_{k=1}^{5} \left\{ {100 \atop k} \right\} = 6.5738 \cdot 10^{67}$ potential partitions that can produce from 1 to 5 AFOs. Thus, $100 \cdot 6.5738 \cdot 10^{67}$ variables are needed to identify the proper subset(s) for all the FOs. Moreover, given an average time flexibility of 5, $5 \cdot 100 \cdot 6.5738 \cdot 10^{67} = 3.2869 \cdot 10^{70}$ variables are needed to identify the aggregation result that maximizes the bidden energy.

Chapter 7

Conclusions and Future Research Directions

Abstract

This chapter summarizes the conclusions and directions for future work presented in Chapter 2 - Chapter 6 and across the whole thesis.

1 Summary of Results

The thesis presents different types of aggregation techniques for flex-offers that enable them to be traded in a flexibility market. According to the Demand Response concept, the energy flexibility of individual prosumers can contribute to increasing the use of energy from RES and confronting the Smart Grid challenges. Subsequently, flex-offers, which capture energy flexibility, become valuable and the TotalFlex project aims to design a market where flex-offers can be traded. However, flex-offers cannot be traded unless they are aggregated. As a result, the Ph.D. project introduces flex-offer aggregation techniques that efficiently aggregate flex-offers, reduce scheduling complexity, and enable flex-offers to be traded in the market. The proposed aggregation techniques focus on energy balancing issues, electrical grid constraints handling, and market trading. In the rest of the section, the outcome of each chapter is summarized and the overall contribution of the Ph.D. thesis is discussed.

Chapter 2 introduces several flexibility measurements to measure and evaluate the flexibility of both individual flex-offers and groups of flex-offers. Flexibility measurements are also used for the design and the evaluation of aggregation techniques and their outcome. Since a flex-offer captures flexibil-

ity in time and energy dimensions, the introduced flexibility measurements take into account the effect of dimensions both individually and combined. Depending on the scenario according to which flexibility is used and the type of the flex-offers that are considered, different flexibility measurements can be used. In cases where mixed flex-offers participate in aggregation and energy balancing is an aggregation requirement, the product, the assignment and the vector flexibility measurements can be used. They all capture the combined effect of time and energy dimensions, and both time and energy flexibility are essential for aggregation techniques to achieve energy balancing. Moreover, the participation of both positive and negative flex-offers is essential for balance aggregation, while the produced aggregated flex-offers are usually mixed flex-offers. All the aforementioned flexibility measurements can efficiently capture flexibility of all types of flex-offers. Under a scenario where flex-offers are traded and the desirable flexibility has to be quantified and compared either on an individual or a grouped level, measurements that capture time and energy flexibility individually might be used, e.g., the time and energy flexibility, the absolute and the relative area-based flexibility.

Chapter 3 presents the base-line start alignment aggregation and the start-alignment aggregation with grouping. An extensive experimental setup of differently sized datasets shows that the techniques efficiently aggregate flex-offers and minimize the flexibility losses. The experiments also show the trade-off between aggregation and flexibility. In particular, the number of aggregated flex-offers is driven by the selected grouping parameters. When the grouping parameters favor aggregation and reduce the number of produced aggregated flex-offers, flexibility losses increase. Chapter 3 also presents a disaggregation technique with processing time that scales linearly with the aggregation processing time. In addition, Chapter 3 introduces 5 balance aggregation techniques and examines them under an energy scenario where zero absolute balance is possible. Both exhaustive and zero terminated exhaustive search are able to identify the minimum absolute balance. However, they have extremely high processing times when the size of the input exceeds the few thousand flex-offers. Similarly, dynamic simulated annealing achieves a low absolute balance, but it also has a very high computational time when the size of the initial flex-offers is increased. On the contrary, the two proposed heuristic techniques, so-called simple and exhaustive greedy, achieve a close to optimum absolute balance and low processing times. In particular, both simple and exhaustive greedy achieve better absolute balance than the start alignment aggregation and exhaustive greedy shows lower flexibility losses than simple greedy. However, simple greedy is faster than exhaustive greedy as it explores a smaller solution space.

Chapter 4 examines the aggregation techniques in terms of scalability and grouping parameters using from 11K to 90K flex-offers. The experimental setup shows that there are cases where the grouping parameters favor start

alignment aggregation and the technique outperforms the greedy approaches in terms of balance. However, there are cases where the datasets include flex-offers with low time flexibility and thus, start alignment has very high flexibility losses. This occurs when the time flexibility tolerance is increased and the number of aggregated flex-offers is also greater than when zero grouping parameters are used. On the contrary, the greedy techniques show a better balance outcome when flex-offers with low time flexibility and long profiles participate in aggregation. Particularly, in cases where grouping parameters have values above zero, the two deviations of the greedy techniques outperform start alignment in terms of balance and flexibility losses. The proposed balance aggregation techniques in Chapter 3 and Chapter 4 reduce the number of flex-offers and, at the same time, produce aggregated flex-offers with desirable balance. Flex-offers are also the scheduling problem input and a high number of flex-offers results in extremely high processing times for scheduling techniques. As a result, the complexity of the scheduling problem is reduced and, at the same time, one of its main goals is partially fulfilled, i.e., to identify schedules that balance out energy demand and supply.

Chapter 5 presents two constraint-based aggregation techniques that take into account technical constraints imposed by the power capacity of the electrical power grid. In particular, Chapter 5 shows the extremely high complexity of the constraint-based aggregation problem and introduces two variations of a greedy algorithm, i.e., the simple and the exhaustive greedy. The experimental setup discussed in Chapter 5 considers a bottleneck of electrical grid where several thousands of devices represented by flex-offers are connected to the grid. Moreover, Chapter 5 compares the proposed constraint-based aggregation techniques to the baseline aggregation techniques introduced in Chapter 3. Part of the experimental setup is also the application of a stochastic scheduling algorithm that is applied on the result of each aggregation technique. The scheduling algorithm schedules the produced aggregated flex-offers, in order to identify a scheduling that respects the electrical grid constraint (valid schedule). The experimental results show that all the aggregation techniques lead to a valid schedule when the size of the input is considerably small. As the input size is increased, start alignment fails to produce a result that can lead to a valid schedule. When the input size exceeds 2K flex-offers, only exhaustive greedy leads to a valid schedule. However, exhaustive greedy explores a larger solution space and it is the slowest among the techniques. Simple greedy is faster than exhaustive greedy and achieves less time flexibility losses, yet it produces more aggregated flex-offers. The reason is that exhaustive greedy exploits the time flexibility of the flex-offers to produce aggregated flex-offers that can lead to a valid schedule. In the case where the input size is 3K flex-offers and the scheduling algorithm is applied on the initial (non-aggregated) flex-offers, it cannot identify a valid schedule even after 10 minutes of execution. On the contrary, when the scheduling

algorithm is combined with exhaustive greedy, a valid schedule is generated in less than 70 seconds taking into account both aggregation and scheduling processing times.

Chapter 6 presents three market-based aggregation techniques. The proposed aggregation techniques consider the market requirements as hard constraints and thus, the produced aggregated flex-offers always fulfill the market requirements. Subsequently, the flex-offers can be transformed to flexible orders and be traded in the day-ahead market. The experimental datasets used in Chapter 6 consist of flex-offers that represent EVs and the size of the datasets is from 5K to 40K. All the proposed aggregation techniques show a very high participation percentage of the flex-offers in aggregation. Hence, the vast majority of the flex-offers participating in aggregation can also be traded in the market via flexible orders. The produced aggregated flex-offers that are traded in the market also capture a very high percentage of energy compared to the energy captured by the initial population of flex-offers. On the contrary, start alignment aggregation with grouping as introduced in Chapter 3 achieves to trade subsequently lower amounts of energy compared to the market-based aggregation techniques. Chapter 6 also financially evaluates the proposed aggregation techniques based on real market prices in 2016. All proposed techniques achieve more than 19% cost reduction in all cases compared to the cost of purchasing energy based on the plug-in times of the EVs. On the contrary, start alignment with grouping aggregation achieves extremely low cost reductions. Moreover, start alignment aggregation achieves negative cost reductions. Furthermore, the market-based aggregation techniques take advantage of the negative prices that appeared late in December 2016. As a result, the BRP, which handles the portfolio of flex-offers, can trade the flex-offers via flexible orders and gain up to 7 times the cost of charging them based on their plug-in time.

To conclude, the Ph.D. thesis demonstrates the prominent role of energy flexibility in the Smart Grid and the importance of aggregation in supporting that role. The Ph.D. thesis shows that through aggregation, energy flexibility can be used to balance out energy demand and supply and to confront electrical grid constraints. Additionally, the proposed aggregation techniques reduce the complexity of the scheduling problem and partitially fulfill the scheduling objectives. Thus, there are cases where aggregation is combined with scheduling and a valid solution is identified, whereas applying scheduling individually fails to provide a valid solution. The output of the aggregation techniques is still energy flexibility but at an aggregated level. Finally, the Ph.D. thesis shows that the trading of such aggregated flexibilities can be realized given current real market scenarios.

2 Future Research Directions

A discussion about the future research directions across the whole Ph.D. thesis follows.

Regarding Chapter 2, a commonly accepted flexibility measurement is still pending. Chapter 2 suggests several flexibility measurements that can be used under different energy scenarios taking into account different devices. However, a generic flexibility measurement that could capture the requirements defined by Chapter 2 is still pending. Moreover, flexibility is also used by the scheduling process. Therefore, the impact that different measurements have on the outcome of the scheduling process shall be investigated.

In the case of Chapter 3, the grouping phase plays an important role in implementing balance aggregation techniques. In particular, it reduces the high complexity of the problem, but at the same time it reduces the chances of identifying the optimum solution. Therefore, more advanced grouping solutions and their integration with the aggregation techniques shall be further examined to reduce the processing time and minimize the absolute balance.

The comprehensive experimental setup in Chapter 4 shows the correlation between the characteristics of the datasets and the performance of aggregation techniques. Depending on the time flexibility and the profile length distributions that the flex-offers follow, the appropriate aggregation technique shall be applied. Thus, sampling techniques in order to identify the distributions that the dataset follows shall be introduced.

The heuristic techniques introduced in Chapter 5 are based on binary aggregations and face challenges identifying a valid solution when the input size is increased considerably. Thus, potential statistical measurements on the overall population shall be used during each execution step of the techniques to improve their efficiency. Moreover, the financial impact of the constraint-based aggregation shall be evaluated and the participation of the flex-offers in aggregation shall also be prioritized. As a result, more flex-offers and subsequently more prosumers will financially benefit from the constraint-based aggregation solutions.

Several future research directions also exist for the market-based aggregation techniques discussed in Chapter 6. In particular, the proposed techniques shall also take into account price signal forecasts so that the cost reduction of the purchase energy is maximized. Moreover, a market scenario where trading of flex-offers might influence the energy price shall also be considered, as the traded volumes of the aggregated flex-offers are relatively high. Finally, there is also uncertainty in flexibility, e.g., in driving patterns, that should be taken into consideration during aggregation as well.

Overall, the Smart Grid faces many challenges and even more due to increasing intermittent RES. Energy flexibility shall be used on a large scale to

support Smart Grid in achieving its goals. As a result, aggregation techniques for energy flexibility should support a very big number of flex-offers from different geographical areas. A hierarchical aggregation approach that could solve local electrical grid congestions at a local level and reassure energy balance on a broader level (e.g., national) will be required. Additionally, potential contradictions among the goals of aggregation at different geographical areas and levels should also be confronted. Finally, due to the high complexity of the aggregation problem, distributed and parallel processing techniques should also be investigated.

References

[1] Totalflex project, link: `www.totalflex.dk`.

[2] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. Wiley, 1 edition, 1988.

[3] J. Acosta, K. Combe, S. Djokic, and I. Hernando-Gil. Performance assessment of micro and small-scale wind turbines in urban areas. *Systems Journal*, 6:152–163, 2012.

[4] M. Aigner. A characterization of the bell numbers. *Discrete Mathematics*, 205:207 – 210, 1999.

[5] M. Albano, L. L. Ferreira, L. M. Pinho, and A. R. Alkhawaja. Message-oriented middleware for smart grids. *Computer Standards & Interfaces*, 38:133 – 143, 2015.

[6] A. Arasu and J. Widom. Resource Sharing in Continuous Sliding-Window Aggregates. In *International conference on Very Large Data Bases (VLDB)*, pages 336–347, 2004.

[7] E. W. E. Association. Creating the internal energy market in europe. Technical report, European Wind Energy Association, 2012.

[8] B. Bach, D. Wilhelmer, and P. Palensky. Smart buildings, smart cities and governing innovation in the new millennium. In *8th IEEE International Conference on Industrial Informatics (INDIN)*, pages 8–14, 2010.

[9] L. Baringo and R. S. Amaro. A stochastic robust optimization approach for the bidding strategy of an electric vehicle aggregator. *Electric Power Systems Research*, 146:362 – 370, 2017.

[10] R. J. Bessa, M. A. Matos, F. J. Soares, and J. A. P. Lopes. Optimized bidding of a ev aggregation agent in the electricity market. *IEEE Transactions on Smart Grid*, 3:443–452, 2012.

[11] B. Biegel, L. H. Hansen, J. Stoustrup, P. Andersen, and S. Harbo. Value of flexible consumption in the electricity markets. *Energy*, 66:354 – 362, 2014.

[12] B. Biegel, M. Westenholz, L. H. Hansen, J. Stoustrup, P. Andersen, and S. Harbo. Integration of flexible consumers in the ancillary service markets. *Energy*, 67:479 – 489, 2014.

[13] M. Boehm, L. Dannecker, A. Doms, E. Dovgan, B. Filipič, U. Fischer, W. Lehner, T. B. Pedersen, Y. Pitarch, L. Šikšnys, and T. Tušar. Data management in the mirabel smart grid system. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 95–102, 2012.

[14] M. H. Böhlen, J. Gamper, and C. S. Jensen. How Would You Like to Aggregate Your Temporal Data? In *International Symposium on Temporal Representation and Reasoning*, pages 121–136, 2006.

[15] M. H. Böhlen, J. Gamper, and C. S. Jensen. Multi-dimensional Aggregation for Temporal Data. In *International Conference on Extending Database Technology (EDBT)*, pages 257–275, 2006.

[16] M. Bucher, S. Chatzivasileiadis, and G. Andersson. Managing flexibility in multi-area power systems. *CoRR*, abs/1409.2234, 2014.

[17] J. Cabot, J.-N. Mazón, J. Pardillo, and J. Trujillo. Specifying aggregation functions in multidimensional models with OCL. In *International Conference on Conceptual Modeling*, pages 419–432, 2010.

[18] H. Cai, A. Hutter, E. Olivero, P. Roduit, and P. Ferrez. Load shifting for tertiary control power provision. In *2015 IEEE 5th International Conference on Power Engineering, Energy and Electrical Drives (POWERENG)*, pages 469–475, 2015.

[19] Y. Cao, S. Tang, C. Li, P. Zhang, Y. Tan, Z. Zhang, and J. Li. An optimized ev charging model considering tou price and soc curve. *IEEE Transactions on Smart Grid*, 3:388–393, 2012.

[20] C.-Y. Chow, M. F. Mokbel, and T. He. Aggregate Location Monitoring for Wireless Sensor Networks: A Histogram-Based Approach. In *International Conference on Mobile Data Management: Systems, Services and Middleware (MDM)*, pages 82–91, 2009.

[21] K. Clement-Nyns, E. Haesen, and J. Driesen. The impact of charging plug-in hybrid electric vehicles on a residential distribution grid. *Power Systems*, 25:371–380, 2010.

[22] L. N. D. Brodén, C. Sandels. Assessment of congestion management potential in distribution networks using demand-response and battery energy storage. In *2015 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–10, 2015.

[23] W. H. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1:7–24, 1984.

[24] I. Diaz De Cerio Mendaza, I. Szczesny, J. Pillai, and B. Bak-Jensen. Demand response control in low voltage grids for technical and commercial aggregation services. *IEEE Transactions on Smart Grid*, 7:2771–2780, 2016.

[25] W. P. Distribution. Generation capacity register. `https://www.westernpower.co.uk/Connections/Generation`. Accessed: 2016-05-05.

[26] H. Farhangi. The path of the smart grid. *IEEE Power and Energy Magazine*, 8:18–28, 2010.

[27] L. L. Ferreira, L. Siksnys, P. Pedersen, P. Stluka, C. Chrysoulas, T. le Guilly, M. Albano, A. Skou, C. Teixeira, and T. Pedersen. Arrowhead compliant virtual market of energy. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–8, 2014.

[28] D. Gao, J. A. G. Gendrano, B. Moon, R. T. Snodgrass, M. Park, B. C. Huang, and J. M. Rodrigue. Main Memory-Based Algorithms for Efficient Parallel Aggregation for Temporal Databases. *Distributed and Parallel Databases*, 16:123–163, 2004.

[29] X. Geng and P. Khargonekar. Electric vehicles as flexible loads: Algorithms to optimize aggregate behavior. In *2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, pages 430–435, 2012.

[30] J. Gordevičius, J. Gamper, and M. Böhlen. Parsimonious temporal aggregation. In *International Conference on Extending Database Technology (EDBT)*, pages 1006–1017, 2009.

[31] R. L. Graham. *Concrete mathematics: a foundation for computer science*. Pearson Education India, 1994.

[32] X. Guan, Q. Zhai, and A. Papalexopoulos. Optimization based methods for unit commitment: Lagrangian relaxation versus general mixed integer programming. In *IEEE Power Engineering Society General Meeting (PESGM)*, volume 2, pages 1095–1100, 2003.

[33] H. Hao, B. Sanandaji, K. Poolla, and T. Vincent. Aggregate flexibility of thermostatically controlled loads. *Power Systems*, 30:189–198, 2015.

[34] H. Hermanns and H. Wiechmann. Future design challenges for electric energy supply. In *IEEE Conference on Emerging Technologies Factory Automation*, pages 1–8, 2009.

[35] S. Hosseini, A. Khodaei, and F. Aminifar. A novel straightforward unit commitment method for large-scale power systems. *IEEE Transactions on Power Systems*, 22:2134–2143, 2007.

[36] G. Hou, R. Yao, J. Ren, and C. Hu. A clustering algorithm based on matrix over high dimensional data stream. In *International conference on Web Information Systems and Mining (WISM)*, pages 86–94, 2010.

[37] J. Hu, H. Morais, T. Sousa, and M. Lind. Electric vehicle fleet management in smart grids: A review of services, optimization and control aspects. *Renewable and Sustainable Energy Reviews*, 56:1207 – 1226, 2016.

[38] C. S. Jensen, D. Lin, and B. C. Ooi. Continuous Clustering of Moving Objects. *Knowledge and Data Engineering, IEEE Transactions on*, 19:1161–1174, 2007.

[39] C. Jin and J. G. Carbonell. Incremental aggregation on multiple continuous queries. In *International conference on Foundations of Intelligent Systems (ISMIS)*, pages 167–177, 2006.

[40] C. Jin, J. Tang, and P. Ghosh. Optimizing electric vehicle charging with energy storage in the electricity market. *IEEE Transactions on Smart Grid*, 4:311–320, 2013.

[41] D. Kaulakienė, L. Šikšnys, and Y. Pitarch. Towards the automated extraction of flexibilities from electricity time series. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 267–272, 2013.

[42] S. Kazarlis, A. Bakirtzis, and V. Petridis. A genetic algorithm solution to the unit commitment problem. *IEEE Transactions on Power Systems*, 11:83–92, 1996.

[43] R. E. Korf. A new algorithm for optimal bin packing. In *Eighteenth National Conference on Artificial Intelligence*, pages 731–736, 2002.

[44] T. K. Kristoffersen, K. Capion, and P. Meibom. Optimal charging of electric drive vehicles in a market environment. *Applied Energy*, 88:1940 – 1948, 2011.

[45] F. Kupzog and C. Roesener. A closer look on load management. In *2007 5th IEEE International Conference on Industrial Informatics*, volume 2, pages 1151–1156, 2007.

[46] J. A. Lee and M. Verleysen. M.: Generalization of the lp norm for time series and its application to self-organizing maps. In *In: COTTRELL, M. (Hrsg.): Proceedings of Workshop on Self-Organizing Maps (WSOM), S. 733–740*, 2005.

[47] G. Lei, X. Yu, X. Yang, and S. Chen. An incremental clustering algorithm based on grid. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 2, pages 1099–1103, 2011.

[48] Z. Liu, Q. Wu, S. Huang, L. Wang, M. Shahidehpour, and Y. Xue. Optimal day-ahead charging scheduling of electric vehicles through an aggregative game model. *IEEE Transactions on Smart Grid*, PP, 2017.

[49] T. Logenthiran, D. Srinivasan, A. Khambadkone, and H. N. Aung. Multiagent system for real-time operation of a microgrid in real-time digital simulator. *IEEE Transactions on Smart Grid,*, 3:925–933, 2012.

[50] T. Logenthiran, D. Srinivasan, and A. M. Khambadkone. Multi-agent system for energy resource scheduling of integrated microgrids in a distributed system. *Electric Power Systems Research*, 81:138 – 148, 2011.

[51] J. Lopes, F. Soares, and P. Almeida. Integration of electric vehicles in the electric power system. *Proceedings of the IEEE*, 99:168–183, 2011.

[52] J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *5th Berkeley Symposium on Math. Stat. and Prob*, pages 281–297, 1967.

[53] E. Malinowski and E. Zimányi. *Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications*. Springer, 1 edition, 2008.

[54] F. Marra, G. Y. Yang, C. Træholt, E. Larsen, C. N. Rasmussen, and S. You. Demand profile study of battery electric vehicle under different charging options. In *Power and Energy Society General Meeting, IEEE*, pages 1–7, 2012.

[55] B. I. Michael Frigge, David C. Hoaglin. Some implementations of the boxplot. *The American Statistician*, 43, 1989.

[56] MIRABEL Consortium. The MIRABEL project, 2010. `http://www.mirabel-project.eu/`.

[57] R. Mitra, V. Arya, B. Sullivan, R. Mueller, H. Storey, and G. Labut. Using analytics to minimize errors in the connectivity model of a power distribution network. In *Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems*, pages 179–188, 2015.

[58] H. Mittelmann. Benchmark of commercial LP solvers. `http://plato.asu.edu/ftp/lpcom.html`, 2016.

[59] A. H. Mohsenian-Rad, V. W. S. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia. Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. *IEEE Transactions on Smart Grid*, 1:320–331, 2010.

[60] B. Moon, I. Fernando Vega Lopez, and V. Immanuel. Efficient Algorithms for Large-Scale Temporal Aggregation. *Knowledge and Data Engineering, IEEE Transactions on*, 15:744–759, 2003.

[61] B. Neupane, T. B. Pedersen, and B. Thiesson. Evaluating the value of flexibility in energy regulation markets. In *Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems*, pages 131–140, 2015.

[62] N. P. Padhy. Unit commitment-a bibliographical survey. *IEEE Transactions on Power Systems*, 19:1196–1205, 2004.

[63] N. H. Park and W. S. Lee. Statistical grid-based clustering over data streams. *ACM Special Interest Group on Management of Data (SIGMOD) Record*, 33:32–37, 2004.

[64] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. A foundation for capturing and querying complex multidimensional data. *Information systems*, 26:383–423, 2001.

[65] M. Petersen, K. Edlund, L. Hansen, J. Bendtsen, and J. Stoustrup. A taxonomy for modeling flexibility and a computationally efficient algorithm for dispatch in smart grids. In *American Control Conference*, pages 1150–1156, 2013.

[66] K. Pollhammer, F. Kupzog, T. Gamauf, and M. Kremen. Modeling of demand side shifting potentials for smart power grids. In *AFRICON*, pages 1–5, 2011.

[67] N. pool AS. Nord pool market. `http://www.nordpoolspot.com`.

[68] S. Rezaee, E. Farjah, and B. Khorramdel. Probabilistic analysis of plug-in electric vehicles impact on electrical grid through homes and parking lots. *Sustainable Energy, IEEE Transactions on*, 4:1024–1033, 2013.

[69] I. Sajjad, G. Chicco, and R. Napoli. Demand flexibility time intervals for aggregate residential load patterns. In *IEEE PowerTech*, pages 1–6, 2015.

[70] F. Salah, J. P. Ilg, C. M. Flath, H. Basse, and C. van Dinther. Impact of electric vehicles on distribution substations: A swiss case study. *Applied Energy*, 137:88 – 96, 2015.

[71] M. R. Sarker, Y. Dvorkin, and M. A. Ortega-Vazquez. Optimal participation of an electric vehicle aggregator in day-ahead energy and reserve markets. *IEEE Transactions on Power Systems*, 31:3506–3515, 2016.

[72] R. Sibson. SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16:30–34, 1973.

[73] L. Šikšnys, M. E. Khalefa, and T. B. Pedersen. Aggregating and disaggregating flexibility objects. In *Scientific and Statistical Database Management: 24th International Conference, Proceedings*, pages 379–396, 2012.

[74] L. Siksnys, C. Thomsen, and T. B. Pedersen. Mirabel dw: Managing complex energy data in a smart grid. In *Proceedings of the 14th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, pages 443–457, 2012.

[75] L. Siksnys, E. Valsomatzis, K. Hose, and T. B. Pedersen. Aggregating and disaggregating flexibility objects. *Knowledge and Data Engineering, IEEE Transactions on*, 27:2893–2906, 2015.

[76] Y. N. Silva, A. M. Aly, W. G. Aref, and P.-A. Larson. SimDB: A similarity-aware database system. In *ACM Special Interest Group on Management of Data (SIGMOD)*, pages 1243–1246, 2010.

[77] Y. N. Silva, W. G. Aref, and M. H. Ali. Similarity Group-By. In *IEEE International Conference on Data Engineering (ICDE)*, pages 904–915, 2009.

[78] D. Srinivasan and J. Chazelas. A priority list-based evolutionary algorithm to solve large scale unit commitment problem. In *International Conference on Power System Technology*, volume 2, pages 1746–1751, 2004.

[79] T. Tusar, E. Dovgan, and B. Filipic. Evolutionary scheduling of flexible offers for balancing electricity supply and demand. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8, 2012.

[80] T. Tušar, L. Šikšnys, T. B. Pedersen, E. Dovgan, and B. Filipič. Using aggregation to improve the scheduling of flexible energy offers. *International Conference on Bioinspired Optimization Methods and their Applications*, pages 347–358, 2012.

[81] S. I. Vagropoulos, D. K. Kyriazidis, and A. G. Bakirtzis. Real-time charging management framework for electric vehicle aggregators in a market environment. *IEEE Transactions on Smart Grid*, 7:948–957, 2016.

[82] E. Valsomatzis, K. Hose, and T. B. Pedersen. Balancing energy flexibilities through aggregation. In *Proceedings of the Second International Workshop on Data Analytics for Renewable Energy Integration (DARE)*, pages 17–37, 2014.

[83] E. Valsomatzis, T. B. Pedersen, A. Abelló, and K. Hose. Aggregating energy flexibilities under constraints. In *2016 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 484–490, Nov 2016.

[84] M. van der Kam and W. van Sark. Smart charging of electric vehicles with photovoltaic power and vehicle-to-grid technology in a microgrid; a case study. *Applied Energy*, 152:20 – 30, 2015.

[85] S. Vandael, B. Claessens, D. Ernst, T. Holvoet, and G. Deconinck. Reinforcement learning of heuristic ev fleet charging in a day-ahead electricity market. *IEEE Transactions on Smart Grid*, 6:1795–1805, 2015.

[86] M. G. Vayá and G. Andersson. Optimal bidding strategy of a plug-in electric vehicle aggregator in day-ahead electricity markets under uncertainty. *IEEE Transactions on Power Systems*, 30:2375–2385, 2015.

[87] R. Weron. *Modeling and Forecasting Electricity Loads and Prices*. John Wiley & Sons Ltd, 2006.

[88] D.-J. Won and S.-I. Moon. Optimal number and locations of power quality monitors considering system topology. *Power Delivery*, 23:288–295, 2008.

[89] J. Yang and J. Widom. Incremental computation and maintenance of temporal aggregates. *The International Journal on Very Large Data Bases (VLDB)*, 12:262–283, 2003.

[90] L. Yang, J. Zhang, and H. V. Poor. Risk-aware day-ahead scheduling and real-time dispatch for electric vehicle charging. *IEEE Transactions on Smart Grid*, 5:693–702, 2014.

[91] D. Zhang. *Aggregation computation over complex objects*. PhD thesis, University of California, Riverside, USA, 2002.

[92] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *ACM Special Interest Group on Management of Data (SIGMOD)*, pages 103–114, 1996.

[93] Z. Zhang, Y. Yang, A. K. H. Tung, and D. Papadias. Continuous k-means Monitoring over Moving Objects. *Knowledge and Data Engineering, IEEE Transactions on*, 20:1205–1216, 2008.

[94] J. Zhao, C. Wan, Z. Xu, and J. Wang. Risk-based day-ahead scheduling of electric vehicle aggregator using information gap decision theory. *IEEE Transactions on Smart Grid*, 8:1609–1618, 2017.

Paper A

Towards Constraint-based Aggregation of Energy Flexibilities

The paper has been published in the
Proceedings of the Seventh International Conference on Future Energy Systems Poster Sessions (e-Energy '16), Waterloo, Canada, 2 pages, 2016.
The layout of the paper has been revised.
DOI: http://dx.doi.org/10.1145/2939912.2942351

Abstract

The aggregation of energy flexibilities enables individual producers and/or consumers with small loads to directly participate in the emerging energy markets. On the other hand, aggregation of such flexibilities might also create problems to the operation of the electrical grid. In this chapter, we present the problem of aggregating energy flexibilities taking into account grid capacity limitations and introduce a heuristic aggregation technique. We show through an experimental setup that our proposed technique, compared to a baseline approach, not only leads to a valid unit commitment result that respects the grid constraint, but it also improves the quality of the result.

1 Introduction

The emergence and wide-spread use of electric vehicles (EVs) and heat-pumps in the transport and heating/cooling sectors pose new challenges to existing distribution grids [21]. The high power requirements of these

new devices can lead to significant congestions in the distribution grids (e.g., at specific peak-hours), which were not originally designed to support such power hungry devices. Such load-sensitive grid locations, i.e., bottlenecks, can be found at different elements of the low and high voltage grid [70], e.g., a distribution transformer (0.4-1kV) with a maximum power value of a few hundred kWs serving from a few (e.g., in North America) to several hundreds of households (e.g., in Europe). Operating the grid over such capacity limits might lead to significant losses of supply quality, breakdown of power equipment, or even power outages.

One idea to mitigate this problem is to reinforce the grid, which can be costly. A more cost-efficient approach [22] is to take advantage of the prosumers' inherent flexibility (e.g., using *vehicle-to-grid*) and off-load electricity demand from peak hours to other hours. Here, business entities called *aggregators* play a crucial role, as they are able to (1) aggregate flexibility from thousands of prosumers, (2) perform local demand/supply balancing, and/or (3) trade aggregated flexibility in a flexibility market [27]. In their day-to-day operations, *flexible load aggregation* is *essential* for being able to (1) produce "large-enough" commodities that are valuable on the market and (2) reduce the complexity of flexibility management and activation, i.e., solving the Unit Commitment (UC) problem [79]. However, to include grid capacity constraints into flexibility management routines (e.g., trading), the aggregators require novel techniques to effectively aggregate flexibility under grid capacity constraints. Unfortunately, no such techniques are currently available.

Consider two EVs, the batteries of which need to be (partially) charged. The first EV needs to be charged between hour 1 and 4, and it requires at least 1kWh and at most 2kWh delivered in 1 hour. Similarly, the second EV needs to be charged between hour 1 and 3, and it requires at least 2kWh and at most 3kWh delivered also in 1 hour. Flexible loads from such EVs can be modelled as two independent *flex-offers* (FOs) [75], shown as f_1 and f_2 in Figure A.1. In the figure, f_1 and f_2 take their basic form, covering only 1 time period (*slice*) of length 1 (hour); $[t_{es}, t_{ls}]$ depicts the *start time* flexibility, specifying the time range in which a load should be activated by the aggregator; and the dashed area depicts the *amount* flexibility, specifying the range of energy amounts in which specific amounts should be activated. Now, assume that the aggregator needs to combine f_1 and f_2 into an aggregated FO (AFO) f_{12}^a by adjusting the respective *time flexibility* and *amount flexibility* bounds. Further, assume that there is a fixed constraint of 2kW which the two EVs should respect at every hour while consuming electricity, see Figure A.1. Clearly, this constraint will be violated if f_{12}^a is produced so that both EVs start charging at the same time (both at hour 1 or 2), as seen in Figure A.1a. However, if f_{12}^a is produced so that the second EV starts charging 1 hour after the first, then, as seen in Figure A.1b, f_{12}^b represents an AFO, which allows

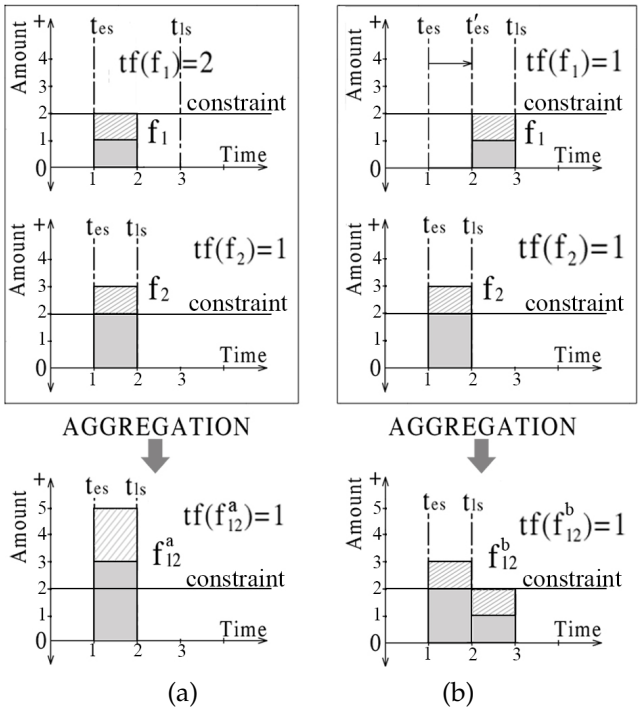


Fig. A.1: Two flex-offers producing two different AFOs

the aggregator to respect the grid capacity constraint while allocating loads of the two EVs. Next, we present an FO aggregation technique that takes into account such grid constraints.

2 Flex-Offer aggregation problem

2.1 Problem definition

In the general case [75], these are N FOs to be aggregated to M AFOs ($N >> M$) and the goal is to reduce M while retaining the flexibility (solution space) of FOs such that there exists at least 1 instantiation of FOs which allows respecting the grid constraints. However, as seen in Figure A.1, there are K possibilities to aggregate just 2 FOs, where K is based on the product of the FO's start time flexibility ($t_{ls} - t_{es} + 1$), making the problem much more harder in the case of N FOs. For instance, given 5 FOs with start time flexibility equal to 3, there exist in total $B_{|5|} \cdot 3^5 = 12636$ combinations to produce AFO sets, where B_n is the n^{th} Bell number. Thus, we introduce a heuristic aggregation technique to aggregate N FOs to M AFOs.

2.2 Heuristic constraint-based aggregation

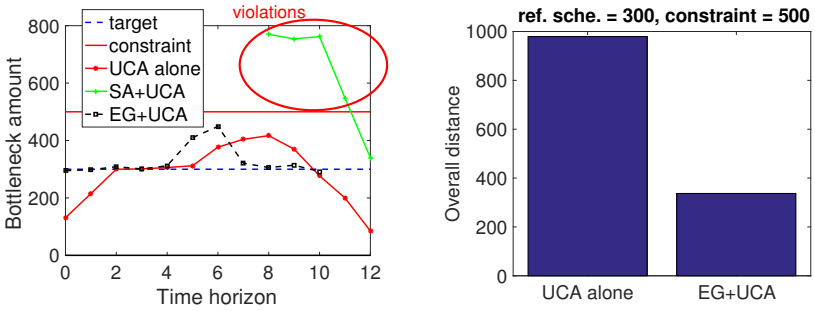
During aggregation, apart from the constraint, we also introduce a *reference schedule* that represents the business objective of the aggregator. We consider the *distance of an FO to the constraint* to be the sum of the minimum absolute differences between all the potential energy amounts of the FO slices and the constraint. Similarly, we compute the *distance of an FO to the reference schedule*. Thus, we consider the *overall distance* of an FO to the constraint and the reference schedule to be the sum of both distances. We prioritize respecting the constraint by using a very high coefficient for the constraint distance compared to a very low one for the reference schedule distance. We propose a greedy aggregation technique named **Exhaustive Greedy (EG)** based on binary aggregations.

Given a set F of FOs, EG starts by selecting and removing from F the FO f_{nom} most distant to the constraint. Then, in each step, it examines all the potential aggregations between f_{nom} and the remaining FOs in F. If there is an AFO, f_a, that reduces the distance of f_{nom}, it removes the FO that participated in the aggregation of f_a and further continues aggregation with f_a. Otherwise, EG inserts f_{nom} in the output set F' and continues aggregation by selecting a new f_{nom} until F is empty. When F is empty, EG returns set F' that contains all the AFOs.

3 Preliminary results

We use a dataset of 100 flex-offers representing a fleet of EVs plugged into a charging park of a workplace. The flex-offers have the same start time flexibility characteristics and similar amount flexibility bounds. Their start time flexibility is equal to 8 and their t_{es} equal to 2. The number of slices and the minimum amount requirements per slice follow a uniform distribution over the interval $[3, 6]$ and $[6, 9]$, respectively. The amount flexibility values of the flex-offers follow a uniform distribution over the interval [0,3].

We examine a case where the business objective of the aggregator is in harmony with the grid constraint, i.e., the reference schedule value (300)<constraint (500). In order to evaluate our technique in terms of constraint respect, we apply a UC algorithm (UCA) [79], both on the initial FOs set and on the aggregation result. UCA forms the amount of the examined bottleneck by activating the FOs (or AFOs) for the examined time horizon. We also use for comparison, the Start Alignment (SA) aggregation introduced in [75]. Our proposed technique prioritizes the constraint respect and leads to a UC result that respects the constraint, see Figure A.2a. On the contrary, we see that SA violates the constraint. Moreover, when EG is combined with UCA, they provide a smaller overall distance to the constraint and the reference schedule, than when UCA is applied individually under a time limit of



(a) UC solution (b) Overall distance

Fig. A.2: Set of 100 flex-offers with similar flexibilities.

20 seconds, see Figure A.2b.

4 Conclusions

This chapter is a first attempt to aggregate energy flexibilities taking into account power capacity constraints imposed by the electrical grid. We show that our proposed aggregation technique can respect the constraint imposed by the grid where a previous technique leads to violations. In the future, we will investigate our technique in more complex scenarios and examine a hierarchical approach taking into account the grid structure.