IT Licentiate theses
2001-005

Market and Resource Allocation Algorithms with Application to Energy Control

PER CARLSSON

UPPSALA UNIVERSITY
Department of Information Technology

UPPSALA UNIVERSITY

Market and Resource Allocation Algorithms with Application to Energy Control

BY

PER CARLSSON

May 2001

DEPARTMENT OF COMPUTING SCIENCE
INFORMATION TECHNOLOGY
UPPSALA UNIVERSITY
UPPSALA
SWEDEN

Dissertation for the degree of Licentiate of Philosophy in Computing Science
at Uppsala University 2001

Market and Resource Allocation Algorithms
with Application to Energy Control

Per Carlsson
Per.Carlsson@cs.lth.se

Department of Computing Science
Information Technology
Uppsala University
Box 337
SE-751 05 Uppsala
Sweden

http://www.it.uu.se/

ISSN 1404-5117
Printed by KFS i Lund AB, Sweden
Printing sponsored by EnerSearch AB

Abstract

The energy markets of today are markets with rather few active participants. The participants are, with few exceptions, large producers and distributors. The market mechanisms that are used are constructed with this kind of a market situation in mind. With an automatic or semiautomatic approach, the market mechanism would be able to incorporate a larger number of participants. Smaller producers, and even consumers, could take an active part in the market. The gain is in more efficient markets, and — due to smaller fluctuations in demand — better resource usage from an environmental perspective.

The energy markets of the Nordic countries (as well as many others) were deregulated during the last few years. The change has been radical and the situation is still rather new. We believe that the market can be made more efficient with the help of the dynamics of the small actors.

The idealised world of theory (of economics) often relies on assumptions such as continuous demand and supply curves. These assumptions are useful, and they do not introduce problems in the power market situation of today, with relatively few, large, participants. When consumers and small producers are introduced on the market, the situation is different. Then it is a drawback if the market mechanism cannot handle discontinuous supply and demand.

The growth in accessibility to computational power and data communications that we have experienced in the last years (and are experiencing) could be utilised when constructing mechanisms for the energy markets of tomorrow.

In this thesis we suggest a new market mechanism, ConFAst, that utilises the technological progress to make it possible to incorporate a large number of active participants on the market. The mechanism does not rely on the assumptions above. The gain is a more efficient market with less fluctuations in demand over the day.

To make this possible there is a need for efficient algorithms, in particular this mechanism relies on an efficient aggregation algorithm. An algorithm for aggregation of objective functions is part of this thesis. The algorithm handles maximisation with non-concave, even noisy, objective functions. Experimental results show that the approach, in practically relevant cases, is significantly faster than the standard algorithm.

Contents

Chapter 1

Introduction

1.1 The Project

The thesis work started in a project, DiMALLOC[1], that was financed by the Swedish National Board for Industrial and Technical Development, NUTEK. As this project ended, the work has been completed within the IT in Energy Programme of the Swedish Research Institute for Information Technology, SITI. This programme is one of four programmes of SITI. The focus of IT in Energy is on (*i*) local communications over the power lines, (*ii*) vulnerability and security in electronic commerce (with focus on power line communication), and (*iii*) electronic energy markets (with focus on economical and computational aspects of electronic power markets). The thesis presents some of the results in the last area.

A Swedish research company, EnerSearch AB, plays a central part in the programme together with Blekinge Institute of Technology, Lund University, and Uppsala University.

The aim of EnerSearch AB — that is owned by interests in the energy sector — is to initiate and coordinate research in areas that in different ways are related to the energy and business sectors. As in the case of the IT in Energy Programme, the research is conducted in close relation with universities. Current owners of EnerSearch AB are: ABB Automation Products, ECN - Netherlands Energy Research Foundation, EDP - Electricidade de Portugal, Iberdrola, IBM Utility Services, E.ON-Energie, and Sydkraft.

1.2 The Problem

Power markets around the world are deregulating. How to design efficient power markets is an issue that is discussed among authorities, energy utilities, and researchers. It is an interdisciplinary problem, with main aspects in economics as

[1] Distributed Market algorithms and resource ALLOCation

well as technical fields. The problem has interesting computational aspects too, and — due to the growing access to computational power and data communication capabilities — the computer science approach is able to open new market possibilities.

A way to enhance the efficiency of power markets is to incorporate a larger number of participants on the active market. Today the end consumer is not present on the market (with few exceptions) and, as a consequence, the market has to rely on more or less accurate predictions of consumption patterns and volumes. The prices at e.g. the Nordic spot market, Nordpool[2], varies significantly from hour to hour, due to fluctuations in demand. On the other hand a large share of the loads has such properties that the time profile of their demand could be utilised to even out a significant part of the fluctuations (if they are incorporated into the active power market), and this can be done with no (or minor) loss in comfort or efficiency for the consumer.

The gains of introducing the small participants on the active market are concerning most of the aspects that we can think of; when they are introduced — supported by a proper market mechanism — the fluctuations in demand, and hence in price, becomes smaller, the planning of production will be based on more accurate numbers, there is less stress on the transmission and distribution grid on high peak hours, we get a better resource usage from an environmental perspective, and so on; a win win situation.

When consumers and small scale producers are introduced on the active market, special care has to be taken concerning discontinuous supply and demand. Market mechanisms of today, based on standard theory of economics, assume that supply and demand is continuous. Due to this assumption it is hard and risk prone to participate on the market for any actor with a non-continuous curve. How to construct a mechanism that handles non-continuous supply and demand is nontrivial.

The possibility of a large scale introduction of the consumption side on the active market is a consequence of the development in computational power and data communications that has taken place during the last years. The accessibility to the Internet is growing and can be utilised, great efforts are put on development of inexpensive local area networks, and more and more equipment is enhanced with (what is often referred as) intelligence. In our homes we have (or soon we are having) computational capacity in washing machines and refrigerators as well as heating/cooling systems. If it is possible to communicate with the equipment, there is a potential to introduce it on the market, represented by small pieces of software. Consumers, other than large ones, will never participate on the market if they have to calculate their demand by hand and express it as mathematical functions. When incorporating consumers on the market we think of them as represented by software agents, i.e. small pieces of computer software, that give their input to the system.

[2] http://www.nordpool.no

1.3 Contributions

This thesis consists of two papers and an implementation of the algorithm introduced in one of them; the first paper introduces a novel market mechanism, ConFAst, a (semi) automatic mechanism with properties that meet the demands that we focused on in the previous section. The second paper is an article which introduces an algorithm for aggregation of a very general class of (separable) objective functions.

The character of the two papers differs significantly. It is not only the topic, but the perspective too. The first one, dealing with power control markets, is on a rather high level, while the second one, focusing on the construction and behaviour of an algorithm, is rather technical.

The third part of the thesis is an implementation of the aggregation algorithm, an implementation that was produced for comparison between our algorithm and the standard algorithm for the problem. The work on the market mechanism included implementation too, but it is not in a form that is suited for publication.

1.3.1 ConFAst, an Approach to Efficient and yet Fast Power Control Markets

The ConFAst mechanism is designed with large computational markets in mind. It is well suited not only for markets where the number of participants counts in tens or hundreds, but it scales to hundreds of thousands or more, both producers, distributors, and consumers. The market computation is distributed over computers in the network, i.e. it is not necessary to gather all information needed at a central spot. It handles non-continuous supply and demand, and hence it reduces the risk of participating with such curves significantly.

The paper has been presented at a seminar on Information and Communication Technology in the Energy Sector, held by the Nordic Energy Research Scientific Program, in Trondheim, Norway, March 8 – 9, 2001. A conference version of the paper is under submission.

The ConFAst mechanism depends on the aggregation of utility functions — functions that express the relative value of consumptions bundles, i.e. a kind of objective functions. Computationally this is the most demanding part of the work and it is essential that the aggregation is performed in an efficient way.

1.3.2 Resource Allocation with Noisy Functions

A lot of efforts have been put on the aggregation of separable concave objective functions[3]. The general case, when nothing can be said about the objectives, is hard. A practically relevant class of functions is separable non-concave objective functions. Standard algorithms for aggregation of this class of functions are basically constructed on a pairwise aggregation of functions and a brute force

[3] Concave in the maximisation version of the problem, convex objectives in the minimisation version.

testing of all combinations of the two functions. More efficient aggregation of separable objective functions that might not be concave is nontrivial.

The article presents an algorithm that focuses on the fact that when aggregating non-concave functions the resulting function very fast becomes almost concave, but noisy. The algorithm utilises regularities in the aggregated functions — when the function is almost concave or is close to another smooth curve — so that the search space can be pruned when aggregating the functions.

The algorithm is generic, but developed with resource allocation and computational markets in mind. The algorithm can be used as a subroutine of the ConFAst market mechanism.

The article is rather technical. For the reader not interested in all technical details it is probably sufficient to read the introduction through the experimental results, and skip the technical description of the algorithm. Hopefully this part gives enough to grasp the main ideas behind the algorithm, and gives an impression of the performance.

The article is submitted for journal publication.

1.3.3 Implementation

Implementations and tests of both the market mechanism and the aggregation algorithm is part of the thesis work.

The Java classes (and test data) needed to run some tests on the aggregation algorithm are available on the Internet and on the CD-ROM included in the thesis. The implementation is done to test and compare the performance of the algorithm on different input (the comparison is done on the standard algorithm for aggregation of non-concave objective functions). The results, that are part of the article, show that the algorithm indeed is competitive. This holds for both adversary data and practically relevant data. It should be noted that the standard algorithm that we use for comparison can be expected to be close to optimal due to its simplicity. Much could probably be done to improve this first test implementation of our algorithm, as it is rather complicated.

The three parts of the thesis are, in some ways, rather disparate. At the same time they are held together by the aim to develop algorithms and mechanisms for resource allocation and markets. As said earlier, the application area that we have in mind is the power markets.

Acknowledgements

The thesis work has been part of two different projects. It started in the DiMalloc project, sponsored by the Swedish National Board for Industrial and Technical Development, NUTEK[4]. The work was finished within the Swedish National Programme on IT in Energy, a programme of the Swedish Research Institute for Information Technology, SITI[5].

The master objective of the programme IT in Energy is to demonstrate how information technology can improve the efficiency in future energy systems.

EnerSearch AB[6], a Swedish research company owned by interests in the energy sector, has played a central part in the projects as coordinating partner of Uppsala University and others.

I want to thank my advisors Fredrik Ygge and Arne Andersson for support, ideas, discussions, and much more, and Thore Husfeldt, that has been a valuable local support in Lund. Both the articles of this thesis are joint work with Andersson and Ygge. Furthermore I want to thank my friends at the Computer Science Department, Lund University, where I live my everyday (working) life. Even though I am not at my home department in Uppsala to often, I appreciate being part of the team.

Finally, I want to thank my family.

[4] Due to restructuring, the part of NUTEK that sponsored the project is now part of the Swedish Agency for Innovation Systems, Vinnova, http://www.vinnova.se.

[5] http://www.siti.se

[6] http://www.enersearch.se

Chapter 2

An Approach to Efficient and yet Fast Power Control Markets

Power markets around the world are deregulating. The issue of how to properly design power markets is a delicate one, intensively discussed among authorities, researchers, energy utilities, etc. In this paper we focus on two major aspects of power market design: (i) management of discontinuous demand/supply curves, and (ii) computational design of markets of huge size with relatively short time frames.

We introduce a mechanism, ConFAst, for highly dynamic power markets that allows for huge numbers of active participants (also consumers) in the market, even if they have discontinuous demand/supply curves. The computational efficiency of the approach enables usage also in markets with short time frames.

We argue that ConFAst significantly can improve the efficiency of energy systems.

2.1 Introduction

2.1.1 The Strains on Market Mechanisms in Extreme Situations

Around January 24, 2000, the weather in all of Sweden was so cold that the demand for electricity was extremely high and the pressure on the whole power system was equally hard. As a consequence the price on the spot market rose to levels where energy utilities with no or small production capacity of their own risked to lose all of their annual profit during one single day.

This was the hardest test of the new market system so far, since the recent deregulation of the energy markets of Finland, Norway, and Sweden. However,

in the times of deregulation the energy system is pushed closer to its limits and we will see more of these types of situations in the future. Skeptics of the deregulation see a new dawn in this phenomenon.

In an extreme situation like this one, new problems arise or come into focus. One such problem is how to release the pressure on the system in a way that affects the society and the economy as little as possible when the demand for energy is too high.

Today, in Sweden, there is an ongoing discussion on how to better cope with this type of situations. One of the main issues is how to properly utilise the dynamics of the energy consumers. This question is an aspect of the issue of how to construct more efficient power markets. The fluctuations in demand and prices on the markets of today are significant even under normal conditions, see Figure 2.1.

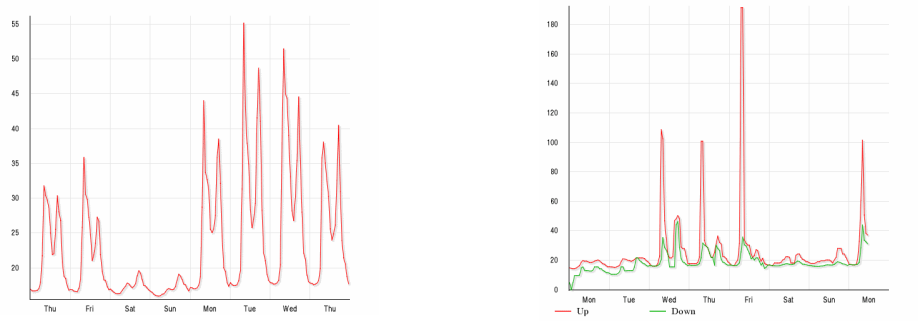

Figure 2.1: *Since demand varies very much from hour to hour the market prices of a deregulated market vary significantly as well. These graphs show the market price of the Nordic spot market (left, peak price 55 Euro/MWh) and the Swedish regulating market (right, peak price close to 190 Euro/MWh) a week in January 2001.*

2.1.2 The Dynamics of the Small Actors

Today the load side of the market, the consumer demand, is essentially taken as exogenous and the suppliers and distributors have to rely on more or less accurate predictions. This article exploits possibilities for the load side, as well as small scale production, to take an active part in the market.

A large portion of the loads within the power grid are *controllable* (in a control system sense). The term controllable load can be illustrated by examples from everyday life: A light bulb or a TV set is not a controllable load — when the user switches it on it is supposed to turn on instantly, and the other way around when it is switched off. On the other hand the heaters of an apartment, or any equipment that supply a comfortable indoor temperature, are typical controllable loads — the time profile of their demand could be changed quite

a lot with no (or minor) loss in comfort for the consumer. An example of the possibilities of changing the consumption pattern of a household is shown in Figure 2.2. For illustration simplicity the examples are taken from the private sphere; there are a lot of loads in industrial settings that have equal or similar properties. The industrial loads are generally much larger and hereby of a much higher economical interest. The larger the loads, the earlier they ought to be incorporated in dynamic power markets. In the industrial sector this is done to some extent today, but there is a large potential to develop the market. In a longer perspective also smaller loads, such as households, are of interest.

Another potentially interesting issue in this respect is local production. There are different forms of local spare plants that could be used in critical situations, but there are also examples of less predictable local production, e.g. wind power, that complicates the dynamics of the energy system.

A small field test that shows interesting results on the possibility to move the power consumption of residential houses that are electrically heated from high demand hours has been performed, see Figure 2.2. The gain (for the system) of doing so with a small number of houses is negligible since their power consumption is low, but a large group of participants has a potential to improve the demand profile of the market.

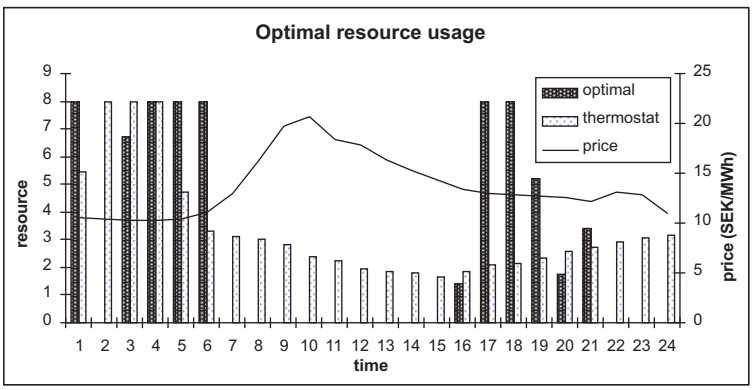

Figure 2.2: *The resource usage of an electrically heated household. The optimal choice — with respect to the spot market — is compared with the use of ordinary thermostats (that do not take price into account). For relevant time periods the indoor temperature deviates less than* $0.5°C$*, but there is a saving in heating costs of approximately 28%.*

In Sweden of today 49% of the electricity is used in the residential and service sectors, and of this 43% is used for heating, 25% is used for household purposes, and the rest for common purposes [11]. These figures show that the residential sector has a potential to influence the market performance in Sweden and countries with similar consumption profiles.

2.1.3 The Technological Progress

During the last years we have experienced an enormous growth in accessibility to computational power and (local area and wide area) data communications, e.g. over the Internet. This is true both in industrial and residential settings, c.f. Section 2.3.2.

One interesting idea that currently is investigated is the possibility to use the power line as a communications medium at a local network level. This gives that any equipment that is connected to the electricity network potentially can be connected to the communication network (without extra wiring), and — if interesting — it can participate on the market. Another interesting communication approach (with no need for wiring) is the Bluetooth industrial standard; cheap, short range communication between equipment, that has potentials to be useful e.g. in households and office buildings. This is only a few examples among many of the changes that we currently are experiencing and that affect the market possibilities.

Another upcoming idea, in the residential setting, is the intelligent or smart house, with quite a lot of possibilities. An interesting possibility is to integrate the smart houses (together with industrial and office settings) into the active power control market.

Altogether, the technical and economical opportunities for providing equipment with computational and communicational capacity can be utilised when developing an electronic market mechanism for power load markets.

2.1.4 Discussion

There are extreme situations in which energy prices go dramatically high. At the same time there is a lot of unused dynamics among the small actors. In the perspective of the technical development, it makes perfect sense to increase the integration and enable more efficient markets. However, as will be discussed below, to design a proper market mechanism is nontrivial.

2.2 Market Mechanisms — Theory and Practice

General equilibrium analysis and computation from economics has inspired the design of a number of market mechanisms, e.g. Walras [14, 4, 15]. The basic idea is to estimate a clearing price such that supply meets demand for every commodity, and then reallocate according to the bids at the clearing price. However, general equilibrium theory of economics generally relies on three classical assumptions:

(i) all producers have increasing marginal cost with increased production,

(ii) all consumers have decreasing marginal utility with increased consumption, and

(iii) all necessary information (i.e. the bids) can be inexpensively communicated to a central point.

As discussed below, these assumptions are often unrealistic.

2.2.1 Increasing Cost and Decreasing Utility

One classical assumption in economics is that marginal cost is increasing with increased production. A related assumption is that marginal utility is decreasing with increasing consumption, see e.g. [10, 13] and Figure 2.3 (top row). These assumptions are useful. First — from a system viewpoint — they are more or less liable to be true. Second, market analysis becomes manageable.

In practice — from the viewpoint of the participant on the market — this assumption is often unrealistic, some examples:

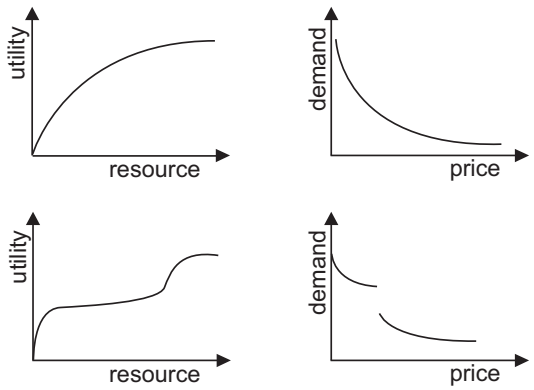

Figure 2.3: *Top row: A classic assumption in economics is that the marginal utility is decreasing with increasing resource. If this is the case, the utility looks as in the left pane. As a consequence the demand looks as in the right pane, a continuous curve. Bottom row: A realistic utility function, e.g. of a water heater; marginal utility is not decreasing (left). The consequence is that the demand is discontinuous (right).*

It is reasonable that a single power generator has an increasing marginal cost, as long as we limit ourselves to look at the interval where it is really efficient. If we, on the other hand, look at the full interval from zero to full capacity, it is more complex. It might e.g. be very costly to run the generator at a low production level (if at all possible). This gives a non-increasing marginal cost in some subintervals. Another aspect is that a true description of a power plant with more than one generator generally implies non-increasing subintervals due to different ways to combine the units.

The situation is similar for the consumers that might be interested in taking part in the market. Many of them have a decreasing marginal utility over the

interval from zero to max consumption, but some do not. A water heater could be a simple example; it has a basic service level where it can provide water e.g. for washing of hands and to do the dishes. When it is possible to take a shower it is on another service level, and so on. In other words, the marginal utility is not decreasing. Another example of loads with non-decreasing marginal utility is any load that has an initial marginal utility of zero up to some breakpoint where it turns positive and decreasing.

Most power markets, e.g. NordPool, ignore the fact that demands may de facto be discontinuous, and require continuous demand curves. For power markets with only very large participants (as currently is the case for many markets) this is rather unproblematic, but if smaller actors are to be incorporated, it is clearly undesired. If someone — as a consequence of acting on the market — gets an allocation that is undesired (c.f. Figure 2.4), it is obvious that this participant would like to buy herself out of the situation as inexpensively as possible. This may be manageable if the time frame of the main market is large enough and if there is some after-market for this regulation. Otherwise she will have to take the undesired allocation as part of the risk of acting on the market. In either case it might not be individually rational for the participant with a non-continuous demand to act on the market. Individual rationality can be expressed as follows: the gain of participating on the market is nonnegative [10].

In earlier work [2, 15] the assumptions concerning increasing marginal cost and decreasing marginal utility where not seriously challenged. Either we restricted ourselves to markets where the net demand is continuous and monotonically decreasing with price [2], or there was just a note that some extra care has to be taken when it is not [15, p.34-35].

The shortcomings of current market approaches has motivated us to construct a new, more general market mechanism that can manage more general utility and cost functions.

2.2.2 Information

Assumption (iii) — all necessary information (i.e. the bids) can be inexpensively communicated to a central point — is the last point of the list in the beginning of Section 2.2. We claim that in practice this too is an unrealistic assumption. To start with, the information is not centrally available from the beginning but distributed over the market participants, and it may be impractical to collect it to a central spot. As long as the market is kept small, when the participants count in tens or hundreds, this is generally no major problem, the information can be gathered when needed. On a market where the participants count in hundreds of thousands or even more the situation is different.

When the market grows large the communication soon becomes a bottleneck for any market mechanism that rely on the property that all information is centrally available. The narrower the time frame of the market is and/or the more expensive the communication, the more essential not to rely on this assumption.

Our market mechanism is constructed with a distributed setting in mind, i.e. it does not rely on this assumption.

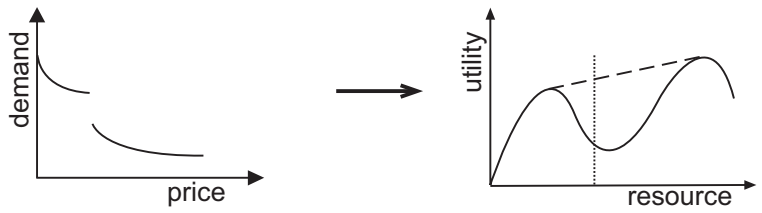

Figure 2.4: *If a market participant has non-continuous demand (left pane) the corresponding utility function is non-concave (right pane). If he, on the other hand, has to give a continuous demand as input to the market the corresponding utility would be a concave approximation of his true utility function (c.f. Definition 2.3.1) — as the upper, dashed line in the right pane. The allocation that he gets on the market might be undesired (e.g. the vertical, dotted line). The participant wants to buy himself out of the situation as inexpensively as possible (buy or sell to get a higher utility). Here he might even be prepared to give resource away for free to get a higher utility.*

2.2.3 The Market of Today and Tomorrow

Today the power market is a rather small marketplace in terms of number of participants. The market mechanisms that are used today may be well suited for this situation (with a small number of participants on the market). On the other hand, with a market mechanism that is made automatic (or semiautomatic), it is possible to let the number of participants grow. The market can be made more efficient and yet fast. A limitation of current market situations is that in practice only the big actors on the production and distribution side take an active part in the market. The consumption side is viewed as exogenous (with maybe a few exceptions). With an approach where the mechanism is made automatic this limitation can be broken and small producers and even consumers might play a new, active part in the market.

The interesting parts of the energy market of today, as it is constructed e.g. in Sweden, is divided into two parts:

- a spot market where the energy prices, supply and demand for the upcoming day (24 hours) are fixed at noon every day, this market is common for the Nordic countries Finland, Norway, (parts of) Denmark, and Sweden, and
- a balancing service organised as a regulating market, an online (semi-) market where the time frame is about 10 to 15 minutes.

In other countries with deregulated power markets the mechanisms are sometimes constructed differently but still face the challenges discussed in this paper.

The changes that are taking place within the sphere of computation and communication capacity can be utilised to build fast and efficient applications

that can handle e.g. a large dynamic regulating market. The computational and communicational capacity already is there in the distribution system and in consumer side systems (and wherever it is not, there is a potential to introduce it as soon as the cost is low relative the gain).

An extreme consequence of the limitations of the old view on the power control market is the red button principle; there is no other means to protect the system under high pressure than to shut of power supply to entire areas. The new view is different: The limitation that only production and distribution acts on the market can be broken. In principle it is even possible to use small loads such as different kinds of heating or cooling systems, freezers etc. in residential buildings to regulate demand, or the other way around, for such loads to act on the market. Loads in industrial and office setting are generally much larger and of a greater economical interest, but in a longer perspective even smaller loads are interesting. Another reason to set some focus on residential settings, even though the loads are relatively small, is that the residential demand during high cost hours probably is very flexible since people are at work. As a result demand is moved from high cost hours to surrounding hours where the price is lower — a win win situation [16].

The main effect of this is a more efficient market, as a side effect the need to use the red button is drastically reduced.

Altogether we set up the goal that a new market mechanism should:

- handle not only the production side, but the consumption side too,
- be suitable for a fast (electronic) market and yet efficient and close to optimal (as far as we are dealing with controllable loads or controllable clusters of loads),
- handle non-continuous demand, i.e. utility functions that are non-concave, and
- be suitable for distributed systems.

2.3 ConFAst, A New Market Mechanism

Our goal is a power control market that is more efficient (economically and computationally) than the markets of today. This is accomplished by a mechanism that allows for a huge number of active participants, consumers as well as producers, while also allowing for participation with non-concave utility (i.e. non-increasing marginal cost of producers and non-decreasing marginal utility among consumers, c.f. Figure 2.3) and also being highly suitable for large and distributed market settings.

2.3.1 Concave and Non-Concave Utility

Concave

When all utility functions are *concave* it is straight-forward to find an equilibrium price of the market based on the given functions. One way to do this (c.f. [4]) is to

1. aggregate the demand or utility functions into one function that describes the optimal utility of the system. For every resource level of the function interval remember the allocations that gave the optimum, and
2. calculate the equilibrium price based on the aggregated function.

The computation can be made in ways more efficient than what is sketched above, e.g. in a binary search fashion, but it is unknown how this can be utilised in a distributed environment.

Non-Concave

Things turn out to be considerably harder when some of the utility functions are *non-concave*. Standard microeconomic theory on competitive equilibria relies on properties that can be expressed in terms of concave utility functions. As stated above, in Section 2.2.1, the restriction to concave utility introduces problems to participants on the market, both consumers and producers.

Generalised Vickrey Auctions [9] is an interesting concept (as an alternative to the one described in this paper) that is able to manage non-concave utility. It has a number of advantages and disadvantages when considered for a power control market. The concept of Generalised Vickrey Auctions

+ has favourable game theoretical properties,
+ produces a Pareto optimal allocation on reported utility, and
+ handles non-concave utility, but
- produces an outcome which is not budget balanced, i.e. the payment does not sum to zero, and
- has major disadvantages when applied to a distributed setting. It is based on a series of computations of the equilibrium; with n participants as many as $\mathcal{O}(n)$ computations of the optimal allocations are needed to establish an equilibrium.

We consider the drawbacks serious in the setting of a power control market and suggest another approach.

Power markets are relatively large, both in terms of resource traded and in number of participants (particularly when consumers and smaller producers are introduced as actors on the market). We can utilise this property when dealing with non-concave utility. As the market grows the aggregated utility very fast

becomes *almost concave*. The aggregated utility function can be described as a concave function with a low amplitude noise on top, and at any point the function is very close to its convex hull. Another way to express the same is that there is a "near equilibrium", i.e. an allocation and price that is close to satisfying an equilibrium [10, p. 629]. This can be utilised to establish a market price and market allocations constituting this "near equilibrium".

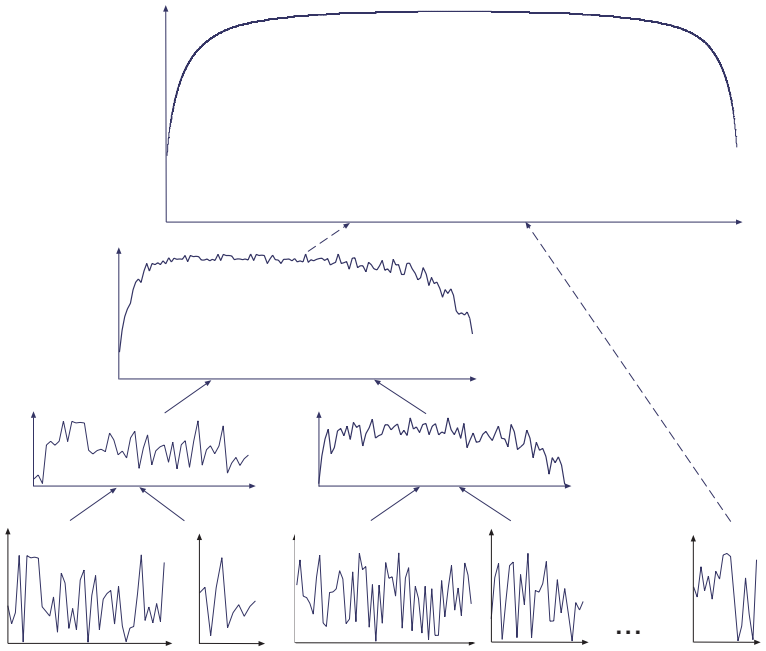

Figure 2.5: *The power load market is a large enough market in the sense that there is always a near equilibrium. This is a basic assumption that we rely on when constructing our market mechanism. Even if aggregating such extreme functions as random noise functions, we get an aggregated function that is very close to concave, and hence — if interpreted as aggregated utility — close to an equilibrium. Here: 100 random noise functions, c.f. [1].*

2.3.2 Outline of the ConFAst Mechanism

Our mechanism, ConFAst[1], calculates the equilibrium price based on the utility functions of the participants. Logically the mechanism can be viewed as a two step algorithm:

(i) assume that all utility functions of the system are concave. This is done approximating the non-concave functions with their convex hull. Calcu-

[1] CONvex hull with Final AdjuSTment

late the (near) equilibrium price and allocations based on the hull of the aggregated utility functions (Section 2.3.2), and

(ii) decide the actual allocations to the participants, the decision is based on the true utility (defined below), see Section 2.3.2. Set the final payments of the participants; this decision is based on the true utility functions and the difference between the allocation in step (i) and in this step (Section 2.3.2)

In this paper we use the expression *true utility* in the following sense:

Definition. 2.3.1 *We define the* true utility *as the utility function that is given by the participant in contrast to the convex hull of the function that is used in step (i) of the algorithm.*

This two step mechanism utilises that we have a near equilibrium in large markets to establish the market price and allocations, and it is able to allocate the resource using the true utility functions (even when they are non-concave). As long as all participants act "price takers", the allocation that ConFAst produces is Pareto optimal, i.e. as it is impossible to make some participants better off without making some other participants worse of [10, p. 307]. This assumption is reasonable from the "near equilibrium" property and e.g. [12]. Up til now we have not put any further social or other demands on the economical outcome.

Input Functions

In principle it is possible to use either utility functions or demand functions as input to the mechanism. In the following we assume that the input functions are utility functions[2].

The utility functions of the participants[3] reflect how useful different amounts of energy are during the coming time period. A consumer gives the information, with a proper resolution, for all resource levels between zero and the maximum consumption. In a similar way a producer gives the corresponding information on the utility of delivering energy in the proper interval. The cost of delivering an amount of energy is simply expressed as the negative utility of a negative allocation (i.e. produced resource). The utility functions do not need to be concave.

Normally a utility function is just an ordinal function [13], that says that a consumption bundle with a higher utility is preferred to one with a lower utility. The function has no cardinal properties. There are certain exceptions from this, e.g. quasi linear utility functions [10, 13] have cardinal properties.

[2] If demand functions are used as input somewhat less information is available, but a reduced utility function could be constructed out of each demand function. The description below is applicable to this case as well.

[3] The participants are represented by software agents that produce the utility functions. The utility of a typical consumer is constructed out of data on customer preferences, load state, market and disturbance predictions, and a load model [15].

The utility functions that we rely on are quasi linear (non-linear in energy, linear in "everything else" which is represented by money). An effect of this is that it is possible to compare two utility functions and to aggregate them in a such a way that, based on the aggregated function, one can allocate a resource between the two in an optimal way (maximising the total utility). Since it is possible to aggregate two, it is possible to aggregate any number of functions.

The Near Equilibrium Price and the Corresponding Allocations

The first step of the mechanism is to establish the (near) equilibrium price for electricity on the market. The computation of the market equilibrium is based on the convex hull of the participants' utility, as if we assume that everyone has a concave utility. As said earlier, Section 2.3.1, we can assume that the majority of the participants have concave utility functions (that equals their convex hull), but it is realistic that some have not.

If the utility functions are concave the equilibrium price equals the first order derivative of the aggregated utility where the excess demand/supply of the system is zero[4]. With participants holding non-concave utility, we establish a market price (a "near equilibrium" price, Section 2.3.1) based on the convex hull of the aggregated utility.

There are two possible outcomes of this:

1. The aggregate true utility of the system equals the hull of the utility at the equilibrium, i.e. the demand function of the system is continuous at the equilibrium. Then we are done and all participants pay according to the equilibrium price for their allocations. All that has to be done is to distribute information on the equilibrium price and the corresponding allocations to the participants.

   Note: It is not enough to broadcast the equilibrium price since some of the participants might have two or more interpretations of their corresponding allocation, see Figure 2.6.

2. There is a difference between the true utility and the hull at the equilibrium, i.e. the demand is not well defined at the equilibrium price, see Figure 2.3. In this situation the mechanism proceeds to establishment of the final allocations and prices of the participants, see Section 2.3.2 and 2.3.2 respectively.

The Final Allocations

The final allocation to the participants is the optimal allocation of the resource given the reported utility functions.

[4] Any other allocation within the interval of the aggregated utility function could be defined as the equilibrium, e.g. if the system is a subsystem of a larger system that decides that this system should deliver a positive or negative excess demand.

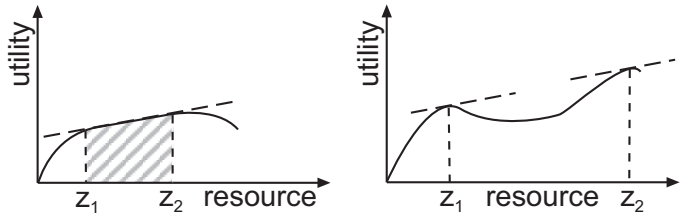

Figure 2.6: *A broadcast of the equilibrium price is not enough to establish an equilibrium allocation if participants have concave utility that is not strictly concave — i.e. there are linear segments — (left), or if they have non-concave utility (right). Some might have more than one interpretation of what resource level to choose (left: any allocation in the grey segment between z_1 and z_2, right: one of the two allocations).*

When some participants have non-concave utility the outcome of the establishment of a "near equilibrium" price and allocation (as described above) often[5] is an allocation that for some participant[6] is undesired.

The participant with an undesired allocation is moved to a new allocation, and as a consequence some other participants have to be moved from their original allocations, and they have to be compensated in some way.

The Final Payment

If there was a feasible allocation (defined below, c.f. Figure 2.4) at the equilibrium, computed as described in Section 2.3.2, the payment of all participants for their allocations is the equilibrium price times their allocation:

$$p^e z_i \tag{2.1}$$

where p^e is the equilibrium price and z_i is the (positive or negative) allocation of participant i. The allocation of a producer is, as said before, a negative allocation — the amount of energy she is to deliver according to the market outcome.

In the following we use the expression *(in)feasible allocation*, that we define in this way:

Definition. 2.3.2 *We define* a feasible allocation *as an allocation such that the utility of every market participant equals the convex hull of her utility function.*

An infeasible allocation *is an allocation where some participant has a utility less than the convex hull of her utility.*

[5] The more actors that have non-concave utility, the more often this will be the case.

[6] By construction (of the aggregation algorithm, c.f. [3]) no more than one sole input function is non-concave at the "near equilibrium", and hence it is only one participant that has to be moved from an undesired allocation.

The notation is chosen since such an allocation would be undefined if the mechanism was based on supply and demand functions (c.f. Figure 2.3).

If there was no feasible allocation the final allocations are computed according to Section 2.3.2 and some participants are moved from their initial allocation so that the result is a feasible and Pareto optimal allocation. The last operation of the mechanism is to compensate for this and decide the final payments for all participants taking the movements into account.

The idea is that:

1. The ones that are moved from initial allocations that were feasible are compensated according to their own utility functions. That is, if for someone of those participants, the utility of the final solution is less than the one of the initial solution she is compensated for the loss of utility, on the other hand, if the utility is higher the participant pays an equally higher price. The payment that those participants pay for their final allocations becomes:
$$p^e z_i + \Delta u_i \tag{2.2}$$
where $p^e z_i$ is as in Equation 2.1 and Δu_i the (positive or negative) change in utility of participant i due to the movement from the infeasible equilibrium solution to a feasible one, and

2. to get a zero cost solution for the total system let the one participant that created the unfeasibility (the jth participant) at the equilibrium get the final payment:
$$p^e z_j + p^m \Delta z_j \tag{2.3}$$
where
$$p^m = \frac{\sum_{i \neq j} \Delta u_i}{\Delta z_j} \tag{2.4}$$
i.e. this participant is not compensated according to her own utility function but has to compensate for the changes that she enforces the other participants by having non-concave utility.

From an economical perspective, the solution is:

1. Pareto optimal on reported utility,
2. individually rational with concave functions, but
3. not individually rational with non-concave functions.

Note that when the market is large, the price that the interesting participant has to pay for the movements is almost the same as the equilibrium price, c.f. the tests in Section 2.4.

This compensation system has two interesting properties:

1. If we count the extra[7] computation for free, the cost for the system for providing the possibility of having participants with non-concave utility is zero, and

[7] Extra relative a mechanism that does not handle non-concave utility.

2. one still takes a small risk by entering the market with non-concave utility. If the "near equilibrium" allocation is infeasible due to the non-concavity of my utility, then I am the one that has to compensate for the changes in utility that is imposed on others.

We find both these properties attractive. The first one is obviously positive, whereas the second one might be subject to some discussion. Since the market runs better the fewer participants have non-concave utility, and the smaller the deviations from concavity are, it is reasonable that the one introducing a non-concave utility takes some risk. At the same time, if the alternative is a mechanism that enforces participants to present concave utility, the risk is much larger for a participant who is unable to express the non-concavity of her utility.

Thus we argue that it can be justified that one takes some risk when participating with non-concave utility. On the other hand, if there is a risk of great loss no one would do so. One of the risk factors is that the one load that created the unfeasibility at the "near equilibrium" might be moved to a zero allocation when the optimal allocation takes place – and still has to pay to compensate the system for the movement of others. If this is fair or not could be argued. We believe this to be a potential problem, but there is a way to solve the problem.

To avoid that someone e.g. has to pay for a zero allocation we separate the distribution of allocations from the distribution of information on final payments (both in phase 2). Then it is possible to decide the optimal allocations in a fast manner and still have a pricing that is considered fair.

The separation of the two operations of phase 2 gives the possibility to take a brokerage from the participants by separating the buying and selling prices on the seldom occasions when someone has to pay all to much[8], e.g. if someone is moved to a zero allocation; When someone, according to Equation 2.2 or 2.3, has to pay more than some factor f times the equilibrium price (or receives less than $1/f$ of it) for its current allocation the buying and selling prices could be separated so that the cost is distributed among the market participants.

We have tried to find out experimentally how large the risk is that one has to separate the buying and selling prices. We have found, as far as our artificial test material is concerned, that there is a small, but maybe not negligible, risk that this will happen now and then. It is clear that the risk is larger the more participants have utility functions with a non-concave character close to zero. Future field tests will provide more knowledge on this.

A Distributed Setting

As said in the introduction, society has experienced an enormous growth in accessibility to computing power and data communication during the last years. There is no longer any need for a centralised computational resource to gather all information on the market and compute the market outcome all by itself. Computers are there, or could be set up, in the network, and most of the com-

[8] If a brokerage is used to finance the system it could be implemented like this too.

putations needed can be done in the network when the information is gathered and distributed, [2].

When a large automated market is constructed, almost all the computation can, and should, take place in the network. Incorporating the consumption side in the active power market we construct such a large market (in number of participants) that the market mechanism has to be constructed with the communicational aspects of computation in mind. There are computational gains if the computation can be done in parallel over the network. Even more essential, there are communicational gains if the number of messages sent over the network could be held low and bottleneck nodes that everyone has to communicate with are avoided.

The basic algorithms of ConFAst are intended for use in a distributed setting. Logically the underlying structure of our mechanism is a balanced tree structure (Figure 2.7), a structure that is very efficient in a distributed setting (no communication chain is longer than $\mathcal{O}(\log n)$ with n participants). For more on the underlying communication sparse aggregation algorithm see our previous work, [2, 3, 1].

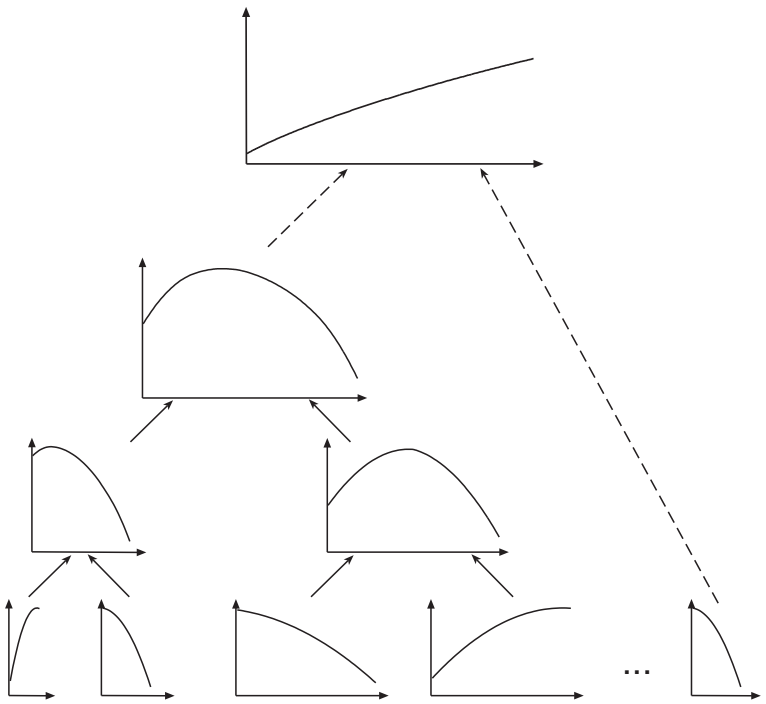

Figure 2.7: *The logical structure of our mechanism is a binary tree structure. In a distributed setting this gives advantages in terms of short communication paths. Moreover, when a system consists of a number of subsystems this is reflected in a natural way (with a subsystem represented by a sub tree).*

Sometimes constraints enforces that the market is divided into sub markets. There are static bottlenecks in the transmission grid and runtime disturbances can result in new bottlenecks. Where there is a bottleneck in the system it is logical to view the two sides as different sub systems[9]. Other logical subsystems are constituted of lower voltage systems. All of this can be handled efficiently by ConFAst. It has a hierarchical tree structure and therefor it adapts very well to partitioning of the power grid into subsystems. The gain is twofold. First, it is possible to take e.g. restrictions in capacity within a subsystem into account and to handle them efficiently. Second, when changes in demand take place in a subsystem — changes that are so small that they do not have to be propagated to higher levels — the mechanism has possibilities to recalculate the optimal allocation (and related prices) locally within the subsystem.

2.4 Implementation

Test implementations on different parts of the mechanisms show that this market algorithm is not only of theoretical interest, but it is a practically useful too. It has potentials to be an efficient alternative to existing market mechanisms, from an economic viewpoint as well as from computational and communicational perspectives.

Tests of the behaviour of ConFAst have been performed, so far in a small scale and with input functions that where not reflecting true preferences, but constructed just to challenge the mechanism. These tests gave some interesting results; one is that there might be a need for diversification of the distribution of information on final allocations on one hand from the distribution of information on final payments on the other, c.f. Section 2.3.2 and Section 2.3.2.

A desirable market property is *individual rationality*. If an actor has non-concave utility there is unfortunately no guarantee that it is individually rational to participate in a market based on ConFAst. This is caused by the construction of the compensation system, c.f. Section 2.3.2 and Section 2.3.2. The final payment for a participant that has to be bought out of an undesired allocation is partly based on the utility of the other participants. An interesting question is how bad the market outcome can be for an actor with non-concave utility. We have performed some tests to investigate this, and we present the results in two ways.

The test data was constructed in the same way in both the tests related to individual rationality. The basis of the input functions was sine waves[10].

In the first test we have measured how expensive it can be, with the worst possible allocation to an actor with non-concave utility, see Figure 2.8. The result for a number of market situations with small number of participants is presented in Table 2.1. The main point is that as the market grows (in number

[9] As is done today e.g. in Scandinavia.

[10] Concave utility functions were based on the sine wave in the interval $[0, \pi]$ (with random scaling factors), non-concave utility on a combination of the sine curve and a linear factor in the interval $[0, 3\pi]$, see Figure 2.8.

of participants) the price paid converges towards the "near equilibrium" price. The price is not a direct measure on individual rationality, but — expressed in an intuitive way — the closer to the "near equilibrium" price the price is that the investigated participant pays, the more obvious it is that it is individually rational to participate on the market.

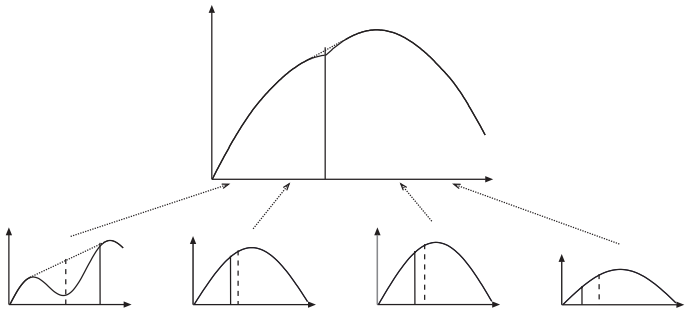

Figure 2.8: *One test related to individual rationality investigated how expensive the most expensive allocation would become for a participant with non-concave utility (in relation to the "near equilibrium price"), c.f. Table 2.1. The curves represent the utility function and the hull function, the dashed lines the initial allocation and the solid lines the final allocation.*

Table 2.1: *Investigation on final payments. The table shows how expensive it can become to participate with non-concave utility on a market using* ConFAst. *The main point is that as the market grows the risk diminishes. The prices are compared to the "near equilibrium price".*

# Market Participants	Highest price paid by the participant with non-concave utility	
	when selling part of her initial allocation	when buying more than her initial allocation
2	208%	139%
3	190%	123%
4	157%	114%
8	134%	107%
25	113%	102%
50	106%	101%
100	102%	100%
200	101%	100%

Note that even with as few as 25 participants on the market the final payment of the participant with non-concave utility is no more than 113% of a payment

according to the "near equilibrium" price.

In the second test related to individual rationality we compare the outcome of ConFAst with an idealised situation where it is possible for a consumer to choose any allocation preferred at the "near equilibrium price", e.g. the maximum amount that she would prefer to consume at this price level. The result of the investigations is presented for three different sizes of markets (in number of participants), see Figure 2.9. The investigated participant has the same utility function in all three tests, it is only the number of participants on the market that varies. The interesting part is how fast the risk diminishes as the market grows.

The measure used, the relative utility, is the quotient

$$\frac{u(z_j) - p_j^f z_j}{u(z_j^*) - p^e z_j^*} \tag{2.5}$$

where z_j is the final allocation of participant j, p^e the "near equilibrium price", p_j^f the final price per unit of participant j, and z_j^* an allocation that participant j would have chosen if she had the opportunity to choose free given the "near equilibrium price" (e.g. the largest one that she would consider). Note that $u(z_j^*) - p^e z_j^*$ is something of an upper bound on how well of a participant can be on the market. This bound can not even always be theoretically supported. The measure is hence a very pessimistic one.

If the relative utility is negative it is not individually rational to participate on the market.

Nevertheless, the results of these small tests are promising. The mechanism performs well and the problems concerning individual rationality are negligible on a market already of moderate size. Even on a small market with 25 participants the risk that the investigated participant took is small, see Table 2.1 and Figure 2.9. Figure 2.9 even shows that it was only when the number of participants was less than a handful[11] that the outcome was negative. In all other market situations it was individually rational for this actor to participate on the market.

2.5 Summary

We have shown how the dynamics of small actors could be utilised to develop the power load market into a more efficient market place. This is made possible by the current development of computational and communicational resources in society, two examples are so called smart equipment and the Internet.

The entire area of electronic commerce is developing; more and more products and services are traded electronically. We focus on the energy market — one of the markets with a great potential — and suggest a mechanism, ConFAst, for this market that has a number of promising properties; it is able to

[11] The outcome was positive for the investigated participant on a market with three actors (not shown in the figure).

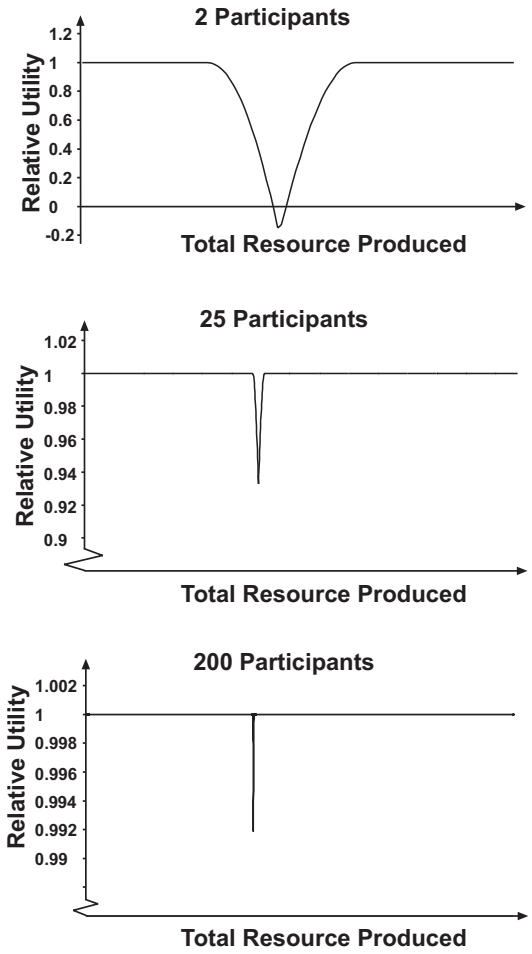

Figure 2.9: *Participating in a market based on* ConFAst *with non-concave utility is not individually rational. There is a (rather small) risk that the outcome for such a participant is worse than it would be if she would be able to buy what she wants paying e.g. the "near equilibrium price". A measure for this is the relative utility above, defined in Equation 2.5. The figure shows how fast the risk of participating with a non-concave utility diminishes when the market grows. It is only on an extremely small market, with no more than two participants that it turns out that it is not individually rational for the investigated participant (with non-concave utility) to take part in the market. The relative utility of the investigated participant is measured for every resource level that could be produced and allocated to the consumers on some markets of different size.*

handle both production and consumption side, it is suitable for fast electronic markets, it handles non-continuous demand, and it is well suited for distributed computing.

With this market mechanism we challenge the fundamental inefficiencies of today's energy systems; demand and prices vary significantly — particularly in shorter time frames — while at the same time many loads can be very flexible, c.f. Figure 2.1 and Figure 2.2.

We believe that this is a win - win possibility for both producers and distributors on one hand and consumers on the other. On top of that there are benefits from an environmental perspective too, due to smaller fluctuations in demand.

When consumers and small producers are introduced as active market participants non-concave utility functions is a reality. ConFAst offers possibilities to participate on the market with such utility. Even though the mechanism does not guarantee that it is individually rational for someone with non-concave utility to participate on the market, the risk of doing so is far less than to participate on a market not supporting the non-concave character of her utility function. Our tests show that on any market of reasonable size, the problem is negligible.

In extreme situations, such as the one sketched in the introduction to the paper, market mechanisms and energy systems are put on the test. We believe that ConFAst — with its dynamic character — has great advantages in such extreme situations as well as during normal conditions.

Chapter 3

Resource Allocation with Noisy Functions

We consider resource allocation with separable objective functions defined over subranges of the integers. While it is well known that (the maximisation version of) this problem can be solved efficiently if the objective functions are concave, the general problem of resource allocation with functions that are not necessarily concave is difficult.

In this article we show that for a large class of problem instances with noisy objective functions the optimal solutions can be computed efficiently. We support our claims by experimental evidence. Our experiments show that our algorithm in hard and practically relevant cases runs up to 40 - 60 times faster than the standard method.

3.1 Introduction

3.1.1 Resource Allocation

We consider resource allocation with separable objective functions defined over sub-ranges of the integers, as given by

$$\begin{array}{ll} \max_{r_1,r_2,\ldots,r_n} & \sum_{i=1}^{n} f_i(r_i) \\ s.t. & \sum_{i=1}^{n} r_i = R, \end{array} \tag{3.1}$$

where each function $f_i(r_i)$ is defined over a range of integers, I_i.

Resource allocation with separable objective functions is a fundamental topic in optimisation. When presenting algorithms for this problem, it is typically assumed that the objective functions are concave; very little is known about the case of non-concave functions. In the general case, where there is no assumption on the shape of the functions, the standard method is to use brute force and — for any pair of functions that are aggregated into one — test all possible solutions.

In this article, we present a new algorithm tailored for the case when there is no prior knowledge about the involved objective functions. The idea is to take advantage of favourable properties of the functions whenever possible. In effect, we obtain a new algorithm for efficient handling of a very general class of non-concave and noisy objective functions.

A typical function with such properties is illustrated in the lower left part of Figure 3.4. The function is noisy, and even with the noise removed, it is non-concave. The function has some regularity though: seen from a distance it looks "smooth". The basic idea of our algorithm is to divide the objective functions into a small number of intervals and to filter out the noise, such that each interval can be treated as either convex or concave. In this way we can use previous techniques [3] for efficient pair-wise aggregation of objective functions in combination with a neighbourhood search. Altogether, we manage to obtain a pruning of the search space; the number of allocations that needs to be tested is significantly reduced.

The robustness of our algorithm makes it very useful in practice. Furthermore, the overhead of the algorithm is small enough to allow for fast implementation. Indeed, it competes very well with the brute force algorithm, although the brute force algorithm has the advantage of being simple and straightforward. This is demonstrated by experimental evidence. It is even the case that we achieve surprisingly good results also for seemingly impossible cases. One such case consists of a set of completely random objective functions. At a first glance, it might seem that there is no hope to improve over the brute force method for such a problem, but our new algorithm offers a significant speedup by taking advantage of smoothness whenever possible.

3.1.2 Aggregating Objective Functions

The brute force algorithm, as well as our algorithm, is based on pair-wise aggregation of objective functions. In a typical implementation of the brute force method, the global objective function is computed by incorporating the functions one by one [8, pp. 47-50], but it is also possible to aggregate the functions in a balanced binary tree fashion [2]. In our experiments, we have chosen the second alternative mostly because this method is more suited for implementation in a distributed environment, which is the case in the application of power load management which we have in mind. We have no reason to believe that the experiments would show any major difference had the other method been used.

In either case, the basic step is the aggregation of two objective functions into one, see Figure 3.1. Given two functions f_1 and f_2 defined over subintervals of length I_1 and I_2, compute the aggregated function f_t, defined over a subinterval of length (at most)[1] $I_1 + I_2$:

$$f_t(r) = \max_{r_1+r_2=r} f_1(r_1) + f_2(r_2), \tag{3.2}$$

[1] If the total available resource to be allocated is smaller than $I_1 + I_2$, then the aggregated function need not be computed over the entire interval $I_1 + I_2$.

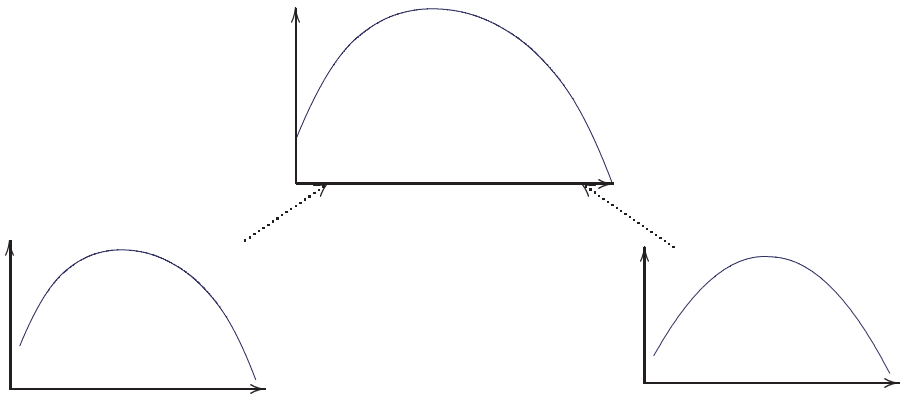

Figure 3.1: *The basic step when aggregating objective functions is the aggregation of two functions into one.*

and for each r in the subinterval compute the corresponding r_1 and r_2.

The complexity of aggregating two objective functions depends on the properties of the functions. Two cases are typically distinguished, e.g. [8, 6]:

- The general case, when no assumptions can be made about the functions. Then an aggregation requires $\Theta(I_1 I_2)$ time.
- The special case of two concave functions. Then the aggregation can be made much faster, in $\Theta(I_1 + I_2)$ time.

As the domain of the objective functions may be very large compared to the number of functions, the difference between these two complexities will be significant in many applications. Despite the fundamental nature of the problem and its practical importance, we have found nothing applicable done by others in recent literature, cf. [7], that tackles the problem with lower complexity than what is presented in the textbook by Ibaraki and Katoh [8], i.e. $\Theta(I_1 I_2)$.

In previous work [3], we have introduced a novel algorithm for pair-wise aggregation of objective functions. The main point of that work is that the complexity of the aggregation is adaptive to the shape of the two functions to be aggregated; the simpler curves the lower complexity.

More precise: Let f_1 and f_2 be two objective functions defined over intervals of length I_1 and I_2. If the two functions can be divided into s_1 and s_2 segments respectively, such that each segment is either convex or concave, see Figure 3.2, then the aggregated objective function and the corresponding optimal allocations can be computed in $\mathcal{O}(I_1 s_2 + I_2 s_1)$ time.

This article generalises the previous work to manage noisy cases. First, we are able to handle functions that are close to concave but noisy (see Figure 3.3) in an efficient way. Furthermore, we handle noisy functions that basically are non-concave but in some sense smooth (as the ones in Figure 3.4).

It is worth pointing out that noisy looking objective functions do not only occur when the input functions are noisy themselves; even with nicely shaped

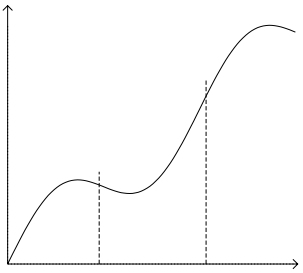

Figure 3.2: *Dividing a function into concave and convex segments.*

input functions the functions occurring after a number of aggregations may become irregular. One example is when the input functions are convex, see the experimental results in Section 3.3 and Figure 3.10.

An important contribution that goes together with the article is our implementation work and the experiments showing the robustness of our algorithm.

3.1.3 A note on the complexity of the general problem

It is worth noting that in most formulations of the resource allocation problem, the goal is not to compute a global objective function over the entire domain, but to compute the optimal allocation (and maybe the total value of this allocation) for a specific total resource. In that case, it may seem like a waste of time to compute the entire function. Indeed, if the objective functions are concave, a binary-search type algorithm solves the problem more efficient. The complexity is $O(n \log(I/n))$, where each objective is assumed to be defined over an interval of length I, and n is the number of objective functions.

However, for the general non-concave case, the situation is harder. In the textbook by Ibaraki and Katoh [8] it is pointed out that the general problem with *non-separable* objective functions is $\mathcal{NP}$-hard. It is also stated that it is unknown whether the problem with separable functions (i.e. the problem described in Eq. (3.1)) is $\mathcal{NP}$-hard or not. However, there is a simple proof that this problem is also $\mathcal{NP}$-hard. A sketch of a proof based on a reduction from the knapsack problem is given in our previous article [3].

It should be noted that a trivial algorithm [8] runs in $O(nR^2)$ time, where n is the number of functions and R is the maximal resource. The dependency on R makes the complexity pseudo-polynomial; the $\mathcal{NP}$-hard instances occur when (i) R is exponential in n and (ii) the description of the objective functions is polynomial in n. In e.g. the above referred reduction from the knapsack problem, each objective function can be described with $O(\log R)$ bits. Therefore, with polynomially sized input, the complexity of any resource allocation algorithm based on aggregation of the objective functions is exponential in n.

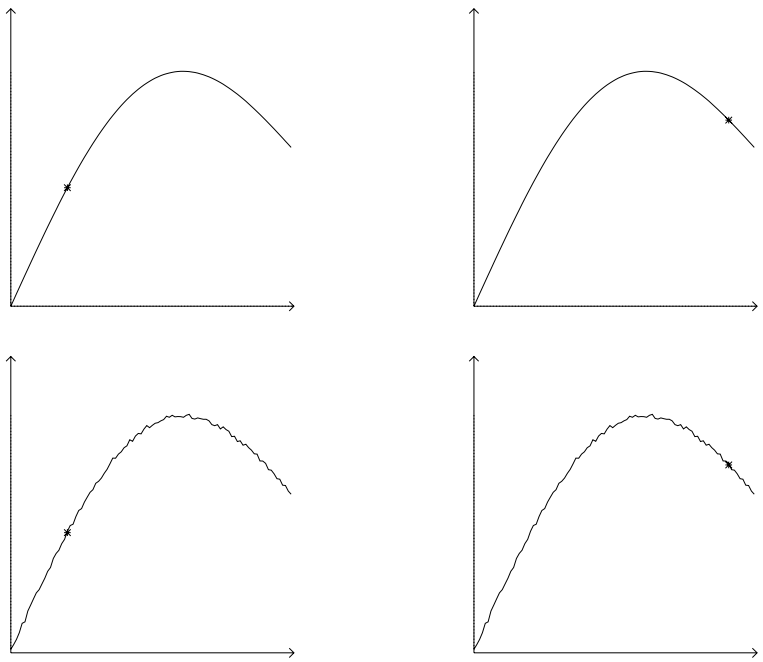

Figure 3.3: *Top row: The allocation of $r = r_1 + r_2$ indicated by the two dots is obviously not optimal and it is well known how to avoid testing it for optimality. Bottom row: It is easy to see that the allocation is not optimal also when the functions are slightly noisy and it is rather easy to avoid testing it for optimality.*

Hence, by the referred reduction from the knapsack problem, we can not hope for any exceptionally good algorithm for the general case, even when the functions are separable if the input is generated by an evil adversary. Nevertheless, we believe that in practice it can be expected that the objective functions will not be generated by an adversary, but rather show some "smoothness", and then the approach introduced in this article can be very efficient as long as the size of the problem instance does not explode. We believe that there are quite a few settings, such as power load management, where the resource R will not be exponential in the number of input functions n, i.e. the function description, with a proper resolution, will not have an exponential size. It might even be that the resolution is more fine-grained in subsystems than in the overall system; e.g. in a power load management setting it might be useful to have a resolution of say 100 W within a building, but when dealing with a nuclear power plant this is not an appropriate resolution. How to rescale an objective function that might have a noisy character is not dealt with in this article.

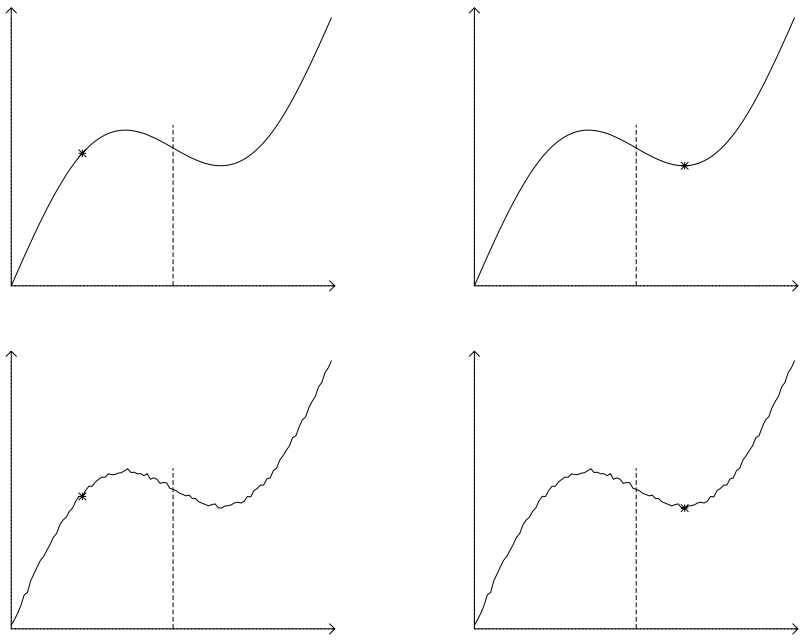

Figure 3.4: *Top row: In previous work [3] we have shown how to efficiently compute the optimal allocation when the objective functions are not restricted to concave functions. The basic idea is to divide the functions into segments. Bottom row: In this article we turn to the practically relevant case when the objective functions can be accurately approximated with fairly well shaped functions. By filtering out the noise the functions can be partitioned into segments as the functions in the top row. Also in this figure it is clear that the marked allocations can not be optimal, though proving it for the bottom case is not trivial.*

3.2 Main Result

In this work we show that it is possible to aggregate two objective functions efficiently and optimally if the objective functions could be divided into a small number of segments that essentially are concave and convex but have some rather low amplitude noise (see Figure 3.3 and 3.4). We present an algorithm, Algorithm 3.4.2, that is based on the algorithm presented in our previous work, [3], but is generalised to manage low amplitude noise.

It is well known, [8], [2], that it is possible to aggregate two functions by testing all combinations (referred to as the brute force method).

Statement 3.2.1 *Let f_1 and f_2 be two objective function (not necessarily con-*

cave) defined over intervals of length I_1 and I_2. The aggregated function and the corresponding allocations can be computed in $\mathcal{O}(I_1 \cdot I_2)$ time by testing of all combinations.

With n objective functions to aggregate this can be done in $\mathcal{O}(nR^2)$ time, where R is the resource to be allocated.

We have shown that when it is possible to divide the functions into a (preferably small) number of segments that are either concave or convex it is possible to do better.

The algorithm is a two step algorithm. First the functions are divided into segments (in linear time), then all segments of the first function are combined with all segments of the second in a search for candidate allocations. We have shown that, for each segment combination, this search can be done in time linear in the size of the segments.

Statement 3.2.2 *Let f_1 and f_2 be two objective functions defined over intervals of length I_1 and I_2. Furthermore, assume that the two functions can be divided into s_1 and s_2 segments respectively, such that each segment is either convex or concave. Then the aggregated objective function and the corresponding optimal allocations can be computed in $\mathcal{O}(I_1 s_2 + I_2 s_1)$ time [3].*

In some situations the growth of the complexity as a function of the number of segments turns out to be a problem since the number of segments often grows large due to low amplitude noise. This noise arises when (some of) the functions are non-concave, cf. Section 3.3. A consequence is that the overall complexity grows and reaches the complexity of testing all combinations.

A way of solving this is to filter out the noise by accepting a small hull distance (defined in Section 3.4.2) when dividing the functions into (concave and convex) segments. Then the neighbourhood of each candidate allocation is searched until the (verified) locally optimal solution has been found. The algorithm combines all segments of the first function with all segments of the second one. Since a globally optimal allocation has to be locally optimal for some segment combination, all globally optimal allocations are found this way.

Statement 3.2.3 *Let f_1 and f_2 be two objective function (not necessarily concave), with relatively small amplitude noise. Furthermore, assume that after a small adjustment of each function value, the functions could be divided into a small number of segments, such that each segment is either convex or concave.*

Then the aggregated objective function and the corresponding optimal allocations can be computed in time considerably less than what is needed for a complete search of all possible combinations.

The complexity is dependent on the number of segments the functions are divided into and the amplitude of the noise.

However, with n objective functions to aggregate we still have not managed to achieve an overall complexity lower than $\mathcal{O}(nR^2)$ (see the following sections).

Still, the algorithm offers a pruning of the search space (compared to a complete search of all combinations) and running times that in practical cases are considerably less than the running times of a the brute force algorithm (see the following section).

3.3 Experimental Results

Before we present the detailed algorithms, we show some illuminating test results.

We have run a series of tests using the algorithm in a tree structured [2] system for resource allocation. The system is implemented in Java and the tests that are referred below were run on a Sun Ultra 10 (tests have been run on other Java systems too and there are some small variations in the results).

3.3.1 Tree-structured Aggregation

As pointed out above, the basic step of solving the resource allocation problem is the aggregation of two objective functions into a new objective function. In our experiments, we have used this basic step within a tree structure. As leaves of the tree, we have the original objective functions, the internal nodes are aggregated functions, the global objective function being the root, cf. Figure 3.5. One great advantage of this is that it is highly suitable for distributed resource allocation. With this structure it is possible to distribute the computational work in the network and to avoid communication bottlenecks, since all the computation does not have to be done at one central point [2].

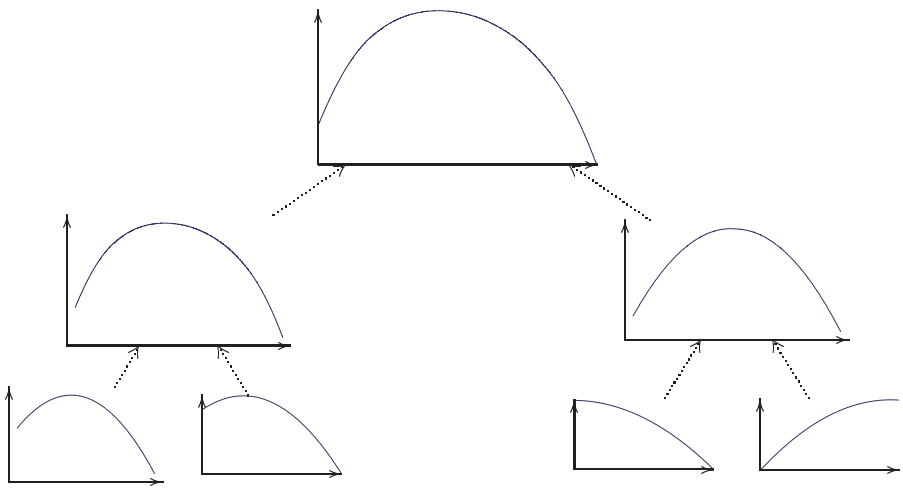

Figure 3.5: *We have presented an algorithm for resource allocation that is designed with distributed resource allocation in mind. In this article we are focusing the pair-wise aggregation that is an essential subroutine of e.g. this algorithm.*

In distributed resource allocation there are a couple of communicational aspects that must be dealt with when constructing the algorithms. First, communication is relatively slow compared to computations, and second, there is an overhead of information in every message sent over the network (e.g. address and error correction information). Therefore, up to a certain size[2] the cost of sending a message is more or less independent of the message size. This gives a trade-off between the number of messages and the message size and it is often preferable to send a few larger messages instead of many small.

For the above reasons, if the object functions of two subsystems are to be aggregated at some node in a distributed resource allocation system, it is often the case that the most efficient way is letting the subsystems compute and communicate all of their respective objective functions as one aggregate function. As a consequence, computing the aggregated objective functions for a subsystem is highly important. Furthermore, it is an advantage to have the aggregated objective function at hand at the top level, as it gives a full picture of how important the resource is within the system, and it gives possibilities to react fast on changes.

3.3.2 The Experiments

In these tests[3] our algorithm for pair-wise aggregation is used within an algorithm that

(i) is tree structured (Figure 3.5),

(ii) in the bottom, where the functions are defined over small intervals, performs a complete search of all combinations, like the brute force algorithm,

(iii) uses our previous algorithm until the number of segments is considered too large, and

(iv) when the number of segments is considered too large turns to using the algorithm introduced in this article.

The main point of this section is to show that our novel algorithm not only prunes part of the search space compared to the more simple algorithm of testing all combinations, but also is faster for practically relevant instances (despite its larger overhead).

Timing Considerations

In our experiments we have measured the total running time as well as the time and number of evaluations used at the top level. The aggregation at the lower levels of the aggregation hierarchy is of minor interest (the functions are typically defined over small intervals and a complete search is of low cost). Therefor we focus on the top level measure in our evaluation (see the tables below).

[2] The actual size depends on the structure of the communication system.

[3] The Java classes needed for running the tests, and a couple of other small test programs, can be downloaded from http://www.csd.uu.se/~perc/papers/noisy/.

Experiment 1: Two Functions Based on a Set of Five Hundred Mixed Functions

The first example is chosen to reflect our application area of power load management. It is realistic to assume that the objective functions of the vast majority of the loads in an electricity grid are concave. However, some functions will be non-concave, e.g. staircase shaped. A few levels up in the aggregation hierarchy the aggregated objective functions could be accurately approximated with a concave function, but a low amplitude noise prevents our previous aggregation algorithm from being highly efficient, see Figure 3.6, (also, because of this noise, standard algorithms for concave functions cannot be applied at all).

The functions that we aggregate in this test are five hundred functions that are randomly generated and ordered. One third of them are concave, one third are convex, and one third are staircase shaped. The length of the intervals that the functions are defined over differs up to a factor five. All variations are randomly distributed over the set of functions. The number of segments accepted before trying to reduce it is chosen so that the algorithm for pair-wise aggregation based on hull functions is used at higher levels of the aggregation hierarchy. A series of tests were run with different hull distances (the number of segments and the neighbourhood searched depends on the hull distance), as described in Table 3.1.

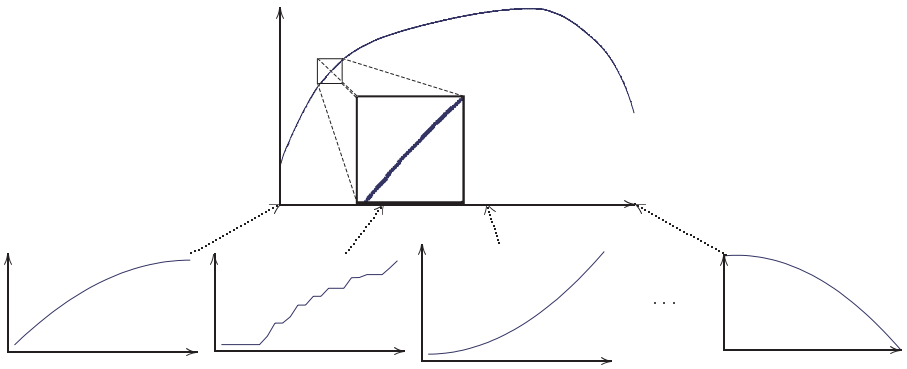

Figure 3.6: *The resulting function when one hundred mixed functions are aggregated is almost concave. (A part of the aggregated function is enlarged so that it is possible to see the noise.)*

With this input and configuration the aggregation at the top level was between three and 44 times faster with our new algorithm compared to the simple algorithm of Statement 3.2.1. The variance is due to the choice of hull distance when constructing the segments, see Section 3.4.3. Counting actual comparisons the difference is even bigger.

Table 3.1: *Aggregating 500 mixed functions. Our algorithm for pair-wise aggregation compared with a complete search of all possibilities. The ϵ is the hull distance (defined in Section 3.4.3). In this case the running time of the top level aggregation becomes independent of the hull distance from $\epsilon = 4$ since the algorithm is able to construct one concave hull function per function, cf. Section 3.4.3.*

Instance	Total Time (s)	#Evaluations (top level)	Time (s) (top level)	Eval. Ratio	Time Ratio
Complete Search	72.1	61,868,250	33.2	1	1
Our Alg., $\epsilon = 0.01$	33.3	8,692,049	11.5	7.1	2.9
Our Alg., $\epsilon = 0.1$	12.3	890,327	2.4	69.5	13.6
Our Alg., $\epsilon = 0.25$	10.4	602,657	1.4	102	23.8
Our Alg., $\epsilon = 0.5$	9.8	557,067	1.2	111	28.8
Our Alg., $\epsilon = 1$	9.6	666,945	1.4	93	24.4
Our Alg., $\epsilon = 2$	9.9	621,584	1.3	99	24.9
Our Alg., $\epsilon = 4$	9.3	482,156	0.7	128	44.7

Experiment 2: A Noisy Concave Function and a Smooth Function

Often the function that is noisy could be accurately approximated with a single concave function, see Figure 3.3 and 3.6. Although this is the normal case, there are situations, e.g. in power load management, where it is not.

Table 3.2: *Aggregating a staircase shaped function, e.g. the objective function of a power plant, and a noisy concave function. Our algorithm for pair-wise aggregation compared with a complete search of all possibilities. The ϵ is the hull distance (defined in Section 3.4.3).*

Instance	#Evaluations	Time (s)	Evaluation Ratio	Time Ratio
Complete Search	8,865,000	4.83	1	1
Our Alg., $\epsilon = 0.01$	874,046	1.66	10.1	2.9
Our Alg., $\epsilon = 0.1$	307,232	0.56	28.8	8.6
Our Alg., $\epsilon = 0.25$	357,254	0.52	24.8	9.3
Our Alg., $\epsilon = 0.5$	479,622	0.92	18.5	5.3
Our Alg., $\epsilon = 1$	558,127	0.67	15.9	7.2
Our Alg., $\epsilon = 2$	794,694	0.95	11.2	5.1

A power plant often has an objective function that is staircase shaped (as the one in Figure 3.7, bottom left). Due to its large impact on the system it is introduced on a rather high level in the aggregation hierarchy (so that it is

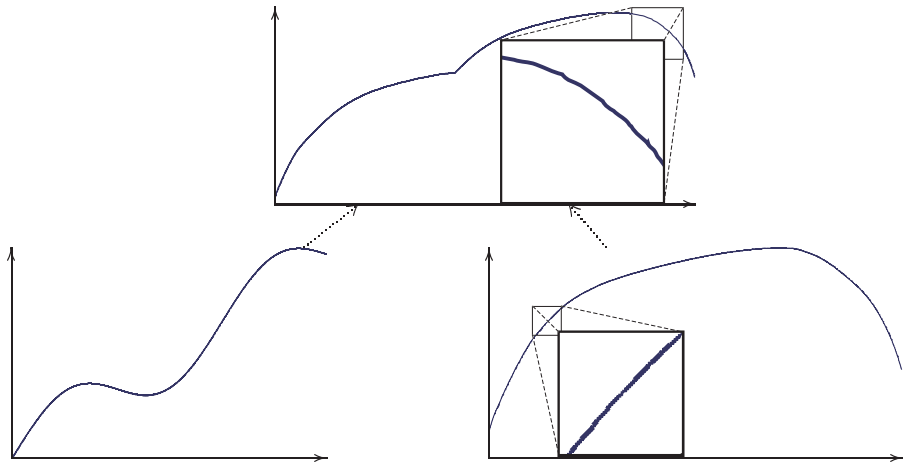

Figure 3.7: *If e.g. a power plant with a staircase shaped objective function is aggregated with a noisy concave function the result is a non-concave noisy function. (Parts of the noisy functions are enlarged so that it is possible to see the noise.)*

in balance with the other nodes on the same level). When a staircase shaped function is aggregated with a noisy concave function the result often is non-concave and noisy, cf. Figure 3.7.

The functions used in this test was a staircase shaped function (e.g. the objective function of a power plant) and a noisy function that was close to concave (a function that was the result of an aggregation of a set of mixed functions, as the set in experiment one). The functions were defined over around 3,000 sample points each. As described in Table 3.2 our algorithm was between three and ten times faster than the brute force algorithm.

Experiment 3: Two Sine Waves With Noise

The next test case appeared to be the hardest one in our series of tests. The output of the preceding test is a non-concave and noisy function, see Figure 3.7. Here we aggregate two such functions, constructed as long sine waves with a low amplitude noise on top.

The construction of the functions was based on sine waves in the interval $[0 \ldots 2\pi]$ stretched to the length of 2,500 sample points with some random noise on top (the maximum amplitude of the noise is 1% of the amplitude of the functions). Except for the noise (that is randomly generated) the two functions are identical, this makes the task harder for our algorithm.

It is harder for the algorithm to solve this instance faster than a complete

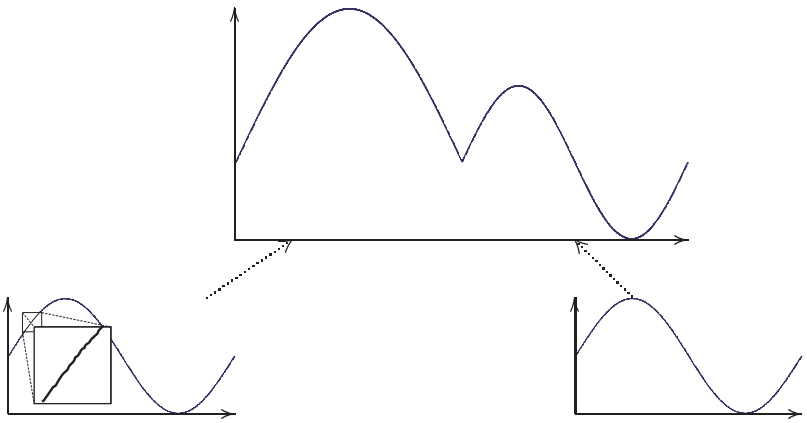

Figure 3.8: *Aggregation of two noisy sine wave functions gives the function at the top. The input in the bottom row. (A part of the left function is enlarged so that it is possible to see the noise.)*

search does. Still, even with the worst performance of the instance (when the algorithm cannot reduce the number of segments and has to perform as many evaluations as the brute force algorithm) the running time does not exceed the running time of the brute force algorithm with more than a factor two. On the other hand, with a good choice of ϵ the algorithm is twice as fast as the brute force algorithm.

The actual output of the preceding experiment could be used as input to a similar test (i.e. two functions shaped as the top function of Figure 3.7). Such a test showed that our algorithm managed to run up to thirteen times faster than the brute force algorithm, and it was not as sensitive to the choice of ϵ as in the test shown in Table 3.3.

Experiment 4: Two Functions Based on a Set of Five Hundred Random Functions

As an adversary test based on a larger set of input functions we have tried random noise functions, see Figure 3.9. However, even here the algorithm behaves surprisingly well.

With a set of five hundred functions constructed with random values in $[0 \ldots 1]$ and a good choice of hull distance the algorithm runs the top level aggregation up to 65 times faster than the complete search algorithm, Table 3.4.

Experiment 5: Two Functions Based on a Set of Five Hundred Convex Functions

As another adversary test, we used a set of five hundred convex functions. With this set of functions the result is slightly less favourable than with the random

Table 3.3: *Aggregating two sine wave functions. Our algorithm for pair-wise aggregation compared with a complete search of all possibilities. The ϵ is the hull distance (defined in Section 3.4.3). Not until $\epsilon = 0.5$ the algorithm is able to reduce the number of segments at all (compared to the number of segments one gets with $\epsilon = 0$).*

Instance	#Evaluations	Time (s)	Evaluation Ratio	Time Ratio
Complete Search	6,250,000	2.2	1	1
Our Alg., $\epsilon = 0.25$	6,250,000	3.49	1	0.6
Our Alg., $\epsilon = 0.5$	744,861	1.00	8.4	2.2
Our Alg., $\epsilon = 1$	1,043,782	1.34	6.0	1.6
Our Alg., $\epsilon = 2$	1.438,994	1.77	4.3	1.2

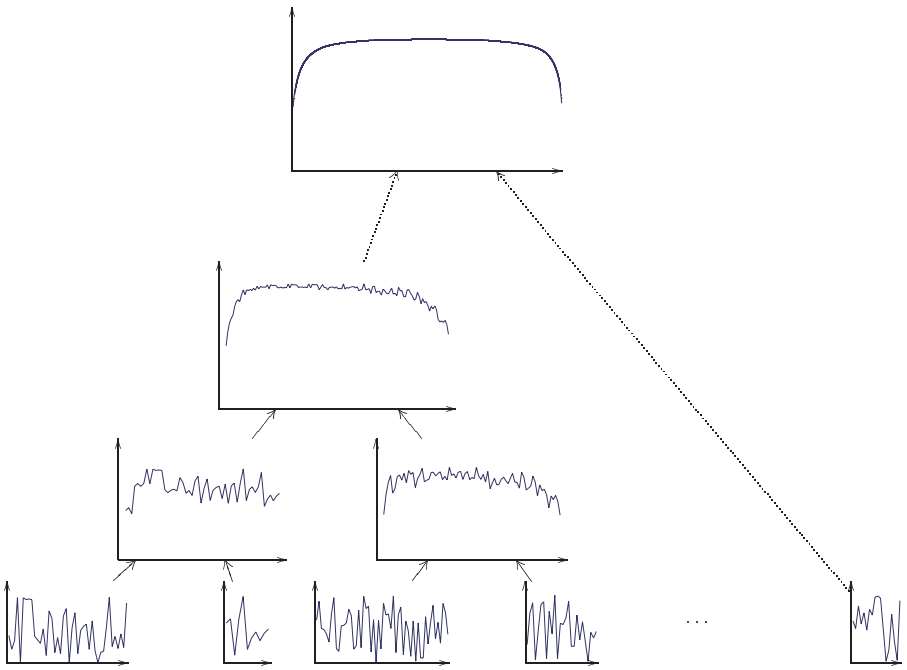

Figure 3.9: *The resulting function when five hundred random functions are aggregated is almost concave. Already at the second aggregation level the almost concave shape of the aggregated function is obvious.*

noise functions and the hull distance has to be larger to achieve running times that are significantly better than with the brute force approach, Table 3.5. This is due to the shape of the functions that gives more jagged functions higher up in the aggregation hierarchy, see Figure 3.10.

Table 3.4: *Aggregating 500 random functions. Our algorithm for pair-wise aggregation compared with a complete search of all possibilities. The ϵ is the hull distance (defined in Section 3.4.3). The running time of the top level aggregation becomes independent of the hull distance from $\epsilon = 0.1$ since the algorithm is able to construct one concave hull function per function, cf. Section 3.4.3.*

Instance	Total Time (s)	#Evaluations (top level)	Time (s) (top level)	Eval. Ratio	Time Ratio
Complete Search	69.9	62,115,388	38.6	1	1
Our Alg., $\epsilon = 0.01$	28.4	893,310	2.45	69	15.7
Our Alg., $\epsilon = 0.1$	6.68	353,444	0.59	175	65.8

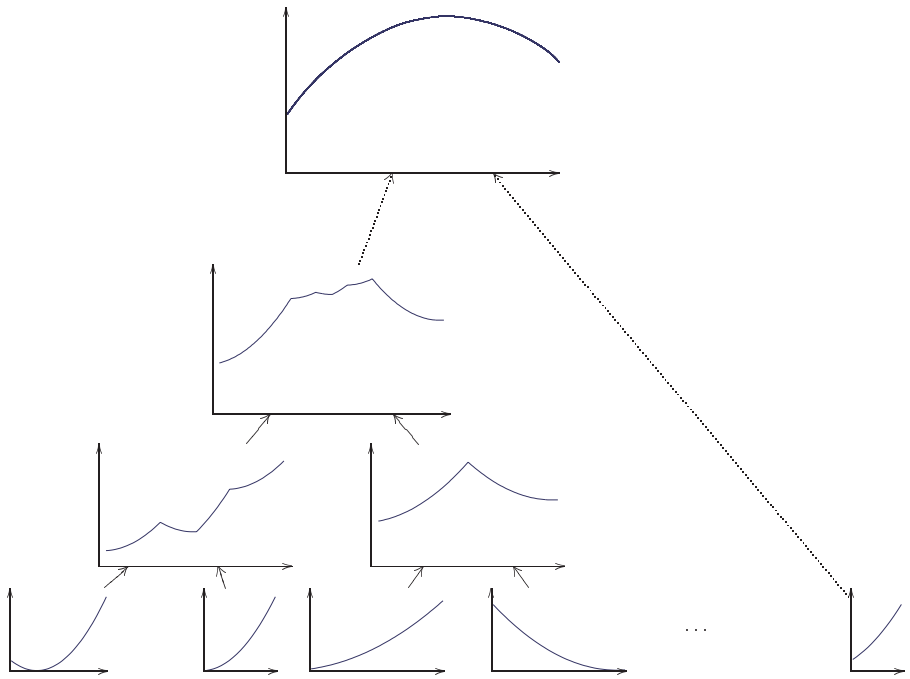

Figure 3.10: *With 500 convex functions as input the shape of the aggregated functions is rather jagged. As the experiments show, the ϵ has to be larger (compared to an input of mixed or random functions) when we want to obtain an aggregation that is as efficient as possible (compare Table 3.5 with Table 3.1 and 3.4).*

3.3.3 Summary of the Experiments

Our new algorithm not only theoretically prunes part of the search space, but also runs faster than the more simple method of testing all combinations (despite

Table 3.5: *Aggregating 500 convex functions. Our algorithm for pair-wise aggregation compared with a complete search of all possibilities. The ϵ is the hull distance (defined in Section 3.4.3). The running time of the top level aggregation becomes independent of the hull distance from $\epsilon = 8$ since the algorithm is able to construct one concave hull function per function, cf. Section 3.4.3.*

Instance	Total Time (s)	#Evaluations (top level)	Time (s) (top level)	Eval. Ratio	Time Ratio
Complete Search	56.0	57,388,068	28.1	1	1
Our Alg., $\epsilon = 0.01$	71.4	56,756,771	3.80	1.0	0.74
Our Alg., $\epsilon = 0.1$	70.0	56,417,480	37.1	1.0	0.76
Our Alg., $\epsilon = 0.25$	64.4	42,068,396	32.5	1.3	0.86
Our Alg., $\epsilon = 0.5$	48.0	18,394,745	19.6	3.1	1.4
Our Alg., $\epsilon = 1$	36.1	9,406,363	13.1	6.1	2.15
Our Alg., $\epsilon = 2$	26.0	5,629,681	8.2	10.1	3.4
Our Alg., $\epsilon = 4$	22.2	4,008,639	6.5	14.3	4.3
Our Alg., $\epsilon = 8$	13.7	1,187,319	1.5	48	19

its larger overhead). It should also be noted that whereas the implementation of the brute force algorithm can be expected to be close to optimised (because of its simplicity) much could probably be done to improve our first test implementation of the more complicated algorithm, particularly regarding the partitioning of the aggregated functions into segments (which is not in focus in this article).

3.4 Technical Details

We solve the problem of noisy functions described above with an algorithm whose basic idea can be described in the following high level way:

(i) Divide the functions to be aggregated into segments that with a given hull distance (Section 3.4.3) could be viewed as convex and concave, and accurately describes the functions, see Figure 3.11 and 3.12,

(ii) follow our previous algorithm (designed without the idea of a hull distance) [3] on the hull functions (that are either concave or convex) over the segments but run a few extra steps in the search to guarantee that the solution is optimal for the original functions. (Hull functions and hull distances are defined in Section 3.4.2.)

3.4.1 Algorithm Without Noise Filtering

The basic idea of our previous algorithm [3] is to divide the functions into (convex and concave) segments and then perform a linear search of each segment

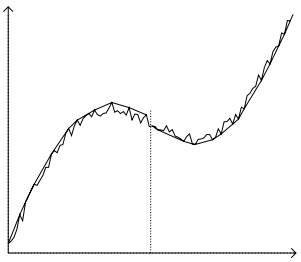

Figure 3.11: *A noisy function is divided into segments with a hull function that is either concave or convex.*

combination for candidate allocations. All candidates of the segment combination are found during the search.

Our algorithm for noisy functions uses the previous algorithm for efficient resource allocation as an essential subroutine. For that reason we start the technical section with a short description of the algorithm.

Recall the definitions of f_1, f_2, f_t, I_1 and I_2 from Equation (3.2). The domain intervals of the two functions f_1 and f_2 to be aggregated are $[R_1^{min} \ldots R_1^{max}]$ and $[R_2^{min} \ldots R_2^{max}]$ respectively, i.e. $I_i = R_i^{max} - R_i^{min} + 1$, $i = 1, 2$. Let f_t be represented by a vector $\mathbf{f}_t$, and let the optimal allocations of r_1 and r_2 for each r be represented by the vectors $\mathbf{r}_1$and $\mathbf{r}_2$ respectively.[4] As an example, $\mathbf{f}_t(7)$ corresponds to the highest value of $f_1 + f_2$ for $r = R_1^{min} + R_2^{min} + 6$, and $\mathbf{r}_1(7)$ and $\mathbf{r}_2(7)$ represent the optimal allocation of this resource.

We use algorithm 3.4.1 to compute $\mathbf{f}_t$, $\mathbf{r}_1$, and $\mathbf{r}_2$.

Algorithm 3.4.1 *Algorithm for aggregation of two functions.*
For every element, i, of the $\mathbf{f}_t$ vector
  $\mathbf{f}_t(i) \leftarrow -\infty$
Divide f_1 and f_2 into concave and convex segments
For every combination of segments with one segment from f_1 and one from f_2
  For every local candidate allocation (defined below), (r_1, r_2), of the segments
    If $\mathbf{f}_t(r_1 + r_2) < f_1(r_1) + f_2(r_2)$
    {
      $\mathbf{f}_t(r_1 + r_2) \leftarrow f_1(r_1) + f_2(r_2)$
      $\mathbf{r}_1(r_1 + r_2) \leftarrow r_1$
      $\mathbf{r}_2(r_1 + r_2) \leftarrow r_2$
    }

We use Δ to denote function differences.

[4] Again, if the total resource which can be allocated is smaller than $I_1 + I_2$, then smaller vectors can be used.

Definition. 3.4.1 *Let* $\Delta f(r) = f(r+1) - f(r)$.

Further, as R^{min} and R^{max} denotes the start and end points of the interval a function is defined over, we define the end points of a segment within a function.

Definition. 3.4.2 *Let* r^{min} *be the start point of the interval of a particular segment and* r^{max} *the end point.*

We define [3] a global candidate as satisfying the following criterion (cf. the Kuhn-Tucker criterion [8]):

Definition. 3.4.3 *An allocation* (r_1, r_2) *is a global candidate if*

$$r_1 = R_1^{min} \text{ or } r_2 = R_2^{max} \text{ or } \Delta f_1(r_1 - 1) \geq \Delta f_2(r_2) \tag{3.3}$$

and

$$r_1 = R_1^{max} \text{ or } r_2 = R_2^{min} \text{ or } \Delta f_1(r_1) < \Delta f_2(r_2 - 1). \tag{3.4}$$

In the same way we define a local candidate:

Definition. 3.4.4 *An allocation* (r_1, r_2) *is a local candidate if*

$$r_1 = r_1^{min} \text{ or } r_2 = r_2^{max} \text{ or } \Delta f_1(r_1 - 1) \geq \Delta f_2(r_2) \tag{3.5}$$

and

$$r_1 = r_1^{max} \text{ or } r_2 = r_2^{min} \text{ or } \Delta f_1(r_1) < \Delta f_2(r_2 - 1). \tag{3.6}$$

Every local optimum is a local candidate. Every global optimum is a global candidate and every global candidate is a local candidate of some segment combination [3].

Lemma 3.4.1 *Algorithm 3.4.1 correctly computes* f_t *and the optimal allocations in* $\mathcal{O}(I_1 s_2 + I_2 s_1)$ *time[3].*

3.4.2 Algorithm Based on Noise Filtering

A problem of using the algorithm above is that the number of segments might grow to size $\mathcal{O}(I)$ with I sample points in the function representations. After a few levels of aggregation the resulting objective function often is rather well shaped, but has some relatively small amplitude noise, see Figure 3.3 and 3.4. The noise originates from the non-concave input functions. Therefore, at some stage the strategy has to be changed. When the number of segments of the functions is considered too big, segments based on concave and convex hull functions (defined in Section 3.4.3) are generated and used.

In this case, if the segments are sufficiently small, one should consider doing a simple complete search of all possibilities[5] in $\mathcal{O}(k_1 k_2)$ time (k_i = the length of the ith segment), due to the overhead of the search strategy below. With large segments and/or small hull distances (defined below) the strategy introduced in this article is appropriate.

The strategy is based on the concepts of hull functions and hull distances that we define as follows:

Definition. 3.4.5 *$\hat{f}$ is the smallest concave function such that $\hat{f}_i(r) \geq f_i(r)$ for all $r, r^{min} \leq r \leq r^{max}$, within a specific segment.*
In the same way $\check{f}$ is the largest function that is convex such that $\check{f}(r) \leq f(r)$ for all $r, r^{min} \leq r \leq r^{max}$.
These are illustrated in Figure 3.12.

We use $\hat{f}$ and $\check{f}$ as functions guiding the search. In this way the number of segments that the function is divided into is reduced.

Definition. 3.4.6 *The* hull function *of a segment is either $\hat{f}$ or $\check{f}$.*

Definition. 3.4.7 *We define the* hull distance, *ϵ, as the maximum distance between f and $\hat{f}$ or $\check{f}$ (depending on which one is used as hull function of the segment), see Figure 3.12.*

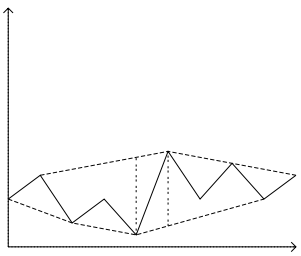

Figure 3.12: *From top down the following functions are shown: $\hat{f}_i(r)$ (dashed line), $f_i(r)$ (solid line), $\check{f}_i(r)$ (dashed line). Either $\hat{f}$ or $\check{f}$ can be used as hull function on the segment if the hull distance, ϵ, (vertical dashed lines) does not exceed the given tolerance.*

The hull functions, $\hat{f}$ and $\check{f}$, are constructed in such a way that the distance between the hull function and the true function never exceeds ϵ in any point, for some small $\epsilon \geq 0$. This ϵ is used in the following section to define the neighbourhoods that have to be searched. There is a trade-off between the number of segments and ϵ; a larger ϵ gives a smaller number of segments but a larger neighbourhood to search in each step (and the other way around).

[5] This is done in the test implementation used in the experiments of Section 3.3.

We now introduce an algorithm for finding the optimal allocations which also is applicable when the functions have some small amplitude noise, the algorithm uses Algorithm 3.4.1 as a subroutine combined with a neighbourhood search.

Algorithm 3.4.2 *Algorithm for aggregation of two functions.*
For every element, i, of the $\mathbf{f}_t$ vector
  $\mathbf{f}_t(i) \leftarrow -\infty$
Divide f_1 and f_2 into segments with concave and convex hull functions
For every combination of segments with one segment from f_1 and one from f_2
  For every local candidate allocation[6]*, (r_1, r_2) of the hull functions*
    {
      Search the neighbourhood[7]
      If $\mathbf{f}_t(r_1' + r_2') < f_1(r_1') + f_2(r_2')$ for any (r_1', r_2') of the neighbourhood of (r_1, r_2)
      {
        $\mathbf{f}_t(r_1' + r_2') \leftarrow f_1(r_1') + f_2(r_2')$
        $\mathbf{r}_1(r_1' + r_2') \leftarrow r_1'$
        $\mathbf{r}_2(r_1' + r_2') \leftarrow r_2'$
      }
    }

Theorem 3.4.1 *Algorithm 3.4.2 correctly computes f_t and the optimal allocations.*

The proof is given i Section 3.4.5.

3.4.3 Constructing Segments Based on Hull Functions

The presentation in the previous section relied on the construction of a number of segments with hull functions that are either convex or concave. The construction of these segments is not in focus of this article, but we give a few reasonable methods here for completeness.

To construct such a hull function the incremental algorithm for building the convex hull of a point set, see e.g. [5], is applied on (a part of) the objective function.

The convex hull is built in linear time since the points describing the function are already sorted. This gives a lower path equal to $\check{f}$ and an upper path equal to $\hat{f}$.

One basic problem is where to set the segment borders to get

(i) a small number of segments, and

(ii) a small hull distance.

[6] The candidates are found by Algorithm 3.4.1 applied to the hull functions of the segment combination.

[7] Described in the lemmas of Section 3.4.4.

Any one of these aims could be set on focus. One possibility is to control the number of generated segments and loosen the ambitions to get a small hull distance, the other to control the hull distance taking the risk that the number of segments gets to high. But it is a lot harder to solve the problem using some tradeoff between the two.

Focus on the Number of Segments

One linear time strategy is to first approximate the whole function with a single concave hull function and then, if a maximum of s segments is acceptable, divide the difference between the integrals of the hull function and the true function by s. The result d could be used as a limit when the segments are constructed in the following step.

The next step is to go over the function once more and construct the segments that are to be used. The difference between the integrals of the hull function and the true one is not allowed to exceed d in any segment. Every time the difference reaches d the current segment is ended and a new one introduced.

The last step is to go over the resulting segments and check the maximum of their respective hull distances and set ϵ to that value.

The strength of this method is that we have full control of the maximum number of segments in the output and that it runs in linear time. A disadvantage is that the hull distance could become very large compared to the ϵ of an optimal segment division. This is a problem that shows up in practice as soon as the function is not essentially concave or convex, e.g. a staircase shaped function as in Figure 3.4 (bottom). The methods used in Section 3.4.4 to find all possible optimal allocations are depending on this ϵ not to be too large, so this may be a major problem.

Focus on the Hull Distance

An alternative is to start with the tolerated hull distance.

First construct a hull on the entire function. If the hull distance is not bigger than tolerated for either $\hat{f}$ or $\check{f}$ we are done. Otherwise divide the segment into two and repeat the process recursively. As long as the segments are divided in a balanced way the total cost of the operation is $\mathcal{O}(I \log I)$.

Dividing the segment could be done e.g. where the distance between the true function and the hull function is largest (a strategy that is not guaranteed to be $\mathcal{O}(I \log I)$). It could also be divided the more simple way, in two equal halves. With this method we win control over the hull distance, on the other hand the cost is higher, and there is no guarantee that the number of segments is acceptable. The quality is hence dependent on a reasonable choice of the hull distance.

The choice of accepted hull distance is somewhat crucial. If it is slightly too small the number of segments could end up close to the original number (i.e. with $\epsilon = 0$). Our tests have shown that a factor two could make the difference between a minimum number of segments and far to many (see Table 3.3). An

error that is a bit too large is not that serious, what happens is that some more work is done searching optimal allocations since the breaking conditions are related to this hull distance (when one or both of the segments are convex).

In many application areas, we believe that an initial choice could be based on experience. If the result of the segmentation with a particular ϵ is not acceptable it has to be changed and the segments rebuilt. A reason that this might be tolerable is that when this aggregation is done over and over again it is plausible that an ϵ that has worked well in the past will do so this time too.

Another possible strategy is to have a small initial ϵ when starting to build the segments and if the number of segments grows large let the ϵ grow too. This may give the different segments of a function different segmentation distances but to a certain extent that is not a big problem (as long as the ϵ does not grow too big).

In our test implementation we have chosen to focus on the hull distance and have a fixed ϵ. Furthermore, we have applied the simple strategy of dividing the segments in equal halves as long as the hull distance is to big. We have found it to commonly generate high quality output.

3.4.4 Finding all Possible Candidates in Segments With Convex and Concave Hull Functions

When a reduced number of segments with an accepted hull distance have been constructed the search for optimal allocations takes place. This is done combining each segment of f_1 with each segment of f_2, see Algorithm 3.4.2.

There are three possible combinations of segments:

- two segments with concave hull functions,
- two segments with convex hull functions, and
- one segment with a concave hull function and one with a convex hull function.

The main principle of the new algorithm is to apply the old algorithm [3] to the hull function and use the candidates as starting points for a neighbourhood search for the optimal solution on f_1 and f_2.

Any globally optimal allocation has to be locally optimal on some segment combination, and we define a locally optimal allocation as follows:

Definition. 3.4.8 *For a given resource r and pair of segments (s_1, s_2) we define a* local optimum *of the segment combination as a resource combination $(r_1, r_2), r_1 + r_2 = r$, with $f_1(r_1) + f_2(r_2) \geq f_1(r'_1) + f_2(r'_2)$ for all $(r'_1, r'_2), r'_1 + r'_2 = r$, within s_1 and s_2 respectively.*

Definition. 3.4.9 *We define ξ_δ such that $f_1(r_1 - \xi_\delta) + f_2(r_2 + \xi_\delta) \geq f_1(r_1 - \chi) + f_2(r_2 + \chi)$ where $0 \leq \xi_\delta \leq \delta,\ 0 \leq \chi \leq \delta$.*

That is, starting at some (r_1, r_2) and searching downwards in r_1 and upwards in r_2 within a segment combination $f_1(r_1 - \xi_\delta) + f_2(r_2 + \xi_\delta)$ is the most valuable of all allocations of $r_t = r_1 + r_2$ found moving δ steps away from (r_1, r_2). The definition is symmetric searching upwards in r_1 and downwards in r_2.

Aggregating Two Segments With Concave Hull Functions

The optimal allocation, (r_1, r_2), of the resource $r_t = r_1 + r_2$ on the hull functions $\hat{f}_1$ and $\hat{f}_2$ is used as the starting point of a search for the optimal allocation (on f_1 and f_2). If $f_1(r_1) + f_2(r_2) = \hat{f}_1(r_1) + \hat{f}_2(r_2)$ there is no need for searching, otherwise move one step a time down in r_1 and up in r_2 searching for the optimal allocation until the condition in Lemma 3.4.2 is fulfilled or a segment border is reached. In a symmetric way, test raising r_1 and lowering r_2.

Lemma 3.4.2 *Assume that (r_1, r_2) is optimal for the concave hull functions on the segments, where $r_1^{min} \leq r_1 \leq r_1^{max}, r_2^{min} \leq r_2 \leq r_2^{max}$. If*

$$\hat{f}_1(r_1 - \delta) + \hat{f}_2(r_2 + \delta) \leq f_1(r_1 - \xi_\delta) + f_2(r_2 + \xi_\delta) \tag{3.7}$$

where $0 \leq \delta \leq min(r_1 - r_1^{min}, r_2^{max} - r_2)$ then there is no (r_1', r_2') with

$$f_1(r_1') + f_2(r_2') > f_1(r_1 - \xi_\delta) + f_2(r_2 + \xi_\delta) \tag{3.8}$$

where $r_1^{min} \leq r_1' < r_1 - \delta,\ r_2 + \delta < r_2' \leq r_2^{max}$.

Proof. For any $(r_1', r_2'), r_1^{min} \leq r_1' < r_1 - \delta, r_2 + \delta < r_2' \leq r_2^{max}$ we have that $\sum_{i=1}^{2} f_i(r_i') \leq \sum_{i=1}^{2} \hat{f}_i(r_i') \leq \hat{f}_1(r_1 - \delta) + \hat{f}_2(r_2 + \delta) \leq f_1(r_1 - \xi_\delta) + f_2(r_2 + \xi_\delta)$. The first and second inequalities are to be proved, whereas the third one is a precondition of the lemma. The first inequality is due to the construction of the hull functions as the smallest concave functions over the segments such that every $\hat{f}_i(r_i) \geq f_i(r_i), r_i^{min} \leq r_i \leq r_i^{max}$. The assumption that (r_1, r_2) is optimal on the concave hull functions and the concavity of the hull functions give the second inequality. □

Lemma 3.4.3 *If the neighbourhood defined by Lemma 3.4.2 is searched for all r, $(r_1^{min} + r_2^{min}) \leq r \leq (r_1^{max} + r_2^{max})$, all locally optimal allocations within the combination of two segments with concave hull functions is found. The complexity equals $\sum_r \delta_r$, where for each r of the interval, δ_r is the smallest number where Equation (3.7) holds.*

Proof. Follows when Lemma 3.4.2 is applied on every allocation that is optimal on the hull functions $\hat{f}_1$ and $\hat{f}_2$. □

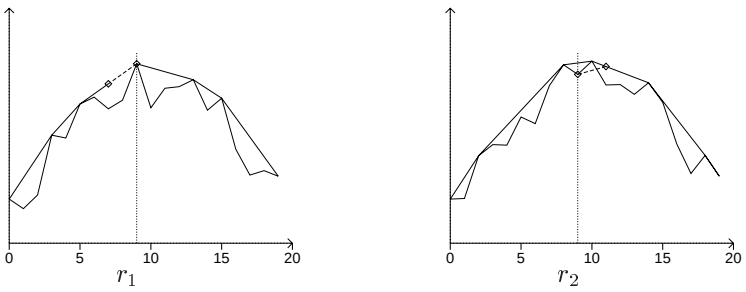

Figure 3.13: *Aggregating two segments with concave hull functions: With* $r_1 = r_2 = 9$*, which is optimal with respect to the hull functions,* $\xi_\delta = 0$*, and* $\delta = 2$*, we have that* $f_1(r_1 - \xi_\delta) - \hat{f}_1(r_1 - \delta) \geq \hat{f}_2(r_2 + \delta) - f_2(r_2 + \xi_\delta)$ *(Equation (3.7) rearranged, the dashed lines with diamonds represent the inequality) and the neighbourhood search of* (r_1, r_2) *is complete (in one direction).*

Aggregating Two Segments With Convex Hull Functions

Aggregating two convex segments is easy [3] since, for each r_t, there are only two possibilities for a candidate:

1. $$r_1 = \begin{cases} r_t - r_2^{min}, & r_t \leq r_1^{max} + r_2^{min} \\ r_1^{max}, & r_t > r_1^{max} + r_2^{min} \end{cases} \quad \text{and } r_2 = r_t - r_1, \tag{3.9}$$

   and

2. vice versa, i.e.

$$r_2 = \begin{cases} r_t - r_1^{min}, & r_t \leq r_1^{min} + r_2^{max} \\ r_2^{max}, & r_t > r_1^{min} + r_2^{max} \end{cases} \quad \text{and } r_1 = r_t - r_2. \tag{3.10}$$

This follows from the fact that if none of the above is fulfilled we have that for two convex segments, either $\Delta f_1(r_1) > \Delta f_2(r_2 - 1)$ or $\Delta f_2(r_2) > \Delta f_1(r_1 - 1)$ or $\big(\Delta f_1(r_1) = \Delta f_2(r_2 - 1)$ and $\Delta f_2(r_2) = \Delta f_1(r_1 - 1)\big)$ holds. In none of these cases the allocation is a candidate, cf. Figure 3.14 (left). If both segments have a hull distance of zero, then the following linear time algorithm finds all candidates [3]:

Algorithm 3.4.3 *For each* r_t*, try the two possibilities above.*

In the following we discuss the second possibility, Equation (3.10), but there is full symmetry between r_1 and r_2. When r_2 is given as much as possible, there is one case when $r_2 = r_2^{max}$ and one when $r_2 < r_2^{max}$. We treat the two cases separately when we define their neighbourhoods.

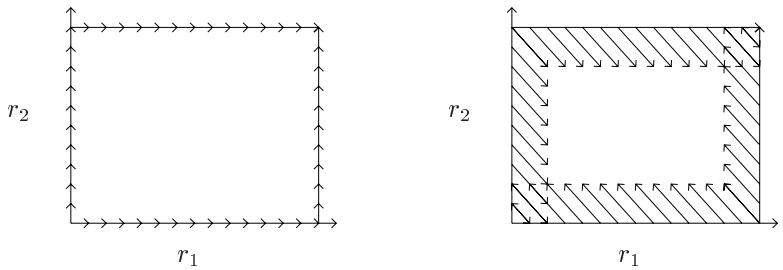

Figure 3.14: *Aggregating two segments with convex hull functions: With $f_1 = \check{f}_1$ and $f_2 = \check{f}_2$ all candidates are found giving as much as possible to one and the rest to the other. With r_1 at the x-axis and r_2 at the y-axis all candidates are found at the border defined by their respective minimum and maximum values (left). With a search based on hull functions (right) the neighbourhood defined by Lemma 3.4.4 and 3.4.5 is searched for the optimal allocation.*

As when working on two segments with concave hull functions, start with finding an optimum on the hull functions and then search the neighbourhood for the optimal allocation, cf. Figure 3.14 (right).

There are occasions when it is enough to start a search at the resource given by Equation (3.10) and no search starting at the resource given by Equation (3.9) is needed (or the other way around). The condition for this is that

$$f_1(r_1) + f_2(r_2) \geq \check{f}_1(r_1') + \epsilon_1 + \check{f}_2(r_2') + \epsilon_2, \tag{3.11}$$

with (r_1, r_2) the combination of Equation (3.10) and (r_1', r_2') the combination of Equation (3.9). If Equation (3.11) does not hold, both are used as starting points.

As will be shown, the following algorithm finds all candidates in two convex segments:

Algorithm 3.4.4 *Algorithm for aggregating two segments with convex hull functions.*
For every r_t
Try the two possibilities given in Algorithm 3.4.3,[8]
Search the neighbourhood for a local optimum

The following lemmas give the pruning conditions for the neighbourhood search (The ξ_δ is again as in Definition 3.4.9). Loosely, with the first equation, the two dashed lines with diamonds in Figure 3.15 are compared. As long as the slope (a, b) of the left pane is greater than the slope (a, b) of the right one it is possible (due to the hull distance) that a local optimum exists with a higher

[8] If none of them can be pruned by use of the condition of Equation (3.11).

r_1 and a lower r_2. On the other hand we have the dashed lines with triangles. If the slope (a, c) of the left pane is greater than the slope (c, d) of the right one the first condition will be true all the way over to the other extreme, and therefore the starting point of the search should be moved over.

Lemma 3.4.4 *Given (r_1, r_2), from Equation (3.10), with $r_1 = r_1^{min}, r_2^{min} \leq r_2 < r_2^{max}$, if*

$$\left(\check{f}_1(r_1^{min} + \delta) + \epsilon_1\right) - f_1(r_1^{min} + \xi_\delta) > f_2(r_2 - \xi_\delta) - \left(\check{f}_2(r_2 - \delta) + \epsilon_2\right) \quad (3.12)$$

and

$$\check{f}_1(r_1^{min} + \delta) - f_1(r_1^{min}) \leq \left(\check{f}_2(r_2) + \epsilon_2\right) - \check{f}_2(r_2 - \delta) \quad (3.13)$$

where $0 \leq \delta \leq min(r_1^{max} - r_1^{min}, r_2 - r_2^{min})$, $0 \leq \xi_\delta \leq \delta$, then $(r_1 + \delta, r_2 - \delta)$ might be a local optimum. The corresponding holds for (r_1, r_2) with $r_1^{min} \leq r_1 < r_1^{max}, r_2 = r_2^{min}$.

Proof. We can write Equation (3.12) as

$$\left(\check{f}_1(r_1^{min} + \delta) + \epsilon_1\right) + \left(\check{f}_2(r_2 - \delta) + \epsilon_2\right) > f_1(r_1^{min} + \xi_\delta) + f_2(r_2 - \xi_\delta),$$

hence, it is also possible that

$$f_1(r_1^{min} + \delta) + f_2(r_2 - \delta) > f_1(r_1^{min} + \xi_\delta) + f_2(r_2 - \xi_\delta).$$

This is true whether Equation (3.13) holds or not, but if not, then due to the convexity of the hull functions the search starting at (r_1^{min}, r_2) should be altered to one starting with giving as much as possible to r_1, cf. Figure 3.15. Symmetry gives that the same holds for (r_1, r_2) with $r_1^{min} \leq r_1 < r_1^{max}, r_2 = r_2^{min}$. □

The next lemma is equivalent to Lemma 3.4.4 except for the starting point of the search, here $r_2 = r_2^{max}$, cf. Equation (3.10).

Lemma 3.4.5 *Given (r_1, r_2), from Equation (3.10), with $r_1^{min} < r_1 < r_1^{max}, r_2 = r_2^{max}$, if*

$$\left(\check{f}_1(r_1 + \delta) + \epsilon_1\right) - f_1(r_1 + \xi_\delta) > f_2(r_2^{max} - \xi_\delta) - \left(\check{f}_2(r_2^{max} - \delta) + \epsilon_2\right) \quad (3.14)$$

and

$$\check{f}_1(r_1 + \delta) - \left(\check{f}_1(r_1) + \epsilon_1\right) \leq f_2(r_2^{max}) - \check{f}_2(r_2^{max} - \delta) \quad (3.15)$$

where $0 \leq \delta \leq min(r_1^{max} - r_1, r_2^{max} - r_2^{min})$, $0 \leq \xi_\delta \leq \delta$, then $(r_1 + \delta, r_2 - \delta)$ might be a local optimum. The corresponding holds for (r_1, r_2) with $r_1 = r_1^{max}, r_2^{min} < r_2 < r_2^{max}$.

Proof. We can write Equation (3.14) as

$$\left(\check{f}_1(r_1 + \delta) + \epsilon_1\right) + \left(\check{f}_2(r_2^{max} - \delta) + \epsilon_2\right) > f_1(r_1 + \xi_\delta) + f_2(r_2 - \xi_\delta),$$

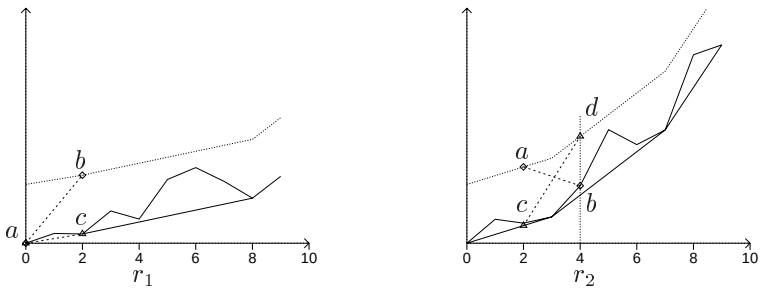

Figure 3.15: *Aggregating two segments with convex hull functions: Equation (3.12) of Lemma 3.4.4 is indicated by the dashed lines with diamonds and Equation (3.13) is indicated with the ones with triangles. We have $f_1(r_1)$ to the left, $f_2(r_2)$ to the right, $r_1 = 0$ and $r_2 = 4$, $\delta = 2$ and $\xi_\delta = 0$, $\check{f}$ and $(\check{f} + \epsilon)$ are indicated as well. Since both equations hold $f_1(r_1^{min} + \delta) + f_2(r_2 - \delta)$ might be a local optimum. The corresponding holds for Lemma 3.4.5.*

hence, it is also possible that

$$f_1(r_1 + \delta) + f_2(r_2^{max} - \delta) > f_1(r_1 + \xi_\delta) + f_2(r_2^{max} - \xi_\delta)$$

This is true whether Equation (3.15) holds or not, but if not, then due to the convexity of the hull functions the search starting at (r_1, r_2^{max}) should be altered to one starting with giving as much as possible to r_1. Symmetry gives that the same holds for (r_1, r_2) with $r_1 = r_1^{max}, r_2^{min} < r_2 < r_2^{max}$. □

Altogether this gives:

Lemma 3.4.6 *If the neighbourhood defined by Lemma 3.4.4 and 3.4.5 is searched Algorithm 3.4.4 finds every locally optimal allocation within the range of $\left[(r_1^{min} + r_2^{min}) \ldots (r_1^{max} + r_2^{max})\right]$ on two segments with convex hull functions. The complexity equals $\sum_r \delta_r$, were for each r of the interval, δ_r is the smallest number such that the conditions of Lemma 3.4.4 and 3.4.5 does not hold.*

Proof. The lemma follows when Lemma 3.4.4 and 3.4.5 are applied to define the neighbourhoods on all allocations within the range. □

Aggregating One Segment With a Concave Hull Function and One Segment With a Convex Hull Function

With one convex and one concave segment it is possible to find all candidates on a path through the segment combination of length linear in the size of the

resources [3]. We are using the method to search the hull functions. Therefor we repeat some of the results from our previous article.

The algorithm for aggregation of a concave and a convex segment is based on the following three lemmas:

Lemma 3.4.7 *If* $r_1^{min} \leq r_1 < r_1^{max}$, $r_2^{min} < r_2 \leq r_2^{max}$ *and*

$$\Delta f_1(r_1) \geq \Delta f_2(r_2 - 1) \tag{3.16}$$

there is no candidate (r_1', r_2'), $r_1 \leq r_1' < r_1^{max}$, $r_2 \leq r_2' \leq r_2^{max}$, *[3].*

Lemma 3.4.8 *If* $(r_1, r_2), r_1^{min} < r_1 < r_1^{max}, r_2^{min} < r_2 \leq r_2^{max}$ *is a candidate then* (r_1', r_2'), $r_1^{min} < r_1' \leq r_1$, $r_2^{min} < r_2' < r_2$, *is not, [3].*

Lemma 3.4.9 *If* $r_1^{min} < r_1 \leq r_1^{max}$, $r_2^{min} \leq r_2 < r_2^{max}$ *and*

$$\Delta f_1(r_1 - 1) < \Delta f_2(r_2) \tag{3.17}$$

there is no candidate (r_1', r_2'), $r_1^{min} < r_1' \leq r_1$, $r_2^{min} \leq r_2' \leq r_2$, *[3].*

One concave segment and one convex segment can be aggregated by the following algorithm in $\mathcal{O}(I_1 + I_2)$ time.

Algorithm 3.4.5 *Algorithm for aggregating one convex and one concave segment*

1. (r_1^{min}, r_2^{min}) *is a candidate. Set* $r_1 \leftarrow r_1^{min}$, $r_2 \leftarrow r_2^{min}$.
2. *Repeat* $r_2 \leftarrow r_2 + 1$ *until* $\Delta f_1(r_1^{min}) \geq \Delta f_2(r_2 - 1)$ *or* $r_2 = r_2^{max} + 1$.
3. *We can now conclude, either because* $r_2 = r_2^{max} + 1$ *or by Lemma 3.4.7, that there is no candidate* (r_1', r_2'), $r_1^{min} \leq r_1' < r_1^{max}$, $r_2 \leq r_2' \leq r_2^{max}$. *In other words: each remaining candidate* (r_1', r_2') *satisfies either*

$$r_1^{min} < r_1' < r_1^{max}, \quad r_2^{min} \leq r_2' < r_2 \tag{3.18}$$

   or

$$r_1 = r_1^{max}. \tag{3.19}$$

4. *In this phase, consider the remaining candidates fulfilling Equation (3.18). Set* $r_1 = r_1^{min} + 1$ *and* $r_2 = r_2 - 1$. *We will now maintain the following invariant: Each remaining (i.e. non-visited) candidate* (r_1', r_2') *must satisfy*

$$r_1 \leq r_1' < r_1^{max}, \quad r_2^{min} \leq r_2' \leq r_2$$

   Initially, the invariant holds.

   While $r_1 < r_1^{max}$ *and* $r_2 \geq r_2^{min}$, *perform one of the four cases:*

   (i) $r_2 = r_2^{min}$, *then* r_1 *can be increased by one; the invariant will still hold,*

(ii) $\Delta f_1(r_1) \geq \Delta f_2(r_2 - 1)$. *Then, by Lemma 3.4.7,* r_2 *can be decreased by one; the invariant will still hold.*

(iii) $r_2 < r_2^{max}$, $\Delta f_1(r_1 - 1) < \Delta f_2(r_2)$. *Then, according to Lemma 3.4.9* r_1 *can be increased by one; the invariant will still hold.*

(iv) If none of the cases above occur, (r_1, r_2) *is a candidate (see below). Then, Lemma 3.4.8 gives that* r_1 *can increased by one; the invariant will still hold.*

5. *When leaving the loop above, the remaining points to investigate are all fulfilling Equation (3.19). Start at* (r_1^{max}, r_2^{max}) *which is a candidate and decrease* r_2 *(implying that* $\Delta f_2(r_2)$ *increases) until* $\Delta f_1(r_1^{max} - 1) < \Delta f_2(r_2)$ *or* $r_2 = r_2^{min}$. *Now, according to Lemma 3.4.9 no further candidates exist.*

We now turn to what is added when the search is based on hull functions, phase by phase.

As in the cases of two concave or two convex segments start by finding the candidates on the hull functions. For each choice of a resource combination (r_1, r_2) perform a local search of the neighbourhood to find all possible local optima, see Figure 3.16. The lemmas below define the neighbourhoods that have to be searched (and what search areas could pruned).

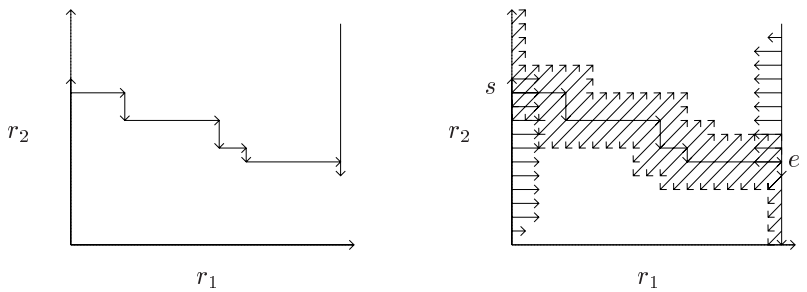

Figure 3.16: *Aggregating one convex and one concave segment: For some particular segment, all candidates are on the path of the left figure. If these are hull functions, then a neighbourhood search for local optima is performed as illustrated in the right figure.*

Phase One

The first step is to evaluate (r_1^{min}, r_2^{min}). For both r_1 and r_2 the true function value is the same as the value of the hull function (due to the way hull functions are constructed), therefor no neighbourhood search is needed.

Phase Two

In Phase 2, r_2 is incremented until Equation (3.16) holds on the hull functions $\check{f}_1$ and $\hat{f}_2$, or $r_2 = r_2^{max}$. At each step the neighbourhood has to be investigated and this is done holding r_2 constant. In Figure 3.16 (right) the horizontal arrows pointing right at the left border show how a neighbourhood search is performed in this phase. The conditions of Lemma 3.4.10 (that defines the neighbourhood) are illustrated in Figure 3.17. Loosely, as long as the slope (a, b) of the left pane is greater than the slope (a, b) of the right one it is possible that a local optimum is found raising r_2. On the other hand, if the slope (c, d) of the left pane is greater than the slope (c, d) of the right one any local optimum further to the right will be found in the other phases, and could be left out.

Using the lemma any local optimum at the left border of Figure 3.16 up to the point where Equation (3.16) holds (s in the right pane on Figure 3.16) is found. All other local optima are left to be found in the search of the following phases.

Lemma 3.4.10 *Given (r_1, r_2) chosen in Phase 2 of Algorithm 3.4.5 (applied on the hull functions), with $r_1 = r_1^{min}, r_2^{min} < r_2 < r_2^{max}$, if*

$$\big(\check{f}_1(r_1^{min} + \delta) + \epsilon_1\big) - f_1(r_1^{min}) > \big(\hat{f}_2(r_2 + \delta) - \epsilon_2\big) - f_2(r_2) \tag{3.20}$$

and

$$\check{f}_1(r_1^{min} + \delta) - \check{f}_1(r_1^{min} + \delta - 1) < \hat{f}_2(r_2 + \delta) - \hat{f}_2(r_2 + \delta - 1) \tag{3.21}$$

for some δ, $0 < \delta \leq min(r_1^{max} - r_1^{min}, r_2^{max} - r_2)$ then $(r_1^{min} + \delta, r_2)$ might be a local optimum.

Proof. We can write Equation (3.20) as

$$\big(\check{f}_1(r_1^{min} + \delta) + \epsilon_1\big) + f_2(r_2) > f_1(r_1^{min}) + \big(\hat{f}_2(r_2 + \delta) - \epsilon_2\big),$$

hence, it is possible that

$$f_1(r_1^{min} + \delta) + f_2(r_2) > f_1(r_1^{min} + f_2(r_2 + \delta).$$

This is true whether Equation (3.21) holds or not, but if not then Equation (3.16) of Lemma 3.4.7 holds on $(r_1^{min} + \delta, r_2)$ and any allocation $(r_1^{min} + \delta', r_2)$, $\delta < \delta' \leq r_1^{max} - r_1^{min}$ is not part of this border. It might still be a local optimum, but then it is left to be found during the neighbourhood search of Phase 4 or Phase 5. □

Phase Three

No new candidates are added in phase three.

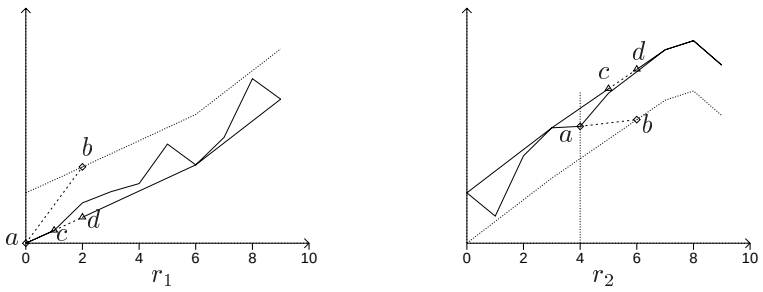

Figure 3.17: *Aggregating one segment with a convex hull function and one with a concave hull function, Phase 2: Equation (3.20) is indicated by the dashed line with diamonds and Equation (3.21) with the ones with triangles. We have* $f_1(r_1)$ *to the left and* $f_2(r_2)$ *to the right,* $r_1 = r_1^{min} = 0$, $r_2 = 4$ *and* $\delta = 2$, $\check{f}_1$, $(\check{f}_1 + \epsilon)$, $\hat{f}_2$ *and* $(\hat{f}_2 - \epsilon)$ *are indicated as well. Since both equations hold,* $(r_1^{min} + \delta, r_2)$ *is part of the border to be investigated in phase two.*

Phase Four

In Phase 4 the aim is to move from r_1^{min} to r_1^{max} and at each step the neighbourhood is searched. The main path of this phase starts were Equation (3.16) holds (s in Figure 3.16 (right)) and ends were r_1^{max} is reached (e in Figure 3.16 (right)). In addition we have to search $(r_1^{min}, r_2 > s)$ and $(r_1^{max}, r_2 < e)$, cf. Figure 3.16 (right). The same lemmas can be used for managing both these borders and the main path.

The following lemmas define the neighbourhood searched in Phase 4. First the lemma giving conditions to end the simultaneous incrementing of r_1 and r_2, cf. Figure 3.18, is introduced, then the corresponding for the simultaneous decrementing, cf. Figure 3.19, is introduced.

Loosely, the first of the lemmas give the possibility to prune a more or less triangular area above the diagonal in Figure 3.16 and the second one prunes a similar area below the diagonal. The path is constructed with Phase 4 of Algorithm 3.4.5 applied to the hull functions. At each step a neighbourhood search is added. In the neighbourhood search both r_1 and r_2 are simultaneously incremented until no more local optima exists $\big($on the search path starting at $(r_1, r_2)\big)$ that are not part of the border searched in Phase 5. In a corresponding way, r_1 and r_2 are simultaneously decremented until no more local optima exist $\big($on the search path starting at $(r_1, r_2)\big)$ that are not part of the border already searched in Phase 2.

Lemma 3.4.11 *Given* (r_1, r_2)*, which is a candidate in Phase 4 of Algorithm 3.4.5 (applied to the hull functions), with* $r_1^{min} \leq r_1 < r_1^{max}, r_2^{min} < r_2 \leq r_2^{max}$*, if*

$$\check{f}_1(r_1 + \delta) - \big(\check{f}_1(r_1) + \epsilon_1\big) \geq \hat{f}_2(r_2 + \delta) - \big(\hat{f}_2(r_2) - \epsilon_2\big), \qquad (3.22)$$

$0 < \delta \leq min(r_1^{max} - r_1, r_2^{max} - r_2)$, *then no* $(r_1', r_2'), r_1 + \delta \leq r_1' < r_1^{max}, r_2 + \delta < r_2' \leq r_2^{max}$ *could be a local optimum if not at the border searched in phase five.*

Proof. By convexity $\Delta\check{f}_1(r_1') \geq \Delta\check{f}_1(r_1 + \delta)$ and by concavity $\Delta\hat{f}_2(r_2') \leq \Delta\hat{f}_2(r_2 + \delta)$. Hence $\check{f}_1(r_1') - \left(\check{f}_1(r_1) + \epsilon_1\right) \geq \hat{f}_2(r_2') - \left(\hat{f}_2(r_2) - \epsilon_2\right)$ and the lemma follows. □

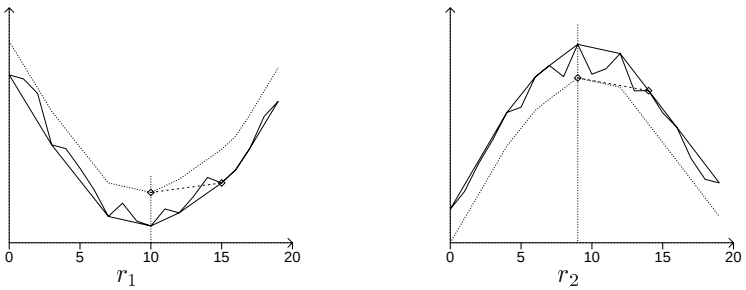

Figure 3.18: *Aggregating one segment with a convex hull function and one with a concave hull function, Phase 4: Equation (3.22) is indicated by the dashed line with diamonds. We have* $f_1(r_1)$ *to the left,* $f_2(r_2)$ *to the right,* $r_1 = 10$, $r_2 = 9$ *and* $\delta = 5$, $\check{f}_1$, $(\check{f}_1 + \epsilon)$, $\hat{f}_2$ *and* $(\hat{f}_2 - \epsilon)$ *are indicated as well. Since the equation holds, no* $(r_1', r_2'), r_1 + \delta \leq r_1' < r_1^{max}, r_2 + \delta < r_2' \leq r_2^{max}$ *can be a local optimum if not being part of the border searched in phase five.*

Lemma 3.4.12 *Given* (r_1, r_2)*, which is a candidate in Phase 4 of Algorithm 3.4.5 (applied to the hull functions), with* $r_1^{min} \leq r_1 < r_1^{max}, r_2^{min} < r_2 \leq r_2^{max}$*, if*

$$\left(\check{f}_1(r_1) + \epsilon_1\right) - \check{f}_1(r_1 - \delta) \leq \left(\hat{f}_2(r_2) - \epsilon_2\right) - \hat{f}_2(r_2 - \delta), \tag{3.23}$$

$0 < \delta \leq min(r_1 - r_1^{min}, r_2 - r_2^{min})$*, then no* $(r_1', r_2'), r_1^{min} < r_1' \leq r_1 - \delta, r_2^{min} < r_2' \leq r_2 - \delta$ *could be a local optimum if not at the border searched in phase two.*

Proof. By convexity $\Delta\check{f}_1(r_1') \leq \Delta\check{f}_1(r_1 + \delta)$ and by concavity $\Delta\hat{f}_2(r_2') \geq \Delta\hat{f}_2(r_2 + \delta)$. Hence $\check{f}_1(r_1') - \left(\check{f}_1(r_1) + \epsilon_1\right) \leq \hat{f}_2(r_2') - \left(\hat{f}_2(r_2) - \epsilon_2\right)$ and the lemma follows. □

With this the search space of the diagonal of Figure 3.16 is defined and all that remains is the border at r_1^{max}.

Phase Five

In Phase 5, r_2 is decremented from (r_1^{max}, r_2^{max}) until Equation (3.17) of Lemma 3.4.9 holds on the hull functions, or $r_2 = r_2^{min}$. At each step, the neighbourhood has

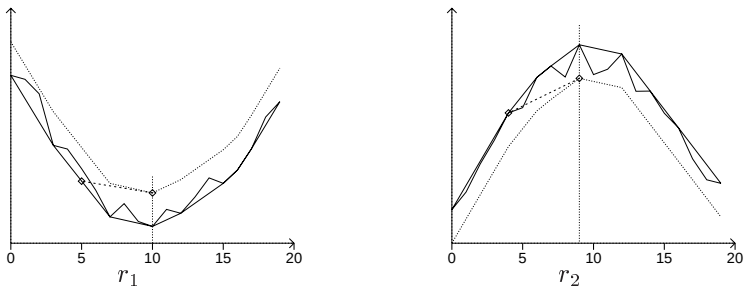

Figure 3.19: *Aggregating one segment with a convex hull function and one with a concave hull function, Phase 4: Equation (3.23) is indicated by the dashed line with diamonds. We have $f_1(r_1)$ to the left, $f_2(r_2)$ to the right, $r_1 = 10$, $r_2 = 9$ and $\delta = 5$, $\check{f}_1$, $(\check{f}_1 + \epsilon)$, $\hat{f}_2$ and $(\hat{f}_2 - \epsilon)$ are indicated as well. Since the equation holds, no $(r_1', r_2'), r_1^{min} < r_1' \leq r_1 - \delta, r_2^{min} < r_2' \leq r_2 - \delta$ can be a local optimum if not being part of the border searched in phase two.*

to be investigated and — as in Phase 2 — this is done holding r_2 constant. We can see in Figure 3.16 (right) how a neighbourhood search is performed in this phase (the horizontal arrows pointing left at the right border). The conditions of Lemma 3.4.13 (that defines the neighbourhood) are illustrated in Figure 3.20. Loosely, as long as the slope (a, b) of the left pane is less than the slope (a, b) of the right one it is possible that a local optimum is found lowering r_2. On the other hand, if the slope (c, d) of the left pane is less than the slope (c, d) of the right one, any local optimum further to the left (in Figure 3.16 (right)) has been found in the other phases, and does not need to be considered.

The following lemma defines the neighbourhood that has to be searched during this phase.

Lemma 3.4.13 *Given (r_1, r_2) chosen in Phase 5 of Algorithm 3.4.5 (applied on the hull functions) with $r_1 = r_1^{max}$, $r_2^{min} \leq r_2 < r_2^{max}$, if*

$$f_1(r_1^{max}) - \left(\check{f}_1(r_1^{max} - \delta) + \epsilon_1\right) < f_2(r_2) - \left(\hat{f}_2(r_2 - \delta) - \epsilon_2\right) \tag{3.24}$$

and

$$\check{f}_1(r_1^{max} - \delta + 1) - \check{f}_1(r_1^{max} - \delta) > \hat{f}_2(r_2 - \delta + 1) - \hat{f}_2(r_2 - \delta), \tag{3.25}$$

$0 < \delta \leq min(r_1^{max} - r_1^{min}, r_2 - r_2^{min})$ then $(r_1^{max}, r_2 - \delta)$ might be a local optimum.

Proof. We can write Equation (3.24) as

$$f_1(r_1^{max}) + \left(\hat{f}_2(r_2 - \delta) - \epsilon_2\right) < f_2(r_2) + \left(\check{f}_1(r_1^{max} - \delta) + \epsilon_1\right),$$

hence, it is possible that

$$f_1(r_1^{max}) + f_2(r_2 - \delta) < f_1(r_1^{max} - \delta) + f_2(r_2).$$

This is true whether Equation (3.25) holds or not. But if it does not hold then Equation (3.17) of Lemma 3.4.9 holds on $(r_1^{max} - \delta, r_2)$ and any allocation $(r_1^{max} - \delta', r_2)$, $\delta < \delta' \leq r_1^{max} - r_1^{min}$ is not part of this border. It might still be a local optimum, but then it has already been found during the neighbourhood search of phase 2 or phase 4. Cf. Figure 3.20. □

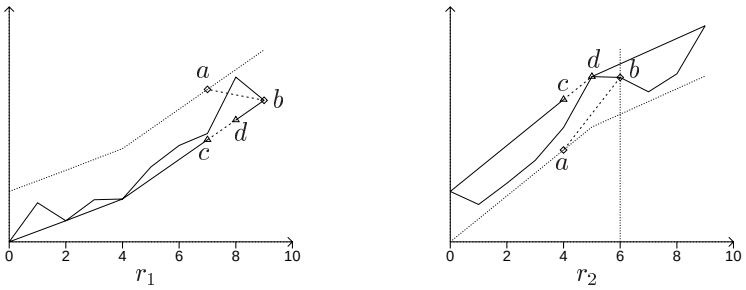

Figure 3.20: *Aggregating one segment with a convex hull function and one with a concave hull function, Phase 5: Equation (3.24) is indicated by the dashed lines with diamonds and Equation (3.25) is indicated with the one with triangles. We have $f_1(r_1)$ to the left and $f_2(r_2)$ to the right, $r_1 = r_1^{max} = 9$, $r_2 = 6$ and $\delta = 2$, $\check{f}_1, (\check{f} + \epsilon), \hat{f}_2$ and $(\hat{f}_2 - \epsilon)$ are indicated as well. Since both equations hold $f_1(r_1^{min} + \delta) + f_2(r_2 - \delta)$ is part of the border to be searched in Phase 5.*

Conclusion on the Aggregation of One Segment With a Concave Hull Function and One Segment With a Convex Hull Function

The aggregation of one concave and one convex segment is non-trivial even when there is no noise to handle, but it is possible to do in time linear in the size of the segments. In this section we have shown how it is possible to prune a significant part of the search space even when we have low amplitude noise on top of segments that essentially are concave and convex respectively. By this it is possible to avoid a complete search of all combinations within the segments.

Lemma 3.4.14 *All locally optimal allocations on one segment with a convex hull function and one segment with a concave hull function are found if*

(i) Algorithm 3.4.5 is applied on $\check{f}_1$ and $\hat{f}_2$, and

(ii) a neighbourhood search is performed at each $r, (r_1^{min} + r_2^{min}) \leq r \leq (r_1^{max} + r_2^{max})$, according to Lemma 3.4.10 ... 3.4.13.

The complexity equals $\sum_r \delta$. *In the case of Lemma 3.4.10 and 3.4.13,* δ *is the smallest number such that the conditions do not hold. In Lemma 3.4.11 and 3.4.12,* δ *is the smallest number such that Equation (3.22) and Equation (3.23) hold.*

Proof. Follows from Lemma 3.4.10 ... 3.4.13. □

3.4.5 Proof of Theorem 3.4.1

By now we have what we need to prove Theorem 3.4.1.

The algorithm searches the neighbourhood of every local candidate of the hull functions. This is done for every possible combination of segments.

There are three possible segment combinations (see the introduction to Section 3.4.4). In all the combinations all local optima are found, Lemma 3.4.3, 3.4.5, and 3.4.14.

Since a globally optimal allocation has to be a local optimum for some segment combination the theorem follows.

3.5 Conclusions

In this article, we have presented a robust algorithm for resource allocation with separable objective functions. It is particularly interesting to note that we can handle seemingly impossible instances, such as random functions, more efficient than the obvious brute force algorithm. Simply said, whenever there is some regularity in the input, either in the original objective functions or in the intermediate aggregated functions, we take advantage of this.

Although our algorithm is more involved than the brute force algorithm, the implementation overhead is affordable, as shown by our experiments. Therefore, our new algorithm is a competitive candidate for practical applications.

Bibliography

[1] A. Andersson, P. Carlsson, and F. Ygge. Resource allocation with noisy functions. Technical Report 2000-017, Department of Information Technology, Uppsala University, July 2000. (Available from www.it.uu.se/research/reports/).

[2] A. Andersson and F. Ygge. Managing large scale computational markets. In H. El-Rewini, editor, *Proceedings of the Software Technology track of the 31th Hawaiian International Conference on System Sciences (HICSS31)*, volume VII, pages 4–14. IEEE Computer Society, Los Alamos, January 1998. ISBN 0-8186-8251-5, ISSN 1060-3425, IEEE Catalog Number 98TB100216. (Available from http://www.enersearch.se/ygge).

[3] Arne Andersson and Fredrik Ygge. Efficient resource allocation with non-concave objective functions. *Accepted for publication in Computational Optimization and Applications*, 1998. (A preprint version is available as a research report from http://www.enersearch.se).

[4] J. Cheng and M. P. Wellman. The WALRAS algorithm—A convergent distributed implementation of general equilibrium outcomes. *Computational Economics*, 12:1–24, 1998. (Available from ai.eecs.umich.edu/people/wellman).

[5] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts London, England, 1989.

[6] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 1987. Second Edition.

[7] R. Horst, P. Pardalos, and N. Thoai. *Introduction to Global Optimization*. Kluwer, Norwell, MA, 1995.

[8] Tochihide Ibaraki and Naoki Katoh. *Resource Allocation Problems — Algorithmic Approaches*. The MIT Press, Cambridge, Massachusetts, 1988.

[9] J. K. MacKie-Mason and H. R. Varian. Generalized vickrey auctions. Technical report, Dept. of Economics, June 1994.

[10] A. Mas-Colell, M. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[11] Å. Leander (project leader). Energy in Sweden. Technical report, The Swedish National Energy Administration, 2000. Available at http://www.stem.se.

[12] T. W. Sandholm and F. Ygge. Constructing speculative demand functions in equilibrium markets. Technical Report WUCS-99-26, Department of Computer Science, Washington University, October 1999. Submitted for journal publication. (Available from www.enersearch.se/ygge).

[13] H. R. Varian. *Intermediate Microeconomics—A Modern Approach*. W. W. Norton and Company, New York, 1999. Fifth Edition.

[14] M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research (www.jair.org/)*, (1):1–23, 1993.

[15] F. Ygge. *Market-Oriented Programming and its Application to Power Load Management*. PhD thesis, Department of Computer Science, Lund University, 1998. ISBN 91-628-3055-4, CODEN LUNFD6/(NFCS-1012)/1-224/(1998) (Available from http://www.enersearch.se/ygge).

[16] F. Ygge. Energy resellers - an endangered species? In *Agent Mediated Electronic Commerce II, Lecture Notes on Artificial Intelligence*, number 1788. Springer Verlag, 2000. (A preprint of the paper is available as a Technical Report from www.enersearch.se/ygge).

Licentiate theses from the Department of Information Technology

2000-001 Katarina Boman: *Low-Angle Estimation: Models, Methods and Bounds*

2000-002 Susanne Remle: *Modeling and Parameter Estimation of the Diffusion Equation*

2000-003 Fredrik Larsson: *Efficient Implementation of Model-Checkers for Networks of Timed Automata*

2000-004 Anders Wall: *A Formal Approach to Analysis of Software Architectures for Real-Time Systems*

2000-005 Fredrik Edelvik: *Finite Volume Solvers for the Maxwell Equations in Time Domain*

2000-006 Gustaf Naeser: *A Flexible Framework for Detection of Feature Interactions in Telecommunication Systems*

2000-007 Magnus Larsson: *Applying Configuration Management Techniques to Component-Based Systems*

2000-008 Marcus Nilsson: *Regular Model Checking*

2000-009 Jan Nyström: *A formalisation of the ITU-T Intelligent Network standard*

2000-010 Markus Lindgren: *Measurement and Simulation Based Techniques for Real-Time Analysis*

2000-011 Bharath Bhikkaji: *Model Reduction for Diffusion Systems*

2001-001 Erik Borälv: *Design and Usability in Telemedicine*

2001-002 Johan Steensland: *Domain-based partitioning for parallel SAMR applications*

2001-003 Erik K. Larsson: *On Identification of Continuous-Time Systems and Irregular Sampling*

2001-004 Bengt Eliasson: *Numerical Simulation of Kinetic Effects in Ionospheric Plasma*

2001-005 Per Carlsson: *Market and Resource Allocation Algorithms with Application to Energy Control*

UPPSALA
UNIVERSITY

KFS i Lund AB, Sweden
Printing sponsored by EnerSearch AB