Karlsruhe Institute of Technology

Solving a Large-Scale Energy Management Problem with Varied Constraints

Diploma Thesis of

Felix Brandt

At the Department of Informatics
Institute of Theoretical Informatics

Reviewer:	Prof. Dr. rer. nat. Dorothea Wagner
Advisors:	Dipl.-Math. Reinhard Bauer
	Dr.-Ing. Andreas Cardeneo
	Dipl.-Inform. Markus Völker

Time Period: 9th February 2010 – 6th August 2010

Zusammenfassung (German Summary)

Diese Arbeit beschäftigt sich mit dem Problem der Wartungs- und Produktionsplanung eines großen konventionellen Kraftwerksparks unter Berücksichtigung vielfältiger Nebenbedingungen und Unsicherheit über einen Zeitraum von bis zu 5 Jahren. Die Aufgabenstellung entstammt dem ROADEF/EURO Challenge 2010, einem Wettbewerb der gemeinsam von der französischen Organisation für Operations Research und Decision Support (ROADEF) und der Europäischen Organisation für Operations Research (EURO) ausgeschrieben wurde. Das im Wettbewerb verwendete Modell und die zur Verfügung gestellten Datensätze stammen vom größten Französischen Energieversorger Electricité de France (EDF). Da diese Arbeit erst nach der Vorrunde des Wettbewerbs begonnen wurde, konnten wir nicht mehr daran teilnehmen.

Das Modell unterscheidet zwei Typen von Kraftwerken, eine allgemein gehaltene Klasse etwa für Kohle- und Gaskraftwerke oder Energieimporte und -exporte sowie eine Klasse detaillierter modellierter Kernkraftwerke. Die Aufgabenstellung enthält drei voneinander abhängige Teilprobleme. Erstens die Erstellung eines zulässigen Wartungsplans für die Kernkraftwerke, das heißt die Festlegung von Zeiträumen, in denen die Kraftwerke vom Netz getrennt sind und gewartet werden. Die Wartungspläne müssen eine Reihe von bereits mathematisch formulierten Bedingungen einhalten um unter anderem logistisch und personell durchführbar zu sein, gesetzlichen Vorgaben zu genügen und die Netzstabilität nicht zu beeinträchtigen. Das zweite Teilproblem ist die Erzeugung von Produktionsplänen für alle Kraftwerke unter verschiedenen Angebots- und Nachfrageszenarien. Aus den Brennstoffvorräten der Kernkraftwerke ergeben sich weitere Einschränkungen in den Produktionsplänen. Die Festlegung der nachzufüllenden Brennstoffmenge bildet das dritte Teilproblem, da das Auffüllen nur während einer Wartung durchgeführt werden kann. Das übergeordnete Ziel ist die Minimierung der gesamten Produktionskosten.

Zu Beginn entwickeln wir zwei untere Schranken, welche wir in unserer Lösung gewinnbringend einsetzen. Wir lösen das Problem in zwei Schritten. In einem ersten Schritt erzeugen wir einen gültigen Wartungsplan mit Methoden der Constraint Programmierung unter Verwendung des Open-Source Frameworks Gecode. Dabei erweitern wir das Modell um zusätzliche Bedingungen, welche die Lösbarkeit der verbleibenden Teilprobleme sicherstellen. Anschließend verwenden wir eine Weiterentwicklung einer der erwähnten Schranken, um mittels einer einfachen Greedy-Heuristik die Produktions- und Bevorratungspläne zu erstellen.

Im weiteren Verlauf der Arbeit entwickeln wir Techniken um unsere Ergebnisse zu verbessern. Ein erfolgreicher, randomisierter Ansatz betrachtet die Entwicklung der durch eine Wartung verursachten Mehrkosten und bevorzugt günstige Wartungszeitpunkte. Weiterhin führen wir eine neue Bedingung ein, welche den Suchraum des Problems auf günstige Wartungspläne beschränkt. Außerdem entwickeln wir eine Bevorratungsstrategie, welche den erwarteten Profit in Relation zum Brennstoffvorrat setzt.

Abschließend vergleichen wir unsere Lösungen der zur Verfügung gestellten Datensätze mit den besten bekannten Lösungen der Wettbewerbsteilnehmer. Unser Ansatz erreicht

für alle Datensätze gute Ergebnisse und entspräche einem 2. Platz unter den 21 Finalisten des Wettbewerbs.

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig angefertigt habe und nur die angegebenen Hilfsmittel und Quellen verwendet wurden.

Karlsruhe, den 06. August 2010

Unterschrift: ....

Acknowledgements

First of all, I want to thank Prof. Wagner for the opportunity to write my thesis at her institute about my self-chosen and not so common topic. Special thanks go to my advisors (in alphabetical order) Reinhard Bauer, for his constant support over the last months, his excellent guidance and his mathematical point of view, Andreas Cardeneo, for his cooperation and especially for pointing me to the topic, and Markus Völker, for his insights to Gecode and lots of proofreading.

Finally, I would like to thank all of the other proofreaders of this thesis.

Contents

1. Introduction

Electric energy is an essential resource in today's life. No matter if we make our first cup of coffee or tea in the morning or run a multi-million euro business, all the time we rely on a secure and inexpensive supply of electricity. In this work we take the perspective of a large utility company, tackling their problems in modeling and planning production assets, i.e., a multitude of power plants. Our goal is to fulfill the respective demand of energy over a time horizon of several years, with respect to the total operating cost of all machinery.

Determining optimal maintenance schedules and production plans is not easy because of the number of alternatives to assess. As the exact electricity demand of each forthcoming day is unknown and depends on a large variety of factors (season, weather, holidays, etc.), this leads to the need of multiple uncertainty scenarios. Additionally, the increasing proportion of renewable energies in today's energy mix makes things more complicated for an utility company, because it has to feed the energy of third-party solar or wind power plants into its electricity networks and regulate its own power plants accordingly.

The problem discussed in this thesis was proposed by the ROADEF/EURO Challenge 2010 [PGJ$^+$], a competition announced by the French Operational Research and Decision Support Society (ROADEF) and the European Operational Research Society (EURO). The model has been developed by the French utility company Electricité de France (EDF). Since we started this work shortly after the challenge's submission deadline for the qualification round passed, we could not take part in the competition. Nevertheless, we will compare our results with the best results obtained by the participants of the challenge.

The examined problem comprises three fields of optimization: maintenance scheduling, production planning and determining refueling amounts. It is a tactical model, neither considering short-term operational restrictions (like intraday load following) nor containing strategic decisions (like adding new power plants). However, the proposed model allows a generic formulation of other concerns like electricity network stability, safety considerations, availability of staff and tools, as well as legal restrictions. All of these limitations can be expressed as mathematical constraints.

Our solution combines a constraint programming formulation of the problem with several heuristics. We decompose the problem and develop different solution strategies ranging from very simple approaches up to sophisticated ones, which are driven by lower bounds we develop. Additionally, we present optimization techniques to improve our obtained results.

Along with the presented solution methods we show our experimental results. All considered problem instances are provided by the competition and are extracted from real world data. Our solution performs competitive and provides good results for all proposed instances.

One will notice the small differences between trivial and the best known solutions. But, as fuel cost is a major cost component in the power industry, it is worth optimizing along this edge. In the end, improving a solution by 0.1% is equivalent to saving several millions of euros per year in operating cost.

1.1 Related Work

To the best of our knowledge, no work has been published yet that addresses the given model. A former model proposed by EDF, comprising a subset of constraints and only considering maintenance scheduling, was solved by a combination of constraint programming and local search [KPB06]. While both models only accept solutions that fulfill all of the problem's constraints, we note that most other models from the literature employ penalties if constraints are violated, rather than looking for exact solutions.

Nevertheless, there exists a vast amount of literature concerning production planning of power generating facilities. Different models have been proposed that track either maintenance scheduling [FMS08],[SN91] or production planning [NKT00],[DR97]. A more general problem addressed frequently in the literature is the unit commitment problem [Pad04],[FKL96],[SK98]. When applied to power systems the objective usually is to find low-cost short-term (typically between 24 and 168 hours) production plans.

For a general introduction to solving combinatorial problems we refer to comprehensive literature about combinatorial optimizations [NW88], networks and network flows [AMO93], and algorithms and data structures [CLRS01]. Methods which have successfully been applied to power generation scheduling include genetic algorithms, ant colony search, tabu search and simulated annealing [LES07]. A detailed introduction to different types and features of power plants can be found in [WW96].

1.2 Overview

In the following we briefly outline the structure of this thesis:

Chapter 2 provides background knowledge of the main concepts of our solution and settles some basic notation. It starts with a short introduction to constraint programming and describes Gecode, the constraint programming toolkit that we use to solve parts of the problem. The chapter concludes with some basic definitions from the field of graph theory.

Chapter 3 gives an extensive introduction to the topic posed by the ROADEF/EURO Challenge 2010. After a first high-level presentation of the problem, we settle some basic terminology and define the model in detail, i.e., its variables, constraint types and the objective function.

Chapter 4 presents our first theoretical approach to the problem. We analyze the proposed model and derive three lower bounds. The first bound is a lower bound on the unit production cost of single power plants, while the other two limit the value of the main objective function. One of the latter two is a fast greedy heuristic, while the other is modeled as a minimum cost flow problem and yields closer bounds.

Chapter 5 introduces our basic solution to the problem. First, we explain how to find feasible outage schedules by transferring the proposed model into a constraint program. Afterwards, we describe the computation of a valid production plan for a given outage schedule. At the end of this chapter, we present first experimental results of our approach.

Chapter 6 discusses several optimization techniques that we apply to our solution and presents the achieved results. In the beginning of this chapter, we present two advanced search strategies that find better outage schedules. Afterwards, we add a new constraint to the problem, which limits the search space to promising outage schedules. Finally, we present an improved refueling strategy.

Chapter 7 describes our experimental setup. It introduces the instances we use to evaluate our solution and compares our results with the best results of the official contest participants. Finally, it presents an overview of how our best solutions evolved with the proposed optimization steps.

Chapter 8 sums up the most important results of this thesis and concludes our work.

2. Fundamentals

In this chapter we introduce the main concepts and tools that will be used in the remainder of this thesis. First, Section 2.1 is dedicated to constraint programming as a programming paradigm and especially introduces the constraint programming toolkit Gecode. Afterwards, Section 2.2 settles some basic notation about graphs and networks.

2.1 Constraint Programming

Constraint programming (CP) is a declarative programming paradigm that evolves since the 1980s and is a powerful method for solving combinatorial problems. While first ideas came from the artificial intelligence field, CP received broader attention from the area of operations research during the 1990s resulting in the first commercial applications for resource planning and other decision support systems. Furthermore, CP has proven to be useful in many other fields of combinatorial optimization such as electrical engineering, molecular biology, and natural language processing.

As a declarative paradigm, CP formulations define the properties of a solution to be found rather than specifying the exact sequence of the steps to execute. These properties can be stated as simple logical conditions (e.g., "$A \Rightarrow B$"), relational constraints (e.g., "$x \leq y$") or much more complex domain specific constraints (e.g., limitations of drivers' working hours). Afterwards, solving a concrete problem is delegated to some specialized CP solver. A solver therefore offers a set of predefined models and constraint types, which can be extended almost arbitrarily to fit any application-specific needs. Furthermore, the model might be modified during the execution of the solver.

Preliminaries

First, we settle some basic notation, mostly following the definitions from [RBW06].

CP models define constraints between variables. The domain of a variable x, denoted $D(x)$, is a set of elements that can be assigned to x. Let $X = \{x_1, \dots, x_n\}$ denote a set of variables. A constraint C can be seen as a subset of the Cartesian product of the variables in X, i.e., $C \subseteq D(x_1) \times \ldots \times D(x_n)$. A tuple $(d_1, \dots, d_n) \in C$ is called a *solution* to C. A solution *assigns* the value d_i to x_i, for all $1 \leq i \leq n$. We also say that this assignment *satisfies* C. If $C = \emptyset$, there exists no valid assignment/solution and we say that C is *inconsistent*.

Constraints can be specified in two different ways: implicitly (e.g., $x_1 \leq x_2$) or explicitly, as the set of solution tuples. We will always use the former variant, since the latter is rather impractical when handling large variable sets or domains.

Definition 1. A *Constraint Satisfaction Problem* (CSP) consists of:

- a finite set of variables $X = \{x_1, \ldots, x_n\}$
- a finite set of constraints $\mathbb{C} = \{C_1, \ldots, C_m\}$ on X

A solution to a CSP is an assignment of a value $d_i \in D(x_i)$ to each $x_i \in X$, such that all constraints from $\mathbb{C}$ are satisfied simultaneously.

Solving a CSP might be useful for several purposes: to determine the feasibility of a problem, to find any valid solution or to find or count all solutions. When looking from an operations research perspective this is not appealing because we are interested in optimal solutions with respect to the given constraints. Hence, we extend the CSP definition:

Definition 2. A *Constraint Optimization Problem* (COP) consists of:

- a CSP P with variables $x_1, \ldots, x_n$
- an *objective function* $f : D(x_1) \times \cdots \times D(x_n) \to \mathbb{Q}$

An optimal solution to a minimization (maximization) COP is a solution d to P which minimizes (maximizes) the value of the objective function $f(d)$.

There are several mature constraint programming libraries in the field, most of them for C++ and Java. Usually, they solve CSPs and COPs by a combination of the following techniques:

- **Constraint Propagation** (reducing variable domains by reasoning from the present constraints and current variable domains)
- **Systematic Search** and backtracking through the space of possible solutions
- **Linear Programming**
- **Stochastic Local Search**

Major parts of the solution presented in this thesis make use of the freely available CP toolkit Gecode. Therefore, we will give a brief introduction to Gecode in the following sections.

2.1.1 Gecode: A Generic Constraint Development Environment

Gecode [Gec10] is a free open source toolkit for developing constraint-based systems and applications. Since its first official release in December 2005, the project has improved constantly and is still under active development. Besides being mature and free, Gecode provides an extensive reference documentation, a lot of examples and is extremely extensible. This not only includes the creation of own models and constraint types, but also customizations of the core CP solver, including custom branching strategies and search engines. Unfortunately, the rather complex tasks of customizing the search engine and adding own variables are not very well documented yet, but there have been considerable improvements in the recent months and the freely available source code offers helpful insights.

Basics

The central object in a Gecode model is a so-called *space*. It describes the current state of our world/model, including all variables, their respective domains and all the constraints a solution has to fulfill. Adding further constraints to a space is called *posting*.

In the remainder of this section we give a short introduction to the different components of a Gecode model. For further reading we recommend the great introduction "Modelling and Programming with Gecode" [STL10] and the dissertation about "Constraint Propagation" [Tac09] by one of the main contributors of Gecode.

2.1.2 Models and Constraints

One of the models offered by Gecode to formulate constraint programs are *finite domain integers* (FDI). An FDI variable can take any integer of its domain – a finite set of 32-bit integers. Note that we can also model logical values using an FDI variable with a domain of $\{0; 1\}$.

The general procedure performed by the CP solver is as follows. In the beginning a space is set up and all its variables are only limited by their initial domain (e.g., $D(x_1) = \{1, \dots, 10\}$, $D(x_2) = \{4, \dots, 10\}$). First, a phase of constraint *propagation* is performed, to limit these domains by reasoning from the given constraints (e.g., $x_1 > x_2 \Rightarrow D(x_1) = \{5, \dots, 10\} \wedge D(x_2) = \{4, \dots, 9\}$). We call a state where no further propagation is possible in the space a *fix point*. A *branching* now splits the search space into (generally) two branches. One branch usually gets one variable assigned to a value of its domain, while this value is excluded from the variables domain in the second branch. After that, constraint propagation can be applied to both branches and the process repeats until there are no more unassigned variables, i.e., a valid solution is found. Usually, this *search* process is organized as a depth first search in the search tree formed by the branching.

The details of propagation, branching and search will be explained in the following sections. Let us first have a look on the constraint types for finite domain integers provided by Gecode, which will be relevant for our upcoming problem formulation:

Domain constraints: The domain of a variable is specified by a lower and upper bound or is defined as a set of integers.

Relational constraints: These constraints can be used to enforce different types of binary relations (e.g., $>, \geq, =, \neq, \leq, <$) between different variables or between a variable and a constant. They can also be used in another way, where the relation is not enforced, but is evaluated and assigned to a boolean-valued FDI variable (e.g., $x < y \Leftrightarrow b$). Most relational constraints can be naturally applied to arrays of variables.

Linear constraints: They can be used to determine a (weighted) sum of an array of variables and model a relation between the result and another variable or constant (e.g., $\sum_i a_i \cdot x_i \leq y$). Similar to the relational constraints, there is an option to evaluate the relation and assign its result (true or false) to another variable.

Arithmetic constraints: A variety of basic arithmetic operations can by applied to variables and constants. This includes basic binary operators ($+$, $-$, $*$, $/$, mod) as well as unary ones (square and square root, absolute value) and operations on arrays of variables (minimum, maximum). The result of this operation is always related to another variable by equality (e.q. $x_1 \sim x_2 = x_3$).

Scheduling constraints: This is one of the more sophisticated constraint types of Gecode. They model resource limits when scheduling a set of unsplittable jobs onto a set of machines. Jobs are given by their possible start/end dates, duration and resource consumption. The machines are given by their respective resource amount per unit of time. Resource limits can be either upper or lower bounds. Note that this constraint does not actively schedule the dates, but limits each jobs start/end domain to feasible values, considering the available machines and states of the other jobs. See [BC02] for details.

Relational constraints on boolean variables (like $\vee$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$) can also be modelled using arithmetic and relational constraints (e.g., $A \Rightarrow B$ is equivalent to $B - A \geq 0$). Additionally, Gecode provides a variety of other constraint types like *array element*, *distinct*, *channel*, *sorted*, *cardinality* and *graph constraints* which we will not introduce here, since they do not appear in our model.

To ease the effort required for modeling, there is a so-called direct modeling support provided by Gecode (technically it is overloading C++ operators). These functions allow us to formulate constraints in a very natural way. For example the constraint $(x + y < 23) \Rightarrow z = 42$ would look like:

```
post(space, tt(imp( ~(x + y < 23), ~(z == 42))))
```

2.1.3 Propagation

In Gecode constraints are implemented as so-called *propagators*. When set up, each propagator subscribes to its affected variables. Later, it is notified on changes of these variables' domains. When called because of changes to a variable x, the propagator checks if the smaller domain of x helps to reduce the domains of other variables. We call this process *propagation* and the reduction of domains *filtering*. Filtering does not change the set of solutions. Furthermore, a propagator manipulating the domain of a variable possibly triggers further propagator execution (including of itself). This notification/execution scheme is repeated until we either reach a fix point or a domain becomes empty. We call the latter a *failed space*.

We can distinguish between different strengths of filtering. A filtering for a constraint C is *complete* if each removal of a value from the residual variable domains would reduce the set of solutions for C. Otherwise, all filterings where additional narrowing of a domain would be possible are called *partial*. We will now formalize two so-called *consistency levels*, which arise from the filtering strengths (cf. [RBW06]):

Definition 3. *(Domain Consistency)* Let C be a constraint on the variables $x_1, \ldots, x_n$ with their respective domains $D(x_i)$ for every $1 \leq i \leq n$. We say that C is domain consistent if for $1 \leq i \leq n$ and $v \in D(x_i)$, there exists a tuple $(d_1, \ldots, d_n) \in C$ such that $d_i = v$. A CSP is domain consistent if each of its constraints is domain consistent.

Definition 4. *(Bound Consistency)* Let C be a constraint on the variables $x_1, \ldots, x_n$ with respective lower and upper bounds $L(x_1), U(x_1), \ldots, L(x_n), U(x_n)$. We say that C is bound consistent if both $(L(x_1), \ldots, L(x_n)) \in C$ and $(U(x_1), \ldots, U(x_n)) \in C$.

We illustrate both consistency types in a short example. Let us assume that we have two variables with domains $D(x_1) = \{1, \ldots, 5\}$, $D(x_2) = \{1, \ldots, 6, 8\}$ and a single constraint $x_1 + x_2 = 10$. A close look reveals that the domain consistent fix point is $D(x_1) = \{2, 4, 5\}$ and $D(x_2) = \{5, 6, 8\}$. Note that it is generally hard (even NP-hard) to find such a fix point. Using the variables' bounds ($L(x_1) = 1$, $U(x_1) = 5$, $L(x_2) = 1$, $U(x_2) = 8$) we can derive that $x_1 \geq 2$ and $x_2 \geq 5$, thus resulting in a slightly weaker bound consistent fix point at $D(x_1) = \{2, \ldots, 5\}$ and $D(x_2) = \{5, 6, 8\}$.

Gecode offers three different notification levels for propagators, derived from these consistency levels. In the strongest, *domain level*, a propagator gets notified of all domain changes of its subscriptions. In *bound level*, the system notifies a propagator if either the lower or upper bound of a subscribed variable is modified. Finally, the weakest level, *value level*, only notifies propagators if a subscribed variable gets assigned, i.e., when its domain size becomes exactly one.

Execution costs between these levels may vary greatly. Although domain consistency yields the strongest filtering, it might be computationally hard to achieve. The weaker levels can provide a significant speedup at the cost of a reasonably poorer filtering. Thus, one can tweak a model by choosing the right notification levels for the propagators.

2.1.4 Branching

After a propagation phase has reached a fix point, no further reasoning is possible. Before propagation can continue, we have to limit some variable's domain or add further constraints manually. Note that the former can be modelled as the latter. An important property of a branching step is *completeness*, which states that everything that follows from our model should still be derivable after branching. Accordingly, to achieve completeness while adding further constraints, we have to split the search space into at least two branches, whose union equals our original search space. To minimize time and effort, the search spaces of these branches should not intersect. Figure 2.1 gives an idea of possible kinds of branching.

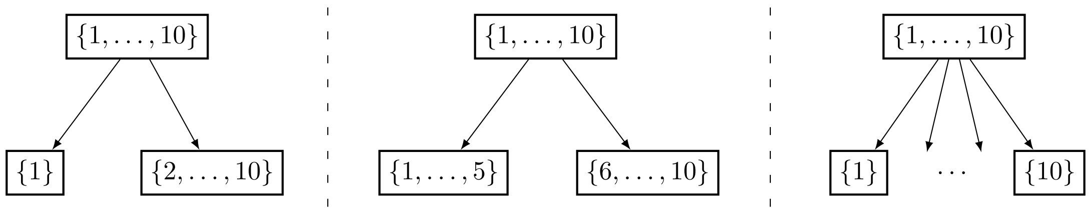

Figure 2.1: We show three different modes of branching a variable's domain. In the left one, the first branch gets a value assigned, while this value is excluded from the domain of the second branch. In the middle the domain is equally distributed among two branches. On the right we have a branch for each value in the variable's domain.

A Gecode branching mainly consists of three functions. First, a *status* function that evaluates if this branching is able to split the current space. Generally this is true if not all variables monitored by this branching are assigned. The second function provides *choices*, i.e., the number of alternatives to branch and how they differ from their parent space. Finally, the third function transforms a parent space into a branched space by applying a given choice. This step is called *commit*.

Gecode offers a variety of standard branchings that are independent of the applications domain. They choose the next variable to be assigned by simple rules, among others: the first variable, the variable with the smallest or largest domain, the variable with the smallest (largest) domain minimum (maximum), a random variable. The assigned value can also be chosen by different strategies: domain minimum/median/maximum or random.

The order of choices of a branching heavily influences the performance of the search process. Assigning the right variables first might dramatically prune the search space. Often it is a good choice to select the variable with the smallest domain. This way it can be shown early if there exists no feasible assignment for the variable, thus resulting in less failed spaces throughout the remaining search process. Accordingly, we can say that the number of failed spaces visited during the search attributes to the quality of a branching.

For most problems there exist good heuristics to decide which variable to assign next and which value to choose. Gecode offers a way to implement such heuristics in custom branchings and to use them in the search process. This way, one might get better performance and branching quality (i.e., less failed spaces).

2.1.5 Search Engines

Gecode offers several search engines which roughly control the search process. Basically, the engine decides about the order of traversal of the search tree formed by the branchings and their respective choices. A common search engine for solving constraint models is a simple depth-first search. At each node of the search tree, the engine commits the first alternative of the choice of one of the present branchings. After each commit a propagation phase takes place. If the domain of at least one variable becomes empty, i.e., a failed space is reached, a backtrack step is performed and the next alternative is chosen. This is repeated until no branching can split the search space anymore – a solution is found – or the whole search tree has been traversed.

One notable problem of a depth-first search on large search spaces is that bad assignments of the first branched variables can hardly be revised. In this case, the whole – possibly huge – subtree has to be evaluated before a decision can be revoked. We will overcome this problem by restarting the search process after a certain number of failed spaces has been reached and by randomizing choices.

2.2 Graphs and Networks

In this section we briefly introduce the main graph theoretic concepts that will appear in the upcoming chapters. For a general introduction to the topic we refer to [CLRS01]. Detailed information about network flows can be found in [AMO93].

Definition 5. *(Digraph)* A directed graph is a pair $G = (V, E)$, where V is a finite set of vertices and $E \subseteq V \times V$ is a set of ordered pairs of vertices, called arcs. An arc from $u \in V$ to $v \in V$ is denoted by (u, v).

Definition 6. *(Flow Network)* A flow network $N = (G, cap, s, t)$ consists of a directed graph $G = (V, E)$, a capacity function $cap : E \to [\mathbb{R}^{\geq 0}, \mathbb{R}^{\geq 0}]$ and two distinguished vertices: a source s and a target t. For each arc the capacity function returns a non-empty interval. If $(u, v) \notin E$, we assume $cap(u, v) := [0, 0]$.

Definition 7. *(Flow)* Let $N = (G, cap, s, t)$ be a flow network. A valid flow is a function $f : V \times V \to \mathbb{R}$, which satisfies the following three properties:

- Capacity constraint: $\forall u, v \in V : f(u, v) \in cap(u, v)$
- Skew symmetry: $\forall u, v \in V : f(u, v) = -f(v, u)$
- Flow conservation: $\forall u \in V \setminus \{s, t\} : \sum_{v \in V} f(u, v) = 0$

Definitions 6 and 7 differ slightly from the definitions found in the mentioned textbooks, because we introduce the additional feature of a lower bound of the capacity function. But, such a network can be transformed into a network without a lower capacity bound (see Figure 2.2), but where any valid flow has to saturate certain arcs.

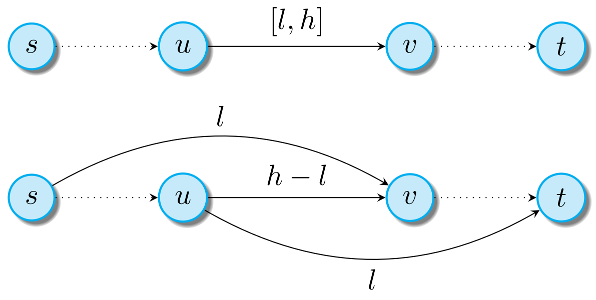

Figure 2.2: Removal of minimum capacities: The arc (u, v) in the upper network has a minimum capacity of l. The lower network does not comprise minimum capacities anymore, but each valid flow has to saturate the new arcs incident to s and t.

Definition 8. *(Minimum Cost Flow Problem)* Let $N = (G, cap, s, t)$ be a flow network, $cost : V \times V \to \mathbb{R}$ a cost function of each arc and d the required amount of flow. The minimum-cost flow problem $P = (N, cost)$ is to find a valid flow f that minimizes the cost function when implementing a flow of size d in the network:

- Required Flow: $d = \sum_{u \in V} f(s, u) = \sum_{u \in V} f(u, t)$
- Objective Function: $\min_f \left(\sum_{(u,v) \in E} f(u, v) \cdot cost(u, v) \right)$

There are several algorithms available to solve the minimum cost flow problem. For our experiments we use an implementation of the cost scaling algorithm [GT90] which is part of the Lemon Graph Library [Lem10]. The algorithm runs in $O(n^3 \log(nC))$ where C is the maximum absolute integer-valued arc cost.

While modeling a part of our problem as a flow network, we identify costs that do not depend on the implemented flow in the network. We will extract these costs into a *global cost offset* and adjust the value of the objective function accordingly:

$$cost(f) := global\ cost\ offset + \sum_{(u,v) \in E} f(u, v) \cdot cost(u, v)$$

3. Problem Statement

This chapter introduces the problem proposed by the ROADEF/EURO Challenge 2010 in [PGJ$^+$] and sets up the model used throughout this thesis. The problem definition was mainly developed by the French utility company EDF and therefore also contains some features of French energy policy. Furthermore, all problem instances are provided by EDF.

We begin with an introduction to the topic in Section 3.1 to the model, followed by some general definitions in Section 3.2. Besides some domain-specific terminology, this section introduces the indices, variables and auxiliary constructs used in the model. Afterwards, Section 3.3 presents the core of the model: various constraint types separated into production related constraints (i.e., technical constraints of each single power plant) and outage scheduling constraints (i.e., constraints between different power plants). Finally, Section 3.4 shows a formal definition of the objective function.

Parts of this chapter replicate the ROADEF/EURO Challenge 2010 subject document. We slightly modify the model proposed there to remove redundancies and to make parameters, function signatures and indexing consistent and easier to understand. There are no conceptual changes.

3.1 Introduction

In the following we introduce a model for medium-term electricity production planning utilizing a large set of power plants. As it is a tactical model, we neither make any strategic decisions (like adding/removing power plants) nor do we care for short-term operational restrictions (like intraday load following). Although the model comprises some peculiarities of the French energy market (see [Kid09]), it should be useful in different contexts.

The model extends over a period of time (e.g., 5 years). This period is split into uniform time steps of configurable length (e.g., 1 day). The two first class entities of our model are a set of various power plants and a set of uncertainty demand scenarios (see Figure 3.1). For each scenario we are looking for a production assignment, such that the sum of energy produced by all available power plants equals the demand during each time step. The need for multiple scenarios arises from the numerous uncertainties that have to be taken into account (e.g., unknown energy demand, generation units availability, spot market prices, quantities that can be bought or sold, . . .).

In our model there are two very different types of facilities. Power plants of the first type can operate continuously and their fuel supply is outside the scope of our problem. During

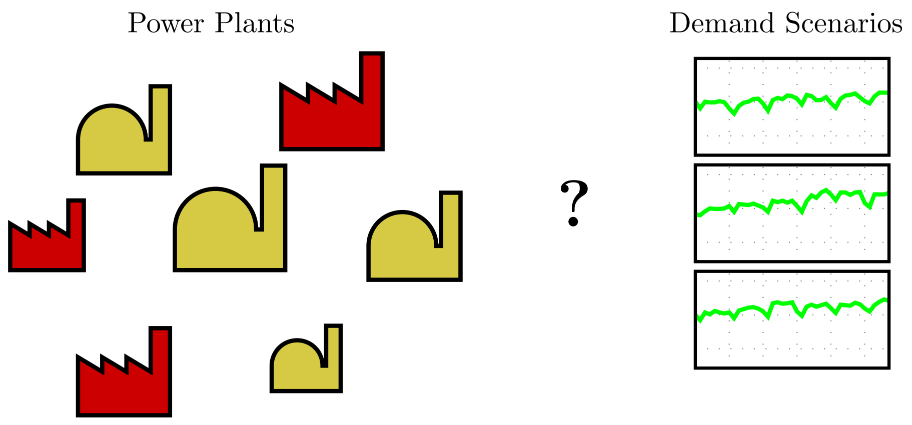

Figure 3.1: On the left, we have an inhomogeneous set of power plants, on the right, a set of demand scenarios. For each scenario we have to generate a production plan distributing the demand among the power plants.

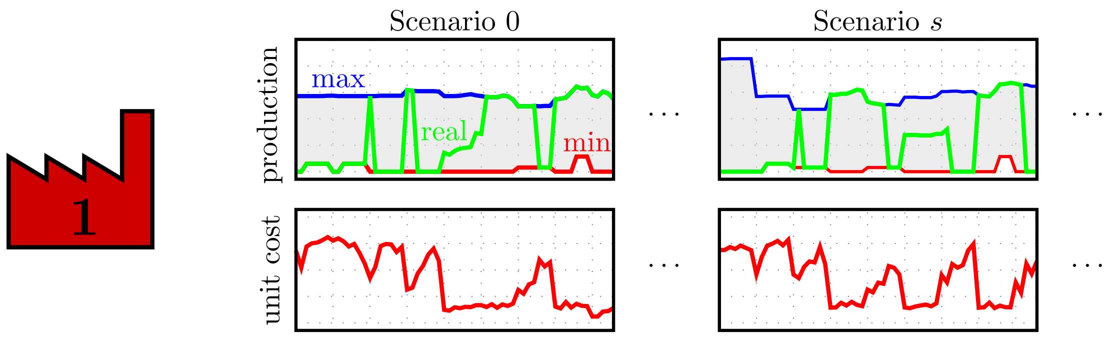

Figure 3.2: Sample plots illustrating the development of production and unit cost over time of a Type-1 power plant for different scenarios. In the upper plots the red (blue) lines show the minimum (maximum) allowed production. The green lines are production plans. In the lower plots the red lines show the cost per unit of energy produced by this plant.

each time step they can produce an amount of energy in an interval depending on the scenario and time step (see Figure 3.2). We call them *Type-1 power plants*. Production at these power plants induces cost that is proportional to the power output of a plant and also depend on the scenario and time step. Power plants of this type might be coal- or gas-fired or even virtual power plants for exporting and importing energy, whose available power levels and unit cost we cannot influence.

The other type of power plants, called *Type-2 power plants*, has to be shut down for refueling and maintenance regularly. Therefore, modeling these facilities requires more effort. As the plants have only limited capacities to store fuel and running out of it would stop the production of energy, these power plants have to be refueled regularly (see Figure 3.3). Refueling can only take place when a power plant is offline for several weeks. Hence, the operation of a Type-2 power plant is organized in cycles – successions of an offline period (an outage with refueling and maintenance) and the following production campaign. It might not be necessary to schedule all cycles of a Type-2 power plant before the end of the time horizon, i.e., we can postpone outages. But the order of a power plant's cycles is fixed and so all successive cycles would have to be postponed, too.

Contrary to Type-1 power plants, production at Type-2 power plants is billed via the amount of fuel reloaded. In the model, the fuel unit cost depends on the cycle and power

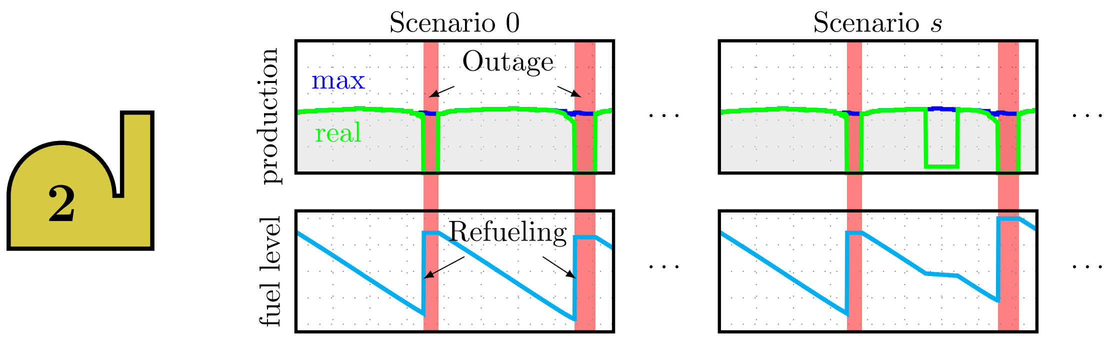

Figure 3.3: Sample plots illustrating the production and fuel levels over time of a Type-2 power plant for different scenarios. In the upper plots the gray areas show the allowed production interval. The green line is a production plan. Note that the outage dates (red area) and refueling amounts (upturns of the cyan line) are equal in all scenarios.

plant. For each Type-2 power plant a number of (fuel-related) production constraints apply. Furthermore, outages of different Type-2 power plants are dependent on each other to fulfill resource, staff, grid stability, production safety and legal restrictions. Power plants of this type are generally nuclear power plants.

To conclude, the proposed subject consists of modeling the production assets and finding an optimal outage schedule. It includes two dependent subproblems:

1. Determine a schedule of plant outages. This schedule must satisfy the given constraints in order to comply with limitations on resources, which are necessary to perform refueling and maintenance operations.
2. Determine an optimal production plan to satisfy demand, i.e., a quantity of energy to produce by each plant at each time step for each scenario.

The objective is to minimize the expected cost of production.

3.2 General Definitions

In this section, we define domain-specific terminology, indices, decision variables and problem parameters, which will be used by the constraint definitions in the next section and all following parts of this thesis.

3.2.1 Terminology

First, we introduce some words with domain-specific meaning:

Time Step: The time horizon is split into discrete periods of time of the same length. A time step is the smallest unit of time we deal with, i.e., our world is constant during a time step.

Week: A (per dataset) fixed number of successive time steps. Note that power plant outages are scheduled in weeks, while production levels will be planned for each time step.

Scenario: The demand to be satisfied by all plants is given in the form of different uncertainty scenarios. Each scenario is given as a vector of concrete production amounts (i.e., one value per time step), which can be seen as the result of some stochastic process.

The following terms apply only to Type-2 power plants:

Outage: A series of weeks during which a plant is offline.

Production Campaign: A series of weeks during which a plant can produce.

Cycle: The succession of an outage and a production campaign.

Decoupling: The first week of an outage.

Coupling: The first week of a production campaign.

Reloading/Refueling: The amount of fuel provided during an outage.

Modulation: The sum of the differences between the maximum allowed and actual production over all time steps of a production campaign.

Imposed Power Profile: A constraint imposing that the production level of a plant has to follow a given profile if the fuel level is below a certain threshold.

3.2.2 Indices

A dataset comprises various sets of entities. Let $\mathcal{S}$ denote the set of the scenarios, $\mathcal{J}$ the set of Type-1 power plants and $\mathcal{I}$ the set of Type-2 power plants. Our timeline consists of T uniform time steps spanning over W weeks in total, where T is always a multiple of W. We define the corresponding sets for time steps as $\mathcal{T} = \{0, \ldots, \mathrm{T}-1\}$ and weeks as $\mathcal{W} = \{0, \ldots, \mathrm{W}-1\}$.

We denote the number of cycles of each Type-2 power plant as K and the corresponding set as $\mathcal{K} = \{0, \ldots, \mathrm{K}-1\}$. K is equal for all Type-2 power plants. Furthermore, we denote the cycle and production campaign each plant starts with (i.e., during time step 0) as the *initial cycle* and use $k = -1$ for it. Similarly, we use K, T and W in the upcoming parameters to denote the end of our time horizon.

The following lower case indices will be used throughout this document to access single entities of the given type: $s \in \mathcal{S}$, $j \in \mathcal{J}$, $i \in \mathcal{I}$, $t \in \mathcal{T}$, $w \in \mathcal{W}$, $k \in \mathcal{K}$.

3.2.3 Decision Variables

The following three types of decision variables make up our problem:

Decoupling Dates: Each Type-2 power plant regularly goes offline for refueling and maintenance. Let $ha_{i,k} \in \{-1\} \cup \mathcal{W}$ denote the week of decoupling of plant i in cycle k. Note that there are two special cases. First, for the initial cycle of each plant we define $ha_{i,-1} := 0$. Second, if a cycle is not scheduled, i.e., its decoupling date is postponed behind the time horizon, then $ha_{i,k}$ will be set to -1.

Refueling Amounts: During each outage a Type-2 power plant can be supplied with a certain of amount of fuel. Let $r_{i,k} \in \mathbb{R}^{\geq 0}$ denote this amount of plant i in cycle k. We define the refueling amount of postponed cycle as 0.

Production Levels: In a solution all power plants (i.e., both types) need to have an absolute real-valued production level assigned for each time step and scenario. Let $p_{j,s,t}/p_{i,s,t} \in \mathbb{R}^{\geq 0}$ denote this level of power plant j/i during time step t in scenario s.

Note that a power plant's production levels may vary between demand scenarios of a dataset, while outage dates and refueling amounts are decision variables of the dataset, i.e, they are equal in all scenarios.

3.2.4 Global Parameters

- $\mathrm{DEM}^{s,t}$: Demand to satisfy for scenario s and time step t
- D: Length of a time step (all time steps have equal length)

3.2.5 Parameters of each Type-1 Power Plant j

- $\mathrm{PMIN}_j^{s,t}$: Minimum production level during time step t of scenario s
- $\mathrm{PMAX}_j^{s,t}$: Maximum production level during time step t of scenario s
- $\mathrm{C}_j^{s,t}$: Cost of production per unit during time step t of scenario s

3.2.6 Parameters of each Type-2 Power Plant i

- PMAX_i^t: Maximum production level during time step t in all scenarios
- XI_i: Initial fuel level (i.e., in time step 0)
- $\mathrm{C}_{i,\mathrm{T}}$: Discount per unit for residual fuel at the end of the time horizon

The following parameters are provided for each cycle k of a Type-2 power plant:

- $\mathrm{DA}_{i,k}$: Duration of the outage in weeks (note the special case $\mathrm{DA}_{i,-1} := 0$)
- $\mathrm{C}_{i,k}$: Cost of refueling per unit during this cycle's outage
- $\mathrm{RMIN}_{i,k}$: Minimum refueling amount
- $\mathrm{RMAX}_{i,k}$: Maximum refueling amount
- $\mathrm{MMAX}_{i,k}$: Maximum modulation over production campaign
- $\mathrm{Q}_{i,k}$: Refueling coefficient (see constraint 10)
- $\mathrm{BO}_{i,k}$: Fuel level threshold activating the imposed power profile for this campaign
- $\mathrm{PB}_{i,k}$: Decreasing power profile (a piece–wise linear concave function: $\mathbb{R}^{\geq 0} \rightarrow [0,1]$)
- ϵ: Tolerance for the imposed power profile
- $\mathrm{AMAX}_{i,k}$: Upper bound of fuel level before refueling
- $\mathrm{SMAX}_{i,k}$: Upper bound of fuel level after refueling

Additionally, the parameters $\mathrm{MMAX}_{i,-1}$, $\mathrm{BO}_{i,-1}$ and $\mathrm{PB}_{i,-1}$ will also be provided.

3.2.7 Auxiliary Constructs

We define some auxiliary constructs for Type-2 power plants, which will come in handy for the upcoming constraint definitions:

Definition 9. Given an arbitrary Type-2 power plant i and a production cycle k, let $t^-_{i,k}$ denote the first time step of this cycle (which is also the first time step of the outage), $t^*_{i,k}$ denote the first time step of this cycle's production campaign and $t^+_{i,k}$ denote the first time step after the end of this cycle. For a not scheduled cycle all three variables equal T. Formally we define:

$$
\begin{aligned}
ha_{i,k} \neq -1 &\Longrightarrow t^-_{i,k} = ha_{i,k} \cdot \mathrm{T} \,/\, \mathrm{W} \\
ha_{i,k} \neq -1 &\Longrightarrow t^*_{i,k} = (ha_{i,k} + \mathrm{DA}_{i,k}) \cdot \mathrm{T} \,/\, \mathrm{W} \\
ha_{i,k+1} \neq -1 &\Longrightarrow t^+_{i,k} = ha_{i,k+1} \cdot \mathrm{T} \,/\, \mathrm{W}
\end{aligned}
$$

Note that $t^+_{i,k}$ generally corresponds to $t^-_{i,k+1}$ with exceptions for the initial cycle, where $t^-_{i,-1} = t^*_{i,-1} = 0$, and the last cycle, where $t^+_{i,\mathrm{K}-1} = \mathrm{T}$.

Definition 10. Let $x(i,s,t)$ denote the fuel level of plant i at the beginning of time step t in scenario s. For each plant and scenario the fuel level of a time step depends on the fuel level of the previous time step and the production level and refueling performed during the previous time step. See the constraints 9 and 10 for the details. Note that $x(i,s,\mathrm{T})$ denotes the fuel level at the end of the time horizon.

3.3 Constraints

In this section we define the constraints that make up our model. We adopt the numbering from the official problem statement of the competition. In the first part we model plant operations (production levels, fuel level variation, fuel level restrictions). The second part presents different types of outage constraints between Type-2 power plants which arise from limited resources (staff, tools, ...), legal restrictions and production safety considerations.

[CT 1] Coupling load and production

The sum of production of all Type-1 power plants and all Type-2 power plants equals the demand $\mathrm{DEM}^{s,t}$ in all scenarios s and time steps t:

$$\sum_{j\in\mathcal{J}} p_{j,s,t} + \sum_{i\in\mathcal{I}} p_{i,s,t} = \mathrm{DEM}^{s,t} \qquad s\in\mathcal{S}; t\in\mathcal{T}$$

3.3.1 Production Related Constraints

[CT 2] Bound of production of Type-1 power plants

The production $p_{j,s,t}$ of a Type-1 power plant j has to stay between its lower bound $\mathrm{PMIN}_j^{s,t}$ and upper bound $\mathrm{PMAX}_j^{s,t}$. These bounds may also model outages which are outside the scope of our problem:

$$\mathrm{PMIN}_j^{s,t} \leq p_{j,s,t} \leq \mathrm{PMAX}_j^{s,t} \qquad j\in\mathcal{J}; s\in\mathcal{S}; t\in\mathcal{T}$$

[CT 3] Offline power

During every time step t where a Type-2 power plant i is on outage, its production $p_{i,s,t}$ equals zero in all scenarios s:

$$p_{i,s,t} = 0 \qquad i\in\mathcal{I}; s\in\mathcal{S}; t\in\bigcup_{k\in\mathcal{K}}[t^-_{i,k}, t^*_{i,k}-1]$$

[CT 4] Bound of production of Type-2 power plants

During every time step t the production $p_{i,s,t}$ of a Type-2 power plant i is between zero and the upper bound PMAX_i^t:

$$0 \leq p_{i,s,t} \leq \mathrm{PMAX}_i^t \qquad i\in\mathcal{I}; s\in\mathcal{S}; t\in\mathcal{T}$$

[CT 5] Maximum power before power profile imposition

We merged this constraint with CT 4.

[CT 6] Maximum power after power profile imposition

If the fuel level $x(i,s,t)$ of a Type-2 power plant i drops below the cycle's threshold $\mathrm{BO}_{i,k}$, then the production $p_{i,s,t}$ is fixed to the given ratio $\mathrm{PB}_{i,k}:\mathbb{R}^{\geq 0}\to[0,1]$ of PMAX_i^t within a tolerance of ϵ, as long as there is enough fuel $x(i,s,t)$ to cover the current consumption:

$$\begin{aligned}&\mathrm{PB}_{i,k}(x(i,s,t))\cdot\mathrm{PMAX}_i^t\cdot\mathrm{D} \leq x(i,s,t) < \mathrm{BO}_{i,k} \Longrightarrow \\ &\qquad (1-\epsilon)(\mathrm{PB}_{i,k}(x(i,s,t))\cdot\mathrm{PMAX}_i^t) \leq p_{i,s,t} \leq (1+\epsilon)(\mathrm{PB}_{i,k}(x(i,s,t))\cdot\mathrm{PMAX}_i^t)\end{aligned}$$

If there is not enough fuel left to fulfill the imposed production amount, then this power plant cannot produce anything:

$$\big(x(i,s,t) < \mathrm{BO}_{i,k}\big) \wedge \big(x(i,s,t) < \mathrm{PB}_{i,k}(x(i,s,t))\cdot\mathrm{PMAX}_i^t\cdot\mathrm{D}\big) \Longrightarrow p_{i,s,t} = 0$$

[CT 7] Bounds of refueling

If a cycle k of Type-2 power plant i is scheduled (i.e., $ha_{i,k} \neq -1$), then the performed refueling $r_{i,k}$ is between its lower bound $\mathrm{RMIN}_{i,k}$ and upper bound $\mathrm{RMAX}_{i,k}$:

$$ha_{i,k} \neq -1 \Longrightarrow \mathrm{RMIN}_{i,k} \leq r_{i,k} \leq \mathrm{RMAX}_{i,k} \qquad i \in \mathcal{I}; k \in \mathcal{K}$$

[CT 8] Initial fuel level

For each Type-2 power plant i, the fuel level $x(i,s,t)$ of the first time step equals the plant's initial fuel level parameter XI_i:

$$x(i,s,0) = \mathrm{XI}_i \qquad i \in \mathcal{I}; s \in \mathcal{S}$$

[CT 9] Fuel level variation during a production campaign

Fuel levels are passed between successive time steps and adjusted by the fuel consumption, which is calculated from the production level $p_{i,s,t}$ and the time steps duration D:

$$x(i,s,t+1) = x(i,s,t) - p_{i,s,t} \cdot \mathrm{D} \qquad i \in \mathcal{I}; s \in \mathcal{S}; t \in \bigcup_{k \in \mathcal{K}} [t^*_{i,k}, t^+_{i,k} - 1]$$

Note that fuel levels cannot be negative:

$$x(i,s,t) \geq 0 \qquad i \in \mathcal{I}; s \in \mathcal{S}; t \in \mathcal{T} \cup \{\mathrm{T}\}$$

[CT 10] Fuel level variation during an outage

In the process of refueling a Type-2 power plant i, a certain amount of unspent fuel has to be removed to make the addition of new fuel possible. The refueling coefficient $\mathrm{Q}_{i,k}$ helps to quantify this amount. Note that the fuel threshold $\mathrm{BO}_{i,k}$ is in reality part of the reload – refueling $r_{i,k}$ and threshold $\mathrm{BO}_{i,k}$ have been separated in the formulation because one is a decision variable and the other is imposed (a technical parameter).

Refueling is performed entirely during the first timestep $t^-_{i,k}$ of an outage:

$$x(i,s,t^-_{i,k}+1) = \frac{\mathrm{Q}_{i,k} - 1}{\mathrm{Q}_{i,k}} (x(i,s,t^-_{i,k}) - \mathrm{BO}_{i,k-1}) + r_{i,k} + \mathrm{BO}_{i,k} \qquad i \in \mathcal{I}; s \in \mathcal{S}; k \in \mathcal{K}$$

There is no further fuel variation during an outage.

[CT 11] Fuel level bounds around refueling

The fuel levels before and after refueling of a Type-2 power plant i and cycle k are restricted by $\mathrm{AMAX}_{i,k}$ and $\mathrm{SMAX}_{i,k}$ respectively:

$$\begin{aligned} 0 \leq x(i,s,t^-_{i,k}) &\leq \mathrm{AMAX}_{i,k} \\ x(i,s,t^-_{i,k}+1) &\leq \mathrm{SMAX}_{i,k} \end{aligned} \qquad i \in \mathcal{I}; s \in \mathcal{S}; k \in \mathcal{K}$$

[CT 12] Maximum modulation over a cycle

While production is not imposed (see CT 6), the production campaign's aggregated deviation from the maximum production PMAX^t_i is limited by $\mathrm{MMAX}_{i,k}$. If the fuel level is below $\mathrm{BO}_{i,k}$ then the imposed power profile determines the exact production level.

$$\sum_{\substack{t^*_{i,k} \leq t < t^+_{i,k} \\ x(i,s,t) \geq \mathrm{BO}_{i,k}}} \left(\mathrm{PMAX}^t_i - p_{i,s,t}\right) \cdot \mathrm{D} \leq \mathrm{MMAX}_{i,k} \qquad i \in \mathcal{I}; s \in \mathcal{S}; k \in \mathcal{K}$$

3.3.2 Outage Scheduling Constraints

The following constraints set up relations between outages of different Type-2 power plants. A dataset might contain multiple instances of the constraint types 14 to 21 for varying subsets of plants and with different parameters. Note that all these constraint instances are independent and do not share any parameters.

[CT 13A] Earliest and latest date of outage

An outage has to start during a given interval or may not be scheduled.

Data:

- **i**: The given Type-2 power plant
- **k**: The given outage/cycle
- $\text{TO}_{i,k}$: First possible week of decoupling
- $\text{TA}_{i,k}$: Latest week of decoupling (or -1 if the cycle can be postponed)

Constraint:

$$\begin{aligned} \text{TA}_{i,k} \neq -1 &\Longrightarrow \text{TO}_{i,k} \leq ha_{i,k} \leq \text{TA}_{i,k} \\ \text{TA}_{i,k} = -1 &\Longrightarrow ha_{i,k} = -1 \vee \text{TO}_{i,k} \leq ha_{i,k} \end{aligned}$$

[CT 13B] Order of outage

Outages of a Type-2 power plant i appear in the same order as presented in the dataset and do not overlap (remember $\text{DA}_{i,k}$ as outage duration). Therefore, if we decide not to schedule an outage, all successive outages of the same plant have to be postponed, too.

Constraint:

$$\begin{aligned} ha_{i,k} \neq -1 &\Longrightarrow ha_{i,k-1} + \text{DA}_{i,k-1} \leq ha_{i,k} \\ ha_{i,k-1} = -1 &\Longrightarrow ha_{i,k} = -1 \end{aligned} \qquad i \in \mathcal{I}; k \in \mathcal{K}$$

3.3.3 Outage Dependency Constraints

The following constraints operate on a subset of Type-2 power plants $\mathcal{A} \subseteq \mathcal{I}$ and only on their scheduled cycles.

Data:

- $\mathcal{A}$: Set of considered Type-2 power plants

3.3.3.1 Spacing Constraints

The following constraints define minimum spacings (or maximum overlappings, respectively) between scheduled cycles from power plants of the set $\mathcal{A}$.

Data:

- Se: Duration in weeks of minimum authorized spacing. Negative values are interpreted as maximum authorized overlapping.

[CT 14] Minimum spacing/Maximum overlapping between outages

This constraint defines a set of Type-2 power plants $\mathcal{A}$ whose outages have to be scheduled with a minimum pairwise spacing (or maximum overlapping) of Se weeks.

Constraint:

$$\begin{aligned}&\big(ha_{i,k} - ha_{i',k'} - \mathrm{DA}_{i',k'} \geq \mathrm{Se}\big) \vee \\ &\qquad (ha_{i',k'} - ha_{i,k} - \mathrm{DA}_{i,k} \geq \mathrm{Se})\end{aligned} \qquad i, i' \in \mathcal{A}; k, k' \in \mathcal{K}; i \neq i'$$

[CT 15] Minimum spacing/Maximum overlapping between outages during a specific period

Outages of a set $\mathcal{A}$ of Type-2 power plants that intersect an interval $[\mathrm{ID}, \mathrm{IF}]$ have to be scheduled with a minimum pairwise spacing (or maximum overlapping) of Se weeks.

Data:

- ID: First week of the specific period
- IF: Last week of the specific period

Constraint:

$$\begin{aligned}&\big(\mathrm{ID} - \mathrm{DA}_{i,k} + 1 \leq ha_{i,k} \leq \mathrm{IF}\big) \wedge \\ &\big(\mathrm{ID} - \mathrm{DA}_{i',k'} + 1 \leq ha_{i',k'} \leq \mathrm{IF}\big) \implies \\ &\qquad \big(ha_{i,k} - ha_{i',k'} - \mathrm{DA}_{i',k'} \geq \mathrm{Se}\big) \vee \\ &\qquad \big(ha_{i',k'} - ha_{i,k} - \mathrm{DA}_{i,k} \geq \mathrm{Se}\big)\end{aligned} \qquad i, i' \in \mathcal{A}; k, k' \in \mathcal{K}; i \neq i'$$

[CT 16] Minimum spacing between decoupling dates

Decoupling dates of outages from a set $\mathcal{A}$ of Type-2 power plants have to be spaced by at least Se weeks.

Constraint:

$$|ha_{i,k} - ha_{i',k'}| \geq \mathrm{Se} \qquad i, i' \in \mathcal{A}; k, k' \in \mathcal{K}; i \neq i'$$

[CT 17] Minimum spacing between coupling dates

Coupling dates of outages of a set $\mathcal{A}$ of Type-2 power plants have to be spaced by at least Se weeks.

Constraint:

$$|ha_{i,k} + \mathrm{DA}_{i,k} - ha_{i',k'} - \mathrm{DA}_{i',k'}| \geq \mathrm{Se} \qquad i, i' \in \mathcal{A}; k, k' \in \mathcal{K}; i \neq i'$$

[CT 18] Minimum spacing between decoupling and coupling dates

Decoupling dates and coupling dates of outages of a set $\mathcal{A}$ of Type-2 power plants have to be spaced by at least Se weeks.

Constraint:

$$|ha_{i,k} + \mathrm{DA}_{i,k} - ha_{i',k'}| \geq \mathrm{Se} \qquad i, i' \in \mathcal{A}; k, k' \in \mathcal{K}; i \neq i'$$

3.3.3.2 Resource Constraints

The following constraints aggregate values if a condition holds. Therefore, we first define an indicator function:

$$\mathbf{1}\big([l,h),x\big) = \begin{cases} 1 & \text{if } x \in [l,h) \\ 0 & \text{else} \end{cases}$$

[CT 19] Limited resources

Each outage of a set $\mathcal{A}$ of Type-2 power plants requires one unit of the given resource during certain weeks of its outage. All outages have to be scheduled with respect to the limited availability of the resource, i.e., there is no week w where the resource limit is exceeded.

Data:

- $\mathrm{L}_{i,k}$: First week of resource usage ($0 \leq \mathrm{L}_{i,k} < \mathrm{DA}_{i,k}$)
- $\mathrm{TU}_{i,k}$: Time of usage of the resource in weeks ($0 \leq \mathrm{TU}_{i,k}$; $\mathrm{L}_{i,k} + \mathrm{TU}_{i,k} \leq \mathrm{DA}_{i,k}$)
- Q: Available quantity of the resource

Constraint:

$$\sum_{\substack{i \in \mathcal{A} \\ k \in \mathcal{K} \\ ha_{i,k} \neq -1}} \mathbf{1}\big([ha_{i,k} + \mathrm{L}_{i,k}, ha_{i,k} + \mathrm{L}_{i,k} + \mathrm{TU}_{i,k}), w\big) \leq \mathrm{Q} \qquad w \in \mathcal{W}$$

[CT 20] Maximum number of outages during a given week

The number of outages among a set $\mathcal{A}$ of Type-2 power plants during a specific week can be limited.

Data:

- H: The considered week
- N: Maximum number of outages during this week

Constraint:

$$\sum_{\substack{i \in \mathcal{A} \\ k \in \mathcal{K} \\ ha_{i,k} \neq -1}} \mathbf{1}\big([ha_{i,k}, ha_{i,k} + \mathrm{DA}_{i,k}), \mathrm{H}\big) \leq \mathrm{N}$$

[CT 21] Maximum offline power capacity during a specific period

During the given period the aggregated maximum production capacity of all plants on outage from a set $\mathcal{A}$ can be limited.

Data:

- ID: First time step of period
- IF: Last time step of period
- IMAX: Maximum offline power capacity

Constraint:

$$\sum_{\substack{i \in \mathcal{A} \\ \bigcup_{k \in \mathcal{K}} [t^-_{i,k}, t^*_{i,k} - 1] \ni t}} \mathrm{PMAX}_i^t \leq \mathrm{IMAX} \qquad t \in \mathcal{T}; \mathrm{ID} \leq t \leq \mathrm{IF}$$

3.4 Objective Function

While satisfying all given constraints (CT 1 – 21) the sum of the following two terms is to be minimized:

- The expected production cost of all Type-1 power plants over all scenarios.
- The total refueling cost of all Type-2 power plants reduced by the expected value of residual fuel at the end of the time horizon over all scenarios.

We can formalize our objective function as follows:

$$\underbrace{\sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{K}} \mathrm{C}_{i,k} \cdot r_{i,k}}_{\text{PP2 refueling cost}} + \frac{1}{|S|} \sum_{s \in \mathcal{S}} \left(\sum_{t \in \mathcal{T}} \left(\underbrace{\sum_{j \in \mathcal{J}} \mathrm{C}_j^{s,t} \cdot p_{j,s,t} \cdot \mathrm{D}}_{\text{PP1 production cost}} \right) - \underbrace{\sum_{i \in \mathcal{I}} \mathrm{C}_{i,\mathrm{T}} \cdot x(i,s,\mathrm{T})}_{\text{PP2 residual fuel refund}} \right)$$

Note that the refueling amounts and refueling unit costs are constant in all scenarios. Contrary, the unit production costs of Type-1 power plants and residual fuel amounts of Type-2 power plants can vary between scenarios and have to be averaged.

4. Lower Bounds

In this chapter we present two lower bounds for the overall solution cost of a given dataset, which have several useful applications. For example, they can be used to evaluate the quality of found solutions and – since our considered search spaces are far too big to be fully explored – to decide when to stop the search. Furthermore, a lower bound that is fast to compute can be used to guide the search into promising regions of the solution space.

Before composing the first bound for our main problem, we develop a lower bound for Type-2 power plant unit production costs, which is presented in Section 4.1. Then, we present the first lower bound for the entire problem – utilizing a cheapest-first auction – in Section 4.2. This bound provides a vantage point for a simple and correct greedy production assignment heuristic presented later.

A more sophisticated lower bound, which also considers most of the fuel level and refueling constraints, is presented in Section 4.3. It employs a flow network to model production levels and refueling amounts.

4.1 Type-2 Power Plant Unit Production Cost

In contrast to Type-1 power plants, the production of Type-2 power plants is not charged via the produced amounts of energy but with the fuel reloaded during each outage. In this section, we set up a simple lower bound for the *unit production cost* (UPC) of each Type-2 power plant in a given dataset. This lower bound will enable us to compare the UPC of all plants and – in a next step – assign production to the cheapest plants.

There are several hurdles to overcome while transforming reloading costs into production costs:

- The initial fuel level is provided for free.
- Refueling unit costs vary with each cycle (CT 7).
- Refueling is not just adding fuel – we can lose or gain fuel for free (CT 10).
- There is a refund for residual fuel at the end of the time horizon.

For this bound we will assume that Type-2 power plants can produce in each time step, i.e., there are no scheduled outage periods. Furthermore, we relax the following constraints:

CT 6 : Maximum power after power profile imposition

CT 7 - 11 : Fuel level tracking constraints

CT 12 : Maximum modulation over a cycle

CT 13 - 21 : Outage dependency constraints

In the remainder of this section, we want to find a lower bound on the UPC c_i of a Type-2 power plant i. Therefore, we first define the total cost C_i of a plant i, which we extract from the objective function (cf. Section 3.4) as the sum of costs from all refuelings reduced by the refund for residual fuel at the end of the time horizon:

$$\mathrm{C}_i := \sum_{k \in \mathcal{K}} \mathrm{C}_{i,k} \cdot r_{i,k} - \mathrm{C}_{i,\mathrm{T}} \cdot x(i, s, \mathrm{T}) \tag{4.1}$$

We can now relate the total cost C_i to the amount of energy produced by plant i and define c_i as follows:

$$\mathrm{c}_i := \frac{\mathrm{C}_i}{\sum_{t \in \mathcal{T}} p_{i,s,t} \cdot \mathrm{D}} \tag{4.2}$$

Since all Type-2 power plant parameters (cf. Section 3.2.6) are independent of the concrete scenario, we ignore s in the following without loss of generality. To transform refueling costs into costs of production, we make an aggregate analysis starting with some helpful definitions.

First of all, we remember the auxiliary variables for the first outage time step ($t^-_{i,k}$), the first production time step ($t^*_{i,k}$) and the first time step after production ($t^+_{i,k}$) of each cycle.

Definition 11. Given an arbitrary Type-2 power plant i and a cycle k, we define the *refueling difference* $d(i,k)$ between the fuel level after refueling and the previous fuel level plus the refueling amount as:

$$d(i,k) := x(i, s, t^-_{i,k} + 1) - \Big(x(i, s, t^-_{i,k}) + r_{i,k} \Big)$$

This is a simplification of the reloading constraint CT 10, where we neglect fuel level thresholds $\mathrm{BO}_{i,*}$ and the refueling ratio $\mathrm{Q}_{i,k}$. By using $d(i,k)$, we can formulate a first lemma which holds for all scheduled cycles:

Lemma 1. Given an arbitrary Type-2 power plant i and a cycle k the amount of fuel gained from all sources (initial fuel level, amount of refueling) equals the amount of fuel delivered to all consumers (production, residual fuel level):

$$x(i, s, t^-_{i,k}) + r_{i,k} + d(i,k) = \sum_{t=t^-_{i,k}}^{t^+_{i,k}-1} p_{i,s,t} \cdot \mathrm{D} + x(i, s, t^+_{i,k}) \tag{4.3}$$

Proof. We will derive this lemma from the given constraints that influence the fuel level during a cycle. Starting with the fuel level variation during a single time step of a production campaign (CT 9), we aggregate over all production time steps of this cycle:

$$x(i, s, t^+_{i,k}) = x(i, s, t^*_{i,k}) - \sum_{t=t^*_{i,k}}^{t^+_{i,k}-1} p_{i,s,t} \cdot \mathrm{D}$$

According to CT 3, we can substitute the time step $t^*_{i,k}$ in the production sum by $t^-_{i,k}$ as there is no production in between these time steps. Similarly, corresponding to CT 10, the fuel level at $t^*_{i,k}$ equals the level at $t^-_{i,k}+1$. We get:

$$x(i,s,t^+_{i,k}) = x(i,s,t^-_{i,k}+1) - \sum_{t=t^-_{i,k}}^{t^+_{i,k}-1} p_{i,s,t} \cdot \mathrm{D}$$

By replacing the fuel level after refueling $x(i,s,t^-_{i,k}+1)$ with the simplified reloading formula from Definition 11, we gain our final equation. □

Equation 4.3 can be summarized over all cycles of a plant i to relate the fuel levels/variation with i's aggregated production. Since generally $t^+_{i,k} = t^-_{i,k+1}$, the initial and residual fuel levels of successive cycles can be canceled down. Furthermore, the initial fuel level can be set according to CT 8:

$$\mathrm{XI}_i + \sum_{k\in\mathcal{K}} r_{i,k} + \sum_{k\in\mathcal{K}} d(i,k) = \sum_{t\in\mathcal{T}} p_{i,s,t} \cdot \mathrm{D} + x(i,s,\mathrm{T}) \tag{4.4}$$

Definition 12. For each Type-2 power plant i, let $\underline{\mathrm{C}}_i$ denote the minimum refueling cost:

$$\underline{\mathrm{C}}_i := \min_k(\mathrm{C}_{i,k})$$

To get rid of varying refueling unit costs, we substitute the unit cost $\mathrm{C}_{i,k}$ of each cycle by $\underline{\mathrm{C}}_i$ in the following. This way, we get a lower bound for the total cost of plant i from Equation 4.1:

$$\mathrm{C}_i \geq \underline{\mathrm{C}}_i \sum_{k\in\mathcal{K}} r_{i,k} - \mathrm{C}_{i,\mathrm{T}} \cdot x(i,s,\mathrm{T}) \tag{4.5}$$

By replacing the refueling amounts of Equation 4.5 with Equation 4.4 (solved for the sum of refuelings) and inserting this to Equation 4.2, we can estimate c_i:

$$\mathrm{c}_i \geq \underline{\mathrm{C}}_i + \frac{\underline{\mathrm{C}}_i \left(-\mathrm{XI}_i - \sum_{k\in\mathcal{K}} d(i,k) \right) - \left(\mathrm{C}_{i,\mathrm{T}} - \underline{\mathrm{C}}_i \right) \cdot x(i,s,\mathrm{T})}{\sum_{t\in\mathcal{T}} p_{i,s,t} \cdot \mathrm{D}} \tag{4.6}$$

We assume a non-negative numerator in Equation 4.6, thereby ignoring all degenerated cases where a plant has negative total cost C_i. Given upper bounds for the sum of refueling differences $d(i,k)$, the sum of production levels $p_{i,s,t}$ and the amount of residual fuel $x(i,s,\mathrm{T})$, we have found a computable lower bound for c_i.

Lemma 2. The residual fuel level can be limited by:

$$x(i,s,\mathrm{T}) \leq \max\left(\mathrm{XI}_i, \max_k(\mathrm{SMAX}_{i,k})\right)$$

Proof. In the beginning the fuel level equals XI_i (see CT 8). During any following campaign, the fuel level never exceeds the maximum fuel level allowed after refueling of this campaign because the fuel level is monotonically decreasing during production according to CT 4 and CT 9. Hence, the residual fuel level cannot exceed the maximum of these upper bounds. □

Lemma 3. The sum of production can be limited by:

$$\sum_{t \in \mathcal{T}} p_{i,s,t} \leq \sum_{t \in \mathcal{T}} \mathrm{PMAX}_i^t$$

Proof. This follows directly from CT 4 when assuming maximum production in each time step. □

Lemma 4. The sum of refueling differences can be limited by:

$$\sum_{k \in \mathcal{K}} d(i,k) \leq \max_{\kappa \in \mathcal{K}} \left(\mathrm{BO}_{i,\kappa} + \left(\sum_{k=0}^{\kappa-1} \frac{\mathrm{BO}_{i,k}}{\mathrm{Q}_{i,k+1}} \right) - \frac{\mathrm{Q}_{i,0} - 1}{\mathrm{Q}_{i,0}} \mathrm{BO}_{i,-1} \right)$$

Proof. Remember the refueling equation from CT 10:

$$x(i,s,t_{i,k}^- + 1) = \frac{\mathrm{Q}_{i,k} - 1}{\mathrm{Q}_{i,k}} \left(x(i,s,t_{i,k}^-) - \mathrm{BO}_{i,k-1} \right) + r_{i,k} + \mathrm{BO}_{i,k} \tag{4.7}$$

We can insert this into Definition 11 to approximate the amount of fuel gained or lost during refueling in the given outage:

$$d(i,k) = \mathrm{BO}_{i,k} - \frac{\mathrm{Q}_{i,k} - 1}{\mathrm{Q}_{i,k}} \mathrm{BO}_{i,k-1} - \frac{1}{\mathrm{Q}_{i,k}} x(i,s,t_{i,k}^-)$$

Assuming that there is no fuel left before refueling, we get an upper bound for $d(i,k)$:

$$d(i,k) \leq \mathrm{BO}_{i,k} - \frac{\mathrm{Q}_{i,k} - 1}{\mathrm{Q}_{i,k}} \mathrm{BO}_{i,k-1}$$

Note that $d(i,k)$ might be negative. We now aggregate $d(i,k)$ for the first κ cycles utilizing a telescoping series:

$$\sum_{0 \leq k < \kappa} d(i,k) = \mathrm{BO}_{i,\kappa} + \left(\sum_{k=0}^{\kappa-1} \frac{\mathrm{BO}_{i,k}}{\mathrm{Q}_{i,k+1}} \right) - \frac{\mathrm{Q}_{i,0} - 1}{\mathrm{Q}_{i,0}} \mathrm{BO}_{i,-1}$$

Since we do not consider any outages here, we cannot determine how many cycles of each plant will be exactly scheduled and as $d(i,k)$ might be negative, the sum over all cycles is probably not the maximum. In order to limit the sum of $d(i,k)$, we choose the maximum reached when only scheduling the first κ cycles. □

By inserting the results of the Lemmas 2, 3 and 4 into Equation 4.6, we get the sought lower bound of the unit production cost c_i of a Type-2 power plant i.

Theorem 1. For an arbitrary Type-2 power plant i we can calculate a lower bound of the unit production cost c_i as

$$\mathrm{c}_i \geq \underline{\mathrm{C}}_i - \frac{\underline{\mathrm{C}}_i \left(\mathrm{XI}_i + \Delta_i \right) + \left(\mathrm{C}_{i,\mathrm{T}} - \underline{\mathrm{C}}_i \right) \cdot \max\left(\mathrm{XI}_i, \max_k (\mathrm{SMAX}_{i,k}) \right)}{\sum_{t \in \mathcal{T}} \mathrm{PMAX}_i^t \cdot \mathrm{D}}$$

where $\underline{\mathrm{C}}_i$ is the minimum refueling cost of i and Δ_i is defined as

$$\Delta_i = \max_{\kappa \in \mathcal{K}} \left(\mathrm{BO}_{i,\kappa} + \left(\sum_{k=0}^{\kappa-1} \frac{\mathrm{BO}_{i,k}}{\mathrm{Q}_{i,k+1}} \right) - \frac{\mathrm{Q}_{i,0} - 1}{\mathrm{Q}_{i,0}} \mathrm{BO}_{i,-1} \right)$$

4.2 An Auction-Based Lower Bound

In this section we present a first lower bound for the objective function (cf. Section 3.4) of a given dataset. It will allow a first (and fast) evaluation of the quality of the found solutions and is the source of a valid production assignment algorithm presented in Section 5.2.

The basic idea is that all plants emit offers of production capacity (and its cost) for each scenario and time step, while we assign production levels in a greedy way to the cheapest plants. We relax the imposed power profile constraint (CT 6), as well as all fuel level tracking (CT 7 - 12) and outage scheduling constraints (CT 13 - 21). Since the unit production costs of the Type-2 power plants is not explicitly given, we will use the lower bound of the unit production cost of each plant presented in Section 4.1.

Definition 13. The gross production amounts (as an interval [pmin, pmax]) and unit cost offered by all plants are defined in the following table:

	Type-1 power plant o	Type-2 power plant o
pmin_o	$\mathrm{PMIN}_o^{s,t}$	0
pmax_o	$\mathrm{PMAX}_o^{s,t}$	PMAX_o^{t}
cost_o	$\mathrm{C}_o^{s,t}$	c_i

Subtracting already assigned production levels gives the net amount of production that can be offered. Since we relax all fuel tracking constraints, we can run the auction for each time step of each scenario independently. Algorithm 1 presents such a single auction consisting of two steps. First, the minimum required production level of each plant is assigned. Afterwards we assign as much production as possible to the cheapest plants until the demand is covered.

```
Algorithm 1: AUCTION (scenario $s$, time step $t$)
Output: Total cost of assigned production for the given scenario and time step
begin
    $demand \longleftarrow \mathrm{DEM}^{s,t}$
    foreach plant $o$ do
        $p_{o,s,t} \longleftarrow \mathrm{pmin}_o$
        $demand \longleftarrow demand - \mathrm{pmin}_o$
    foreach plant $o$ ascending by $\mathrm{cost}_o$ do
        $produce \longleftarrow \min(demand, \mathrm{pmax}_o - p_{o,s,t})$
        $p_{o,s,t} \longleftarrow p_{o,s,t} + produce$
        $demand \longleftarrow demand - produce$
    return $\sum_{j\in\mathcal{J}} \mathrm{cost}_j \cdot p_{j,s,t} \cdot \mathrm{D} + \sum_{i\in\mathcal{I}} \mathrm{cost}_i \cdot p_{i,s,t} \cdot \mathrm{D}$
end
```

Lemma 5. Given an arbitrary scenario s and a time step t, then Algorithm 1 finds a cheapest possible production assignment with respect to CT 1, 2 and 4.

Proof. The conformance with CT 1,2 and 4 can easily be shown by using invariants on Algorithm 1. We will proof the remaining statement by contradiction, assuming that Algorithm 1 produced an assignment p, but there exists a cheaper assignment p'. In this case, there have to be two plants φ and φ' whose production in p and p' differs and which have different unit costs as follows:

$$p_{\varphi,s,t} > {p'}_{\varphi,s,t} \tag{4.8}$$

$$p_{\varphi',s,t} < {p'}_{\varphi',s,t} \tag{4.9}$$

$$cost_{\varphi'} < cost_{\varphi} \tag{4.10}$$

Since both solutions have to meet the minimum offers, Algorithm 1 must have assigned different values in the maximum offer step. According to Equation 4.10, the cheaper plant φ' is processed before plant φ. From Equation 4.9 follows that $p_{\varphi',s,t}$ is strictly less than the maximum amount offered by φ', which implies that the assigned production level was bounded by the demand and all more expensive plants (especially φ) do not get any production assigned. But, this is a contradiction to Equation 4.8 because $p_{\varphi,s,t}$ has to be strictly greater than 0. □

We can now execute Algorithm 1 for all scenarios and time steps, which yields a cheapest production plan for our model without outages and fuel tracking. Algorithm 2 does this and returns the average total cost of production per scenario.

Algorithm 2: AUCTIONS

```
Output: Average cost of assigned production across all scenarios
begin
    $cost \longleftarrow 0$
    foreach $s \in \mathcal{S}$ do
        foreach $t \in \mathcal{T}$ do
            $cost \longleftarrow cost + \text{AUCTION}(s,t)$
    return $\frac{cost}{|\mathcal{S}|}$
end
```

Lemma 6. Given production intervals from Definition 13, then Algorithm 2 returns a lower bound of the objective function's value for a given dataset.

Proof. From Lemma 5 we can assume that Algorithm 2 aggregates lower bound production assignments for each time step and thus finds a global cheapest production assignment for the offers of Definition 13. We now want to show that offers of Definition 13 also form a lower bound according to the original objective function (from Section 3.4). This is done by formalizing the result of Algorithm 2 and transforming it into the original objective function:

$$\begin{aligned}
&\frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\left(\sum_{t\in\mathcal{T}}\left(\sum_{j\in\mathcal{J}}\mathrm{C}_j^{s,t}\,p_{j,s,t}\,\mathrm{D}+\sum_{i\in\mathcal{I}}\mathrm{c}_i\,p_{i,s,t}\,\mathrm{D}\right)\right)\\
=&\frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\left(\sum_{t\in\mathcal{T}}\left(\sum_{j\in\mathcal{J}}\mathrm{C}_j^{s,t}\,p_{j,s,t}\,\mathrm{D}\right)+\sum_{i\in\mathcal{I}}\left(\mathrm{c}_i\sum_{t\in\mathcal{T}}p_{i,s,t}\,\mathrm{D}\right)\right)\\
\overset{\text{Eq 4.1/4.2}}{\leq}&\frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\left(\sum_{t\in\mathcal{T}}\left(\sum_{j\in\mathcal{J}}\mathrm{C}_j^{s,t}\,p_{j,s,t}\,\mathrm{D}\right)+\sum_{i\in\mathcal{I}}\left(\sum_{k\in\mathcal{K}}\mathrm{C}_{i,k}\,r_{i,k}-\mathrm{C}_{i,\mathrm{T}}\,x(i,s,\mathrm{T})\right)\right)\\
=&\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}}\mathrm{C}_{i,k}\,r_{i,k}+\frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\left(\sum_{t\in\mathcal{T}}\left(\sum_{j\in\mathcal{J}}\mathrm{C}_j^{s,t}\,p_{j,s,t}\,\mathrm{D}\right)-\sum_{i\in\mathcal{I}}\mathrm{C}_{i,\mathrm{T}}\,x(i,s,\mathrm{T})\right)
\end{aligned}$$

□

Since we neither care for outages nor for fuel levels, this bound is not as close to real solutions as one might wish and we will present a more sophisticated bound in the next section.

4.3 A Flow-Based Lower Bound

In this section, we present a flow network, which models outage restrictions as well as fuel sources and consumption to deduce a lower bound of the objective function for a given dataset. The network is instantiated for each scenario of a dataset. We first calculate a lower bound on the cost of each scenario. In the end, the average cost over all scenarios provides a lower bound for the production cost of a dataset.

We first present some basic intuition about the structure of our network. After that, we explain some specialties of not scheduled cycles. Next, we give a formal definition of the network and, in the last part of this section, we prove the lower bound property of a minimum cost flow in our network.

The previously shown auction-based bound has two major drawbacks, which we overcome using the flow-based bound:

1. Unit production costs of Type-2 power plants are strongly underestimated. This is not a big problem when distributing demand (because all Type-2 power plants are nearly equally underestimated) but clearly distorts any lower bound.
2. Type-2 power plant production capacity is overestimated, because we neither care for outage scheduling nor restrict the amount of fuel consumed over time.

4.3.1 Intuition

The network consists of two logical parts (see Figure 4.1). One part handles the distribution of demand levels to the power plants, a simple matching approach. The other part models the internals of Type-2 power plants, more precisely the sources of fuel which is consumed for production. The networks commodity is fuel, i.e., production levels and demands have to be multiplied by their duration (see CT 9). Arcs are annotated with minimum and maximum capacities, as well as the cost per unit of flow.

In each time step, there is an amount of fuel required to cover the demand level. The demand is distributed among all facilities. Since costs of Type-1 power plants are charged with the produced amounts of energy, an arc between the time step and the Type-1 power plant can capture this cost.

To charge the production of a Type-2 power plant, we have to map the consumed fuel to its source. In the second part of the network, we first map any production of a Type-2 power plant i onto a cycle of i that may be active at the given time step. From a cycle's point of view, there are two sources of fuel: the residual fuel of the previous cycle and the refueling done during the cycle's outage. To model the residual fuel in our network, we introduce an arc with negative cost (the refund) from the source directly into the last cycle of each Type-2 power plant. For technical reasons, we add a bypass arc of zero cost. Its capacity equals the sum of capacities from all residual arcs. This way, we get a minimum cost flow problem to solve.

For this bound, we relax the following constraints:

CT 6: We only track the fuel levels before and after each production campaign. Since we do not determine the fuel levels for each time step in between exactly, we cannot reduce the maximum production capacity during time steps where an imposed power profile would apply.

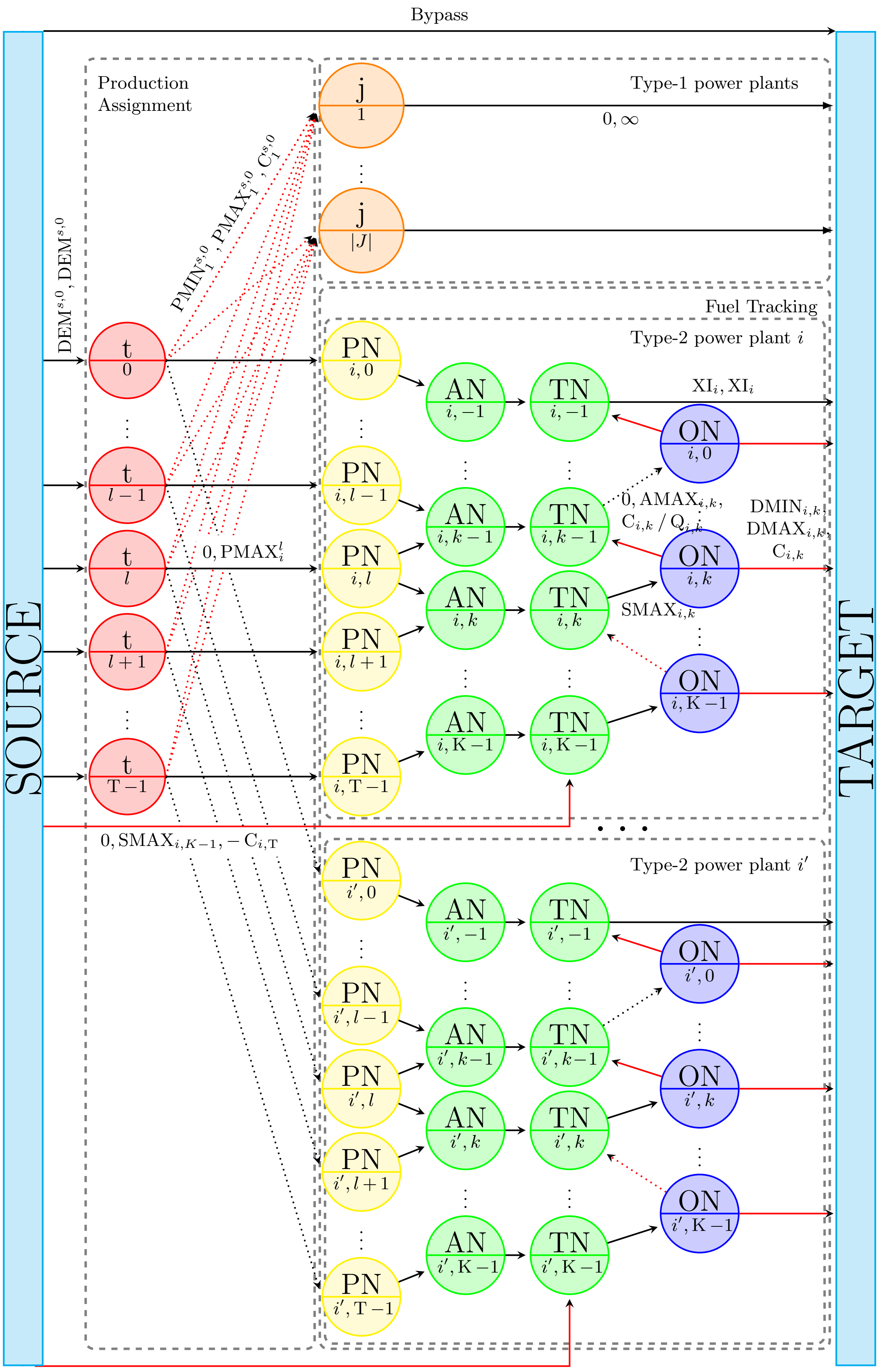

Figure 4.1: Sketch of the flow network: The left part performs the production assignment, while the lower right parts assign the fuel consumed by Type-2 power plants to a potential cycle, where it is charged. Only red arcs incur costs.

CT 14 - 21: All dependencies between outages of different plants are ignored.

In our flow network we cannot handle the rather complex refueling formula (see CT 10) directly. Therefore, to model a weaker form of this restriction, we define the simpler *reloading difference*, which can be embedded in the network.

Definition 14. Given an arbitrary Type-2 power plant i and a cycle k, we define the *reloading difference* $\delta(i,s,k)$ between the fuel level after refueling and the previous fuel level as:

$$\delta(i,s,k) := x(i,s,t^-_{i,k}+1) - x(i,s,t^-_{i,k})$$

Lemma 7. For an arbitrary Type-2 power plant i and cycle k, the reloading difference can be limited to $\delta(i,s,k) \in [\mathrm{DMIN}_{i,k}, \mathrm{DMAX}_{i,k}]$, where the interval is defined as follows:

$$[\mathrm{DMIN}_{i,k}, \mathrm{DMAX}_{i,k}] := \left[\mathrm{RMIN}_{i,k} - \frac{\mathrm{AMAX}_{i,k}}{\mathrm{Q}_{i,k}}, \mathrm{RMAX}_{i,k}\right] + \mathrm{BO}_{i,k} - \frac{\mathrm{Q}_{i,k}-1}{\mathrm{Q}_{i,k}}\mathrm{BO}_{i,k-1}$$

Proof. We derive this from the refueling constraint CT 10 and Definition 14, relating both by $x(i,s,t^-_{i,k}+1)$:

$$\delta(i,s,k) = r_{i,k} - \frac{1}{\mathrm{Q}_{i,k}}x(i,s,t^-_{i,k}) - \frac{\mathrm{Q}_{i,k}-1}{\mathrm{Q}_{i,k}}\mathrm{BO}_{i,k-1} + \mathrm{BO}_{i,k} \tag{4.11}$$

We get the minimum of $\delta(i,s,k)$ by performing minimum refueling ($r_{i,k} = \mathrm{RMIN}_{i,k}$, see CT 7) at the maximum possible fuel level ($x(i,s,t^-_{i,k}) = \mathrm{AMAX}_{i,k}$, see CT 11). Analogously the maximum is reached by setting $r_{i,k} = \mathrm{RMAX}_{i,k}$ and $x(i,s,t^-_{i,k}) = 0$. □

Note that we will only consider the reloading difference in our network and do not care explicitly for the bounds of refueling. As a result, the concrete amount of charged fuel is hidden inside the reloading difference and with it the exact cost of the refueling operation. We can overcome this problem with the following lemma.

Lemma 8. For an arbitrary Type-2 power plant i and a cycle k, we can calculate the exact cost of refueling, given the reloading difference $\delta(i,s,k)$ and the fuel level before refueling $x(i,s,t^-_{i,k})$ as:

$$\underbrace{\frac{\mathrm{C}_{i,k}}{\mathrm{Q}_{i,k}}\cdot x(i,s,t^-_{i,k})}_{\text{fuel level}} + \underbrace{\mathrm{C}_{i,k}\cdot\delta(i,s,k)}_{\text{reloading difference}} + \underbrace{\mathrm{C}_{i,k}\cdot\left(\frac{\mathrm{Q}_{i,k}-1}{\mathrm{Q}_{i,k}}\mathrm{BO}_{i,k-1} - \mathrm{BO}_{i,k}\right)}_{\text{global cost offset}}$$

While the first two factors can be embedded in the flow network, the latter is just a global cost, which can be handled in a post-processing step.

Proof. According to the objective function, reloaded fuel is charged by $\mathrm{C}_{i,k}\cdot r_{i,k}$. Replacing $r_{i,k}$ by Equation 4.11 yields the final formula. □

Besides the refueling issues, there are two constraints whose compliance we cannot fully enforce in our flow network, especially because the outage dates are not fixed yet. Nevertheless, we will model these constraints to improve the bound:

CT 3: As the outage dates are not determined yet, it is unknown during which time steps a power plant is shut down. Therefore, we focus on each cycle and its earliest and latest possible production time step. Remember that to estimate the cost, it is only important during which cycle the consumed fuel was provided. So, we allow the fuel consumed during a time step to be gathered from all cycles that might be active during this step, thereby indirectly modeling this constraint if no cycle is available for the respective time step.

CT 12: Remember that coupling dates are uncertain and this constraint applies in each cycle only until the fuel level threshold $\mathrm{BO}_{i,k}$ is under-run. Furthermore, fuel levels are also not determined at the time of instantiation of our flow network. Therefore, we can only apply very weak limits to the amount of production for each cycle. We get the minimum production, when assuming the shortest production campaign length (latest start and earliest end) while exhausting the available maximum modulation $\mathrm{MMAX}_{i,k}$ completely. Analogously, the maximum is reached in a long-lasting production campaign with no modulation.

4.3.2 Handling Non-Scheduled Cycles

In this section we address several issues that arise from postponed cycles in our flow network. At building time of a concrete flow network instance, the number of scheduled cycles performed by a power plant in a final (and maybe optimal) solution is unknown. But, since parts of the network will be set up per cycle, we have to take care that the cost induced by modelled but postponed cycles does not destroy the lower bound property. Therefore, we want to group cycles with similar characteristics (considering their possible decoupling dates). We identify three categories of cycles:

Mandatory cycles have to be scheduled in any valid solution. This generally holds for cycles with a last possible decoupling date set explicitly, i.e., $\mathrm{TA}_{i,k} \neq -1$.

Optional cycles might, but need not, be scheduled in a solution. We can identify them by their earliest possible decoupling date $\mathrm{TO}_{i,k} < \mathrm{W}$ and an unset latest possible decoupling date: $\mathrm{TA}_{i,k} = -1$ (see CT 13).

Impossible cycles cannot be scheduled in any valid solution. They can be characterized by their earliest possible decoupling date being past the time horizon ($\mathrm{TO}_{i,k} \geq \mathrm{W}$). This might be stated as a dataset parameter, but is more likely not to be revealed until we perform a basic reasoning (cf. Section 5.1) on the given dataset.

Definition 15. For an arbitrary Type-2 power plant i let K_i^{min} denote the number of the last mandatory cycle and K_i^{max} denote number of the last optional cycle.

When building a concrete network instance, we omit all impossible cycles and model only the cycles up to K_i^{max} of each power plant. Furthermore, optional cycles will have to be modelled in a special way that preserves the lower bound property if such a cycle is not scheduled in the optimal solution. In Lemma 8 we identified three components which assemble the refueling cost of Type-2 power plants in our network. These cost factors have to be designed to form a lower bound, no matter how many of the optional cycles are scheduled in a solution:

Fuel Level: This factor models the opportunity cost of lost (or not gained free) fuel during an outage. Since there is no such loss for not scheduled cycles and this factor is always positive, we cannot insert these cost for optional cycles into the network.

Furthermore, according to our network, the final fuel level of a Type-2 power plant might have to pass several optional cycles. Hence, we have to relax the maximum fuel level before and after refueling (see CT 11) of each optional cycle and allow a capacity range up to the maximum achievable fuel level until this time step, instead of $\mathrm{AMAX}_{i,k}$ and $\mathrm{SMAX}_{i,k}$. We define appropriate parameters for all optional cycles:

$$\mathrm{AMAX}'_{i,k} := \begin{cases} \mathrm{XI}_i & \text{if } k = 0 \\ \max(\mathrm{AMAX}_{i,k}, \mathrm{SMAX}_{i,k-1}) & \text{if } k = \mathrm{K}_i^{min} + 1 \\ \max(\mathrm{AMAX}_{i,k}, \mathrm{SMAX}'_{i,k-1}) & \text{else} \end{cases} \tag{4.12}$$

$$\mathrm{SMAX}'_{i,k} := \max(\mathrm{AMAX}'_{i,k}, \mathrm{SMAX}_{i,k}) \tag{4.13}$$

Reloading Difference: As there will be no refueling in a not scheduled outage, there would be no extra cost when charging the refueling amount of an optional outage. But such a solution might not satisfy the bounds of refueling. Therefore, we relax CT 7 for optional cycles to: $0 \leq r_{i,k} \leq \mathrm{RMAX}_{i,k}$. Furthermore, we adapt the bounds of the reloading difference $\delta(i,s,k)$:

$$[\mathrm{DMIN}'_{i,k}, \mathrm{DMAX}'_{i,k}] := [\min(0, \mathrm{DMIN}_{i,k}), \mathrm{DMAX}_{i,k}]$$

Global Cost: These offsets (one per cycle) enable us to model the rather complicated refueling formula in a simple way. Costs might be positive or negative values. We choose the minimum subsum when scheduling any possible number of optional cycles, as the real global cost of a power plant i:

$$\mathrm{cost}_i := \min_{\mathrm{K}_i^{min} \leq \kappa \leq \mathrm{K}_i^{max}} \left(\sum_{k=0}^{\kappa} \mathrm{C}_{i,k} \cdot \left(\frac{\mathrm{Q}_{i,k} - 1}{\mathrm{Q}_{i,k}} \mathrm{BO}_{i,k-1} - \mathrm{BO}_{i,k} \right) \right)$$

4.3.3 Formal Network Definition

In this section we formally define the flow network $N = (G, cap, \text{source}, \text{target})$ shown in Figure 4.1. It consists of a directed graph $G = (V, E)$, arc capacities $cap : E \to [\mathbb{R}^{\geq 0}, \mathbb{R}^{\geq 0}]$ and arc cost function $cost : V \times V \to \mathbb{R}$. Besides the source and target nodes, a node for each time step $t \in \mathcal{T}$ and a node for each Type-1 power plant $j \in \mathcal{J}$, there are several categories of nodes:

- A *production node* $PN_{i,t}$ exists for each Type-2 power plant i and time step t to limit the plant's production during this time step:

$$V_{PP2} \mathrel{\widehat{=}} \{PN_{i,t} | i \in \mathcal{I}; t \in \mathcal{T}\}$$

- An *aggregation node* $AN_{i,k}$ exists per Type-2 power plant i and cycle k (including the initial cycle as -1), aggregating the production during this cycle:

$$V_{AGG} \mathrel{\widehat{=}} \{AN_{i,-1} | i \in \mathcal{I}\} \cup \{AN_{i,k} | i \in \mathcal{I}; k \in \mathcal{K}\}$$

- A *transfer node* $TN_{i,k}$ exists per Type-2 power plant i and cycle k (including the initial cycle as -1), to connect fuel levels to the previous and next cycles:

$$V_{TRA} \mathrel{\widehat{=}} \{TN_{i,-1} | i \in \mathcal{I}\} \cup \{TN_{i,k} | i \in \mathcal{I}; k \in \mathcal{K}\}$$

- An *outage node* $ON_{i,k}$ will connect two successive campaigns of a Type-2 power plant i, to perform refueling operations:

$$V_{OUT} \mathrel{\widehat{=}} \{ON_{i,k} | i \in \mathcal{I}; k \in \mathcal{K}\}$$

The set of nodes can be written as:

$$V = \{\text{source}; \text{target}\} \uplus \mathcal{T} \uplus \mathcal{J} \uplus V_{PP2} \uplus V_{AGG} \uplus V_{TRA} \uplus V_{OUT}$$

In our network, arcs generally connect different node groups to distribute fuel from the source, via nodes representing time steps, production, aggregation, transfer and outages, to the target node. We now introduce the different arc categories and define the capacity function *cap* for each category. Note that a lot of categories have a minimum required flow.

- For each time step t of the given scenario an arc carries the production demand:

$$\begin{aligned} E_{DEM} &= \{(\text{source}, t) | t \in \mathcal{T}\} \\ cap(e) &= [\text{DEM}^{s,t} \cdot \text{D}, \text{DEM}^{s,t} \cdot \text{D}] && e \in E_{DEM} \end{aligned}$$

- For each time step t, an arc carries the assigned production level of Type-1 power plant j:

$$\begin{aligned} E_{PP1} &= \{(t, j) | t \in \mathcal{T}; j \in \mathcal{J}\} \\ cap(e) &= [\text{PMIN}_j^{s,t} \cdot \text{D}, \text{PMAX}_j^{s,t} \cdot \text{D}] && e \in E_{PP1} \end{aligned}$$

- For each Type-1 power plant j an arc hauls the aggregated production to the target:

$$\begin{aligned} E_{PPT} &= \{(j, \text{target}) | j \in \mathcal{J}\} \\ cap(e) &= [0, \infty) && e \in E_{PPT} \end{aligned}$$

- For each time step t and Type-2 power plant i an arc carries the assigned production:

$$\begin{aligned} E_{PP2} &= \{(t, i) | PN_{i,t} \in V_{PP2}; t' \in \mathcal{T} : t = t'\} \\ cap(e) &= [0, \text{PMAX}_i^t \cdot \text{D}] && e \in E_{PP2} \end{aligned}$$

- For each production node $PN_{i,t}$ of a Type-2 power plant i there are arcs to all aggregation nodes $AN_{i,k}$, whose cycle k might be active at that time step. As the decoupling dates might still be uncertain when we build the flow network, let us denote $\min(t^*_{i,k})$ as the first possible coupling date of cycle k of power plant i and $\max(t^+_{i,k})$ as the latest possible decoupling date of the following cycle.

$$\begin{aligned} E_{AGG} &= \{(PN_{i,t}, AN_{i,k}) | PN_{i,t} \in V_{PP2}; AN_{i,k} \in V_{AGG} : \min(t^*_{i,k}) \le t < \max(t^+_{i,k})\} \\ cap(e) &= [0, \infty) && e \in E_{AGG} \end{aligned}$$

- For each aggregation node $AN_{i,k}$ there is an arc to the corresponding transfer node $TN_{i,k}$, which limits the production over the cycle:

$$\begin{aligned} E_{LIM} &= \{(AN_{i,k}, TN_{i,k}) | AN_{i,k} \in V_{AGG}; TN_{i,k} \in V_{TRA}\} \\ cap(e) &= [0, \infty) \cap \left([-\text{MMAX}_{i,k}, 0] + \sum_{t=t^*_{i,k}}^{t^+_{i,k}-1} \text{PMAX}_i^t \right) && e \in E_{LIM} \end{aligned}$$

- Each outage node $ON_{i,k}$ is connected to the transfer node $TN_{i,k-1}$ of the previous cycle (to receive the fuel level before refueling):

$$\begin{aligned} E_{AMAX} &= \{(ON_{i,k}, TN_{i,k}) | ON_{i,k} \in V_{OUT}; TN_{i,k-1} \in V_{TRA}\} \\ cap(e) &= \begin{cases} [0, \text{AMAX}_{i,k}] & \text{if } k \le \text{K}_i^{min} \\ [0, \text{AMAX}'_{i,k}] & \text{else} \end{cases} && e \in E_{AMAX} \end{aligned}$$

- Each transfer node is connected to the outage node of the same cycle (to receive the fuel level after refueling):

$$\begin{aligned} E_{SMAX} &= \{(TN_{i,k}, ON_{i,k}) | TN_{i,k} \in V_{TRA}; ON_{i,k} \in V_{OUT}\} \\ cap(e) &= \begin{cases} [0, \text{SMAX}_{i,k}] & \text{if } k \le \text{K}_i^{min} \\ [0, \text{SMAX}'_{i,k}] & \text{else} \end{cases} && e \in E_{SMAX} \end{aligned}$$

- Initial fuel levels are injected from the *target* into the initial cycle ($k = -1$) of each Type-2 power plant:

$$\begin{aligned} &E_{INIT} = \{(i, \text{target}) | TN_{i,-1} \in V_{TRA}\} \\ &cap(e) = [\mathrm{XI}_i, \mathrm{XI}_i] \qquad e \in E_{INIT} \end{aligned}$$

- The reloading difference of each outage flows from each *outage node* into the *target*:

$$\begin{aligned} &E_{REF} = \{(ON_{i,k}, \text{target}) | ON_{i,k} \in V_{OUT}\} \\ &cap(e) = \begin{cases} [\mathrm{DMIN}_{i,k}, \mathrm{DMAX}_{i,k}] & \text{if } k \leq \mathrm{K}_i^{min} \\ [\mathrm{DMIN}'_{i,k}, \mathrm{DMAX}'_{i,k}] & \text{else} \end{cases} \qquad e \in E_{REF} \end{aligned}$$

- Residual fuel levels are provided from the *source* into the last cycle ($k = \mathrm{K} - 1$) of each Type-2 power plant:

$$\begin{aligned} &E_{RES} = \{(\text{source}, TN_{i,k}) | TN_{i,k} \in V_{TRA} : k = \mathrm{K} - 1\} \\ &cap(e) = cap((TN_{i,\mathrm{K}-1}, ON_{i,\mathrm{K}-1})) \qquad e \in E_{RES} \end{aligned}$$

- To be able to fix the amount of flow through the network, we introduce a bypass arc. This arc carries all fuel which is not present in any Type-2 power plant at the end of the time horizon:

$$\begin{aligned} e_{BYP} &= \{\text{source}, \text{target}\} \\ cap(e_{BYP}) &= \sum_{e \in E_{RES}} cap(e) \end{aligned}$$

The set of arcs now consists of a disjoint union of all the categories above.

Finally, we need to define our arc cost function *cost* for the given arc groups:

- Type-1 power plant production cost is applied to the corresponding production arc:

$$cost(t, j) = \mathrm{C}_j^{s,t} \qquad (t, j) \in E_{PP1}$$

- Type-2 power plant refueling cost is applied to the corresponding reloading arc:

$$cost(ON_{i,k}, \text{target}) = \mathrm{C}_{i,k} \qquad (ON_{i,k}, \text{target}) \in E_{REF}$$

- Additional cost for refueling (see Lemma 8) is applied to the corresponding fuel level before refueling arc:

$$cost(ON_{i,k}, TN_{i,k}) = \begin{cases} \mathrm{C}_{i,k} \,/\, \mathrm{Q}_{i,k} & \text{if } k \leq \mathrm{K}_i^{min} \\ 0 & \text{else} \end{cases} \qquad (ON_{i,k}, TN_{i,k}) \in E_{AMAX}$$

- The refund for residual fuel is applied to the residual arc of each Type-2 power plant:

$$cost(\text{source}, TN_{i,k}) = -\,\mathrm{C}_{i,\mathrm{T}} \qquad (\text{source}, TN_{i,k}) \in E_{RES}$$

All other arc categories have zero costs. The global cost offset for each scheduled cycle (see Lemma 8) can be added in a post-processing step.

4.3.4 Proof of the Lower Bound

In the following we prove the lower bound property of the presented flow network. This is done in two steps. First, we show how a given solution can be transformed into a network flow. Afterwards, we analyze the cost of a fixed flow in the network and show that this does not exceed the value of the objective function of the corresponding solution.

Embedding a given solution into our flow network is almost straightforward. We remember that a solution contains the exact production levels of all power plants during all time steps ($p_{j,s,t}$ and $p_{i,s,t}$) as well as the decoupling dates ($ha_{i,k}$) and refueling amounts ($r_{i,k}$) for all outages. From these values we can deduce the fuel levels $x(i,s,t)$ of all Type-2 power plants for all time steps (see CT 8 – 10).

We describe the embedding of a given solution by the induced flow amounts $f : E \rightarrow \mathbb{R}^{\geq 0}$ for each arc type:

- flow on demand arcs E_{DEM} can be derived from the dataset parameters:
$$f(\text{source}, t) = \text{DEM}^{s,t} \cdot \text{D} \qquad\qquad t \in \mathcal{T}$$
- production of Type-1 power plants is directly assigned to the E_{PP1} arcs:
$$f(t, j) = p_{j,s,t} \cdot \text{D} \qquad\qquad j \in \mathcal{J}; t \in \mathcal{T}$$
- the sum of production per Type-1 power plant is assigned to the E_{PPT} arcs:
$$f(j, \text{target}) = \sum_{t \in \mathcal{T}} p_{j,s,t} \cdot \text{D} \qquad\qquad j \in \mathcal{J}$$
- production of Type-2 power plants is assigned to the arc of the time step:
$$f(t, PN_{i,t}) = p_{i,s,t} \cdot \text{D} \qquad\qquad i \in \mathcal{I}; t \in \mathcal{T}$$
- among the aggregation arcs E_{AGG} only those carry flow, which connect a production node to the active cycle of the time step:
$$f(PN_{i,t}, AN_{i,k}) = \begin{cases} p_{i,s,t} \cdot \text{D} & \text{if } t^*_{i,k} \leq t < t^+_{i,k} \\ 0 & \text{else} \end{cases} \qquad\qquad i \in \mathcal{I}; k \in \mathcal{K}; t \in \mathcal{T}$$
- the sum of production per cycle of a Type-2 power plant is assigned to the E_{LIM} arcs:
$$f(AN_{i,k}, TN_{i,k}) = \sum_{t=t^*_{i,k}}^{t^+_{i,k}-1} p_{i,s,t} \cdot \text{D} \qquad\qquad i \in \mathcal{I}; k \in \mathcal{K}$$
- flow on E_{AMAX} arcs corresponds to the cycle's fuel level before refueling:
$$f(ON_{i,k}, TN_{i,k-1}) = x(i, s, t^-_{i,k}) \qquad\qquad i \in \mathcal{I}; k \in \mathcal{K}$$
- flow on E_{SMAX} arcs corresponds to the cycle's fuel level after refueling:
$$f(TN_{i,k}, ON_{i,k}) = x(i, s, t^*_{i,k}) \qquad\qquad i \in \mathcal{I}; k \in \mathcal{K}$$
- flow on initial fuel arcs E_{INIT} can be derived from dataset parameters:
$$f(TN_{i,-1}, \text{target}) = \text{XI}_i \qquad\qquad i \in \mathcal{I}$$

- we derive the reloading difference E_{REF} from the pre- and post-refueling fuel levels:

$$f(ON_{i,k}, \text{target}) = x(i, s, t^-_{i,k} + 1) - x(i, s, t^-_{i,k}) \qquad i \in \mathcal{I}; k \in \mathcal{K}$$

- the final fuel levels are put on the residual arcs E_{RES}:

$$f(\text{source}, TN_{i,\mathrm{K}-1}) = x(i, s, \mathrm{T}) \qquad i \in \mathcal{I}$$

- to fit the fixed flow, all fuel that is not pushed along the residual arcs flows on our bypass arc:

$$f(\text{source}, \text{target}) = \sum_{i\in\mathcal{I}} \max(cap((source, TN_{i,\mathrm{K}-1}))) - x(i, s, \mathrm{T})$$

Lemma 9. A given solution implements a valid flow in the flow network.

Sketch of proof. We show that flow conservation ($\forall u : \sum_{v\in V} f(u, v) = \sum_{v\in V} f(v, u)$) holds for all nodes and the capacity constraints ($\min(e) \leq f(e) \leq \max(e)$) hold for all arcs. We begin with the flow conservation for each node category:

- **Time step t:** Follows directly from CT 1, which has to hold for a valid solution.
- **Type-1 power plant j:** Follows from the definition of the outgoing flow.
- **Production node $PN_{i,t}$:** Follows from the definition of the outgoing flow, as there is only one k, such that $t^*_{i,k} \leq t < t^+_{i,k}$ (i.e., the cycle is well defined). Note that there is no production between $t^-_{i,k}$ and $t^*_{i,k}$ (see CT 3).
- **Aggregation node $AN_{i,k}$:** Follows from the definition of the outgoing flow.
- **Transfer node $TN_{i,k}$:** Follows from CT 9, which has to hold for a valid solution.
- **Outage node $ON_{i,k}$:** Follows from the definition of the E_{REF} arcs.

According to the flow network definition, the overall amount of flow that has to be pushed from the source to the target node equals:

$$f(E) = \sum_{t\in\mathcal{T}} \mathrm{DEM}^{s,t} + \max(cap(E_{BYP}))$$

Now we have to show that each solution, which implements this flow, fulfills the minimum and maximum capacity constraints on all arcs. Again we iterate over all arc types, skipping arcs with infinite capacity:

- **Demand E_{DEM}:** Follows from the definition of the embedding.
- **Type-1 production E_{PP1}:** Follows from CT 2.
- **Type-2 production E_{PP2}:** Follows from CT 4/5.
- **Transfer E_{LIM}:** Follows from the definition of $cap(E_{LIM})$ and CT 12.
- **Initial E_{INIT}:** Follows from CT 8.
- **Before refueling E_{AMAX}:** Follows from CT 11.
- **After refueling E_{SMAX}:** Follows from CT 11.
- **Reloading E_{REF}:** Follows from Lemma 7 and the network definition. The boundaries proposed by the lemma are chosen as arc capacity bounds in the flow network.
- **Residual E_{RES}:** Follows from the definition of $cap(E_{RES})$ and the fact that $\mathrm{SMAX}_{i,k}$ is an upper bound of the fuel level of cycle k.

- **Bypass E_{BYP}:** Follows from the definition of the fixed network flow. The bypass capacity is just the difference between the flow and the minimum capacity of all demand arcs.

□

Theorem 2. The cost of a minimum cost flow with flow amount

$$cap(e_{BYP}) + \mathrm{D} \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}} \mathrm{DEM}^{s,t}$$

in our network reduced by the global cost offset (see Lemma 8), yields a lower bound of production cost for the given scenario.

Proof. We prove this lemma by analyzing the cost of a minimum cost network flow. In the following, we aggregate the cost of all arc categories (plus the global cost offset) and transform this sum into the objective function. Let us start with the average cost above all scenarios:

$$\frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \Bigg(\underbrace{\sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} \mathrm{C}_j^{s,t} \cdot p_{j,s,t}}_{E_{PP1}} + \underbrace{\sum_{i \in \mathcal{I}} \sum_{k=0}^{\mathrm{K}_i^{min}} \frac{\mathrm{C}_{i,k}}{\mathrm{Q}_{i,k}} \cdot x(i,s,t_{i,k}^-)}_{E_{AMAX}} + \underbrace{\sum_{i \in \mathcal{I}} \sum_{k=0}^{\mathrm{K}-1} \mathrm{C}_{i,k} \cdot \delta(i,s,k)}_{E_{REF}}$$

$$- \underbrace{\sum_{i \in \mathcal{I}} \mathrm{C}_{i,\mathrm{T}} \cdot x(i,s,\mathrm{T})}_{E_{RES}} + \underbrace{\sum_{i \in \mathcal{I}} \min_{\mathrm{K}_i^{\min} \le \kappa \le \mathrm{K}_i^{\max}} \left(\sum_{k=0}^{\kappa} \mathrm{C}_{i,k} \left(\frac{\mathrm{Q}_{i,k} - 1}{\mathrm{Q}_{i,k}} \mathrm{BO}_{i,k-1} - \mathrm{BO}_{i,k} \right) \right)}_{\text{global cost}} \Bigg)$$

We now insert the number of the last scheduled cycle $\widehat{\mathrm{K}}_i$ of each power plant into the E_{AMAX}, E_{REF} and global cost subsums, only getting (not strictly) higher cost. Note that $\mathrm{K}_i^{min} \le \widehat{\mathrm{K}}_i \le \mathrm{K}_i^{max}$. The E_{AMAX} sum is extended from K_i^{min} to $\widehat{\mathrm{K}}_i$ elements, which are all non-negative. In the E_{REF} sum we cut away all elements of not scheduled cycles. Remember that the refueling of postponed cycles is 0 and so is the flow on the reloading arc $\delta(i,s,k)$. Therefore, cost is not reduced when we ignore the not scheduled cycles. At the global cost subsum, we also considered the global cost for $\widehat{\mathrm{K}}_i$ scheduled cycles at the minimum construction. Therefore, we can safely replace the minimum sum by the sum of $\widehat{\mathrm{K}}_i$ elements. We get:

$$\le \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \Bigg(\underbrace{\sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} \mathrm{C}_j^{s,t} \cdot p_{j,s,t}}_{E_{PP1}} + \underbrace{\sum_{i \in \mathcal{I}} \sum_{k=0}^{\widehat{\mathrm{K}}_i} \frac{\mathrm{C}_{i,k}}{\mathrm{Q}_{i,k}} \cdot x(i,s,t_{i,k}^-)}_{E_{AMAX}} + \underbrace{\sum_{i \in \mathcal{I}} \sum_{k=0}^{\widehat{\mathrm{K}}_i} \mathrm{C}_{i,k} \cdot \delta(i,s,k)}_{E_{REF}}$$

$$- \underbrace{\sum_{i \in \mathcal{I}} \mathrm{C}_{i,\mathrm{T}} \cdot x(i,s,\mathrm{T})}_{E_{RES}} + \underbrace{\sum_{i \in \mathcal{I}} \sum_{k=0}^{\widehat{\mathrm{K}}_i} \mathrm{C}_{i,k} \left(\frac{\mathrm{Q}_{i,k} - 1}{\mathrm{Q}_{i,k}} \mathrm{BO}_{i,k-1} - \mathrm{BO}_{i,k} \right)}_{\text{global cost}} \Bigg)$$

For each cycle, the corresponding elements of the E_{AMAX}, E_{REF} and global cost sums put together equal the cost of refueling (see Lemma 8). We aggregate these sums to get the real refueling cost:

$$=\frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\left(\underbrace{\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}}\mathrm{C}_j^{s,t}\cdot p_{j,s,t}}_{E_{PP1}}+\underbrace{\sum_{i\in\mathcal{I}}\sum_{k=0}^{\widehat{\mathrm{K}}_i}\mathrm{C}_{i,k}\cdot r_{i,k}}_{\text{refueling cost}}-\underbrace{\sum_{i\in\mathcal{I}}\mathrm{C}_{i,\mathrm{T}}\cdot x(i,s,\mathrm{T})}_{E_{RES}}\right)$$

We now extend the sum of refueling costs by all not scheduled cycles. Since $r_{i,k}$ equals 0 for these cycles, this does not change the sum. Finally, by moving the refueling costs out of the average above all scenarios, we get the objective function:

$$=\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}}\mathrm{C}_{i,k}\cdot r_{i,k}+\frac{1}{|S|}\sum_{s\in\mathcal{S}}\left(\sum_{t\in\mathcal{T}}\left(\sum_{j\in\mathcal{J}}\mathrm{C}_j^{s,t}\cdot p_{j,s,t}\cdot\mathrm{D}\right)-\sum_{i\in\mathcal{I}}\mathrm{C}_{i,\mathrm{T}}\cdot x(i,s,\mathrm{T})\right)$$

$\square$

4.4 Evaluation

In this section we evaluate the proposed lower bounds. For an introduction of our general experimental setup see Chapter 7.

The results that we obtained for all instances can be seen in Table 4.1. Both bounds yield stable deviations from the best known results. Obviously, the network-based lower bound gives far better results than the auction-based bound. The reason is that the network is designed to overcome the disadvantages of the auction (underestimated unit cost of Type-2 power plants and overestimated production capacity). In the flow network, the cost of production by Type-2 power plants is really mapped onto a concrete outage and billed with the refueling. Furthermore, the network models outages by limiting the production of the power plants over a certain period of time, while outages are not considered in the auction approach.

Therefore, the remaining gap between the network-based lower bound and an optimal solution originates mainly from the unconsidered spacing and resource constraints (CT 14 - 21) and the caveats against optional outages.

The lower bounds for instances from the sets B and X differ much more from their best known solution than instances from the set A do. This probably results from the larger number of outages and spacing constraints in the respective instances. Consequently, the influence of the spacing constraints, which are not modeled in the lower bound, grows and makes real solutions close to the lower bound hard to obtain.

As the obtained *scores* (deviation from the best known solution, cf. Section 7.2) of our lower bounds are similar among the instances from the sets B and X, we assume that these instances are somewhat equally hard to solve. Thus, a good solution should also yield similar scores for all instances.

	Auction		Network	
Instance	Value	Score	Value	Score
A00	8 676 507 008 664	-0,66%	8 701 730 192 910	-0,34%
A01	160 847 599 728	-5,13%	165 560 766 563	-2,35%
A02	130 148 129 044	-10,09%	139 991 463 398	-4,15%
A03	137 744 660 926	-10,80%	148 454 212 611	-3,87%
A04	82 428 942 096	-23,45%	102 326 409 223	-8,30%
A05	94 379 648 874	-24,99%	112 467 964 426	-10,61%
B06	36 338 357 898	-56,44%	69 592 490 826	-16,58%
B07	38 263 622 745	-52,86%	68 528 158 937	-15,58%
B08	28 410 840 209	-65,32%	62 594 950 271	-23,60%
B09	30 000 446 786	-63,30%	63 991 967 201	-21,72%
B10	30 389 023 703	-60,92%	63 747 909 586	-18,03%
X11	33 377 981 846	-57,81%	66 931 120 300	-15,40%
X12	36 740 208 840	-52,65%	66 558 720 126	-14,22%
X13	25 789 835 024	-66,27%	62 155 964 336	-18,70%
X14	26 903 399 271	-64,68%	63 045 557 475	-17,23%
X15	28 444 392 103	-62,13%	61 866 985 938	-17,62%

Table 4.1: Lower bounds: We compare both lower bounds with respect to the obtained objective function *value* and its *score* (deviation from the best known solution, cf. Section 7.2).

5. Basic Solution

In this chapter we present a first solution to the proposed problem. Since there is probably no way to solve even medium-sized instances of this problem exactly in reasonable time, we develop several heuristics. We decompose the problem into an outage scheduling and a power assignment phase. Outage dates $ha_{i,k}$ are fixed in the first phase, while power levels $p_{j,s,t}/p_{i,s,t}$ and refuelings $r_{i,k}$ are calculated in the second phase for a given outage schedule. Throughout this chapter we primarily care about getting a valid solution – postponing optimization to Chapter 6.

First, Section 5.1 deals with the outage scheduling phase. Due to the large number of constraint types (see Section 3.3) and their dependencies we use a CP formulation to find a feasible schedule. In Section 5.2, we present a valid production assignment heuristic for a given set of outages. Therefore, we utilize a demand auction derived from the lower bound presented in Section 4.2.

In Table 5.1 we give an overview at which point of the solution process we take care of each constraint type of the problem. We mark parts of our heuristics with an orange square if the consideration of the given constraint is necessary, i.e., not checking this property in the particular step might result in invalid solutions. A green dot stands for a sufficient mean to achieve this constraint, i.e., after performing this step the remaining solution space cannot fail this constraint.

5.1 Outage Scheduling

In this section we show a heuristic that constructs an outage schedule (i.e., explicit values for all $ha_{i,k}$ decision variables) for a given problem instance (see Chapter 3). The resulting schedule is valid with respect to the outage scheduling constraints CT 13 to CT 21. We add further constraints to our model, to ensure the feasibility of the remaining production assignment problem, which is solved in Section 5.2. As Type-1 power plants are not scheduled for outages, we only consider Type-2 power plants here.

We use the constraint programming toolkit Gecode for the creation of outage schedules and therefore present a suitable CP formulation of our problem in the upcoming sections. The CP model is presented in a bottom-up approach. In Section 5.1.1.1, we define the different variables for a single cycle of a Type-2 power plant and set up their relations among each other. After that, we establish the relations between consecutive cycles of a power plant in Section 5.1.1.2. Therefore, we also introduce two custom constraints, which propagate fuel

CT	Outage Scheduling			Production Assignment				
	Domain	Constraint	Custom	Preproc	Min	Max	Auction	Refueling
1	○	○	🟧	○	○	○	🟢	○
2	○	○	○	○	🟢	🟢	○	○
3	○	○	🟧	🟧	🟢	🟢	○	○
4/5	○	○	🟧	🟧	🟢	🟢	○	○
6	○	○	🟧	🟧	🟢	🟢	○	○
7	🟢	○	○	🟧	○	○	○	🟢
8	🟢	○	○	○	○	○	🟢	○
9	○	○	🟧	🟧	🟢	○	🟢	○
10	○	🟢	○	○	○	○	○	🟢
11	🟢	○	○	🟢	○	○	○	○
12	○	○	🟧	○	🟢	🟢	○	○
13A	🟢	○	○	○	○	○	○	○
13B	○	🟢	○	○	○	○	○	○
14	○	🟢	○	○	○	○	○	○
15	○	🟢	○	○	○	○	○	○
16	○	🟢	○	○	○	○	○	○
17	○	🟢	○	○	○	○	○	○
18	○	🟢	○	○	○	○	○	○
19	○	🟢	○	○	○	○	○	○
20	○	🟢	○	○	○	○	○	○
21	○	🟢	○	○	○	○	○	○

Table 5.1: Where do we take care for the different constraint types? In the outage scheduling part, we restrict the *domains* of variables, add the given *constraints* to our CP model or add further *custom* constraints. The production assignment part consists of a *preprocessing* step, the calculation of *minimum* and *maximum* offers for the *auctions* of each time step and the fixation of *refueling* amounts. Orange squares symbolize a step, where ignoring the constraint might result in invalid solutions. Green dots mark a step, whose entire resulting search space fulfills the constraint, i.e., we do not need to care anymore about it.

levels between successive cycles and enforce minimum durations of production campaigns. In Section 5.1.3 we model the various outage scheduling constraints, emphasizing the more complex constraints that cannot be modelled one-to-one from their definition in Section 3.3. Finally, Section 5.1.4 presents different branching strategies and explains how the search for a schedule works.

5.1.1 Basic Modelling

We model all variables using the finite domain integer model provided by Gecode. Note that problems may arise from the real-valued fuel levels. Hence, when implementing this part, we have to model fuel level constraints with tolerance for rounding errors.

5.1.1.1 Modelling a single cycle

We start by modeling a single cycle of a Type-2 power plant. The key variable of each cycle is the decoupling date. Although one variable would technically suffice, we add two other auxiliary variables for convenience.

Definition 16. Given an arbitrary Type-2 power plant i and cycle k (including the initial cycle -1), we define the boolean-valued finite domain integer variable $sch_{i,k}$ as an indicator ($1/true$ if this cycle is scheduled before the end of the time horizon) and an integer-valued variable $dec^*_{i,k}$ as the week of decoupling of power plant i in cycle k. Furthermore, we define $dec_{i,k}$ as the result of $ha_{i,k}$. Note that $dec_{i,k}$ equals $dec^*_{i,k}$ as long as the cycle is scheduled. Only if a cycle is postponed, the values differ. $dec_{i,k}$ then contains the published result -1, while $dec^*_{i,k}$ represents the internally available date range of this outage (which is after the time horizon). We set the limits of these variables as follows (remember $\mathrm{TO}_{i,k}$ and $\mathrm{TA}_{i,k}$ from CT 13):

$$sch_{i,k} \in \{0,1\} \tag{5.1}$$

$$\mathrm{TO}_{i,k} \leq dec^*_{i,k} \leq \begin{cases} \mathrm{TA}_{i,k} & \text{if } \mathrm{TA}_{i,k} \neq -1 \\ \infty & \text{else} \end{cases} \tag{5.2}$$

$$dec_{i,k} \in \{-1, 0, 1, \ldots, \mathrm{W}-1\} \tag{5.3}$$

For these three variables we can come up with a basic constraint set, which arises from CT 13: If the cycle is scheduled, $dec^*_{i,k}$ and $dec_{i,k}$ are equal. If it is postponed, $dec_{i,k}$ has to be -1 while $dec^*_{i,k}$ is only limited by $\mathrm{TO}_{i,k}$. Equations 5.4 to 5.6 give a formalization of this construct:

$$sch_{i,k} \iff dec_{i,k} = dec^*_{i,k} \tag{5.4}$$

$$sch_{i,k} \iff dec^*_{i,k} < \mathrm{W} \tag{5.5}$$

$$\neg\, sch_{i,k} \iff dec_{i,k} = -1 \tag{5.6}$$

Note that only the $dec_{i,k}$ variables need to be assigned in a solution, because they determine the $sch_{i,k}$ variables. The $dec^*_{i,k}$ are unimportant in a solution. Either $dec_{i,k}$ equals -1 (i.e., the cycle is postponed) or it lies in between the earliest and latest allowed decoupling dates. Because the special meaning of $dec_{i,k} = -1$, we do not use $dec_{i,k}$ in the following. Instead we use $dec^*_{i,k}$, which is simply not assigned in a solution if the cycle is not scheduled.

The reason for duplicating the decoupling date into the variables $dec_{i,k}$ and $dec^*_{i,k}$ is not obvious. But, modeling this way has three advantages:

- **Standard Branching:** Gecode's standard branching techniques work towards assigning all decision variables. In our case we do not care about the specific date of an outage, if it is scheduled after the time horizon. So, when using a standard branching, we only consider $dec_{i,k}$ as a decision variable.

- **Smaller Domains:** A later cycle is more likely to be postponed ($dec_{i,k} = -1$). So, until we decide if a cycle is scheduled, its domain ranges from -1 to $\mathrm{W} - 1$ (scheduled for the last week), which results in poor constraint propagation. However, $dec^*_{i,k}$ always represents the true scheduling options.
- **Easier Modelling:** By using $dec^*_{i,k}$ we can get rid of any exceptions arising from postponed cycles, when modeling outage dependency constraints (CT 14 to CT 21). Furthermore, this can be seen as a more natural model, since we really move the cycle's decoupling date after the time horizon.

Although we do not plan any production levels in this phase, it is crucial to track fuel levels in order to keep the remaining production assignment problem feasible. We give a short example: Let us take the minimum achievable fuel level at the beginning of a production campaign and the maximum allowed fuel level at the beginning of the next outage. Their difference imposes a minimum amount of fuel that has to be consumed during the given production campaign. Since fuel consumption is limited (see PMAX_i^t and CT 9), this effectively generates a spacing constraint between two successive outages. We model the fuel tracking by three more finite domain integer variables for each cycle:

Definition 17. Given an arbitrary Type-2 power plant i and cycle k (including the initial cycle -1), we define the following variables: $pre_{i,k}$ as the fuel level before refueling, $ref_{i,k}$ as the amount of fuel provided during this outage and $post_{i,k}$ as the fuel level after refueling but before any production. We set the limits of these variables as follows:

$$0 \leq pre_{i,k} \leq \mathrm{AMAX}_{i,k} \tag{5.7}$$

$$\mathrm{RMIN}_{i,k} \leq ref_{i,k} \leq \mathrm{RMAX}_{i,k} \tag{5.8}$$

$$0 \leq post_{i,k} \leq \mathrm{SMAX}_{i,k} \tag{5.9}$$

The limits of $ref_{i,k}$ can be naturally taken from the bounds of refueling (CT 7). Similarly, we can extract the limits of $pre_{i,k}$ and $post_{i,k}$ from the maximum allowed fuel levels before and after refueling (CT 11). For the initial campaign we consider $post_{i,-1} = \mathrm{XI}_i$, while $pre_{i,-1}$ and $ref_{i,-1}$ can be ignored.

We now add the refueling constraint (CT 10) to our model:

$$post_{i,k} = \frac{\mathrm{Q}_{i,k} - 1}{\mathrm{Q}_{i,k}}(pre_{i,k} - \mathrm{BO}_{i,k-1}) + ref_{i,k} + \mathrm{BO}_{i,k} \tag{5.10}$$

So far, we established a model for a single cycle. Table 5.2 gives an overview of the introduced variables, their values for the initial cycle ($k = -1$) and domain boundaries for the following cycles. In the next section we assemble these cycles to model a complete Type-2 power plant.

5.1.1.2 Modelling Type-2 Power Plants

Right now our CP model can handle decoupling dates, scheduled states (i.e., scheduled or postponed) and fuel levels of each cycle independently. In this section, we first fix the order of cycles of a Type-2 power plant. Afterwards, we augment our model with constraints that guarantee the feasibility of the production assignment problem for a single power plant in the second phase.

According to CT 13B all cycles of a Type-2 power plant have to be scheduled in the same order as they appear in the dataset. Therefore, we introduce Equation 5.11. Note that we do not need to care if the outage is scheduled, because we use $dec^*_{i,k}$ instead of $dec_{i,k}$.

$$dec^*_{i,k+1} \geq dec^*_{i,k} + \mathrm{DA}_{i,k} \tag{5.11}$$

Variable	**k = −1**	**k > −1**		**Description**
	Assign	**Min**	**Max**	
$sch_{i,k}$	1	0	1	Is this campaign scheduled?
$dec_{i,k}$	0	−1	W −1	Date of decoupling
$dec^*_{i,k}$	0	0	∞	Auxiliary variable for date of decoupling
$ref_{i,k}$	0	$\mathrm{RMIN}_{i,k}$	$\mathrm{RMAX}_{i,k}$	Amount of fuel provided
$pre_{i,k}$	0	0	$\mathrm{AMAX}_{i,k}$	Fuel level before refueling
$post_{i,k}$	XI_i	0	$\mathrm{SMAX}_{i,k}$	Fuel level after refueling

Table 5.2: Finite domain integer variables instantiated for each cycle of a Type-2 power plant i. The *Initial* column comprises the predefined values of this variable for the initial cycle $k = -1$, while the *Min* and *Max* columns represent the domain limits for all successive cycles.

Equation 5.11 implies that there is no scheduled outage after the first not scheduled outage of a Type-2 power plant.

Lemma 10. It holds:

$$sch_{i,k+1} = 1 \Longrightarrow sch_{i,k} = 1$$

Proof. We derive this lemma from the previously defined constraints/equations in this section:

$$sch_{i,k+1} = 1 \overset{5.5}{\Longleftrightarrow} dec^*_{i,k+1} < \mathrm{W} \overset{5.11}{\Longrightarrow} dec^*_{i,k} < \mathrm{W} \overset{5.5}{\Longleftrightarrow} sch_{i,k} = 1$$

□

The constraints provided yet would suffice to create a valid outage schedule for a single Type-2 power plant. Unfortunately, it is not sufficient to guarantee feasible production assignments later, since we might schedule two outages within a time distance that is too short to consume enough fuel to comply with CT 11 in the next cycle. Furthermore, from an opposite perspective we discover that fixing two successive outage dates $dec^*_{i,k}$ and $dec^*_{i,k+1}$ imposes restrictions on the amount of refueling $ref_{i,k}$.

5.1.2 Additional Constraints

In this section we add constraints to our CP model, which solve the problems shown in the last paragraph. I.e., they guarantee the feasibility of the production assignment phase for a yet to find outage schedule. Technically, we have to model further dependencies between the fuel levels and the decoupling dates of consecutive cycles (see Figure 5.1). On the one hand, the minimum difference between the fuel levels before ($post_{i,k-1}$) and after ($pre_{i,k}$) a production campaign imposes a minimum spacing between the decoupling dates $dec_{i,k-1}$ and $dec_{i,k}$. On the other hand, the maximum distance between the decoupling dates imposes a maximum difference between the fuel levels.

In the following we present a CP formulation in the form of two custom propagators, which enforce the additional constraints. Both propagators heavily rely on the estimation of maximum fuel levels in forward and backward direction in time. Forward propagation works straightforward (see CT 9). How to perform backward propagation is explained in Section 5.1.2.1. There we show how to deduce an upper bound of the fuel level of a previous time step from the given fuel level of the present time step. Afterwards in Section 5.1.2.2,

$$\begin{tikzcd}
 & post_{i,k-1} & \\
dec_{i,k-1} & {} \arrow[l] \arrow[r] \arrow[u] \arrow[d] & dec_{i,k} \\
 & pre_{i,k} &
\end{tikzcd}$$

Figure 5.1: Conflicting variables of consecutive cycles.

we present a propagator that limits the fuel levels before and after a production campaign, based on the possible outage dates. Finally, we present the second propagator in Section 5.1.2.3, which sets up a minimum spacing constraint between consecutive outages of a plant, considering the fuel levels and dataset parameters.

5.1.2.1 Fuel Level Back Calculation

In this section we present an inverse function that calculates an upper bound of the fuel level $x(i,s,t)$ from a given future fuel level $x(i,s,t+n)$ and the available production intervals in between. This function is utilized in the upcoming additional constraint definitions. Note that the back calculation is easier for time steps t where the fuel level $x(i,s,t)$ is above the fuel level threshold $\mathrm{BO}_{i,k}$ (see CT 6). In this case we exactly know the maximum production levels (and with it the maximum possible fuel consumption) of all time steps and can apply CT 9:

Lemma 11. For an arbitrary Type-2 power plant i and a given fuel level at time step $t+1$ we can bound the fuel level at time step t as:

$$x(i,s,t) \leq x(i,s,t+1) + \mathrm{PMAX}_i^t \cdot \mathrm{D}$$

Note that Lemma 11 holds for all time steps. As long as the production level is not imposed, this bound even guarantees that future fuel limits are achievable if the current fuel level does not exceed its bound. To give this guarantee while $\mathrm{BO}_{i,k}$ is under-run requires additional effort. We do not know the available ratio of the maximum production level (see Figure 5.2), because it depends on the previous fuel level – the value we are looking for. From CT 6 and CT 9 we derive:

$$x(i,s,t) = x(i,s,t+1) + \mathrm{PB}_{i,k}(x(i,s,t)) \cdot \mathrm{PMAX}_i^t \cdot \mathrm{D} \tag{5.12}$$

To transform this equation into a computable upper bound, we first introduce an alternative notation of $\mathrm{PB}_{i,k}$. While the original problem defines this piecewise linear function by its data nodes (fuel level, ratio), we use a different representation to ease the calculation:

Notation 1. Let the imposed power profile $\mathrm{PB}_{i,k} : \mathbb{R}^{\geq 0} \to [0,1]$ for a Type-2 power plant i and cycle k be denoted as follows. Let $s_0, s_1, \ldots, s_n$ denote the fuel level values of the data nodes in descending order (i.e., $s_0 = \mathrm{BO}_{i,k}$ and $s_n = 0$). Besides, let m_l denote the slope of the ratio between s_l and s_{l-1} and n_l denote the offset of this linear function (see Figure 5.2). For a given fuel level x less or equal s_0 we calculate the available fuel ratio by $\mathrm{PB}_{i,k}(x) := m_l \cdot x + n_l$ where l is chosen such that $s_l \leq x < s_{l-1}$. Furthermore, we define $m_0 := 0$ and $n_0 := 1$ so that, for $x \geq s_0$ it holds $\mathrm{PB}_{i,k}(x) = 1$.

By using Notation 1 we can rewrite Equation 5.12:

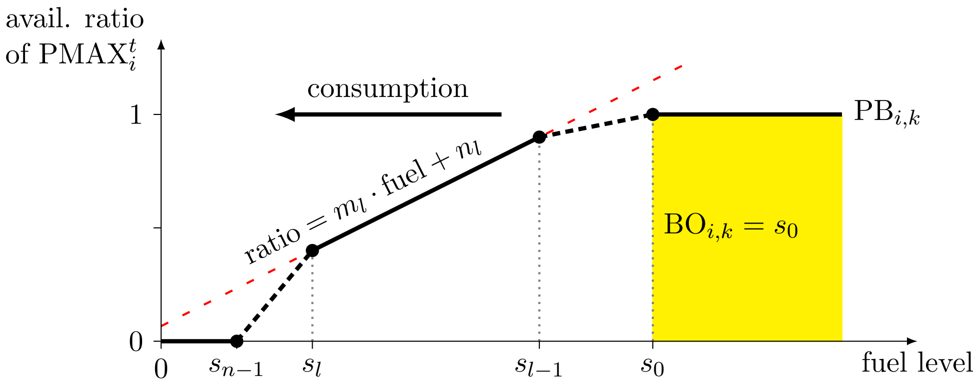

Figure 5.2: Available production ratio during a time step in relation to the current fuel level. As long as there is more than $\mathrm{BO}_{i,k}$ fuel, we can choose the ratio freely (yellow area). Otherwise the ratio is fixed by the piecewise linear concave function $\mathrm{PB}_{i,k}$. Note that we "move" left in this chart, since we consume fuel between two time steps.

Lemma 12. For an arbitrary Type-2 power plant and a given fuel level at time step $t+1$, there exists an l such that we can bound the fuel level at time step t to:

$$x(i,s,t) \leq x(i,s,t+1) + (m_l \cdot x(i,s,t) + n_l) \cdot \mathrm{PMAX}_i^t \cdot \mathrm{D} \tag{5.13}$$

and it holds $s_l \leq x(i,s,t) < s_{l-1}$.

By solving Lemma 12 for $x(i,s,t)$ we can present the core of the inverse function, which works out the initial fuel level of a time step.

Lemma 13. For an arbitrary Type-2 power plant i with a fuel level $x(i,s,t+1)$ at the end of time step t, we can limit the fuel level $x(i,s,t)$ at the beginning of this time step. There exists a (not necessarily unique) l such that

$$x(i,s,t) \leq \frac{x(i,s,t+1) + \mathrm{PMAX}_i^t \cdot \mathrm{D} \cdot n_l}{1 - \mathrm{PMAX}_i^t \cdot \mathrm{D} \cdot m_l}$$

and $s_l \leq x(i,s,t) < s_{l-1}$ or production during t was unbound ($x(i,s,t) \geq \mathrm{BO}_{i,k}$).

Proof. We assume that production is bound, i.e., $x(i,s,t) < \mathrm{BO}_{i,k}$ (otherwise the equation can be reduced to Lemma 11). The existence of an l that fulfills $s_l \leq x(i,s,t) < s_{l-1}$ follows directly from Notation 1. Taking a suitable l, we can transform Lemma 12 into the sought formula. □

Although we know that a suitable segment l exists, it is not trivial to forecast. A good choice might be the segment of the $x(i,s,t+1)$ value, but dependent on the granularity of $\mathrm{PB}_{i,k}$ the segment of $x(i,s,t)$ might still be several steps away. Algorithm 3 takes a simpler approach, iterating over all segments of $\mathrm{PB}_{i,k}$ and stopping at the first matching segment – thus maximizing the available production ratio.

Finally, note that our back calculation does generally not provide a lower bound of the fuel level before production in cases where the residual fuel level is so low that production might have been ceased for several time steps (see CT 6). In this case we cannot determine the last time step of production.

Algorithm 3: PREVIOUS (plant i, time step t, power profile $\mathrm{PB}_{i,k}$, fuel level $post$)

```
Input: Plant parameters and fuel level after production of time step $t$
Output: Upper bound on the fuel level before production of time step $t$
begin
    $l \longleftarrow 0$
    // Loop over segments, stop at the first matching segment
    repeat
        $l \longleftarrow l + 1$
        // Perform calculation of Lemma 13
        $pre \longleftarrow (post + \mathrm{PMAX}_i^t \cdot \mathrm{D} \cdot n_l)/(1 - \mathrm{PMAX}_i^t \cdot \mathrm{D} \cdot m_l)$
    until $s_l \leq pre$
    return $pre$
end
```

5.1.2.2 Fuel Level Propagation

In this section we model the connection between fuel levels of consecutive cycles, specifically the difference between the level after refueling $post_{i,k}$ and the level before the refueling of the next campaign $pre_{i,k+1}$. We focus on the estimation of the amount of production during the campaign. Technically, we realized this section by adding a custom propagator to the CP model. We show the concept of this propagator, while omitting implementation specific details.

In our CP model fuel levels of consecutive cycles are not related yet. But, since these fuel levels effectively impose spacing constraints between outages, we cannot ignore them here. From the problem formulation we know that fuel levels depend on production levels (CT 9). Besides, other constraints apply during a production campaign, for example, the allowed maximum modulation (CT 12) or power profile imposition (CT 6) if we under-run the threshold $\mathrm{BO}_{i,k}$.

When modeling the overall production of a campaign, the following degrees of freedom arise from the problem formulation:

- start of production campaign ($dec^*_{i,k} + \mathrm{DA}_{i,k}$)
- opening fuel level ($post_{i,k}$)
- end of production campaign ($dec^*_{i,k+1}$ or time horizon)
- final fuel level ($pre_{i,k+1}$)
- maximum modulation ($\mathrm{MMAX}_{i,k}$)

To limit the fuel level variables we consider a minimum and a maximum production scenario. Maximum production is obviously done with PMAX_i^t. Similarly we model minimum production by adding the maximum modulation $\mathrm{MMAX}_{i,k}$ to the fuel level and by performing maximum production at all time steps. This works because the accumulated difference between the maximum and minimum production cannot exceed $\mathrm{MMAX}_{i,k}$. One might imagine these additional units of fuel as *no-production* units, which, when consumed by a plant, reduce the plant's real production level.

In the following we introduce the forward and the backward propagation of this constraint. First, we limit the opening fuel level of the next cycle $pre_{i,k+1}$ by performing a maximum and minimum production starting with $post_{i,k}$ (*Forward Propagation*). After that we simulate maximum production starting with $pre_{i,k+1}$ and going back in time (*Backward Propagation*), thus finding an upper bound of $post_{i,k}$ for any feasible production assignment problem.

Forward Propagation

Since we want to constrict $pre_{i,k+1}$ as much as possible, we have to choose the concrete values from the remaining degrees of freedom accordingly. To reach the maximum final fuel level we take the upper domain limit of the opening fuel level, choose the shortest duration of the production campaign and fully exhaust modulation. Contrary, to reach the minimum final fuel level we take the lower domain limit of the opening fuel level, choose the longest duration and no modulation. Table 5.3 gives an overview of the used parameters, while Figure 5.3 illustrates the process.

	Minimum	Maximum
start	$\min(dec^*_{i,k}) + \mathrm{DA}_{i,k}$	$\max(dec^*_{i,k}) + \mathrm{DA}_{i,k}$
end	$\max(dec^*_{i,k+1})$	$\min(dec^*_{i,k+1})$
fuel	$\min(post_{i,k})$	$\max(post_{i,k})$
modulation	0	$\mathrm{MMAX}_{i,k}$

Table 5.3: Forward propagation parameters to get an upper limit (*Maximum*) and lower limit (*Minimum*) of $pre_{i,k+1}$.

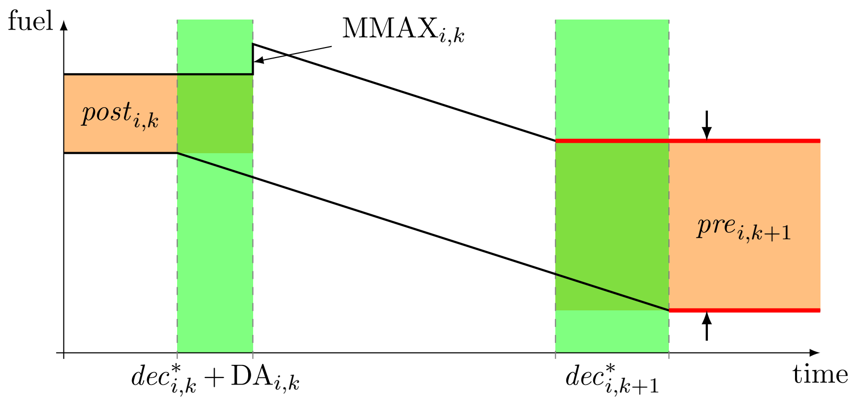

Figure 5.3: Forward propagation: From $post_{i,k}$ we simulate maximum production (lower line) and minimum production (upper line) to limit the domain of $pre_{i,k+1}$.

Both maximum and minimum calculation can be performed using CT 9 for all time steps of the considered campaign range and initializing the first fuel level properly.

Backward Propagation

While the forward propagation of fuel levels can be seen just as completion of the CP model, backward propagation is crucial to keep the production assignment problem of the next phase feasible. Assuming that $post_{i,k}$ would not be limited this way, we might load more fuel than can be consumed in the following production campaign and fail to reach the maximum allowed fuel level before refueling of the next campaign $\mathrm{AMAX}_{i,k+1}$ (see CT 11). Technically, we can also provide a lower bound for $post_{i,k}$, but this is not a hard constraint, since less fuel would just result in ceased production of this plant.

Backward propagation should constrict $post_{i,k}$, so we again have to choose values from the remaining degrees of freedom (see Table 5.4). To determine the maximum allowed fuel level at the beginning of the production campaign we take the upper domain limit of the final fuel level, choose the longest duration of the production campaign and no modulation.

	Minimum	Maximum
start	$\min(dec^*_{i,k+1})$	$\max(dec^*_{i,k+1})$
end	$\max(dec^*_{i,k}) + \mathrm{DA}_{i,k}$	$\min(dec^*_{i,k}) + \mathrm{DA}_{i,k}$
fuel	$\min(pre_{i,k+1})$	$\max(pre_{i,k+1})$
modulation	$\mathrm{MMAX}_{i,k}$	0

Table 5.4: Backward propagation parameters to get an upper limit (*Maximum*) and lower limit (*Minimum*) of $post_{i,k}$.

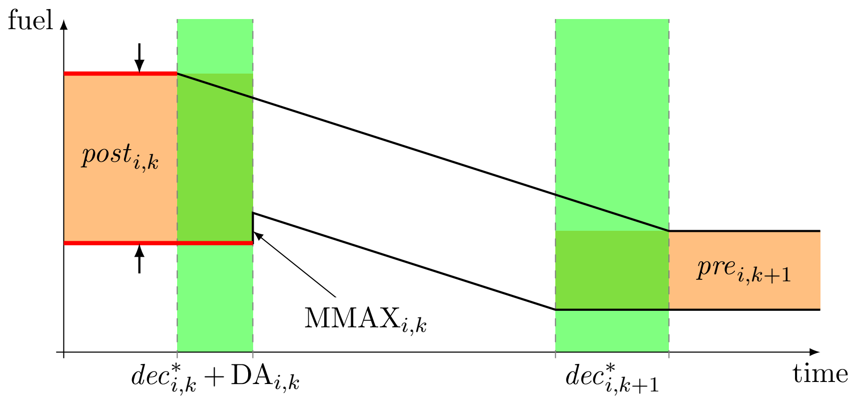

Figure 5.4: Backward propagation: From $pre_{i,k+1}$ we simulate backward maximum production (upper line) and backward minimum production (lower line) to limit the domain of $post_{i,k}$. Note that the minimum production is more of a suggestion than a hard constraint.

For the sake of completeness Table 5.4 also lists the parameters to get a lower limit of a "useful" $post_{i,k}$. Again Figure 5.4 illustrates the process.

Both maximum and minimum calculation can be performed using Algorithm 3 for all time steps of the considered campaign range in descending order and initializing the last fuel level properly.

5.1.2.3 Outage Date Propagation

In the last section we limited the fuel levels before ($post_{i,k}$) and after ($pre_{i,k+1}$) a production campaign. Given these levels and the power plant's maximum production for all time steps of the production campaign, we can deduce the minimum period of time the plant needs to consume this fuel difference. Note that scheduling the neighboring outages in a shorter distance would not yet fail our outage scheduling phase but surely result in an infeasible production assignment problem.

In this section we set up a spacing constraint between consecutive outages of a given Type-2 power plant. Similar to the last section, we simulate maximum production (in forward and backward direction) to constrain the latest start and earliest stop date of a production campaign. Again, we realized this as a custom propagator in Gecode, while this document just shows the concept.

Contrary to the fuel propagation constraint, there are no real degrees of freedom in this constraint. The considered fuel levels can be fixed (minimum before and maximum after production), while production itself is always assumed to be maximal. Only the decoupling dates are variable.

During forward propagation we determine the earliest possible stop date of a production campaign (or the earliest date of decoupling of the next campaign, respectively). Therefore, we try to reach the maximum fuel level at the end as early as possible, which means: starting at the earliest date of coupling and performing maximum production. Figure 5.5 illustrates this concept.

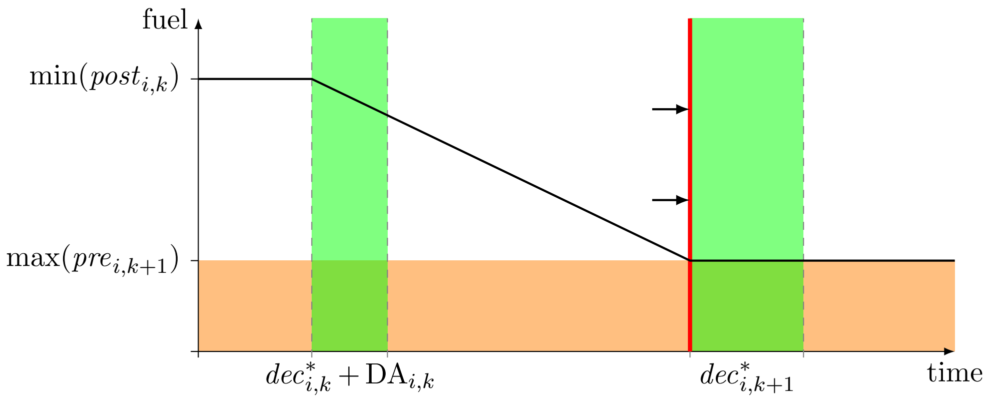

Figure 5.5: Outage Forward Propagation: Starting maximum production at the earliest possible date with the minimum fuel level ($\min(post_{i,k})$), we can limit the earliest date of the next outage to the time step, where we reach the maximum allowed fuel level at the end ($\max(pre_{i,k+1})$).

Backward propagation works in a similar way and is illustrated in Figure 5.6. We are looking for the latest possible start date of a production campaign. We assume that production reaches the maximum allowed fuel level before refueling $\mathrm{AMAX}_{i,k+1}$ at the latest possible time step, while maximum production has been performed throughout the whole production campaign. The time step where the fuel level back calculation reaches the minimum pre-production fuel level is an upper bound for the coupling date.

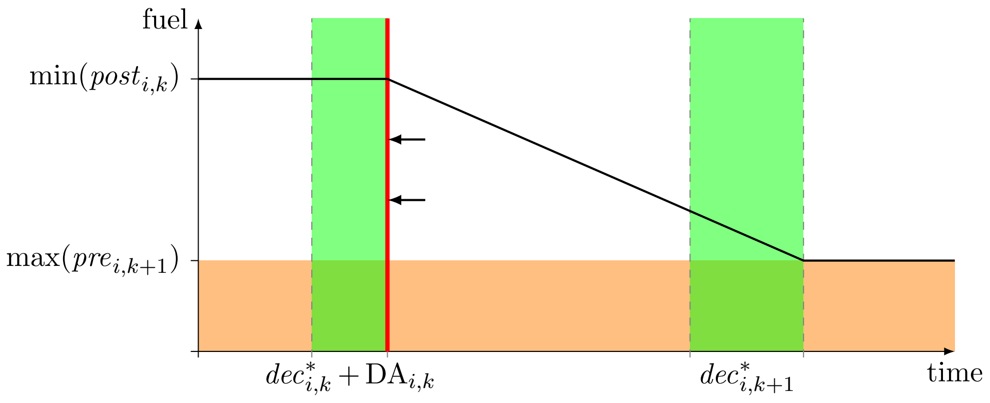

Figure 5.6: Outage Backward Propagation: Assuming maximum production until the latest possible date, reaching the maximum allowed fuel level at the end ($\max(pre_{i,k+1})$), we can limit the latest date of the beginning of the production campaign to the time step where we reach the minimum fuel level before production ($\min(post_{i,k})$).

5.1.3 Modelling Constraint Types

In the last section we worked out a basic model of Type-2 power plant operations that cares for feasibility of the second phase – the production assignment. All constraints considered yet (CT 3 - 13) applied to a single power plant. In this section we model the remaining constraints CT 14 - 21.

Note that CT 14 - 18 are all spacing constraints with large similarities. In Section 5.1.3.1 we first show how to formulate these spacing constraints for our CP model. After that, the constraints CT 19 -21 are handled separately in Sections 5.1.3.2 to 5.1.3.4, because all of them are special on their own.

5.1.3.1 Handling Spacing Constraints

The spacing constraints CT 14 - 18 operate on a set $\mathcal{A}$ of Type-2 power plants. For all pairs of cycles k and k' from two different power plants i and i' in $\mathcal{A}$ they impose a certain minimum spacing between these cycles' outages. We can transform the constraints from their definition in Section 3.3.3.1 in a canonical way and add them to our model for all pairs of cycles from different Type-2 power plants :

[CT 14] Minimum spacing between outages

$$\big(\,dec^*_{i,k} - dec^*_{i',k'} - \mathrm{DA}_{i',k'} \geq \mathrm{Se}\big) \vee \big(\,dec^*_{i',k'} - dec^*_{i,k} - \mathrm{DA}_{i,k} \geq \mathrm{Se}\big)$$

[CT 15] Minimum spacing between outages during a specific period

$$\begin{aligned}&(\mathrm{ID} - \mathrm{DA}_{i,k} < dec^*_{i,k}) \wedge (dec^*_{i,k} \leq \mathrm{IF}) \wedge (\mathrm{ID} - \mathrm{DA}_{i',k'} < dec^*_{i',k'}) \wedge (dec^*_{i',k'} \leq \mathrm{IF})\\ &\implies (dec^*_{i,k} - dec^*_{i',k'} - \mathrm{DA}_{i',k'} \geq \mathrm{Se}) \vee (dec^*_{i',k'} - dec^*_{i,k} - \mathrm{DA}_{i,k} \geq \mathrm{Se})\end{aligned}$$

[CT 16] Minimum spacing between decoupling dates

$$|\,dec^*_{i,k} - dec^*_{i',k'}\,| \geq \mathrm{Se}$$

[CT 17] Minimum spacing between coupling dates

$$|\,dec^*_{i,k} + \mathrm{DA}_{i,k} - dec^*_{i',k'} - \mathrm{DA}_{i',k'}\,| \geq \mathrm{Se}$$

[CT 18] Minimum spacing between decoupling and coupling dates

$$|\,dec^*_{i,k} + \mathrm{DA}_{i,k} - dec^*_{i',k'}\,| \geq \mathrm{Se}$$

Note that we do not need to care if the cycles are scheduled or not, since we always use the $dec^*_{i,k}$ variables. If a cycle may not be scheduled, the upper bound of $dec^*_{i,k}$ is infinity and therefore all the just formulated constraints hold.

5.1.3.2 [CT 19] Resource Constraint

To model this constraint type we make use of the *cumulatives* constraint (see [BC02]) of Gecode, which schedules tasks onto limited resources. In our setup we have a single constrained resource per instance of CT 19 and a task corresponds to the usage of this resource during an outage. Note that the outage's duration and the time of resource consumption are not equal – resource consumption starts $\mathrm{L}_{i,k}$ weeks after decoupling and lasts for $\mathrm{TU}_{i,k}$ weeks. Therefore, we have to calculate the dates for each task relative to the decoupling date of each outage. We have to set the *start* and *duration* of each task, where $start_{i,k}$ is again a finite domain integer variable while $duration_{i,k}$ is a constant. The *height* is set to 1 for each task, since CT 19 states that each outage consumes exactly one unit of the resource.

The global resource limit is given as Q and can be used directly. Algorithm 4 gives a pseudo-code notation of the approach stated above.

Algorithm 4: POST#19 ($\mathcal{A}$, starts L, durations TU, limit Q)

```
begin
    foreach $i \in \mathcal{A}$ do
        foreach $k \in \mathcal{K}$ do
            $start_{i,k} \longleftarrow dec^*_{i,k} + \mathrm{L}_{i,k}$
            $duration_{i,k} \longleftarrow \mathrm{TU}_{i,k}$
            $height_{i,k} \longleftarrow 1$
    $cumulatives(start, duration, height, \mathrm{Q})$
end
```

5.1.3.3 [CT 20] Maximum number of outages during a given week

We model this constraint type utilizing a linear constraint. A boolean-valued FDI vector marks if an outage takes place during the given week. The sum of this vector is limited to the given number of maximum parallel outages. See Algorithm 5.

Algorithm 5: POST#20 ($\mathcal{A}$, week H, limit N)

```
begin
    $v \longleftarrow$ vector of boolean-valued FDI variables
    foreach $i \in \mathcal{A}$ do
        foreach $k \in \mathcal{K}$ do
            $(dec^*_{i,k} \leq \mathrm{H}) \wedge (dec^*_{i,k} + \mathrm{DA}_{i,k} > \mathrm{H}) \Leftrightarrow v_{i,k} = 1$
    $linear(v, \mathrm{IRT\_LQ}, \mathrm{N})$
end
```

5.1.3.4 [CT 21] Maximum offline power capacity during a specific period

Similarly to CT 20 we use a linear constraint here. But instead of limiting the sum of the vector itself, we limit the sum of $v \cdot \mathrm{PMAX}$, hence limiting the maximum power capacity on outage. Since this constraint applies over a certain period of time, we have to post this constraint for each time step in between. Although outage dates only vary in weeks, posting one constraint per week would not suffice, since the maximum capacity of a Type-2 power plant can change during a week. See Algorithm 6.

Algorithm 6: POST#21 ($\mathcal{A}$, weeks [ID, IF], limit IMAX)

```
begin
    foreach $ID \leq$ time step $t \leq IF$ do
        $v \longleftarrow$ vector of boolean-valued FDI variables
        foreach $i \in \mathcal{A}$ do
            foreach $k \in \mathcal{K}$ do
                $(dec^*_{i,k} \leq week_t) \wedge (dec^*_{i,k} + \mathrm{DA}_{i,k} > week_t) \Leftrightarrow v_{i,k} = 1$
        $linear(v, \mathrm{PMAX}^t, \mathrm{IRT\_LQ}, \mathrm{IMAX})$
end
```

5.1.4 Branching

The employed branching strategy is not crucial to find any solutions, but important to find good solutions fast. As introduced in Section 2.1.4 branching always consists of two parts: choosing an unassigned variable and deciding what to do with it.

The constraints set up now limit all variable domains as strong as possible. When propagation reaches a fix point (i.e., no further domain limiting is possible), we have to add further constraints and branch the resulting search space, to allow further constraint propagation. Branching and propagation are alternating steps. We repeat them until all cycles are either scheduled at a specific date or postponed.

In this section we present three different branching strategies and show their respective advantages and disadvantages. The first two are standard Gecode branchings. Remember that we designed our model in a way ($dec_{i,k}$ vs. $dec^*_{i,k}$) that standard branchings work for our needs.

The first branching strategy, presented in Section 5.1.4.1, schedules outages with the smallest domain first. Our second branching strategy introduces a randomized approach in Section 5.1.4.2. Finally, a branching that schedules outages in the order in which they appear is presented in Section 5.1.4.3, overcoming a problem of the branchings before: the decision to postpone all optional outages.

No matter which branching technique we use, searching the whole search tree is out of scope. Even if we assume that we only try out the lower and upper date bounds of each outage, the search space is still of size $2^{|outages|}$. However, medium sized datasets have around 60 outages to be scheduled (the biggest ones have around 300). Therefore, randomizing the search and continuously limiting of the domains towards promising regions should yield good options to achieve cheap schedules.

In Chapter 6 we develop more sophisticated branching techniques to improve the results found. This includes branchings augmented with lower bound data.

5.1.4.1 Smallest Domain Branching

In our first shown branching technique we fix the outages with the smallest domain first, i.e., outages with a small difference between $\max(dec^*_{i,k})$ and $\min(dec^*_{i,k})$. Although we do not care for the cost incurred this might be reasonable to find first solutions. Outages with small domains are more likely to become infeasible if other outages are assigned first.

Choosing a good value is not simple: if an outage is scheduled too early we loose fuel during the outage's refueling, but if the outage is scheduled too late we run out of fuel and are effectively offline longer than necessary. Since we set up constraints that ensure feasible fuel levels, we keep this branching simple and choose from three options: minimum, median, maximum decoupling date.

$$branch(dec, \text{INT\_VAR\_SIZE\_MIN}, \text{INT\_VAL\_MIN})$$
$$branch(dec, \text{INT\_VAR\_SIZE\_MIN}, \text{INT\_VAL\_MED})$$
$$branch(dec, \text{INT\_VAR\_SIZE\_MIN}, \text{INT\_VAL\_MAX})$$

5.1.4.2 Random Branching

Although deterministic branching would sooner or later explore all possible solutions, it has the huge disadvantage that we are nearly unable to revoke early decisions, due to the size of our search tree. To overcome this problem we present a randomized branching in this section, which chooses an outage and the assigned value randomly from all unassigned outages and their respective domains.

Choosing outage dates randomly often results in infeasible spaces. Therefore, we restart this branching frequently, at the latest when a solution has been found. Especially the decision to schedule or postpone a specific cycle often leads to this problem because we cannot predict if the resulting spacing constraints can be fulfilled. Therefore, we use a preprocessing before applying this branching:

1. perform an initial propagation step
2. postpone all cycles whose *sch*-variable's domain contains 0

This way we only schedule the mandatory cycles, which results in less constraints and less failed spaces. Unfortunately, this also results in poor solutions for weak constrained datasets, since here most cycles are optional, but not scheduling them leads to Type-2 power plants running out of fuel.

$$branch(dec, \text{INT\_VAR\_RND}, \text{INT\_VAL\_RND})$$

5.1.4.3 Sweep Branching

Remember that the quality of a branching can be seen as how early infeasible regions are detected. Both previously presented branchings fail from this point of view, because the cycles assigned one after another are seldom closely related. This results in situations which look feasible until we try to assign a cycle, whose predecessor and successor have already been assigned. For this reason it might be useful to assign variables of similar dates consecutively, as this might result in early detection of failed states and therefore finding more results in the given time.

Our third proposed branching overcomes this problem. It always chooses the cycle that minimizes $\min(dec^*_{i,k})$ and assigns $dec^*_{i,k}$ to its minimum. So, when scheduling a cycle, we can be sure that its previous cycle has already been scheduled to a fixed date. If the minimum above all unassigned $dec^*_{i,k}$ is behind the time horizon, i.e., greater or equal W, the remaining cycles are not scheduled.

5.2 Production Assignment

In this section we present a power assignment heuristic derived from the auction-based lower bound presented in Section 4.2. Power assignments are performed for a given outage schedule (i.e., all $ha_{i,k}$ are fixed) and assign the resulting decision variables: production levels $p_{j,s,t}/p_{i,s,t}$ and refueling amounts $r_{i,k}$. We use the same auction method and Type-1 power plant offers as presented in Chapter 4 about lower bounds but augment the offers of Type-2 power plants by their current fuel level and consumed modulation (i.e., the difference between the maximum and realized production throughout the current campaign).

The auction is run in ascending order of the time steps and for all scenarios in parallel. This way, it is easy to adjust the fuel levels from the previous time step and fix the refueling amounts consistently, when we run the auction for the second time step of an outage.

This chapter is organized as follows. First, we present a special preprocessing step in Section 5.2.1 which is performed for each Type-2 power plant to ensure production feasibility. After that we show how the offers of each Type-2 power plant are compiled in Section 5.2.2. Finally, we introduce the calculation of the refueling amounts in Section 5.2.3.

Although cheap production is the ultimate goal, sometimes we have to utilize a Type-2 power plant at any cost to comply with the given constraints. Fortunately, the provided datasets suggest that Type-2 power plant production is generally cheaper than Type-1 power plant production. But, this is nowhere stated explicitly and cannot be taken for granted. Nevertheless, we assume in this chapter that it is desirable for each Type-2 power plant to produce as much as possible and adjust our refueling strategy accordingly. This means that we always choose the maximum possible refueling amount and thus increase Type-2 power plant utilization. A more sophisticated refueling strategy is presented in Section 6.6.

5.2.1 Type-2 Power Plant Preprocessing

During each production campaign a certain amount of fuel has to be consumed to reach the fuel level limits that apply before or after the upcoming refueling (see CT 11). In Section 5.1.2.2 we introduced a custom constraint that assured this property between successive outages. In this preprocessing step we define this limit for each time step, relying on the custom constraint that the assignment problem is feasible.

We start at the last time step and move backward in time. Since we know all outage dates, we can assign a maximum allowed fuel level to each time step by assuming maximum production in each step and minimum refueling during each outage. See Algorithm 7 for a pseudo-code illustration of this preprocessing step.

Note that these maximum fuel levels impose a minimum production if the current fuel level comes close to its maximum.

Algorithm 7: PREPROCESS (plant i)

```
Output: Vector of upper bounds of fuel levels after production for each time step
begin
    // initialize the last element of the result vector
    $fuelmax_{\mathrm{T}} \longleftarrow \infty$
    foreach scheduled $k \in \mathcal{K}$ in descending order do
        // backward propagation of the maximum fuel level
        for $t = t^{+}_{i,k} - 1$ to $t^{*}_{i,k}$ do
            $fuelmax_t \longleftarrow \mathrm{PREVIOUS}(i, t, fuelmax_{t+1})$
        // apply fuel level limit after refueling
        $fuelmax_{t^{-}_{i,k}+1} \longleftarrow \min(\max(post_{i,k}), fuelmax_{t^{*}_{i,k}})$
        // reconstruct the fuel level before a minimum refueling
        $fuelmax_{t^{-}_{i,k}} \longleftarrow \frac{\mathrm{Q}_{i,k}}{\mathrm{Q}_{i,k} - 1}(fuelmax_{t^{-}_{i,k}+1} - \min(ref_{i,k}) - \mathrm{BO}_{i,k}) + \mathrm{BO}_{i,k-1}$
        // apply fuel level limit before refueling
        $fuelmax_{t^{-}_{i,k}} \longleftarrow \min(\max(pre_{i,k}), fuelmax_{t^{-}_{i,k}})$
    return $fuelmax$
end
```

5.2.2 Type-2 Power Plant Offers

Besides the maximum allowed fuel levels, we have to take care of several other constraints when assembling the offered production amounts of a Type-2 power plant. Firstly, offers have to comply with imposed power profiles if we under-run the fuel level threshold $\mathrm{BO}_{i,k}$. This is achieved by setting an imposed power level as the minimum offer. Since we track the fuel level of each time step, we can easily calculate this production level. Secondly, the maximum modulation over each cycle has to be respected. Since the auction should be done in ascending order of the time steps, the production levels of previous time steps are known, and with them the currently consumed modulation. Given the maximum allowed fuel levels of each time step, we can now formulate the offers.

Type-2 Power Plant Minimum Offers

Minimum offers ensure the validity of the found assignment. Therefore, we consider the following constraints here:

CT 6: If the power level is imposed, we calculate the exact level and set it as the minimum offer.

CT 11: If the current fuel level exceeds the maximum allowed fuel level of this time step, we are not able to reach the maximum fuel level before or after refueling of the next cycle. Therefore, we set a minimum offer, such that the fuel level is reduced to its maximum allowed value.

CT 12: If the consumed modulation of the current cycle (including the time step of the auction) would exceed the allowed maximum modulation, the minimum offer is set, such that the modulation equals $\mathrm{MMAX}_{i,k}$.

Algorithm 8 gives a pseudo-code notation of this process.

Algorithm 8: MINOFFER (plant i, cycle k, scenario s, time step t)

```
Output: Minimum required amount of production
begin
    $amount \longleftarrow 0$
    // Is this a production time step?  [CT3]
    if $t^*_{i,k} \leq t$ then
        // Imposed power profile active?  [CT6]
        if $fuellevel_{i,t} < \mathrm{BO}_{i,k}$ then
            $amount \longleftarrow \mathrm{PB}_{i,k}(fuellevel_{i,t}) \cdot \mathrm{PMAX}^t_i$
        else
            // Stay below maximum fuel level [CT11]
            if $fuellevel_{i,t} > fuelmax_{i,t}$ then
                $amount \longleftarrow (fuellevel_{i,t} - fuelmax_{i,t}) / \mathrm{D}$
            $consumed\_modulation = \sum_{\tau=t^*_{i,k}}^{t} (\mathrm{PMAX}^\tau_i - p(i,s,\tau)) \cdot \mathrm{D}$
            // Stay below maximum modulation [CT12]
            if $consumed\_modulation > \mathrm{MMAX}_{i,k}$ then
                $amount \longleftarrow \max(amount, (consumed\_modulation - \mathrm{MMAX}_{i,k}) / \mathrm{D})$
    return $amount$
end
```

Type-2 Power Plant Maximum Offers

Determining the maximum offers is easy. For time steps where a Type-2 power plant is offline or its production is imposed, the maximum production was already set by the minimum offer. During all other time steps, the only constraints limiting maximum production are CT 4, stating the maximum production of the plant, and CT 9, limiting the fuel level to non-negative values. Algorithm 9 assembles the maximum offered production amount.

5.2.3 Refueling Amounts

Finally, we have to decide how much fuel to reload during each outage. If the auction is run for a time step in which a refueling for plant i has to be performed, we know i's current fuel levels in all scenarios, because the previous auctions have already been executed. Since we calculated the maximum allowed fuel levels of all scenarios in the preprocessing step, we can determine a possible refueling interval for each scenario. The intersection of these intervals is the globally possible refueling interval of the considered outage.

Algorithm 9: MAXOFFER (plant i, cycle k, scenario s, time step t)

```
Output: Maximum available amount of production
begin
    $amount \longleftarrow 0$
    // Is this a production time step?  [CT3]
    if $t^*_{i,k} \leq t$ then
        // production not imposed?  [CT6]
        if $fuellevel_{i,t} \geq \mathrm{BO}_{i,k}$ then
            // choose minimal upper bound from [CT4/5] and [CT9]
            $amount \longleftarrow \min(fuellevel_{i,t}/\,\mathrm{D}, \mathrm{PMAX}^t_i - p(i,s,t))$
    return $amount$
end
```

Lemma 14. Given an arbitrary Type-2 power plant i and cycle k, the intersection of the scenarios' refueling intervals $r_{i,k}$ is not empty.

Proof. Since the realized fuel level in each scenario is less or equal its upper bound found in the preprocessing step, we can at least perform a minimum refueling without exceeding the maximum allowed fuel level of the next time step (see Algorithm 7). $\square$

We still have to decide which value to take from the global refueling interval of the outage. A higher amount of refueling probably results in higher utilization of the plant. This seems desirable, as the provided instances suggest that Type-2 power plant production is generally cheaper than production by Type-1 power plants. On the other hand, a larger refueling results in more residual fuel and thus a worse refueling difference, i.e., more lost fuel during refueling (see CT 10).

For now, we always choose the maximum possible refueling amount and postpone optimization to Section 6.6. Algorithm 10 demonstrates the calculation of the concrete value.

Algorithm 10: REFUELING (plant i, cycle k, time step t)

```
Output: The reloading amount to perform
begin
    $a \longleftarrow \mathrm{RMAX}_{i,k}$
    foreach scenario $s$ do
        $a \longleftarrow \min\left(a, fuelmax_{i,t^-_{i,k}+1} - \mathrm{BO}_{i,k} - \frac{\mathrm{Q}-1}{\mathrm{Q}}(fuellevel_{s,t^-_{i,k}} - \mathrm{BO}_{i,k-1})\right)$
    return $a$
end
```

5.3 Evaluation

In this section we present the first results of our approach. We evaluate the proposed branching strategies with respect to the solution quality. As the ROADEF/EURO Challenge 2010 limits program running times to 30 minutes (for A instances) and 60 minutes (for B and X instances), we always exploit this time. Thus, deterministic branchings (smallest domain branching, sweep branching) iterate through the CP solutions (i.e., outage schedules) and run the production assignment phase for each found schedule, always keeping the best solution. Contrary, random branching is restarted from the root of the search tree after it visited a certain number of failed spaces. As this maximum failure count, we choose twice the number of outages to assign. This number is big enough to

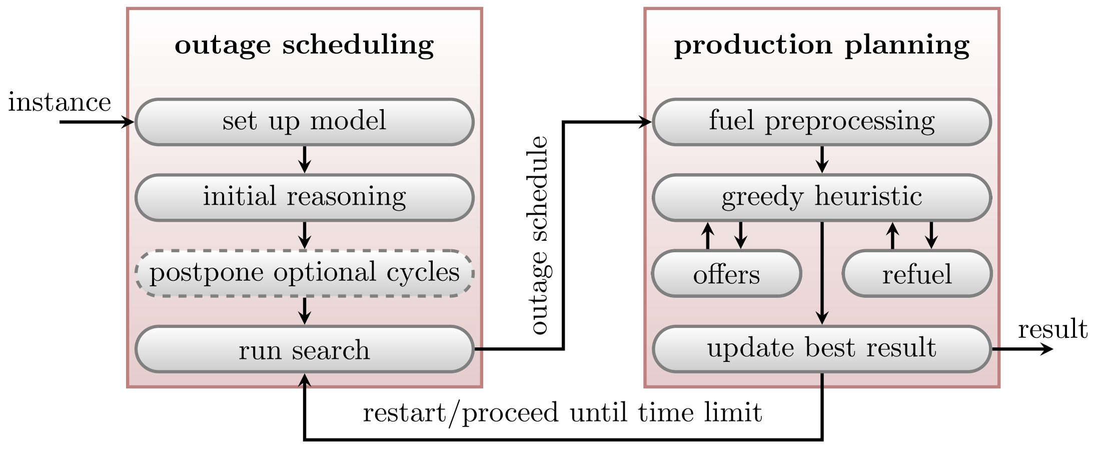

Figure 5.7: Illustration of the overall solution process. *Postpone optional cycles* is not applied when using sweep branching. Additionally, only random branching restarts the search from the beginning, while both other branchings iterate to the next solution.

ignore single failures and small enough to restart soon if it seems we search an infeasible region of the search tree. If the CP yields a valid outage schedule, we run the production assignment phase for the found schedule and also restart the CP from its root. See Figure 5.7 for an illustration of the overall process. All three presented branching techniques have very low computational overhead, as their choice depends solely on the domains of the decision variables ($dec^*_{i,k}$). For an introduction of our general experimental setup and the evaluated instances see Chapter 7.

Let us first have a look at the results of smallest domain branching. We evaluate all three options: minimum, median, maximum branching. The results obtained for all instances can be seen in Table 5.5. For most instances we get a fair *score* (the deviation from the best known solution, see Section 7.2 for a precise definition), with exceptions for the less constrained instances B08 and B09.

In 6 of 48 cases the branching was not able to find a solution in the given time. We suppose that in these cases an infeasible variable assignment was made early by the branching and there was not enough time to backtrack up to this point. As smallest domain branching works deterministic, repeating the experiment does not improve the situation.

Note that 9 of 15 best solutions have been obtained using the median value strategy. This is comprehensible, since neither very early nor very late decoupling dates are desired. Scheduling an outage too early leads to large amounts of residual fuel and more expensive refueling (cf. CT 10), while too late outages result in Type-2 power plants running out of fuel. There are also three (two) instances where late (early) branching outperforms the other strategies. In these cases, other factors like the number of parallel outages or the current unit cost of other power plants might influence the result stronger than the plants' fuel levels. Finally, we cannot generally say that it is good to schedule outages after a certain proportion of their available decoupling interval.

The second analyzed technique is random branching. Compared to both other branchings it provides pretty good results. We attribute this to the fact that our random branching is restarted regularly and therefore visits many more different regions of the search tree. Unfortunately, this technique fails quite often to find a feasible outage schedule (see Table 5.6). This means, we waste a lot of time evaluating ”dead” parts of the search tree.

As repeated random branching results in different outage schedules, we analyze again if there is a promising direction of decoupling dates (i.e., early or late). Therefore, we sum

Instance	MIN	MED	MAX
A00	0.16%	0.19%	**0.12%**
A01	2.11%	1.03%	**0.56%**
A02	2.37%	**0.89%**	0.96%
A03	1.96%	**1.41%**	INFEASIBLE
A04	**2.54%**	INFEASIBLE	3.27%
A05	INFEASIBLE	INFEASIBLE	INFEASIBLE
B06	**9.93%**	10.01%	13.86%
B07	18.52%	12.68%	**11.57%**
B08	4 716.79%	**4 647.15%**	4 805.49%
B09	4 605.46%	**4 027.56%**	4 224.70%
B10	11.45%	**9.53%**	9.94%
X11	10.35%	**7.01%**	7.89%
X12	10.57%	**6.82%**	8.27%
X13	14.37%	**11.23%**	INFEASIBLE
X14	18.75%	**12.65%**	15.66%
X15	9.38%	6.87%	**6.51%**

Table 5.5: Smallest Domain Branching: We compare the achieved scores (deviations from the best known results, cf. Section 7.2) by the strategies *MIN*, *MED* and *MAX* for all proposed instances. The best results are in bold font. Runs that did not give a valid solution in 30 minutes (set A) or 60 minutes (set B and X) are marked as infeasible.

Instance	Best Score	Iterations	Failure Rate
A00	0.12%	>1000	0%
A01	0.37%	>1000	8%
A02	0.61%	>1000	30%
A03	0.71%	>1000	17%
A04	1.83%	909	28%
A05	2.26%	>1000	64%
B06	8.16%	176	23%
B07	9.39%	178	12%
B08	4 554.66%	71	30%
B09	3 840.82%	66	20%
B10	8.25%	59	12%
X11	6.54%	217	37%
X12	6.04%	230	29%
X13	10.29%	189	65%
X14	12.00%	84	44%
X15	5.61%	90	28%

Table 5.6: Random Branching: We show the best achieved *score* (cf. Section 7.2) for all instances. Column *iterations* lists the number of restarts of our search procedure during the given time. *Failure rate* lists the proportion of iterations that did not provide a valid result before the next restart.

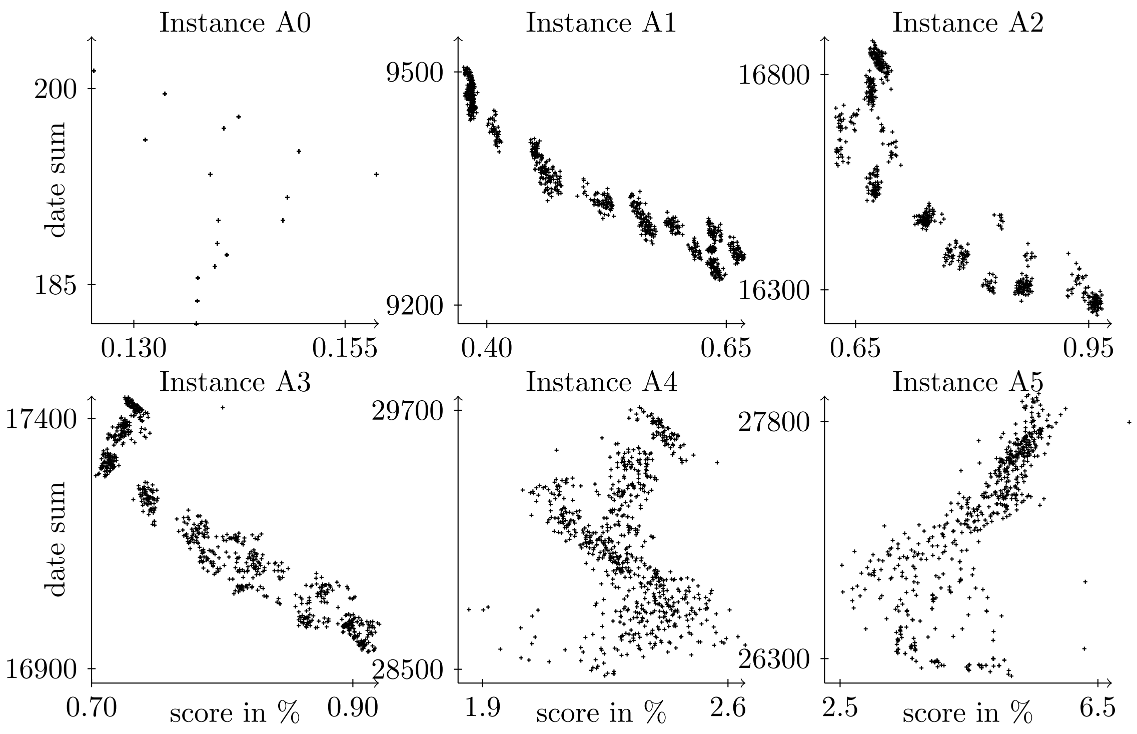

Figure 5.8: Random Branching: Results from around 1000 iterations on the A instances. We plot the *sum of decoupling dates* (not scheduled outages are set to W) against the obtained *score* (cf. Section 7.2). The higher the sum of decoupling dates the later the outages have been scheduled.

up all single decoupling dates and add a dummy value of W (number of weeks in the instance) for each not scheduled outage. This way, a higher sum can be interpreted as scheduling outages at a later date. The results are plotted in Figure 5.8.

Although there seem to be trends for each instance, these trends vary heavily. Later dates give better results for instance A1. Contrary, earlier dates provide cheaper schedules for instance A5. There is even one instance – A4 – without a clear trend. Note that these results coincide with the results obtained from smallest domain branching.

Next, we look at the results of sweep branching (see Table 5.7). Although it does generally not perform as well as random branching, we note that it is the only technique providing reasonable results for all instances. Its main advantage is that we can postpone the decision which outages/cycles we want to schedule and which not. Furthermore, this technique branches into very little failed spaces, since the decision variables are assigned in ascending order (i.e., $dec^*_{i,k}$ before $dec^*_{i,k+1}$). This is the same order in which these variables depend on each other and therefore, each decision immediatelly takes effect on all the "later" variables of a plant. This results in better propagation and thus less failed spaces.

Finally, we compare the results of all three branching techniques in Table 5.7. We notice that 14 out of 16 best results have been obtained using random branching. Both, smallest domain branching and random branching, fail at instances B08 and B09. These instances have a lot of optional cycles, which are not scheduled by these branchings. Sweep branching solves these instances much better, since optional cycles are only not scheduled if they become impossible cycles while searching. Therefore, we will put further effort into determining promising decoupling dates.

Instance	Domain	Random	Sweep
A00	**0.12%**	**0.12%**	0.16%
A01	0.56%	**0.37%**	2.12%
A02	0.89%	**0.61%**	2.34%
A03	1.41%	**0.71%**	1.95%
A04	2.54%	**1.83%**	2.50%
A05	INFEASIBLE	**2.26%**	6.54%
B06	9.93%	**8.16%**	11.53%
B07	11.57%	**9.39%**	16.53%
B08	4 647.15%	4 554.66%	**17.88%**
B09	4 027.56%	3 840.82%	**27.90%**
B10	9.53%	**8.25%**	11.90%
X11	7.01%	**6.54%**	10.57%
X12	6.82%	**6.04%**	10.58%
X13	11.23%	**10.29%**	14.86%
X14	12.65%	**12.00%**	20.43%
X15	6.51%	**5.61%**	9.94%

Table 5.7: Overview of the three basic branching techniques: smallest domain branching (*Domain*), random branching (*Random*) and sweep branching (*Sweep*). We show the best scores for all instances achieved by each branching.

6. Optimization

From the last chapter we learned that the considered search spaces are far too big to be fully explored. Thus, it is not satisfactory to rely on the current model and basic branching techniques to receive good results. In this chapter we examine strategies that provide more promising outage schedules and improve production assignment.

In the first three sections, we focus on the branching part and do not change the model set up in Section 5.1. We reuse ideas about the lower bounds presented in Chapter 4. In Section 6.1 we introduce a branching that schedules outages to somehow cheap dates, by considering the additional cost of each outage. A similar idea is presented in Section 6.2 but not further elaborated, since first results have been rather bad. Afterwards we present our most competitive branching technique so far, sweep margin branching, in Section 6.3. It combines ideas from the Sections 5.1.4.3 and 6.1.

In the remaining sections we restrict our model to a smaller search space or alter the way of search. In Section 6.4 we propose a new constraint that limits decoupling dates to reasonable intervals. Furthermore, we refine the calculation of the additional cost in Section 6.5. Finally, Section 6.6 introduces an improved refueling strategy that repeats the production assignment phase and utilizes knowledge from the first run throughout the second.

Each proposed optimization technique is added to our model and employed in the following optimization steps unless otherwise stated. Along with the presented optimizations we will shortly evaluate their solution quality, focussing on improving our best achieved results.

6.1 Incremental Cost Branching

Although all branchings presented yet are able to find an optimal solution given unrestricted time, the best results obtained in Section 5.3 come from a bulk random search. Obviously, choosing outage dates randomly is unlikely to provide a best choice. Therefore, we present a new randomized branching that selects promising outage dates with higher probability. It augments the random selection by the additional cost incurred, when scheduling an outage, i.e., making a Type-2 power plant unavailable. To model additional cost we will use a simplification of the power assignment phase, with adaptive calculation for each outage fixation.

The basic idea of this branching is to fix outages first that potentially increase the solution cost the most and schedule them to their cheapest date. We assume that a Type-2 power

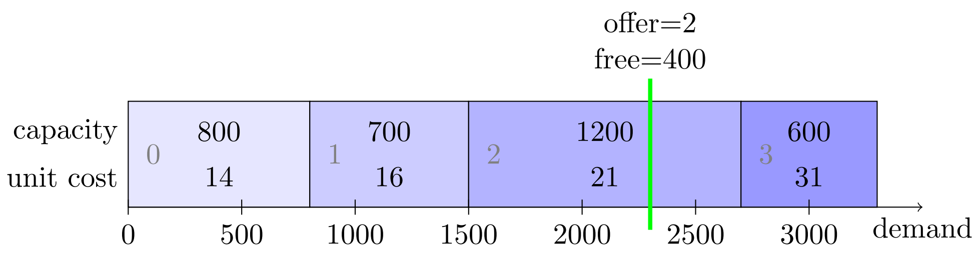

Figure 6.1: Ordered and stacked list of offers for a single time step and scenario. The green line marks the margin offer for a demand of 2300 units. We store the index of the offer and the remaining capacity.

plant, whose outage has to be scheduled, is among the cheapest plants and its production level is maximal in all time steps, i.e., equals PMAX_i^t. Although this is nowhere stated explicitly, the provided datasets support this assumption. Scheduling this outage to any specific decoupling date would now increase the overall cost of production, because the plant's production during the shut down time steps has to be shifted to other – more expensive – plants. So, it is desirable to assign outages to dates where this additional cost is small.

By calculating the least additional cost for each yet unassigned outage, we can identify the critical outages (i.e., outages which might lead to high production cost in the end) and schedule them first. This is appropriate, because scheduling this outage later will result in even higher additional cost, as more Type-2 power plants will not be available. Note that an outage with high additional cost for all possible decoupling dates is not necessarily a critical one, since we have no real option to achieve a good result.

The branching is set up as follows. First, to reduce the number of failures, we only schedule mandatory cycles (cf. Section 4.3.2). Afterwards we preprocess the additional cost and start the search. Creating a choice of this branching is a two-step process. First, for each unassigned outage, we estimate the additional cost of each possible decoupling date and select a cheap date. Each outage is now annotated with a good decoupling date and the corresponding additional cost. In the second step, we choose an outage with high additional cost. This outage and its selected decoupling date form the choice of the branching step.

In the following we present all parts of the branching. First, we introduce the preprocessing step and the concept of margin offers in Section 6.1.1. After that, we present the choice of the best decoupling dates and best outage in the Sections 6.1.2 and 6.1.3. Finally, in Section 6.1.4 we show how to adapt data structures when committing a choice.

6.1.1 Preprocessing Margin Offers

To estimate the additional cost that arise from scheduling an outage, we extend the auction–based approach, which was presented in Section 4.2. Remember that we distributed the demand among the cheapest power plants there. This time, we are rather interested in the most expensive power plant still utilized for production than in the globally resulting cost. We will call this most expensive offer that we need to produce the required demand a *margin offer*. Information about this offer (or power plant respectively) and its current utilization is the key to determine cost differences when the demand changes. Of course, demand does not really change, but putting a Type-2 power plant on outage is basically the same as increasing the demand by the plant's production capacity, assuming that the plant was fully utilized before.

When this branching is initialized, we collect the maximum production offers of all power plants for each time step and scenario. We sort these lists ascending by their unit cost and

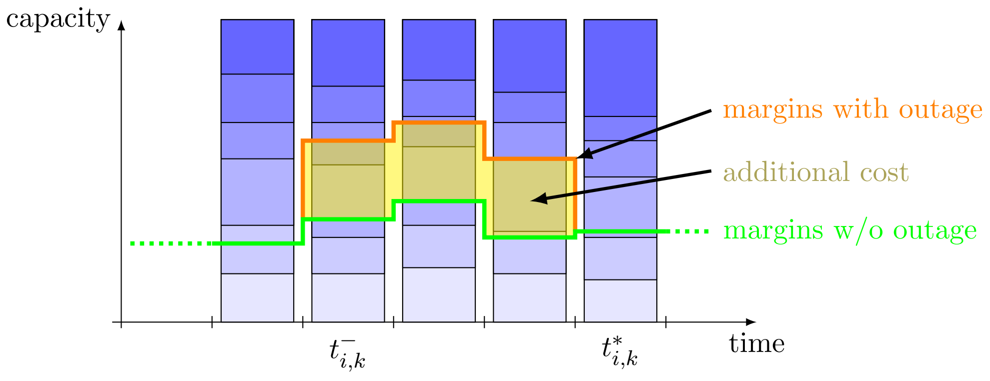

Figure 6.2: Illustration of additional cost that arise when scheduling an outage. Offers are represented by blue boxes ordered by ascending cost (light blue = cheap, dark blue = expensive). Offers below the green line are used to produce the demand. When putting i on outage between the time steps $t^-_{i,k}$ and $t^*_{i,k}$, its production capacity has to be transfered to the next more expensive offers (yellow region), resulting in new margin offers (orange line).

calculate the margin offers by a greedy distribution of production at each time step and scenario among the cheapest facilities. Figure 6.1 illustrates this for a single time step.

Using margin offers, we want to quantify the additional costs, when scheduling an outage at a certain week. Additional costs arise because the plant's production levels during the offline time steps have to be shifted to the next (i.e, more expensive) plants. This is illustrated in Figure 6.2. Aggregating these additional costs over all affected time steps of a specific decoupling date gives the additional costs of the whole outage.

Calculating the additional costs of a specific demand increment is easy, given an ordered list of offers and the current margin offers. We have to utilize the more expensive and yet unused plants. Technically we increase the margin offer by the additional demand and sum up the induced cost. Algorithm 11 gives a pseudo-code notation of this process.

Algorithm 11: INCREMENT (*demand*, *offer*, *free*)

```
Input: Current margin offer and additional demand
Output: Cost of additional demand cost and resulting margin offer (offer, free)
begin
    $cost \longleftarrow 0$
    while $demand > free$ do
        $cost \longleftarrow cost + unit\_cost_{offer} \cdot free$
        $demand \longleftarrow demand - free$
        $offer \longleftarrow$ next expensive offer
        $free \longleftarrow capacity_{offer}$
    $cost \longleftarrow cost + unit\_cost_{offer} \cdot demand$
    $free \longleftarrow free - demand$
end
```

Note that this way of calculating the additional cost tends to give higher costs for long lasting outages. Consequently our heuristic fixes the decoupling dates of these cycles before those with shorter outages. This is even advantageous, because it becomes harder to find a feasible decoupling date for a long lasting outage as more outages have already been scheduled to a fixed date.

6.1.2 Outage Date Selection

In this section we focus on the selection of a best date of decoupling for a single outage. Furthermore, we add a cost value to this date, making it comparable to other outages.

We perform Algorithm 11 for each potential offline time step and aggregate the results for each decoupling date (i.e, all weeks from the interval $[ha(i,k), ha(i,k)+\mathrm{DA}_{i,k}])$. Hence, we get a set of tuples consisting of decoupling dates and the corresponding additional costs.

Among these dates we are interested in selecting dates, which incur low additional costs, with higher probability. Therefore, we choose the k-th best out of n decoupling dates with probability $\mathbf{P}[k] := 0.5^k$. The worst date is chosen with the residual probability $\mathbf{P}[n] := 1 - \sum_{k=1}^{n-1} \mathbf{P}[k] = 0.5^{n-1}$.

To choose the cost of a selected decoupling date, we will try four measures. Remember that outages, which are more likely to produce higher additional cost, should get a high cost value assigned now. As the cost of the outage represents the set of possible dates, we calculate it independently of the chosen decoupling date. The proposed measures are:

MAX: The most expensive decoupling date. Note that this measure does not care if all decoupling date options are equally bad.

MED: The median of all available decoupling dates' costs. This measure is resistant against outliers, but requires the most computational effort.

RANGE: The cost difference between the cheapest and most expensive decoupling date.

REGRET: The difference between the average and minimum costs. This measure provides a good estimation of the proportion of cheap decoupling dates.

6.1.3 Outage Selection

In the previous section we set up the choice of a best decoupling date for each outage and an associated cost value. We now have to choose one of these outages for branching. Since our primary goal is to save cost, we have to choose the outage with the highest cost here. Remember that additional costs are monotonically increasing while we schedule more outages, hence an outage with already high costs matters most. Again, we order all outages by their cost values and choose the more expensive outages with higher probability. Technically, we reuse the distribution function presented in Section 6.1.2.

6.1.4 Committing a Choice

Until now, we showed how to choose a probably good outage and one of its decoupling dates for a branching step. In this section we illustrate how the search space is splitted and changed when committing a choice.

A choice splits the search space into two branches. In the first we assign the chosen decoupling date to the specific outage, while we exclude this date from the domain of $dec^*_{i,k}$ in the second branch (see Figure 6.3). Technically this is done by adding further constraints (e.g., $dec^*_{i,k} = x$) to the copied search space, when the `commit` procedure is run for a specific alternative.

After adapting our model, we need to update the internal state of the margin offers in both branches. In the first branch, where we assigned the decoupling date, we execute Algorithm 11 for all time steps of the scheduled outage, increment the demand by the now shut down production capacity and adjust the margin offers accordingly. In the second branch we do not change the margin offers, as there was no outage actively scheduled. Note that we will miss outages, which become assigned by constraint propagation.

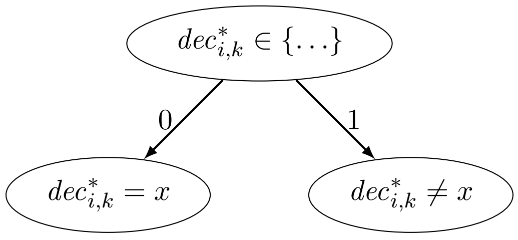

Figure 6.3: Branching alternatives by adding constraints. While we assign the chosen outage date x in the left branch, we exclude it from the search space of right branch.

Instance	Best Score	Iterations	Failure Rate
A00	**0.05%**	>1000	0%
A01	**0.12%**	>1000	1%
A02	**0.19%**	>1000	11%
A03	**0.37%**	>1000	58%
A04	**0.62%**	315	4%
A05	**1.19%**	177	47%
B06	**7.36%**	47	24%
B07	**8.50%**	32	9%
B08	4 674.42%	25	0%
B09	4 011.19%	21	0%
B10	**6.17%**	10	2%
X11	**5.72%**	38	5%
X12	**3.88%**	40	0%
X13	**9.73%**	8	35%
X14	**9.67%**	8	0%
X15	**2.84%**	15	0%

Table 6.1: Incremental Cost Branching: We show the best achieved *score* (cf. Section 7.2) for all instances. *Iterations* lists the number of restarts, *failure rate* the proportion of iterations without a result. New best results are in bold font.

6.1.5 Evaluation

In this section we evaluate our results obtained by using incremental cost branching instead of the previously presented branching techniques. We start with the analysis of the proposed cost measures. In Figure 6.4 we show our experimental results on the A instances. Obviously, none of the measures has outstanding performance when compared to the others. Accordingly, we assume that of order of the variable assignments (which is the effect of the cost measures) has much less influence on the quality of an outage schedule than the choice of a good decoupling date. From this point of view we might be able to reduce the complexity of the choice of the next outage to assign and still get results of a similar quality.

The best results obtained for all instances by using incremental cost branching are shown in Table 6.1. The results are generally better than those presented in Section 5.3, but suffer the same problems as random branching with the less-constrained instances B08 and B09. Having a look into a concrete solution of B08 (see Figure 7.1), this can clearly be attributed to our decision to postpone all optional outages.

The calculation of additional costs requires considerable effort, i.e., the number of rounds that can be performed in the given time drops by a factor of around 5 when compared to

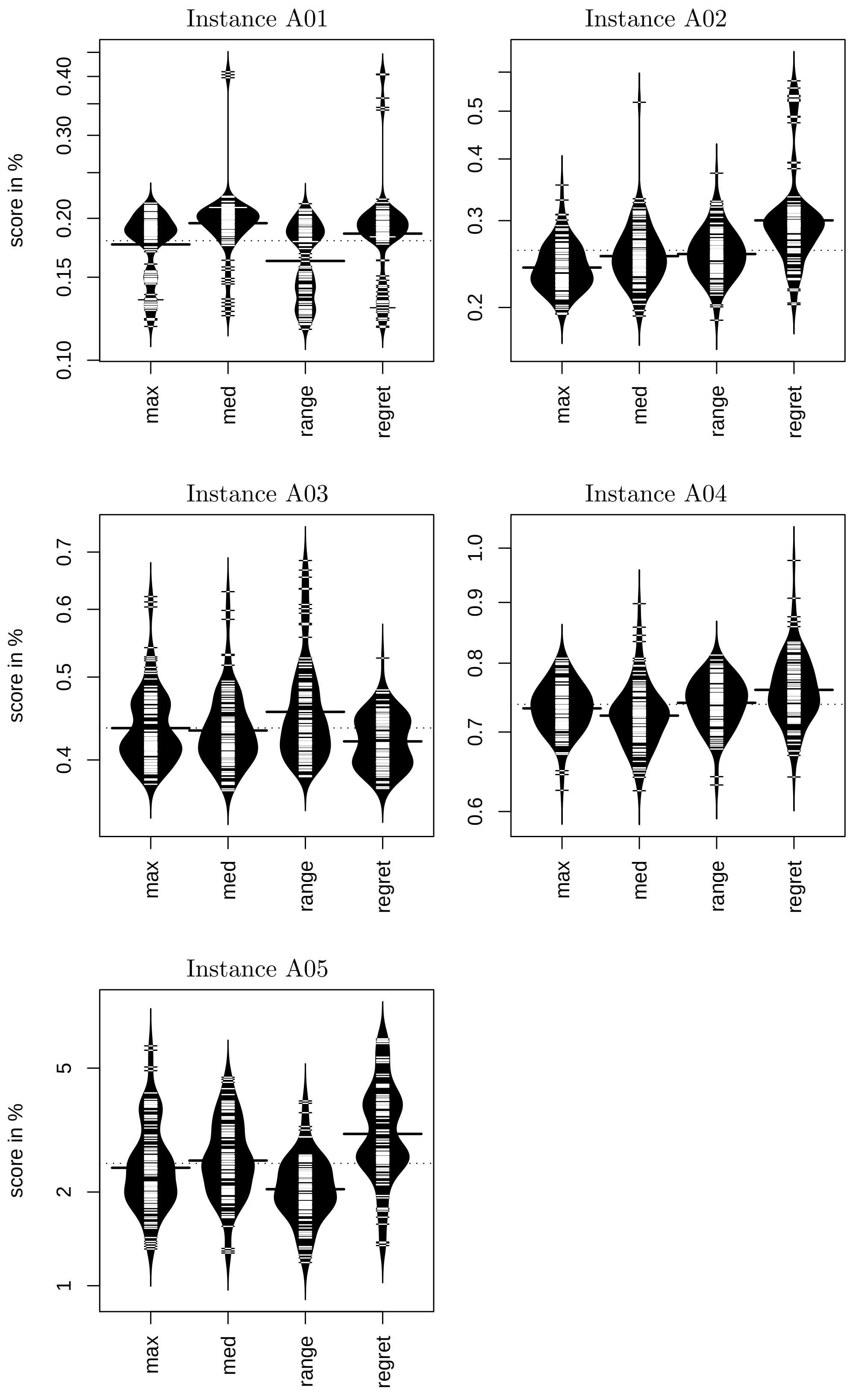

Figure 6.4: Quality of cost measures: We compare the results obtained by the 4 proposed cost measures: Maximum, Median, Range and Regret. The plots show the distribution of the obtained scores from 100 runs. The centered ticks at each measure mark single results. Thick horizontal lines mark the mean achieved by each strategy. We omit the dummy-instance A00.

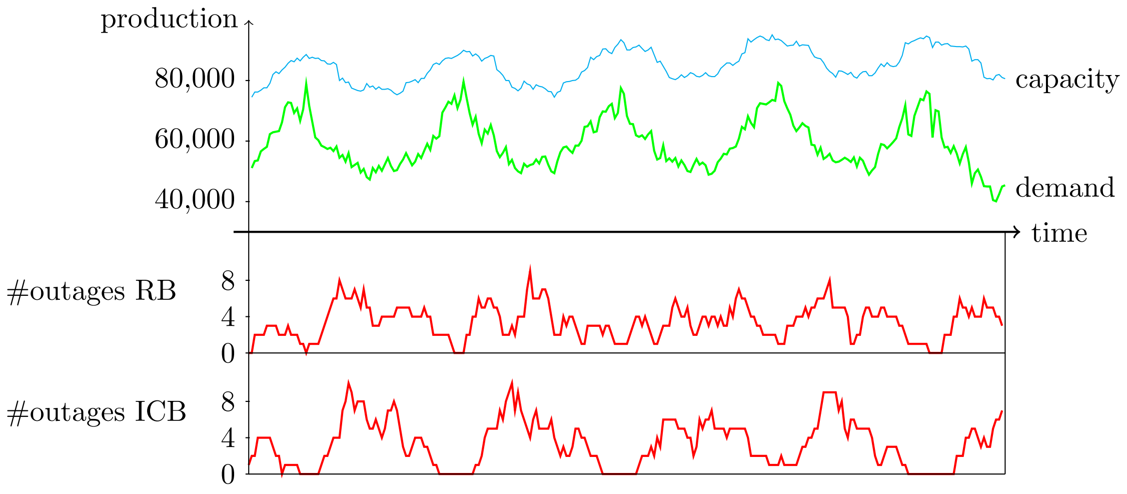

Figure 6.5: Histograms of the best outage schedules found by random branching (RB) and incremental cost branching (ICB), compared to the capacity of all power plants and the demand. We see that ICB schedules outages towards times of low demand. The data are taken from scenario 0 of instance A05.

random branching. Furthermore, this branching technique results in less failures than pure random branching. We guess this is caused by the fact that several outages which might be performed in an expensive region are assigned by the branching shortly after another. This results in earlier detection of infeasible states, because the majority of constraints – the spacing constraints – apply to outages which are scheduled within a short date range. Remember the sweep branching (SB, cf. Section 5.1.4.3) that assigned outages to their earliest possible date in ascending order, i.e., it also schedules outages with similar ranges of their potential decoupling dates shortly after another. As SB also shows the smallest failure rate among our basic branchings, we will combine SB and ICB in Section 6.3.

Finally, we analyze why this branching performs better. All previously presented branchings select outage dates without considering anything else than the variable's domain. Therefore, these dates become somewhat equally distributed. Contrary, ICB considers margin offers when selecting a date, which tend to have higher values when there is little unused production capacity. In Figure 6.5 one can clearly see how ICB scheduled the outages towards times with low demand and thus high unused capacity.

6.2 Decremental Cost Branching

Similar to incremental cost branching, we developed decremental cost branching, which works in opposite direction. In the beginning, we assume each Type-2 power plant is not available for production during all time steps where it might be offline, i.e., between $min(dec^*_{i,k})$ and $max(dec^*_{i,k})+\mathrm{DA}_{i,k}$. We set up the margin offers accordingly, i.e., they are higher than in a final solution. Branching again limits the domains of the $dec^*_{i,k}$ variables, thereby reducing the number of time steps a Type-2 power plant is not available. For all newly available time steps we decrement the margin offers by the production capacity of the plant. In each branching step, we choose the outage that reduces the margin offers the most.

In our first experiments this branching did not perform as well as the incremental cost branching and was therefore dismissed.

Instance	Best Score	Iterations	Failure Rate
A00	0.10%	>1000	0%
A01	0.22%	>1000	0%
A02	0.38%	>1000	1%
A03	0.53%	>1000	0%
A04	0.93%	653	52%
A05	3.60%	521	0%
B06	8.60%	129	1%
B07	9.87%	123	0%
B08	1 941.21%	46	0%
B09	1 318.26%	45	0%
B10	8.69%	48	1%
X11	5.97%	126	19%
X12	5.14%	129	0%
X13	**8.82%**	46	0%
X14	10.25%	47	3%
X15	4.59%	49	0%

Table 6.2: Sweep Margin Branching: We show the best achieved *score* (cf. Section 7.2) for all instances. *Iterations* lists the number of restarts, *failure rate* the proportion of iterations without a result.

6.3 Sweep Margin Branching

Up to now we created two branching strategies with competitive features. The first one, sweep branching (SB, cf. Section 5.1.4.3), is able to provide comparably fair results for all datasets in a short time. The other one, incremental cost branching (ICB, cf. Section 6.1), gives the best results yet, but is slow and often gets stuck in failed spaces, thus wasting a lot of time. Most failed spaces arise when we schedule the $k-1$ and $k+1$ cycles of a Type-2 power plant and constraint propagation does not fail until we try to fix the k-th cycle. When fixing the cycles in ascending order, we evade this trap and might receive far less failed spaces.

In this section we propose a new branching that combines the beneficial features of SB and ICB. Again we create our choice in two separate steps. First, we select the outage to be branched as proposed by SB, i.e., we choose the outage with the minimal domain minimum. Next, we choose the decoupling date of this outage according to ICB. Therefore, decoupling dates are ranked by their additional cost and cheaper dates are chosen with higher probability.

Remember that the computational effort of ICB is quite high, because all additional cost values have to be recomputed in each iteration. For the current branching we will dramatically simplify this approach. We still collect the margin offers in a preprocessing step, but do not adapt them during branching. This way the additional costs for a decoupling date stay the same for the whole search process. As the additional costs do not change, they can also be calculated and ordered once before the search.

Another advantage is that we do not need to decide which outages have to be scheduled before search. We simply postpone a cycle as soon as its first possible date is restricted to a week behind the time horizon.

6.3.1 Evaluation

In this section we present our results obtained by using sweep margin branching (SMB) and compare it to both its ancestors RB and ICB. As can be seen in Table 6.2 SMB generally yields better results when compared to sweep branching (see Table 5.7). This can be attributed to the consideration of margin offers. Contrary, we receive worse results compared to ICB, which is caused by not updating the offers. Therefore, the more outage dates we assign the more our fixed margin offer's cost differs from the real margin offer's cost, thus resulting in a worse choice.

We note that SMB is around three times as fast as ICB. We attribute this to the reduced computational overhead of the branching's choice. The achieved failure rate is negligible for most instances and thus we can evaluate much more outage schedules in the given time (which is probably the only reason why we could improve the result of X13). We attribute the low failure rate to the order of variable assignments, which this branching inherited from SB. Again, propagation can take full effect when we assign the decoupling dates of a plant one after another. However, there are two instance (A04 and X11) with very high failure rates. This most probably results from some structural properties of the instances, i.e., an early cycle with an infeasible outage date that is not properly detected. Consequently, all runs of the solver that branch into this subtree will fail. An idea to approach this problem might be a statistical analysis of the decoupling dates of successful and failed traversals for each variable.

Again, the instances B08 and B09 are not satisfactory solved. Remember their large number of optional outages. As optional outages tend to have more possible decoupling dates (their upper bound equals the time horizon), there are probably also later dates with low margin costs. As SMB prefers cheap dates, the plants run out of fuel far before the scheduled outage and are not available anymore. Since this peculiarity is probably also present in the other instances, we will track this problem in the upcoming sections.

6.4 Online Propagator

Up to now the solution process is affected only by the instance's constraints. In this section we propose a new instance-independent constraint for our model that limits decoupling dates to promising intervals. Note that by adding this constraint the CP model might become infeasible. Although this problem did not occur in any instance, we keep it in mind. So, if we do not find a single solution in reasonable time, we just rerun our CP model without this constraint.

Remember the Outage Date Propagator presented in Section 5.1.2.3. For each scheduled outage, this propagator limits the earliest decoupling date of the next outage. The limit is chosen such that there is enough time to reduce the fuel level to the allowed upper bound at the beginning of the next outage. We will now also limit the latest decoupling date of this next outage.

Let us assume that there is a Type-2 power plant during a production campaign and its next decoupling date is not limited by an upper bound. Sooner or later this plant will run out of fuel and not be available for production anymore. But, if the plant is offline anyway, we can schedule an outage at this date without further loss. Consequently, we set the first week where we definitely run out of fuel as an upper bound for the next decoupling date (see Figure 6.6).

Note that our currently chosen latest possible decoupling date is a very conservative assumption. Most of the time Type-2 power plants are heavily utilized. Hence, they will run out of fuel earlier, because they do not exhaust their allowed maximum modulation

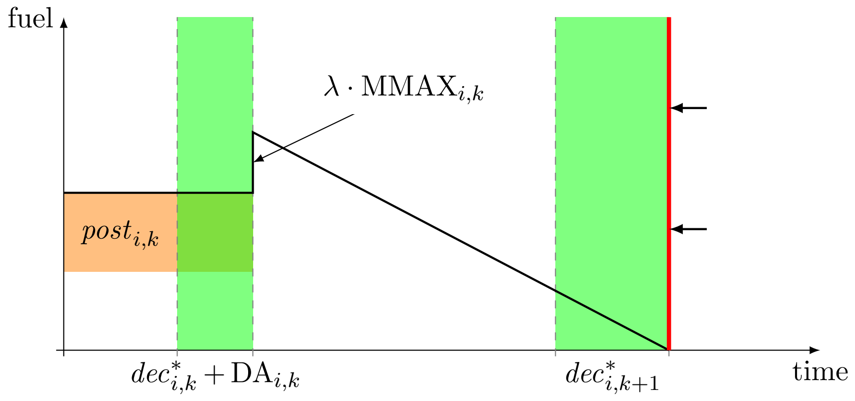
fuel
$\lambda \cdot \mathrm{MMAX}_{i,k}$
$post_{i,k}$
$dec^*_{i,k} + \mathrm{DA}_{i,k}$
$dec^*_{i,k+1}$
time

Figure 6.6: Reasoning by the Online Propagator: The latest possible decoupling date of the upcoming outage is limited to the week where the plant runs definitely out of fuel.

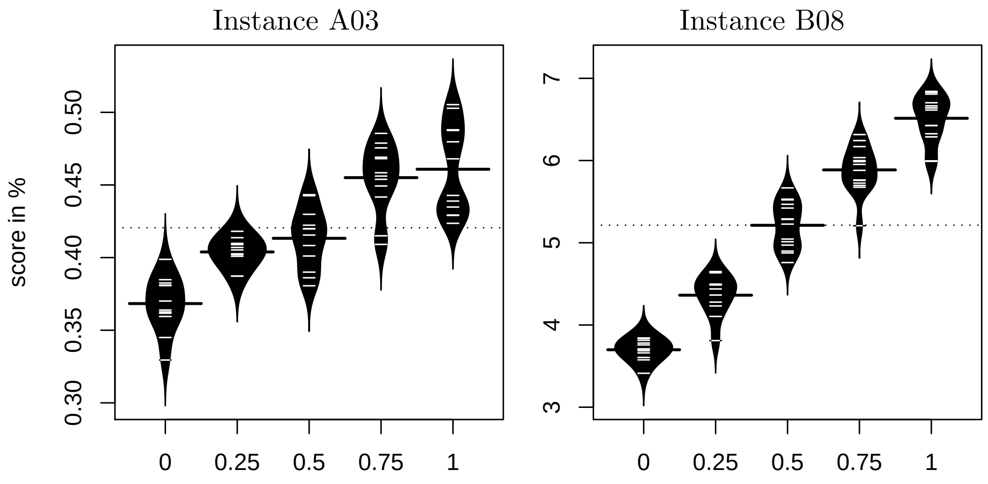

Figure 6.7: Online Propagator: We compare the results obtained for different values of $\lambda \in \{0, 0.25, 0.5, 0.75, 1\}$. The plots show the distribution of the obtained *scores* (cf. Section 7.2) from around 15 runs per λ.

$\mathrm{MMAX}_{i,k}$ completely. Therefore, we add a parameter $\lambda \in [0, 1]$ to this constraint which represents the exhausted proportion of $\mathrm{MMAX}_{i,k}$. In our experiments we will evaluate different values for λ.

To determine the latest decoupling date, i.e., $\max(dec^*_{i,k+1})$, we perform a fuel level forward propagation (cf. Section 5.1.2.2). As the maximum production of a plant is achieved with a post-refueling fuel level of $\max(post_{i,k})$, we choose $\max(post_{i,k}) + \lambda \cdot \mathrm{MMAX}_{i,k}$ as the initial fuel level of the production campaign and perform maximum production, which starts at the latest possible coupling date $\max(dec^*_{i,k}) + \mathrm{DA}_{i,k}$. We set the first week where production is ceased (see CT 6) as the upper bound of $dec^*_{i,k+1}$.

We notice that there is one special case, where the current setup will result in failed spaces. This happens if the plant runs definitely out of fuel before the earliest possible next decoupling date. So, in this case we do not commit the infeasible upper bound but assign $dec^*_{i,k+1}$ to its domain minimum.

6.4.1 Evaluation

In this section we discuss the results obtained after adding the Online Propagator to our model. First we analyze the impact of λ. In Figure 6.7 we picked two sample instances

Instance	Best Score	Iterations	Failure Rate
A00	0.05%	>1000	0%
A01	0.17%	>1000	0%
A02	0.28%	>1000	0%
A03	**0.33%**	>1000	0%
A04	0.72%	606	48%
A05	**0.63%**	529	0%
B06	**5.92%**	123	2%
B07	**4.34%**	119	0%
B08	**6.60%**	47	0%
B09	**6.63%**	45	3%
B10	**5.51%**	48	2%
X11	**4.94%**	73	75%
X12	**3.76%**	126	1%
X13	**5.35%**	47	0%
X14	**5.37%**	45	3%
X15	2.85%	47	0%

Table 6.3: Online Propagator: We show the best achieved *score* for all instances. *Iterations* lists the number of restarts, *failure rate* the proportion of iterations without a results. New best results are in bold font.

and evaluate their performance for varying proportions of modulation. We identify a clear trend: less modulation yields better results. There are probably two reasons for this effect. Firstly, Type-2 power plants are fully utilized most of the time and thus consume their fuel as fast as possibe. Secondly, the chosen fuel level before production $\max(post_{i,k})$ is an overestimation. This seems plausible because the upper bound of $post_{i,k}$ assumes minimum production in the previous cycles, which is probably not the case.

In the following experiments we will always assume $\lambda = 0$.

Before, let us have a look on the results obtained after adding the propagator (see Table 6.3). First of all, we note that we improve the results of B08 and B09 up to a similar quality as our other solutions. Additionally, we can considerably improve the results of most other instances. The computational effort is nearly unchanged. So, although the new constraint requires additional effort for propagation, this can be compensated by the smaller search space.

6.5 Residual Fuel Penalty

Choosing outage dates based on their additional cost yields our best results yet. This is why we further analyze this strategy. Looking at the refueling formula CT 10 we realize that we do not consider fuel levels before refueling as a factor of additional cost. But, the higher this fuel level is, the more fuel is lost in the next refueling operation.

In the following we will analyze the influence of the fuel level before refueling to the additional cost and increase this cost by an appropriate *penalty*. First, we set up a lower bound of residual fuel (see Figure 6.8). We choose the minimum initial fuel level before production and perform maximum production, thus determining the minimum residual fuel level for each week of the domain of $dec^*_{i,k+1}$.

We now analyze the consequences on the objective function caused by different fuel levels before refueling. Let us assume two different fuel levels $f_1 = 0$ and $f_2 = x(i, s, t^-_{i,k})$ before

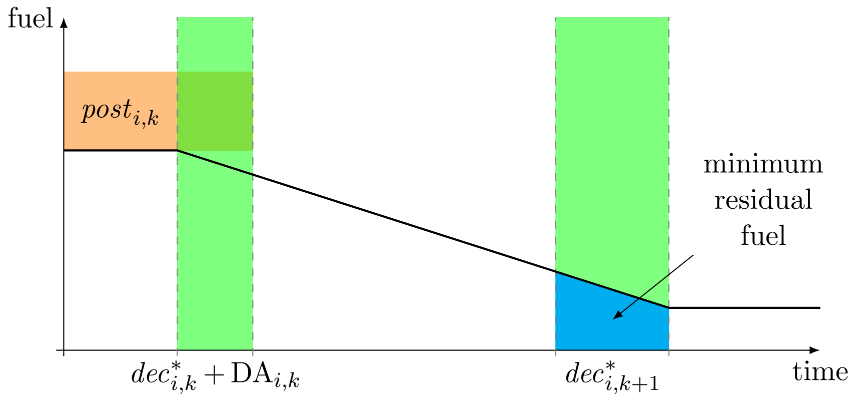

Figure 6.8: For each possible decoupling date of $dec^*_{i,k+1}$ we determine the minimum residual fuel level.

refueling. After reloading the same amount of fuel, the difference between both fuel levels reduced by $x(i, s, t^-_{i,k}) / \mathrm{Q}_{i,k}$ (see CT 10). We call this difference the *loss* of fuel. According to the objective function each unit of fuel is worth $\mathrm{C}_{i,k}$. Hence, we add the following penalty to the additional cost of each week with a residual fuel amount of f:

$$penalty(f) := \frac{\mathrm{C}_{i,k}}{\mathrm{Q}_{i,k}} \cdot f$$

After adding the penalty to the additional cost and resorting the decoupling dates, the used branching technique proceeds as normal. Note that this penalty calculation is only practical for sweep margin branching, because there the penalties have to be calculated only for the outage chosen for branching. With incremental cost branching we would have to calculate the penalties for all outages because the outage to branch is selected after evaluating all decoupling dates.

6.5.1 Evaluation

In this section we present the results obtained after adding residual fuel penalties to the choice of sweep margin branching (see Table 6.5). There is no significant impact on the failure rate because the penalty just modifies the probabilities of the decoupling dates and does not further limit the search space. Besides, this technique hardly influences the running time because the calculation is run only once at each branching step.

We can improve 4 of 16 solutions, but improvements are rather marginal. Contrary, the solution quality of all other instances becomes worse. We assume that the impact of the realized outage schedules on the objective function is much stronger than the impact of the residual fuel levels. Accordingly, we should not consider the amount residual fuel in the outage scheduling phase, but optimize the fuel levels afterwards. In the following, we will not make further use of the residual fuel penalty.

6.6 Tuning Fuel Levels

Remember our basic refueling strategy (presented in Section 5.2.3) to always choose the maximum possible refueling amount. This is fast to compute, but not desirable for several reasons. First, if the plant is operating at its maximum allowed fuel level, its production level is imposed in order to stay below the maximum allowed fuel level at the end of the production campaign (cf. Section 5.2.2). So, the plant might have to produce although it is not among the cheapest plants. Second, reloading more fuel is expensive and as we

Instance	Best Score	Iterations	Failure Rate
A00	0.05%	>1000	0%
A01	0.19%	>1000	0%
A02	0.28%	>1000	1%
A03	0.34%	>1000	0%
A04	0.76%	640	41%
A05	0.91%	585	0%
B06	**5.75%**	125	0%
B07	4.53%	122	0%
B08	**5.35%**	47	0%
B09	**6.53%**	44	2%
B10	5.73%	48	2%
X11	5.34%	55	60%
X12	4.05%	126	0%
X13	**5.10%**	45	0%
X14	6.43%	47	0%
X15	2.98%	49	0%

Table 6.4: Residual Fuel Penalty: We show the best achieved *score* (cf. Section 7.2) for all instances. *Iterations* lists the number of restarts, *failure rate* the proportion of iterations without a results.

reload more fuel, we probably get a higher fuel level at the end of the production campaign, thus resulting in a higher penalty for residual fuel (see CT 10 and Section 6.5). On the other hand, reloading too little amounts of fuel results in less production, and hence also higher cost.

To determine a good refueling amount, we set up a model to forecast the profit induced by a certain refueling amount. We will estimate this profit from the initial cost of refueling, the production amounts and a penalty for the residual fuel level. Note that the cost of the initial and residual fuel levels can be easily determined. The key idea to transfer production amounts into revenue is to run our production assignment phase twice. In the first run, we store the margin cost realized for each time step. We will use these cost in the second run as the price of each produced unit of energy.

```
Algorithm 12: PROFIT ($s$, $i$, $k$, fuel level $f$, refueling $r$, margins $c$)
Output: Estimated profit of the cycle when refueling $r$
begin
    $profit \leftarrow -r \cdot \mathrm{C}_{i,k}$
    $f \leftarrow$ fuel level, when refueling $r$
    for time step $t \leftarrow t^*_{i,k}$ to $t^+_{i,k}$ do
        $prod \leftarrow$ production with respect to the fuel level, modulation, unit cost, ...
        $profit \leftarrow profit + prod \cdot c_t$
    if next cycle is scheduled then
        $profit \leftarrow profit - \frac{\mathrm{C}_{i,k+1}}{\mathrm{Q}_{i,k+1}} \cdot f$
    else
        $profit \leftarrow profit + f \cdot \mathrm{C}_{i,\mathrm{T}}$
    return $profit$
end
```

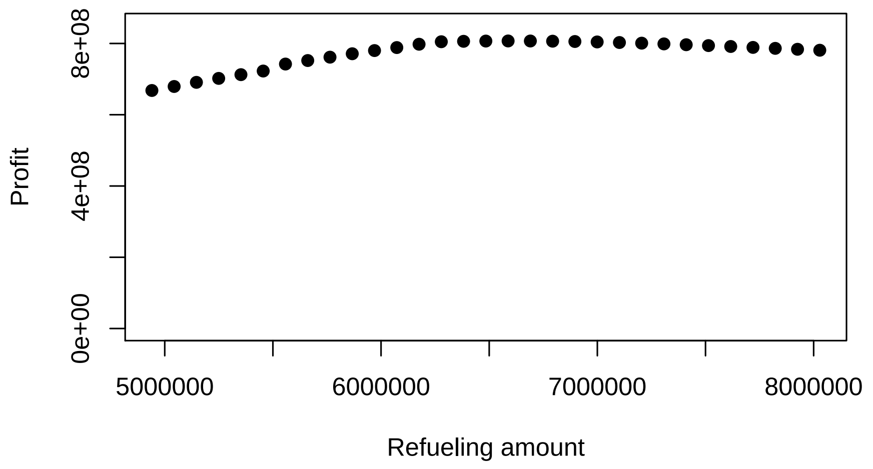

Figure 6.9: Fuel Level Tuning: A sample plot of the profit gained for different refueling amounts. The data is taken from A05, powerplant 0, cycle 1.

Note that at the instant we decide about the refueling amount, we already fixed the decoupling dates $dec^*_{i,k}$ and know the fuel level before refueling of the plant in each scenario. Algorithm 12 gives a pseudo-code notation, how to determine the profit for a given refueling amount.

As an analytical approach is hard, our heuristic will just simulate the profit for different refueling amounts (see Figure 6.9). We assume that the revenue function is concave. So, we perform a binary search to choose the refueling amount with the highest expected profit.

6.6.1 Evaluation

As this phase is independent of the outage scheduling phase it can be applied to all solutions obtained yet. Furthermore, our experiments show that this phase runs in less than a minute for the largest datasets (see Table 6.5). Therefore, we quit the outage scheduling and production assignment loop early enough and run this optimization step on the best found solution once before we reach the time limit.

We can improve the results of all but 4 instances. However, there are two instances (B08 and X13) where the refueling amounts are all fixed, such that this optimization technique can take no effect. We note that the improvements seem not to depend on the chosen outage schedule, as we achieve similar improvements when running this step for several found production plans.

As this is the final approach of our work, we give an overview of the robustness of the solutions obtained. In Figure 6.10 we plot the distributions of 50 single iterations (using SMB, OP and fuel level tuning) per instance. The variance is rather small and the median solutions yield an average *score* (see Section 7.2) of around 5.8%. Thus, our approach provides comparably good results already after the first few iterations.

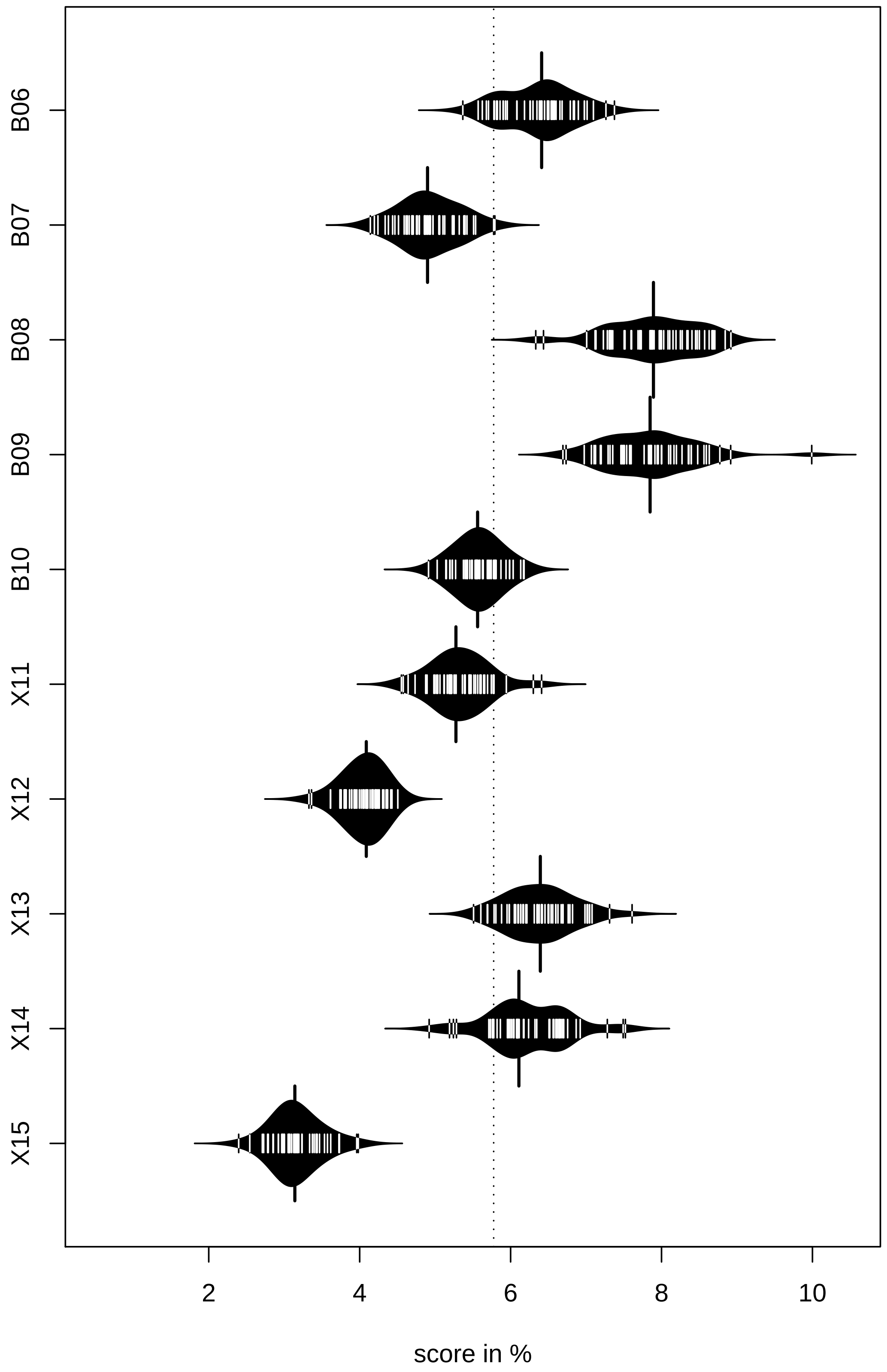

Figure 6.10: Robustness of our solution: We show the *scores* (cf. Section 7.2) of 50 iterations on the B and X instances. Ticks mark single solutions. The thick vertical lines mark the median solution of each instance. Note that the average expected score (dotted line) after one iteration of each instance is less than 6%.

Instance	Best Score	Running Time [sec]	Variability	Improvement
A00	0.05%	1	28%	0.00%
A01	**0.07%**	1	20%	-0.03%
A02	**0.12%**	2	21%	-0.04%
A03	**0.22%**	2	32%	-0.10%
A04	**0.46%**	5	24%	-0.14%
A05	**0.55%**	5	14%	-0.07%
B06	**5.37%**	17	21%	-0.13%
B07	**4.14%**	16	39%	-0.06%
B08	6.60%	44	0%	0.00%
B09	6.63%	43	22%	0.00%
B10	**4.91%**	42	38%	-0.52%
X11	**4.55%**	18	21%	-0.30%
X12	**3.33%**	16	38%	-0.34%
X13	5.35%	43	0%	0.00%
X14	**4.92%**	45	22%	-0.12%
X15	**2.40%**	44	38%	-0.37%

Table 6.5: Fuel Level Tuning: *Score* (cf. Section 7.2) presents the best solutions obtained. Column *variability* lists the average distance between the maximum and minimum allowed refueling amounts ($\mathrm{RMAX}_{i,k}$ and $\mathrm{RMIN}_{i,k}$) in the instance. The other columns show the average *running time* of the extra auction round, and the average *improvement* when applying this technique to results obtained without fuel level tuning.

7. Evaluation

We implemented the algorithms and methods which we describe in this thesis in C++ and evaluated their performance by using the instances provided by the ROADEF/EURO Challenge 2010. In this chapter we introduce the general experimental setup and give an overview of our experiments. As detailed evaluation is presented along with the development of our solution, we reference to the corresponding sections.

This chapter is organized as follows: First, we specify our implementation and testing environment in Section 7.1. Afterwards, in Section 7.2 we give an overview of the instances that have been used for evaluation and highlight their features. Section 7.3 summarizes the results obtained by our presented methods and optimization steps. We conclude with the best solutions found by our approach in Section 7.4.

7.1 Implementation and Testing Environment

We implemented of our solution in C++ and utilize two external libraries. First, for the solution of the outage scheduling phase (cf. Section 5.1) we extend the constraint programming toolkit Gecode 3.3.2 [Gec10]. A detailed introduction into Gecode is given in Section 2.1.1. To solve the Minimum-Cost Flow Problem of the network-based lower bound (cf. Section 4.3) we use the implementation of the Cost-Scaling Algorithm from the Lemon Graph Library 1.2 [Lem10].

The mentioned libraries and our code have been compiled with GCC 4.3, using optimization level 3. Our presented experiments have been performed on one core of a 2x dual-core AMD Opteron Processor 2218 clocked at 2.6 GHz, equipped with 1 MB L2 Cache per core and 32 GB RAM. We notice that this setup is slightly less powerful than the reference setup used in the competition (2x quad-core Intel Xeon 5420 at 2.5 GHz, 12 MB L2 Cache, 8 GB RAM). Our solutions are run as a single thread and do not consume more than 8 GB of memory per process.

7.2 Provided Datasets

Three datasets (A, B and X) have been provided by the ROADEF/EURO Challenge 2010. While the first two sets have always been available to us, the X set – which was meant for final evaluation of the challenge – was published two weeks before we finished this work. The A set contains 6 small instances, while the B and X sets comprise 5 big instances each.

Instance	# Power Plants		Scenarios	Weeks	Timesteps	# Constraints				
	Type-1	Type-2				13	14–18	19	20	21
A00	1	2	2	89	623	4	1	0	0	0
A01	11	10	10	250	1750	46	11	1	1	1
A02	21	18	20	250	1750	84	17	1	1	1
A03	21	18	20	250	1750	80	18	1	1	1
A04	31	30	30	250	1750	122	23	1	1	1
A05	31	28	30	250	1750	120	22	1	1	3
B06	25	50	50	277	5817	222	77	1	50	5
B07	27	48	50	265	5565	192	70	1	50	5
B08	19	56	121	277	5817	114	86	1	50	5
B09	19	56	121	277	5817	114	86	1	50	5
B10	19	56	121	265	5565	235	86	1	50	5
X11	25	50	50	277	5817	239	77	1	50	5
X12	27	48	50	263	5523	207	70	1	50	5
X13	19	56	121	277	5817	260	86	1	50	5
X14	19	56	121	277	5817	256	86	1	50	5
X15	19	56	121	263	5523	245	86	1	50	5

Table 7.1: Overview of the instances provided by the ROADEF/EURO Challenge 2010.

Their key figures are shown in Table 7.2. Note that the ratio between Type-1 and Type-2 power plants is nearly 1 : 1 in set A, while the sets B and X comprise 2-3 times more Type-2 power plants than Type-1 power plants. As most of our constraint types are related to Type-2 power plants the B and X instances require much more effort. Additionally, some instances contain more than 120 different demand scenarios, therefore resulting in more than 50 million decision variables in a single instance.

All instances – except the dummy-instance A00 – contain 6 cycles for each Type-2 power plant. From the original problem description we derive that a time step's duration in real time is between 8 and 24 hours. The demand scenarios show similar characteristics and seem realistic (e.g., clear seasonal oscillation). Besides, there is no artificial variability, as would anyway be unrealistic from a power network control point of view. Instances B08 and B09 are a bit special as they contain less outage date range constraints CT 13, which dramatically increases the search space, making it harder to find the best solution. Furthermore, in B08 and X13 all refueling amounts are fixed (i.e., $\text{RMIN}_{i,k} = \text{RMAX}_{i,k}$).

All instances contain one Type-1 power plant that is able to cover the complete demand – at extremely high cost though. This plant can be seen as a backup plant to keep the problems feasible. As it is never desirable to utilize this plant, we will omit it in all plots and figures.

To make solution qualities comparable, the ROADEF/EURO Challenge 2010 defines a so called *score* on each result, which corresponds to the result's deviation from the best known solution in percent. Assume that R is our solution and R' the best known solution so far, then the score is calculated as:

$$score(R) := \frac{cost(R)}{cost(R')} - 1$$

where *cost* returns the result's value of the objective function. In the competition teams are ranked by their average score for all B and X instances. Each unsolved instance gives a penalty of two times the score of the worst submitted solution.

Note that the objective function measures solutions in abstract monetary units. Since these are huge numbers and rather unhandy, we will always use the score (given in percent) when comparing our results.

7.3 Evolution of Results

In this section we show how our best solutions evolve while we add the presented optimization techniques (see Table 7.2). Remember that we rerun our solver until we reach the time limit for the given instance. In each iteration we first search for a valid outage schedule. Afterwards, we perform a production assignment for the found schedule and update our best solution if necessary. If we do not find a valid outage schedule within a certain time, we also proceed to the next iteration.

Our basic solution from Chapter 5 performs one iteration with sweep branching (SB) to get a first reasonable result. Afterwards random branching (RB) is performed. The basic solution provides fair results with an average score of 11.21%. Difficulties arise in the instances B08 and B09 because they comprise a lot of optional cycles, which we do not schedule efficiently. In Section 6.1 we replace RB by incremental cost branching (ICB) and can improve most results because the consideration of additional cost for the respective decoupling dates yields more promising outage schedules. Since ICB is rather slow, we combine it with SB in Section 6.3. The resulting sweep margin branching (SMB) obtains better results than those of SB and runs much faster than ICB. However, its results are not as good as those of ICB. The remaining optimization techniques work all with SMB. In Section 6.4 we add the online propagator (ON) to our model, which limits the domains of outage dates to promising values, thus dramatically reducing the search space. We are able to find comparable good results for the instances B08 and B09 for the first time and can improve most other best solutions, resulting in an average score of 5.13%. In Figure 7.1 we present some charts of our best production plans achieved by ICB, SMB and OP. One can see that the available production capacity of the Type-2 power plants sinks over time because we postpone all optional schedules, while SMB utilizes these cycles and keeps the production capacity at a fair level. With ON the plan becomes more regular and the available Type-2 power plant production capacity can be increased again.

Afterwards, in Section 6.5 we modify the search order of SMB by introducing penalties for undesirable residual fuel levels of the production campaigns. Although this technique can improve the results of 4 instances the overall solution quality becomes worse. We dismiss this technique and add a deterministic postprocessing of the best found solutions in Section 6.6. It repeats the production assignment phase and utilizes an improved refueling strategy that is based on knowledge from the first run of the production assignment phase. This technique again improves most of our results.

7.4 Best Solutions

In this section we present the best solutions found by our approach. First of all, we note that our obtained solution quality for A instances is much better than for B and X instances. This probably originates from some nice properties of the A instances: they comprise less spacing constraints and outages to schedule and thus the search spaces are smaller.

Among the B and X instances we achieve good results with a robust deviation from the best known solution (see Figure 6.10). Unfortunately, we have not been able to improve the best known results for the provided instances. Remember that only the B and X instances are considered for the ranking of the ROADEF/EURO Challenge 2010. Across these instances we gained an average score of 4.66% which corresponds to the second best rating in the competition.

	Sections					
Method	5	6.1	6.3	6.4	6.5	6.6
1× SB	●	●	●	●	●	●
n× RB	●	○	○	○	○	○
n× ICB	○	●	○	○	○	○
n× SMB	○	○	●	●	●	●
ON	○	○	○	●	●	●
PEN	○	○	○	○	●	○
FUEL	○	○	○	○	○	●

Instance	5	6.1	6.3	6.4	6.5	6.6
A00	0.12%	**0.05%**	0.10%	0.05%	0.05%	0.05%
A01	0.37%	**0.12%**	0.22%	0.17%	0.19%	**0.07%**
A02	0.61%	**0.19%**	0.38%	0.28%	0.28%	**0.12%**
A03	0.71%	**0.37%**	0.53%	**0.33%**	0.34%	**0.22%**
A04	1.83%	**0.62%**	0.93%	0.72%	0.76%	**0.46%**
A05	2.26%	**1.19%**	3.60%	**0.63%**	0.91%	**0.55%**
B06	8.16%	**7.36%**	8.60%	**5.92%**	**5.75%**	**5.37%**
B07	9.39%	**8.50%**	9.87%	**4.34%**	4.53%	**4.14%**
B08	17.88%	17.88%	17.88%	**6.60%**	**5.35%**	6.60%
B09	27.90%	27.90%	27.90%	**6.63%**	**6.53%**	6.63%
B10	8.25%	**6.17%**	8.69%	**5.51%**	5.73%	**4.91%**
X11	6.54%	**5.72%**	5.97%	**4.94%**	5.34%	**4.55%**
X12	6.04%	**3.88%**	5.14%	**3.76%**	4.05%	**3.33%**
X13	10.29%	**9.73%**	**8.82%**	**5.35%**	**5.10%**	5.35%
X14	12.00%	**9.67%**	10.25%	**5.37%**	6.43%	**4,92%**
X15	5.61%	**2.84%**	4.59%	2.85%	2.98%	**2.40%**
$\max(B+X)$	27.90%	27.90%	27.90%	6.63%	6.53%	6.63%
$avg(B+X)$	11.21%	9.97%	10.77%	5.13%	5.18%	4.82%

Table 7.2: Evolution of results: The upper table shows the utilized methods of each section. We show sweep branching(*SB*, cf. 5.1.4.3), random branching(*RB*, cf. 5.1.4.2), incremental cost branching(*ICB*, cf. 6.1), sweep margin branching(*SMB*, cf. 6.3), the online propagator(*ON*, cf. 6.4), residual fuel penalty(*PEN*, cf. 6.5) and tuning fuel levels(*FUEL*, cf. 6.6). In the lower table we list the best scores obtained in each section. Improvements to previous steps are in bold font.

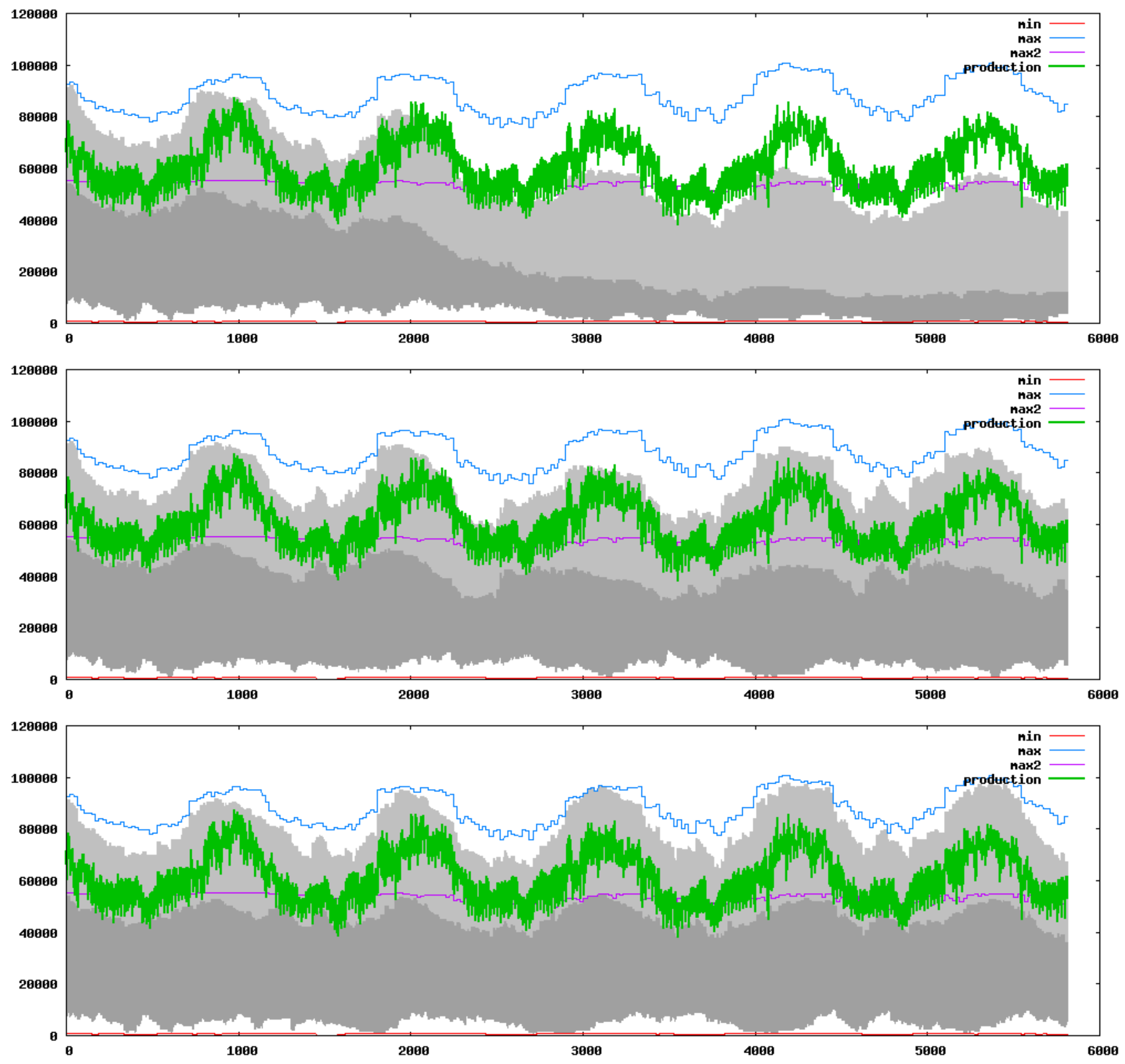

Figure 7.1: Sample Production Plans: We show three different solutions to instance B08 aggregating all power plants. The upper solution was found by ICB, while both other have been found by SMB. The lower solution was found after adding the online propagator (OP) to our model. There are lines for the gross capacity of all plants (blue), the demand (green), the gross capacity of all Type-2 power plants (purple) and the minimum capacity of all plants (red). The light (dark) gray region marks the available production capacity of all Type-1 (Type-2) Power Plants. We clearly see that the plants run out of fuel and become unavailable when using incremental cost branching, because all optional cycles are postponed. While SMB performs better the online propagator again increases the available capacity.

Instance	Best Known Solution	Our Best Solution	Score	
A00	8 730 985 367 093	8 735 652 150 125	0.05%	
A01	169 538 386 903	169 661 789 155	0.07%	
A02	146 048 433 973	146 226 213 771	0.12%	
A03	154 429 888 940	154 775 064 397	0.22%	
A04	111 591 350 516	112 106 431 505	0.46%	
A05	125 822 236 385	126 509 328 020	0.55%	
B06	83 424 716 217	87 901 956 744	5.37%	
B07	81 174 243 138	84 535 993 963	4.14%	
B08	81 926 206 073	86 308 981 283	5.35%	*
B09	81 750 858 197	87 092 422 958	6.53%	*
B10	77 767 024 999	81 587 212 103	4.91%	
X11	79 116 772 289	82 718 992 355	4.55%	
X12	77 589 910 940	80 171 684 438	3.33%	
X13	76 449 207 715	80 345 858 956	5.10%	*
X14	76 172 998 633	79 921 258 048	4.92%	
X15	75 101 398 439	76 901 330 176	2.40%	

Table 7.3: Best solutions: We compare the costs of the best known results to our best achieved results. Note that the starred solutions (B08, B09, X13) have been found using the residual fuel penalty from Section 6.5. All other best solutions have been found by tuning fuel levels instead (cf. Section 6.6).

8. Conclusion

In this thesis we solved the problem posed by the ROADEF/EURO Challenge 2010, which contained an industry scale power plant production planning problem in an uncertain and highly constrained environment. The overall objective function was to reduce the total operating cost.

We developed two lower bounds for the problem. One is a fast greedy heuristic, the other is based on a flow network and yields closer bounds at the cost of high computational effort. We decomposed the main problem into two stages: outage scheduling and production assignment. For the first stage we formulated the given model as a constraint program and added further constraints which ensure the feasibility of the remaining production assignment problem. In the second stage we utilized a greedy heuristic – developed from our greedy lower bound – to assign production levels and refueling amounts for a given outage schedule.

We applied several optimization techniques to our basic solution. First, we improved the search for good outage schedules by randomization, identifying promising outage dates and choosing them with a higher probability. We further analyzed the problem and developed a new constraint which removes undesirable outage schedules from the search space. At last we improved our production assignment stage by running it twice, utilizing knowledge from the first run throughout the second.

We implemented all the proposed methods and evaluated their performance on 16 instances which have been provided by the French utility company EDF and are extracted from real world data. For all experiments we kept as close as possible to the specifications of the competition.

Finally, we related this work to the results obtained by the participating teams of the ROADEF/EURO Challenge 2010. Our approach is able to solve all proposed instances, which only 4 of 21 finalist teams did achieve. In our experiments we gained very competitive results with an average deviation from the best known results of less than 5%. This corresponds to the second best score achieved in the competition.

List of Figures

List of Tables

92

List of Algorithms

94

Bibliography

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, Inc., 1993.

[BC02] N. Beldiceanu and M. Carlsson, "A New Multi-resource cumulatives Constraint with Negative Heights," in *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming.* Springer-Verlag, 2002, pp. 63–79.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT Press, 2001.

[DR97] D. Dentcheva and W. Römisch, "Optimal Power Generation under Uncertainty via Stochastic Programming," in *Stochastic Programming Methods and Technical Applications, Lecture Notes in Economics and Mathematical Systems 458*, K. Marti and P. Kall, Eds. Springer-Verlag, 1997, pp. 22–56.

[FKL96] S. Feltenmark, K. C. Kiwiel, and P. Lindberg, "Solving Unit Commitment Problems in Power Production Planning," in *Operations Research Proceedings 1996.* Springer-Verlag, 1996, pp. 236–241.

[FMS08] W. K. Foong, H. R. Maier, and A. R. Simpson, "Power Plant Maintenance Scheduling Using Ant Colony Optimization: An Improved Formulation," *Engineering Optimization*, vol. 40, pp. 309 – 329, 2008.

[Gec10] "GECODE - A Generic Constraint Development Environment," http://www.gecode.org, 2010.

[GT90] A. V. Goldberg and R. E. Tarjan, "Finding Minimum-Cost Circulations by Successive Approximation," *Mathematics of Operations Research*, vol. 15, no. 3, pp. 430–466, 1990.

[Kid09] S. Kidd, "Nuclear in France - What Did They Get Right?" Nuclear Engineering International Magazine, 2009. [Online]. Available: http://www.neimagazine.com/story.asp?storyCode=2053355

[KPB06] M. O. I. Khemmoudj, M. Porcheron, and H. Bennaceur, "When Constraint Programming and Local Search Solve the Scheduling Problem of Electricité de France Nuclear Power Plant Outages," in *Principles and Practice of Constraint Programming, Lecture Notes in Computer Science 4204*, 2006, pp. 271–283.

[Lem10] "LEMON Graph Library," http://lemon.cs.elte.hu, 2010.

[LES07] K. Y. Lee and M. A. El-Sharkawi, *Modern Heuristic Optimization Techniques: Theory and Applications to Power Plants.* Wiley-Interscience, 2007.

[NKT00] J. M. Ngundam, F. Kenfack, and T. T. Tatietse, "Optimal Scheduling of Large-Scale Hydrothermal Power Systems Using the Lagrangian Relaxation Technique," *International Journal of Electrical Power & Energy Systems*, vol. 22, no. 4, pp. 237 – 245, 2000.

[NW88] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization.* Wiley-Interscience, 1988.

[Pad04] N. Padhy, "Unit Commitment - A Bibliographical Survey," *Power Systems, IEEE Transactions on*, vol. 19, no. 2, pp. 1196 – 1205, 2004.

[PGJ$^+$] M. Pocheron, A. Gorge, O. Juan, T. Simovic, and G. Dereu, "Challenge ROADEF/EURO 2010: A Large-Scale Energy Management Problem with Varied Constraints." [Online]. Available: http://challenge.roadef.org/2010/

[RBW06] F. Rossi, P. v. Beek, and T. Walsh, *Handbook of Constraint Programming (Foundations of Artificial Intelligence).* Elsevier Science Inc., 2006.

[SK98] S. Sen and D. P. Kothari, "Optimal Thermal Generating Unit Commitment: A Review," *International Journal of Electrical Power & Energy Systems*, vol. 20, no. 7, pp. 443 – 451, 1998.

[SN91] T. Satoh and K. Nara, "Maintenance Scheduling by Using Simulated Annealing Method [for Power Plants]," *Power Systems, IEEE Transactions on*, vol. 6, pp. 850 – 857, 1991.

[STL10] C. Schulte, G. Tack, and M. Z. Lagerkvist, "Modeling and Programming with Gecode," www.gecode.org/doc-latest/MPG.pdf, 2010, online handbook.

[Tac09] G. Tack, "Constraint Propagation – Models, Techniques, Implementation," Doctoral Dissertation, Saarland University, 2009.

[WW96] A. J. Wood and B. F. Wollenberg, *Power Generation, Operation, and Control*, 2nd ed. Wiley-Interscience, 1996.