Marko Vistica

Value of stochasticity in hydropower planning optimization

Master thesis

EEH – Power systems laboratory
Swiss Federal Institute of Technology (ETH) Zürich

Expert: Prof. Dr. Göran Andersson
Supervisor: Huber Abgottspon

Zürich, August 22nd, 2012
Acknowledgment

Firstly, I would like to thank Hubert Abgottspon, my supervisor at ETH, for agreeing to supervise this project and for all the guidance and help during these past six months.

A very special thanks goes to Yelena Vardanyan, my supervisor at KTH, who gave me plenty of valuable advice. Also, I have to thank my examiner, prof. Lennart Söder for accepting this thesis proposal.

Finally, I would like to thank my family for the constant support they provided throughout my studies.
Abstract

With respect to market liberalization, efficient use of resources is becoming more important for players in the market. In order to achieve that different optimization techniques were developed which enable better operational efficiency. These techniques can be segmented into two different categories, depending on their time horizon:

- Yearly time horizon – mid-term hydropower scheduling
- Daily time horizon – short-term hydropower scheduling

These two time horizons account for two case studies presented in this thesis.

In the first case study (mid-term planning), the focus is on determining power plant’s optimal operating strategy, while taking into account the uncertainty in inflows and prices. Stochastic dynamic programming has been chosen as a mid-term optimization technique. Since stochastic dynamic programming calls for a discretization of control and state variables, it may fall under the curse of dimensionality and therefore, the modeling of stochastic variables is important.

By implementing a randomized search heuristic, a genetic algorithm, into the existing stochastic dynamic programming schema, the optimal way of using the stochasticity tries to be found. Two price models are compared based on the economic quality of the result.

The results give support to the idea of using search heuristics to determine the optimal stochasticity setup, however, some deviations from the expected results occur.

Second case study deals with short-term hydropower planning, with a focus on satisfying the predefined demand schedule while obtaining maximum profit. With short-term hydropower planning being a nonlinear and nonconvex problem, the main focus is on the linearization of unit performance curves, as well as satisfying technical constraints from the power plant perspective. This optimization techniques also includes the water value in the solution. The problem has been solved by means of mixed integer linear programming.

The results from the second case study are fully in line with the expectations and it is shown that mixed integer linear programming approach gives good results with good computational time.

Suggested improvements to the model and potential for future work can be found in the final chapter of this thesis.
Contents
Acknowledgment ........................................................................................................ ii
Abstract ...................................................................................................................... iii
List of figures ............................................................................................................... vii
List of tables ............................................................................................................... viii
List of Acronyms ........................................................................................................ ix
List of Symbols .......................................................................................................... x
Chapter 1. Motivation and outline ............................................................................... 1
  1.1. Motivation ........................................................................................................... 1
  1.2. Objective ........................................................................................................... 2
  1.3. Outline ............................................................................................................... 2
Chapter 2. Literature survey ....................................................................................... 3
  2.1. Stochastic dynamic programming ...................................................................... 3
  2.2. Modeling of stochastic parameters .................................................................... 3
  2.3. Evolutionary algorithms ................................................................................... 4
  2.4. Short-term optimization - mixed integer programming ...................................... 4
Chapter 3. Model overview and previous work done .................................................. 7
  3.1. Basic formulation ............................................................................................. 7
  3.2. Model of the hydropower plant ....................................................................... 7
     3.2.1. Hydropower plant description ................................................................... 7
     3.2.2. Modeling of the hydropower plant ........................................................... 9
  3.3. Modeling of stochastic values .......................................................................... 11
  3.4. Stochastic dynamic programming .................................................................... 12
     3.4.1. Introduction to dynamic programming ................................................... 12
     3.4.2. Profit-to-go function (PTG) ................................................................... 12
     3.4.3. Backward recursion ................................................................................. 13
     3.4.4. Forward step ............................................................................................. 14
     3.4.5. Drawbacks of SDP .................................................................................. 15
     3.4.6. Advantages of SDP .................................................................................. 15
  3.5. Introduction to short-term planning .................................................................. 16
  3.6. Power plant model ............................................................................................ 17
  4. Genetic algorithm ................................................................................................. 19
     4.1. Introduction to evolutionary algorithms ....................................................... 19
     4.2. Genetic algorithm .......................................................................................... 21
     4.3. Genetic algorithm components ....................................................................... 22
8.2. Future work .........................................................................................................................................................66

Appendix 1. Power plant data ........................................................................................................................................67
A.1. Case study 1 data ....................................................................................................................................................67
A.2. Case study 2 data ....................................................................................................................................................67

Appendix 2. Short-term hydropower planning using the genetic algorithm – basic concept and implementation .........................................................................................................................................................68
A2.1. Implementation of the solution ......................................................................................................................................68
A2.2. Optimization results ................................................................................................................................................71
List of figures

Figure 1. Inflows to the upper reservoir ................................................................. 8
Figure 2. Power plant model .................................................................................. 9
Figure 3. Decision making under uncertainty ....................................................... 11
Figure 4. Power plant model - case study 2 ........................................................... 18
Figure 5. Global and local maximum .................................................................... 19
Figure 6. Genetic algorithm flowchart .................................................................. 21
Figure 7. Single point crossover ........................................................................... 24
Figure 8. Mutation ................................................................................................. 25
Figure 9. Performance curve of a genetic algorithm .............................................. 26
Figure 10. Branch-and-cut algorithm .................................................................... 31
Figure 11. Feasible solution for LP and MILP problems ....................................... 32
Figure 12. Implementing GA and MILP procedures .............................................. 33
Figure 13. Some Monte Carlo inflow scenarios ...................................................... 35
Figure 14. Price scenarios .................................................................................... 36
Figure 15. Price scenarios .................................................................................... 39
Figure 16. Stochastic modeling as Markov chain ................................................... 40
Figure 17. Creating price intervals ....................................................................... 41
Figure 18. Implementing GA into SDP framework ................................................ 43
Figure 19. Schedule and price for the short-term planning ..................................... 44
Figure 20. Unit performance curves for different head values .............................. 45
Figure 21. Unit performance curve ....................................................................... 46
Figure 22. Piecewise linearization of unit performance curve ............................... 46
Figure 23. Water value curve ............................................................................... 49
Figure 24. Yearly schedule of the power plant ...................................................... 51
Figure 25. Upper reservoir solution ..................................................................... 52
Figure 26. Lower reservoir solution ..................................................................... 53
Figure 27. Optimal stochasticity setup over 56 phases .......................................... 56
Figure 28. Analysis of different stochasticity setups ............................................ 56
Figure 29. Fitness function evolution .................................................................... 57
Figure 30. 24 h schedule of designated power plants .......................................... 59
Figure 31. Operating schedule for Stalden 2 ......................................................... 59
Figure 32. Operating schedule for Zermeiggern 1 ............................................... 60
Figure 33. Operating schedule for Zermeiggern_pump_2 ... .............................. 60
Figure 34. Discharge data for Zermeiggern1 turbine ........................................... 61
Figure 35. Hourly profit with associated prices .................................................... 62
Figure 36. Reservoir level Mattmark within a 24 hour period ......................... 63
Figure 37. Reservoir level Zermeiggern within a 24 hour period ...................... 63
Figure 38. 24 hour schedule with GA ................................................................. 71
Figure 39. Mattmark content using GA ............................................................... 72
Figure 40. Zermeiggern content using GA .......................................................... 72
# List of tables

Table 1. Different coding possibilities ........................................................................................................... 22
Table 2. Data concerning different units ........................................................................................................... 49
Table 3. Profits over different scenarios .......................................................................................................... 53
Table 4. Analysis of different stochasticity setups .......................................................................................... 55
Table 5. Piecewise linearization data ............................................................................................................. 58
Table 6. Daily inflows into the reservoirs ......................................................................................................... 58
Table 7. Case study 1 power plant data ........................................................................................................... 67
Table 8. Case study 2 power plant data ........................................................................................................... 67
Table 9. Case study 2 reservoir data .............................................................................................................. 67
# List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA</td>
<td>genetic algorithm</td>
</tr>
<tr>
<td>MILP</td>
<td>mixed integer linear program</td>
</tr>
<tr>
<td>SDP</td>
<td>stochastic dynamic programming</td>
</tr>
<tr>
<td>PTG</td>
<td>profit-to-go function</td>
</tr>
<tr>
<td>EEX</td>
<td>European Energy Exchange</td>
</tr>
<tr>
<td>HPFC</td>
<td>Hourly price forward curve</td>
</tr>
<tr>
<td>GBM</td>
<td>Geometric Brownian motion</td>
</tr>
<tr>
<td>STHS</td>
<td>Short-term hydro scheduling</td>
</tr>
</tbody>
</table>
List of Symbols

Sets

\( G \)  
set of generating power plants

\( P \)  
set of pumping power plants

\( I \)  
set of indices of a plant

\( T \)  
number of periods in the horizon

\( M \)  
set of scenarios

Parameters

\( c_{p,t} \)  
electricity price at stage \( t \)

\( c_{st} \)  
cost of spilling water

\( u_{i,tech\_min} \)  
technical minimum of a power plant

\( P_{0i}(i) \)  
minimum power output of power plant \( i \)

\( U_i \)  
minimum water discharge of power plant \( i \)

\( \bar{U}_i \)  
maximum water discharge of power plant \( i \)

\( \bar{U}_i(i) \)  
maximum water discharge of block \( l \) of power plant \( i \)

\( \rho_i(i) \)  
slope of the block \( l \) of power plant \( i \)

\( \gamma \)  
value of stored water in €/m³

\( D_t \)  
schedule at stage \( t \)

Variables

\( \theta_t \)  
profit-to-go-function

\( v_t \)  
reservoir filling at stage \( t \)

\( q_t \)  
inflow at stage \( t \)

\( u_t \)  
turbined energy at stage \( t \)

\( b_t \)  
pumped energy at stage \( t \)

\( s_{lt} \)  
spilled energy from the lower reservoir at stage \( t \)

\( s_{ut} \)  
spilled energy from the upper reservoir at stage \( t \)

\( c_{2nd,t} \)  
secondary control reserve bid price

\( bid_t \)  
amount of secondary control offered

\( \Pi \)  
profit

\( p_i(t) \)  
power output from power plant \( i \) in time period \( t \)

\( u_i(t) \)  
discharge from the power plant \( i \) in time period \( t \)

\( u_i(i,t) \)  
water discharge from block \( l \) of plant \( i \) in time period \( t \)

\( x_i(t) \)  
binary variable stating on/off status of power plant \( i \) in period \( t \)

\( w_i(i,t) \)  
binary variable stating if water discharge exceeded the maximum amount in block \( l \)

\( a_i(t) \)  
shut down status at time \( t \)

\( n_i(t) \)  
start-up status at time \( t \)
Chapter 1. Motivation and outline

1.1. Motivation

With respect to market liberalization and increasing market competition, efficient use of resources is of big importance for hydropower producers. Failing to utilize these resources might lead to low profits, unreliable supply and being overpowered by the competition[1]. That is why there is an increasing effort put into hydropower planning optimizations.

This is usually done in different time intervals, typically ranging from one hour to a couple of years. In the first part of the thesis, the main focus will be on midterm planning, which refers to a yearlong planning period with weekly or monthly intervals. The aim of the midterm planning is to maximize the profit in the time span, while taking into account seasonal evolutions of parameters like inflows or prices. The key decisions are how much water to store in the reservoir and how much to discharge from the power plant in each stage. It also serves as a frame for the short-term planning that’s performed in a period from a day to a couple of days. The aim of the short-term planning is satisfying the load demand.

The planning is performed on a typical hydropower plant that is described in more detail in Chapter 3 of this thesis. When considering midterm planning, one has to realize that not all information is known at the moment of decision making and thus the decisions are made under uncertainty. The variables causing uncertainty are:

- Uncertain water inflow into the reservoirs
- Uncertain market prices

These parameters, because of their nature, will from now on be referred as stochastic parameters. There are several methods suitable for solving stochastic problems (e.g. stochastic linear programming, stochastic dynamic programming, dual dynamic programming and similar) and the method of stochastic dynamic programming is chosen for this thesis, as it can handle non-linearity and non-convexity of the problem, as well as take into account randomness of stochastic variables.

Extensive research has been done on stochastic dynamic programming, part of which can be seen in Chapter 2 of this thesis. Based on the research, one can conclude that the modeling of stochastic variables is important, since if it is done incorrectly it can lead to either poor quality of results or computational intractability. By implementing some heuristic methods into the stochastic dynamic programming scheme, one tries to find the optimal way of discretizing input data (prices) with respect to some evaluation criteria (e.g. profit, execution time etc.). This would account for a more efficient use of stochasticity. That is what this thesis is going to deal with by implementing the genetic algorithm into the stochastic dynamic programming frame.
Short-term planning uses the data obtained by mid-term planning to form an operating schedule on a day ahead basis. This will be the focus in the second part of the thesis. The planning will be performed for a real Swiss power plant, whose description can be found in Chapter 3. The main goal of short-term planning is to satisfy the load curve, while maintaining feasible operation. The load curve is a plot of load variation during time. It can be done in several time intervals, but a daily load curve is of interest in this thesis.

This operating schedule is going to be obtained by implementing the same method as in the first case study – mixed integer linear programming.

1.2. Objective

The goals of this thesis are:

- Implementing the genetic algorithm into the existing stochastic dynamic programming frame used for mid-term hydropower planning
- Determining the optimal stochasticity setup with respect to the final profit
- Use MILP to perform a short-term optimization of a typical Swiss hydropower plant

As an additional task, using genetic algorithm to do the short-term planning was planned. However, due to time limitations, the task wasn’t performed completely. The implementation part can be found in Appendix 2. and can serve as a basis for future research.

1.3. Outline

Chapter 2 gives an overview of the relevant literature. The description of the model and previous work done in the project is presented in Chapter 3. It includes basic information about the power plant as well as stochastic processes and stochastic dynamic programming. Chapter 4 gives an introduction to genetic algorithm (GA). Mixed integer programming is introduced in Chapter 5. How case study problems are implemented is seen in Chapter 6. Chapter 7 provides simulation results. Finally, Chapter 8 will present the conclusion as well as possible improvements and further research possibilities.
Chapter 2. Literature survey

This chapter provides a review of literature that could be of interest when researching stochasticity in mid-term hydropower optimization. Firstly, an overview of literature concerning stochastic dynamic programming is given, followed by the modeling of stochastic processes. Afterwards, literature focusing on evolutionary algorithms is introduced, as well as some examples of its use with other optimization methods. Finally, literature on mixed integer programming is presented.

2.1. Stochastic dynamic programming

P. Kall and S.W. Wallace (1994)[2] wrote a basic textbook on stochastic programming that introduces the topic, ranging from basic concepts to more advanced views. It starts off by presenting basic concepts, followed by the introduction of dynamic systems. Here, Bellman’s principle of optimality is introduced, as well as scenario trees and stochastic dynamic programming. Chapter 3 covers recourse problems – decomposition methods for stochastic programs, such as L shaped decomposition method. Chapters on probabilistic constraints and preprocessing are following and the final chapter deals with network problems.

A. Eichorn (2010)[3] in his workshop presentation gives an overview of stochastic programming application in power systems. He starts with an overview of the power system and the introduction of mid-term planning. He then introduces the concept of stochastic programming, starting with linear programming and afterwards introducing two stage and multi-stage stochastic programming. Following this, a dynamic approach is introduced as he covers stochastic dynamic programming and stochastic dual dynamic programming.

2.2. Modeling of stochastic parameters

T.G. Siqueira, M. Zambelli, M. Cicogna, M. Andrade and S. Soares (2006)[28] compare different streamflow models which are used as input data in hydropower planning optimization. Three models were introduced – one based on average values of historic data, one based on probability distribution functions and the final one adopts a Markov chain based on a lag-one periodical auto-regressive model. The results have shown that all models lead to similar results, with only minor differences and thus the deterministic approach (first introduced model) can be used for optimizing complex multi-reservoir problems.

C. Blanco, S. Choi and D. Soronow (2001)[24] give an example of how to model stochastic price processes. They focus on one method – geometric Brownian motion. Two main principles of this method are that the price changes are independent of each other and that the price changes have a constant mean and volatility. Except for the explanations, the article list possible pitfalls of using this method. This article is the first of three where the authors are explaining different modeling approaches.
M. Birger, A.Gjelsvik, A.Grundt and K. Karesen (2001)[23] deal with price modeling in mid-term hydropower operation optimization. The optimization technique used is SDP and the price model is based on a Markov chain principle, where in each step different number of price nodes are identified and transition probabilities are calculated. A method for inclusion of extreme prices is offered, as is the method for long-term price uncertainty modeling.

2.3. Evolutionary algorithms

A.E. Eiben and J.E. Smith (2003)[8] in their textbook dedicate a chapter to evolutionary algorithm. It starts with an introduction to the algorithm; it is explained in the form of flowchart as well as pseudo-code. It then continues to describe specific stages of the algorithm, including the explanation of the fitness function. This is followed by a set of examples, for instance the solution to the Knapsack problem. This chapter is finalized with the application of this algorithm in optimization problems.

E. Alba and C.Cotta (2004)[4] introduce the concept of evolutionary algorithms. They start from the origin in biology and connect it with the optimization problems. The overview of main types of evolutionary algorithm is given as well as the applications in industry. Special focus is put on explaining different stages in the algorithm, with accompanying examples.

G. Jones (2002)[5] in the chapter of the book talks about the evolutionary algorithms and their application. He starts by introducing genetic algorithms and gives a canonical code of the algorithms. Afterwards, he introduces evolutionary strategies and evolutionary programming. Finally, he deals with applications of those algorithms in computational chemistry.

T. Back, D.B. Fogel and Z. Michalewicz (1997)[10] in their book “Evolutionary Algorithms and Their Standard Instances”. Here, he explains the basic of genetic algorithms such as the principle of operation and provides the pseudocode. He also covers the theory and application of genetic algorithm processes, like crossover, representation and mutation.

2.4. Short-term optimization - mixed integer programming

G-W. Chang et. Al. (2001)[21] present experiences with mixed integer linear programming in short term hydro scheduling. They believe MILP is a powerful tool for solving large scale short-term planning problems. In this paper, they introduce the model with explanations of the variables, objective function and the constraints. They tested the model on two power systems and it gave satisfying results in reasonable time.

J. Garcia-Gonzalez and G.Alonso-Castro (2001)[26] test the MILP approach on three cascaded reservoir belonging to the same hydro chain. Since this is normally a non-linear problem, after the introduction, they present the linearization of a two variables function, which enables efficient solving of the problem. Afterwards, the model is introduced and
simulated and results are presented. It is shown that this method is not suitable for very large systems since the accuracy of results decline.

A.Borghetti, C. D’Ambrosio, A. Lodi and S. Martello (2008)[25] in their paper focus on short-term hydro planning of a pumped-storage power plant with head dependency. The focus is on model explanation and introduction of linearization process by introducing binary variables. Afterwards, they introduced an enhanced linear model that they later compared to the previous one. It is shown that the enhanced mode, though having higher computational times, gives better results.

A.J. Conejo et. al. (2002)[22] in their paper focus on the self-scheduling problem of the hydro producer in an electricity market. They take into account the head dependency and the nonconcavity of the unit performance curve and formulate the problem through the use of binary variables. The system in question is a cascaded system with multiple reservoirs and the MILP approach proves to give satisfying results.
Chapter 3. Model overview and previous work done

This chapter explains the models used in the thesis. In case study 1, the hydropower plant in question will be described, followed by the modeling of the stochastic variable and theoretical background on stochastic dynamic programming. For case study 2, a brief introduction to short-term scheduling is given, followed by an explanation of the model used.

Case study 1 – determining optimal stochasticity setup

3.1. Basic formulation

Mid-term scheduling is performed in this case study in a time horizon of one year, with a time resolution of one week. As already mentioned, it has to take into account the stochasticity of inflows and prices during this period. The aim of the optimization is to maximize the profit, while taking into account the value of stored water in the reservoir.

Mid-term scheduling problems are large scale, nonlinear and nonconvex problems, that normally would require large computational effort if solved directly. SDP method has been chosen for this thesis, as the complexity of the problem is easily overcome, since only one stage is considered at the time. Also, in comparison to other stochastic solving methods like SDDP, it gives similar quality results[27].

Discretization and proper modeling of stochastic variables is important for result accuracy. By implementing the genetic algorithm into the SDP frame and using it as a tool, the optimal discretization setup of prices tries to be found.

3.2. Model of the hydropower plant

3.2.1. Hydropower plant description

The hydropower plant in use is a typical Swiss pumped-storage hydropower plant with two reservoirs. The upper reservoir is the seasonal reservoir and the water from the upper reservoir is used to produce electricity. The lower reservoir serves as balancing reservoir that has a purpose of, basically, refilling the upper reservoir during off peak hours. There is also the possibility of spilling water from the reservoirs because of physical (full reservoir, large inflows) or economic reasons.

The inflow to the upper reservoir of the hydropower plant in question can be seen in Fig 1.
Since the actual data is confidential, Fig 1. shows an example of an inflow so the correlation and seasonality can be better understood. One can notice that there is almost no inflow in the first third of the year and the last couple of months, due to the fact that most water is accumulated as snow or ice. Somewhere at the beginning of May, the snow starts melting and the inflow increases rapidly. The inflow period between May and late October basically accounts for the year round inflow. This strong seasonality is the reason for building large reservoirs, so the production is not dependent on the inflow itself.

This information is valuable when determining starting dates of the optimization. Due to climate factors mentioned above, it is assumed that the reservoir level at the beginning of May is zero and this time is thus considered to be perfect since a yearly horizon is taken into account and one doesn’t have to account for the water value at the end of the optimization.

For more technical information about the power plant, one should consult Appendix 1.

The model of the power plant can be found in Fig. 2.
3.2.2. Modeling of the hydropower plant

As one can see from Fig 2., the reservoir content is denoted with a $v$, while the turbined and pumped energy are denoted by $u$ and $b$ respectively. The inflow is denoted with $q$ and the spilled water is denoted by $s$. All the parameters shown are expressed in MWh/h.

Physically, reservoir filling, inflows and spillage are calculated in m$^3$, but in order to simplify building the optimization problem, a conversion factor (kWh/m$^3$) is introduced:

$$cf = 3.8 \text{ kWh/m}^3$$

The efficiencies of the turbine and the pump also need to be included in the calculations and they are:

$$\eta_t = \eta_p = 0.85$$

The efficiencies of the turbine and pump need to be included in the model when pumping possibilities exist. If one unit of energy needs to be pumped, $\frac{1}{\eta_p}$ units of energy have to be bought.
In order for the solutions to be within boundaries of what is physically and technically possible, a set of constraints need to be determined. Two types of constraints are applied to this model:

- Equality constraints
- Variable bounds

**Equality constraint**

\[ v_{t+1} = v_t + q_t + b_t - u_t - s_t \]  

(3.1)

As one can see from (3.1) the equality constraint links the state variable \( v_t \) in successive time steps and how the state is achieved. Reservoir is considered a state variable as it mathematically describes the state of a dynamic system. It is said the reservoir level at time period \( t+1 \) depends on the inflow between \( t \) and \( t+1 \) \( (q_t) \), the amount of water that was spilled during the same period \( (s_t) \) and the decisions on the amount of turbined and pumped water \( (u_t, b_t) \) that were made at the beginning of stage \( t \).

**Variable bounds**

Variable bounds impose a technical or physical limit to introduced variables. A set of inequality constraints can be seen below:

\[ v_{\text{min}} \leq v \leq v_{\text{max}} \]
\[ s_{\text{min}} \leq s \leq s_{\text{max}} \]  
\[ u_{\text{min}} \leq u \leq u_{\text{max}} \]
\[ b_{\text{min}} \leq b \leq b_{\text{max}} \]  

(3.2)

As one can see, the example of a physical limit is the level of water in a reservoir that cannot exceed the reservoir dimensions. On the other hand, a technical limit is the amount of turbined water that depends on the installed power of the power plant.
3.3. Modeling of stochastic values

There are two stochastic variables that cause uncertainty in this optimization problem and they are water inflows and market prices. Modeling of these stochastic variables will be shown later in this chapter, while for now the focus is on introducing the concept of stochasticity.

If one has to optimize a process lasting through several time stages, uncertain nature of the variables will greatly influence on decision making processes. One can assume that all variables are known when period $t$ in time is reached and the system is in state $S$. Based on the current state of variables, but also future expectations, a decision has to be made. An optimal decision to this problem is obtained by the process of optimization. This decision is then applied to the system in state $t$, which will lead to system moving into a different state $S_1$.

After reaching $S_1$, the operator of the process has no control over the system as stochastic processes begin to occur and they could lead the system into several different states at time $t + 1$. The probability of reaching these states can be estimated from the probability distributions of stochastic processes.

When the system reaches time stamp $t + 1$, it is in the state $S_2$, where another decision has to be made by the operator, same as at stage $t$, leading into state $S_3$.

To illustrate this process better, Fig. 3 is presented.

![Figure 3. Decision making under uncertainty](image-url)
3.4. Stochastic dynamic programming

3.4.1. Introduction to dynamic programming

The term dynamic programming was introduced by Bellman, when describing the theory dealing with multi-stage decision processes. When these decision processes account for uncertainty, the term stochastic dynamic programming is used. The principle and the main idea behind stochastic dynamic programming is given with the following example.

A system that evolves over T time periods is considered. At time period \( t \), \( x_t \) and \( u_t \) are used to represent the state of the system and the control action respectively. That means that, in period \( t \), the state of the system is determined by its history:

\[
x_t = f_{t-1}(x_{t-1}, u_{t-1})
\]

If the goal of the optimization is to maximize the profit over the entire time horizon, the objective function might look as (3.3)

\[
\max_{\varphi} E_x \{ f_r(x_T) + \sum_{t=0}^{T-1} f(x_t, u_t) \}, \quad t = 0, \ldots, T
\]  

(3.3)

This is, of course, a quite elaborate problem and a tail subproblem of maximizing the profit from time \( i \) to time \( T \) can be considered.

\[
\max_{\varphi} E_x \{ f_r(x_T) + \sum_{t=i}^{T-1} f(x_t, u_t) \}
\]  

(3.4)

According to Bellman, no matter how state \( i \) has been reached, the remaining decisions must be optimal for the tail subproblem. Stochastic dynamic programming will first solve all tail subproblems for the final stage, followed by the previous one and so on. The original problem is solved at the final step by using the solutions of all tail subproblems.

This approach proved to give good results as it breaks down a big problem into smaller, easily solvable subproblems.

3.4.2. Profit-to-go function (PTG)

Stochastic dynamic programming is one of the more common optimization techniques used with stochastic problems. For a multi-stage problem like a mid-term optimization problem, one decision is going to be made for each state in each stage. It should be noted, that decisions for all but the first stage, depend on the outcome of stochastic variables[2]. The output of the SDP optimization problem is the so called “profit-to-go” function. As it can be seen from (3.3), PTG describes how much profit will is expected to be obtained in the future depending on the current state of state variables, if the optimal policy is applied to the system.

Before writing the PTG expression, a concept of state variables and control variables needs to be introduced. The state variables in the model describe the mathematical state of the dynamic system, while the control variables are the ones whose manipulation influences the state variables.
In a hydropower planning optimization problem, the profit depends entirely on the state variables and can be written in an recursive form, like shown in (3.3)

\[
\theta_t(v_t, c_{t+1}) = \max c_{p,t} * u_t - c_{p,t} * b_t + E \left[ -c_{s,t} * s_t + \theta_{t+1}(v_{t+1}, c_{t+1}) \right]
\]

subject to

\[
v_{t+1} = v_t - u_t + b_t + q_t - s_t
\]

\[
v_{t+1} \in V
\]

\[
u_t \in U
\]

\[
b_t \in B
\]

\[
s_t \in S
\]

As it is seen, the profit only depends on the reservoir content and the current prices. It should also be mentioned that the PTG at stage T+1 can be calculated and is thus considered zero.

3.4.3. Backward recursion

The aim of backward recursion is to find a PTG function for each stage in the optimization process. As this type of simulation can’t deal with continuous variables, since that would mean that it would have to “try out” for every single combination of state and control variables, those variables need to be discretized into a finite number of possibilities. As this is stochastic dynamic programming, the distributions of stochastic variables need to be taken into account.

There are several steps that need to be followed when performing the backward recursion:

1. Backward recursion starts at the end of the optimization period – at time step T. Stochastic variables need to be discretized into a predetermined number of values.

2. For each of the possible reservoir fillings, energy prices and control variables (turbining and pumping) the profit is calculated. The combination of values that leads to the best profit is saved in a “look up” table that the operator uses to determine the optimal control action. This procedure is then repeated for every stage until the first one.

3. When the optimization comes into stage T-n, it performs steps 1 and 2, meaning that stochastic variables need to be discretized and the variable combinations are tried out.
However, the optimization also takes into consideration the information from the stage \(T-n+1\), which contains information from previous stage and so on. This way, the optimization time span as a whole is taken into account.

As it can be seen, the backward recursion basically gives an optimal policy of how a power plant should be operated. In order to test this policy, the so-called forward step is introduced.

The pseudocode for the SDP formulation used in this thesis is as follows:

\[
\theta_{T+1}(v_t, c_{p,t}) = 0
\]

\begin{align*}
\text{for each stage } t &= T, T-1, \ldots, 1 \\
\text{for each reservoir content level } v_t \\
\text{for each turbining possibility } u_t \\
\text{for each pumping possibility } b_t \\
\text{for secondary control possibility } bid_t \\
\text{for each spot price possibility } c_{s,t} \\
& \quad \text{calculate profit for selected values using } \theta_{T+1} \\
& \quad \text{calculate expected profit over all realizations of the stochastic process } c_{s,t} \\
\end{align*}

\begin{align*}
\text{select } u_t \text{ and } b_t \text{ giving maximum profit for selected reservoir content level} \\
\text{determine maximum profits for all reservoir content levels}
\end{align*}

3.4.4. Forward step

After the backward recursion, the full PTG will be obtained. If this optimal policy wants to be tested, a sample set of data is introduced (e.g., historic data from one of the previous years) and a forward step is performed.
At $t=0$, the initial state of the system is known, meaning the reservoir filling as well as the price is known. The operator then checks the look up table and implements the optimal decision for the systems state. Based on the stochastic inflows, the system will end up in one of the possible states at $t=1$ (as explained in Fig. 3.). There, an optimal decision is made again based on the current state and this procedure is performed until the final stage $T$ is reached.

3.4.5. Drawbacks of SDP

The main drawback of using SDP is the computational intractability, which is also known as the curse of dimensionality. Since the optimization algorithm requires discretization of all state and control variables, if a detailed modeled is wanted, the computational effort might become too high[3]. For instance, if 4 variables ($v_1, v_2, v_3, v_4$) are discretized into 50 values, that means that in each stage $v_1 \cdot v_2 \cdot v_3 \cdot v_4$ possibilities need to be evaluated in order to find the optimal solution.

This discretization problem and the effort to find an optimal stochasticity setup is further addressed in this thesis, as it presents the core problem of Case study 1.

3.4.6. Advantages of SDP

The main advantage of SDP is its robustness. This means they can account for change in parameters and still give good results. A good overview of the this characteristic is given in [27]. Linear programming solution techniques have an exponential growth with the number of stages, which makes it suitable for small scale problems, but complex problems cause computational issues. On the other hand, stochastic dynamic programming, since it is using the backward recursion in stages, only presents a linear growth in execution time when the number of stages is increased. Since SDP is solving one stage at the time, there is virtually no restrictions on complexity of the problem as whole, as it is divided into smaller sub-problems. Finally, there is also no limitations on how precise the discretization of the stochastic variables can be, though, using to many discretisations might lead to increased computational efforts.
Case study 2 – short-term hydropower planning

3.5. Introduction to short-term planning

As mentioned previously, with respect to market liberalization, a lot of power producers face new challenges with the final goal of maximizing their profits and maintaining their position in the market. This problem is also dealt with in short-term optimization that is formulated as an optimization problem where the aim is to determine the unit commitment, maximize the profit and meet the system demand, while taking into account various constraints[26].

The purpose of STHS is to develop schedules for the hydro plants for a period ranging from several hours to a couple of days[25]. In this case, an optimization period of 24 hours will be taken into account.

STHS problems are nonlinear, discrete, nonconvex and large-scale problems. Nonlinearity and nonconvexity come from the relation between the output power and the discharges, while the discreetness comes from the on/off status of the power plants[21].

A Mixed Integer Linear Programming (MILP) approach has been chosen in this thesis as it enables for easy addition of constraints and also, the nonlinearities can be incorporated into the model by piecewise linearizing the unit performance curves. In addition, by using binary variables, the discreteness of the problem can be modeled easily.

By using advanced solvers like CPLEX, fast solutions speeds are achieved, which in combination with advantages mentioned above, make MILP one of the most used optimization techniques.
3.6. Power plant model

The plant in question is an existing Swiss power plant with two reservoirs. The plant configuration can be seen in Fig. 4.

The power plant is a pumped-storage power plant, where the lower reservoir serves as a balancing reservoir, while the upper reservoir is a seasonal reservoir. It has five turbines and two pumps:

- Stalden 1
- Stalden 2
- Zermeggern 1
- Zermeggern 2
- Saas Fee
- Zermeggern_1_pump
- Zermeggern_2_pump

turbines

The power plant data can be found in the Appendix 1. It should be noted that turbines Stalden 1 and Stalden 2, have larger available power then turbines Zermeggern 1 and Zermeggern 2.

As is the case with power plant from Case study 1., this power plant is subject to seasonal inflows and one should consult Fig. 1, for an inflow representation. Equations (3.1) and (3.2) are valid for this system as well.

All the parameters are modeled as follows:

- Reservoir content – m$^3$
- Inflows – m$^3$/s
- Power output – MW
- Water value – EUR/ m$^3$
- Discharge – m$^3$/s

Since inflow and discharge units are not the same, a multiplication factor is used to bring them to the same scale:

\[ 1 \text{ m}^3/\text{s} = 3600 \text{ m}^3/\text{h} \]

Unlike in mid-term planning, the relation between the discharge and the power output has to be modeled precisely and this is done through the use unit performance curves. The concept behind unit performance curves will be explained in later chapters.
Figure 4. Power plant model - case study 2
4. Genetic algorithm

The aim of this chapter is to introduce basic concepts of genetic algorithms. All the concepts are purely theoretical and are not directly applied in the implementation of this thesis. For the practical implementation, Chapter 6. should be consulted.

4.1. Introduction to evolutionary algorithms

The ability of living creatures to live in most remote and isolated areas and adopt to most hostile environments is the result of a Nature’s mechanisms called evolution. The efficiency of evolution as an optimization process has sparked interest among scientist who deal with optimization techniques and a whole branch of techniques were developed based on Darwinian theory and called evolutionary algorithms. These algorithms try to mimic the process of evolution as closely as possible in order to find good solutions to a problem.

There are several different evolutionary algorithm techniques discussed in [5] and the most common are:

- Genetic algorithm
- Evolutionary programming
- Evolution strategies

All three algorithms can yield optimal solutions given complex, multimodal and discontinuous search spaces[5].

The main focus in this thesis is going to be on the genetic algorithm, as it can be implemented into the existing framework easily and also handles large search spaces very well.

It is worth mentioning that evolutionary algorithms fall into the category of stochastic optimizers, which means they operate with a degree of randomness. When going through the search space, they are sampling wide variety of areas while also trying to find areas for future sampling [7].

![Figure 5. Global and local maximum](image-url)
Figure 5. provides an example of a search space with some distinct points. Finding an optimal solution is trying to find a minimum or a maximum of the function, depending on the requirements. What differs a good optimization method from a not so good one is the possibility to find a global optimum, as opposed to being stuck at a local optima. This will be discussed later in more detail.

**Terminology**

Since evolutionary algorithms are based on biological theory, it is necessary to introduced proper terminology that is used in these optimizations[4][7][8]:

- *population* – a group of P individuals that the algorithm manipulates. Each individual is composed of one or more chromosomes

- *fitness value* – measure of solution quality

- *parent selection* – process of selecting best individuals based on fitness value that then form the basis for the new generation

- *crossover* – a form of recombination where parents produce children

- *mutation* – process that changes information in the genome of parents, based on some predefined probability distribution
4.2. Genetic algorithm

As previously mentioned, genetic algorithm (GA) is one of more common evolutionary algorithm that is based on Darwinian theory of evolution. A simple flowchart of GA and its phases can be seen in Fig. 6.

A simple flowchart of GA and its phases can be seen in Fig. 6. Algorithm starts with a set of solutions called the initial population. This population can be determined randomly or be predetermined. Based on a fitness function, all the individuals in the population are assigned fitness value which describes the individuals quality of solution. Several individuals are then chosen as parents, which means they serve as a basis for creation of next generation. This process is done based on the fitness value, so the higher the fitness value, the higher is the possibility of an individual to become a parent. By performing crossover on the parents, children are created and a new generation is formed. Mutation is then performed on the new population. This process is repeated until a stopping criteria is reached.

One can find a pseudo-code for the genetic algorithm below.
Generate an initial population of n individuals \( X = \{X_1, ..., X_n\} \)
Evaluate the fitness value \( f(X) \) of each individual in the generation
While (Stopping criteria are not satisfied)
   Select parents from current population based on their fitness
   Perform crossover on selected parents in order to generate children and form new population
   With a mutation probability \( P \) perform mutation on the new population
   New population becomes current population
End
Return best value

4.3. Genetic algorithm components

4.3.1. Representation

Representation is the first step in any GA since its purpose is to translate the real world problem into a formulation that’s computationally solvable.

The most common representation is in the form of string of numbers, where each part of the string represents a piece of information from the original setup. Usually, bit-string representation is used, where all the elements are either 0 or 1. A simple example of this would be if a potential solution to the GA problem is an integer 10. This integer would be represented by a string of bits 1010.

There are several bit coding possibilities[8] and the most famous one are:

- Gray code
- Standard binary coding

The difference between the two is that with Gray code, all successive values are differentiated by only one bit, while it may not be the case with standard binary coding[11].

<table>
<thead>
<tr>
<th>Gray code</th>
<th>Standard binary coding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Binary</td>
</tr>
<tr>
<td>7</td>
<td>0100</td>
</tr>
<tr>
<td>8</td>
<td>1100</td>
</tr>
</tbody>
</table>

As seen from Table 1, to change states with Gray code, only 1 bit needs to be changed, while the same change of state requires with standard binary coding requires all 4 bits to changes. This way, Gray’s code considerably reduces the potential of error while changing states and is therefore used more frequently.
4.3.2. Initialization

Initialization stands for the process of generating the first population of individuals, where each individual represents a solution to the problem. When defining the initial population, one should take into account several parameters:

- Population size – the numbers of individuals in the population
- Population diversity – based on individual’s location in the search space

One would generally like to have as big of an initial population as possible, however, that would significantly increase the computational speed, as more iterations need to be done, so a balance needs to be found. The size initially depends on the representation, genetic operators used and similar[5].

There are also two possibilities of forming the initial population, with more common one being creating the initial population randomly so the individuals are scattered all over the search space. The other option is manually setting the initial population with worthy values that were obtained from similar research or experience.

The population diversity is extremely important for a GA optimization. Having a diverse population means having solutions in all areas of the search space, which then makes it easier to search for regions with higher quality solutions[8]. Ideally, one would rather have a smaller population that is spread out, but a large population concentrated in one region of the search space.

4.3.3. Parent selection

After the initial population is generated and the fitness values of all the individuals are known, it is time to perform a parent selection process. This process, alongside crossover, is responsible for the very basic concept of GA and that is the survival of the fittest.

This process selects the best individuals in a population, based on their fitness value and they are called parents. The parents then undergo a change and create offspring that forms the next generation.

An example of how this can be done is called a tournament selection. Tournament selection is rather simple method, also based on fitness value of individuals in a population. Firstly, a number $\mu$ is chosen that represents the tournament size. That is the number of individuals from a population that are going to be selected for a tournament. These individuals are selected randomly and the one with the best fitness value is selected to be the parent. Depending on the number of parents, this process is done multiple times. Of course, the higher the number $\mu$, the tougher the competition[3][5].
4.3.4. Crossover

After the parents have been selected, a crossover operator is applied on them and this operator takes characteristics from both parents and creates two offspring individuals. The crossover forces the children to have parts of chromosomes from the parents, thus improving combinatorial diversity that could lead to exploring new areas of search space with fitter results then current. Parts of parent chromosomes that are used are based on random drawing and crossover is thus considered to be a stochastic operator [3].

An example of how a crossover can be performed is a single point crossover. This is a simple way of creating crossover. A cut point in parents chromosomes is chosen randomly and the portions after the cut are exchanged between parents, thus forming two new children. This is shown in Fig 7.[17].

![Figure 7. Single point crossover](image)

One should also notice that this method is biased towards the low order schemas. This is so, because if the fixed positions in the schema are far apart, there is a high probability that some binary values will be altered, thus losing quality characteristics. This disadvantage is reduced in the following two methods [5].

4.3.5. Mutation

Mutation operator is introduced to complement the effect of recombination. There is one drawback of the recombination process and that is that there is no “inflow” of new data into the system, but rather already existing data is managed in different ways. This could lead to the optimization process to converge to a local optima, as opposed to overcoming this and converging toward the global optimum[6].
However, too much new information in the system would destroy the quality solution (schemas) and that’s why the mutation probability $P_m$ is set to low values (typically of 1%) [5].

Mutation is done after crossover, so it’s performed on children, rather than parents.

The principle behind the mutation is that a random number is drawn and if the number falls below the mutation probability, the mutation will occur.

![Mutation Example](image)

**Figure 8. Mutation**

### 4.3.6. Fitness function

Once the population is created, the respective solutions need to be evaluated in order to determine their fitness in the environment. The environment in optimization problems is the objective function. So basically, the fitness function evaluates how good a certain solution is, for the problem of interest. This makes the fitness function one of the more crucial components of GA.

Setting the evaluation criteria can vary from simple numbers (e.g. profit) to complex calculations, depending on the scope of the problem. What is important is that the fitness function shouldn’t be exclusive in a sense that it is a 0 or 1 choice, because then the algorithm searches for a single solution in the whole search space, which is equivalent to searching for a needle in a haystack. It is better that a function as such can “grow” gradually, i.e. going through the search space through desirable areas in search for a near-optimal solution and approach it with satisfaction of some criteria[3].
4.3.7. Performance

Figure 9. shows a typical performance curve of a genetic algorithm.

![Performance curve of a genetic algorithm](image)

It shows that most progress concerning finding the optimal solution is done in the first iterations, while it flattens out later on. From this figure some conclusion concerning initial population and stopping criteria can be drawn. Firstly, the before discussed possibility of setting the meaningful initial population, based on some other simulation, now seems rather unreasonable. This is because it will only take a couple of iterations for the GA with a random initial population to “catch up” to the one that started with a meaningful initial population.

Also, having long optimizations might prove unnecessary since the solutions obtained at the very end of the graph in figure 12, will only slightly differ from those obtained at the half way point.

This phenomenon is occurring because of the effectiveness of crossover and mutation functions. They enable effective movement through the search space and quick elimination of large non-optimal areas[3].
5. Mixed integer linear programming

5.1. General introduction

Most commonly used optimization models are linear models. Those are the models in which all functional relations are linear and methods used to solve these problems are generally known as linear programming[12].

However, the assumption that all variables can obtain real values is not always valid in practice. There are many problems where the integrality of the variables is very important and this leads us to another group of methods commonly known as mixed integer linear programming (MILP). Unlike linear programming, some of the variables considered are integers (or binaries, which then can be considered mixed binary programming).

MILP has established itself as a popular optimization technique for problems that are making decisions under uncertainty, whether it is in operations control, scheduling problems, artificial intelligence and similar. One can find relevant literature about various applications in [18][19].

MILP as such is a subset of a larger field of mathematical programming. In mathematical programming, one has a model that represents a real life system and a set of variables who’s manipulation tries to mimic the actual happening in the system. This model is also represented by a function that serves as a connector between input variables and output data. By trying to optimize this function, one tries to evaluate the quality of the solution obtained by decision making in the process. In order to ensure that the system is behaving as it would in real life, a set of constraints and boundaries is introduced[12].

MILP usually deals with large scale complex problems who’s variables are interdependent. Therefore, an approach where each possible solution is explicitly examined would be computationally untraceable. Therefore, MILP uses the technique called implicit enumeration. According to [12] implicit enumeration is: “A method of solving integer programming problems, in which tests that follow conceptually from using implied upper and lower bounds on variables are used to eliminate all but a tiny fraction of the possible values, with implicit treatment of all other possibilities.”

What this basically means is that a subset of solutions is examined and the optimization theory tries to prove that no other solution is better than the best one found[14].
5.2. MILP modeling

As said previously, MILP is a minimization or a maximization of a linear function that is subject to linear constraints. Most common form can be seen below:

\[
\text{Maximize } \sum_{j=1}^{n} c_j x_j \tag{5.1.}
\]

subject to

\[
\sum_{j=1}^{n} a_{i,j} x_j \leq b_i, i = 1, \ldots, m_1 \tag{5.2.}
\]

\[
\sum_{j=1}^{n} a_{i,j} x_j = b_i, i = m_1 + 1, \ldots, m \tag{5.3.}
\]

\[
x_j \geq 0, \forall j = 1, \ldots, n \tag{5.4.}
\]

\[
x_1, \ldots, x_k \text{ integer } \tag{5.5.}
\]

As one can see, by introducing (5.5.) a normal linear problem is turned into a mixed integer problem. Equation (5.1.) represents the objective function of the model and that is the functions who’s solutions needs to be optimized.

In general, in order to build a proper MILP model, three steps are needed:

1. Determine the variables that represent the model \( X = \{x_1, \ldots, x_n\} \)
2. Build the objective function that represents the wanted solution to the problem
3. Build constraints and boundaries so the model behaves within set limitations

Different solution techniques exist that solve these kind of problems and the most popular ones are:

- Branch and bound
- Cutting planes
- Branch and cut

All of these techniques are exact techniques. This means that if a feasible solution exist they will find it and if it doesn’t exist, they will prove it doesn’t exist[14]. Branch and cut is of interest for this thesis and will be examined in more detail. If one would like to know more about the other two techniques, [14][15] provide more details.
5.3. Building matrices

In order to implement a MILP problem into MATLAB, some thought has to be put into building the constraints into matrices suitable for the solvers. CPLEX, solver used in this thesis supports the use of both equality and inequality constraints. That means the the equations (5.2) and (5.4) will form an inequality matrix, while the equation (5.3) will form an equality matrix. The size of these matrices depends exclusively on the number of time stages used in the optimization.

In order to explain the formation of the matrices, a simple example is provided below.

Example

A pumped-storage power plant is examined. The reservoir level is dependent on the turbining rate as well as the pumping rate. The inflow is only accounted for in the upper reservoir. The aim of the optimization is to maximize the profit over a period of one month with a daily time step.

Now the setup of the problem is as follows:

\[
\max C^T \times X = \max \sum_{t \in T} c_{sp,t} \times u_t - c_{p,t} \times b_t
\]

Subject to

\[
\begin{align*}
v_{t,up} &= v_{t-1,up} - u_t + b_t + q_t \\
v_{t,low} &= v_{t-1,low} + u_t - b_t
\end{align*}
\]

where \( c_{sp,t} \) is the spot price, while \( c_{p,t} \) is the price paid for turbining.

It is seen that the state vector \( X \) is composed of 4 variables:

- Turbinning rate
- Pumping rate

\{ Control variables \}

- Content of the upper reservoir
- Content of the lower reservoir

\{ State variables \}

Now, matrices \( X \) and \( C \) are as follows:

\[
X = [u_1, u_2, \ldots, u_{30}, b_1, b_2, \ldots, b_{30}, v_{1,up}, v_{2,up}, \ldots, v_{30,up}, v_{1,low}, v_{2,low}, \ldots, v_{30,low}]
\]

\[
C = [c_{sp,1}, c_{sp,2}, \ldots, c_{sp,30}, c_{p,1}, c_{p,2}, \ldots, c_{p,30}, 0, 0, 0, 0, 0, 0, 0, 0]
\]

When discussing equality constraints, they have a form of

\[
A \times X = b
\]

where the matrix \( b \) represents the constant values in the equation. An example of equality equations over different time stages is shown below.
Branch and cut method has been extensively used in many commercial solvers (for instance CPLEX) and has good results in reaching optimal solutions.

The general algorithm for a branch and cut method can be seen in Fig. 10.

\[ v_{1,up} = v_{ln,up} - u_1 + b_1 + q_1 \]
\[ v_{2,up} = v_{1,up} - u_2 + b_2 + q_2 \]
\[ \vdots \]
\[ v_{t,up} = v_{t-1,up} - u_t + b_t + q_t \]

Finally, after the equations for all time stages are written, the suitable form is built:

\[
\begin{pmatrix}
-1 & 0 & 0 & \ldots & 0 & 1 & 0 & 0 & \ldots & 0 & 1 & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\
0 & -1 & 0 & \ldots & 0 & 0 & 1 & 0 & \ldots & 0 & -1 & 1 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\
0 & 0 & -1 & \ldots & 0 & 0 & 0 & 1 & \ldots & 0 & 0 & -1 & 1 & \ldots & 0 & 0 & 0 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & -1 & 0 & 0 & 0 & \ldots & 1 & 0 & 0 & \ldots & -1 & 1 & 0 & 0 & \ldots & 0 \\
\end{pmatrix} \times \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_{30} \end{pmatrix}
\]

Since these matrices are large in size and contain very few non-zero elements, sparse-matrices in MATLAB are used, to prevent RAM memory problems. Other equality and inequality matrices are built in the same way.

5.4. Branch and cut method

Branch and cut method has been extensively used in many commercial solvers (for instance CPLEX) and has good results in reaching optimal solutions.
As seen from Fig. 10, this method is based on a search tree that is made up of nodes. Each node stands for a linear programming sub-problem that needs to be evaluated. The starting point for this method is performing linear programming relaxation of the MILP problem. LP relaxation is a relatively straightforward process in which all integrality constraints are removed and replaced with continuous equivalents. A small example is shown below.

**Integral constraint**

\[ x_i = \{0,1\} \]

**Continuous constraints**

\[ x_i \geq 0 \text{ and } x_i \leq 1 \]

In this way the LP representation of the MILP problem can be solved and the obtained solution mostly doesn’t contain integer values (if it did, it means the process is done and the solution was found immediately). Also, since the area of feasible solution for LP is larger than for MILP, the optimal solution of MILP problem will be included in the LP solution as well. This is nicely shown on the Fig. 11., that represents a simple example of a LP problem that has already been relaxed[13].
Figure 11. Feasible solution for LP and MILP problems

In this figure, straight red lines represent the continuous constraints and the area between them is a feasible solution for an LP problem. Blue crosses represent the feasible solution for an integer problem and now it is clear how the feasible area of an LP problem is bigger than MILP problem.

After the LP relaxation has been done, cutting planes for the first (root) node are introduced and an incumbent solution is found. This solution is the current best solution that satisfies all the integrality requirements. The cuts are added to the node until solutions are breaking the cuts. Once no violated cuts are found, the process is stopped.

Once the cutting process is stopped, branching occurs. Here, the main problem is divided into sub-problems, thus generating new nodes in the node tree. For each sub-problem, the integrality constraints are again neglected and cutting planes occur. Once this phase is done, the obtained node solution is checked against the integrality constraints. If the solution satisfies all the constraints and its value is bigger than the current incumbent value, it becomes the new incumbent. If the value is less than the current incumbent, the node will be branched further. If, at any time during this process, the node becomes infeasible, it is removed from the tree [20].

This process is then performed until there are no more active nodes and the optimal solution has been found.
6. Implementation of the solution

Case study 1

6.1. General introduction – case study 1

As said previously, the goal of this case study is to determine the optimal stochasticity setup in mid-term hydropower planning. This is done for two main reasons:

- Computational traceability
- Quality of the results

Having presented the theoretical introduction to all the optimization and programming techniques in previous chapters, the implementation of solution will be given.

A flowchart of the solution can be seen in Fig.12:

Figure 12. Implementing GA and MILP procedures
Based on historical values, one can generate a certain number of testing scenarios and use them to determine the “ideal profit” (by means of MILP), meaning the profit a company would obtain if it knew all the inflows and all the prices for the upcoming year. This is, of course, not reasonable and therefore one tries to get as close as possible to this desired value. This is where the stochastic dynamic programming steps in. The aim of stochastic dynamic programming is to determine the optimal policy, almost like a look up table, where decisions are suggested based on the current state of the system.

This policy is then tested with the same testing scenarios previously used in MILP calculation. The profits of both techniques are then compared and optimal solution tries to be found.

The role of genetic algorithm is to generate the initial population of the stochastic solution and based on the phases presented in Ch. 4., generate new populations that are closer to the optimal value.

More detailed look on how these were implemented can be found in the following chapters.

6.2. Mixed integer linear programming

6.2.1. Generating testing scenarios

In order to obtain results that are as close to the optimal one as possible, several inflow and prices scenarios need to be created. The starting point for creating scenarios is available historic data for inflows and prices.

Since the price data is available in hourly increments, while the inflows are known as daily values, it was assumed that the distribution of inflows during the day is constant:

\[ \text{Inflow}_{\text{day}} = 24 \text{m}^3 \Rightarrow \text{Inflow}_{\text{h}} = \frac{24}{24} = 1 \text{m}^3/\text{h} \]

As the reservoir is large in proportion, the hourly inflow changes shouldn’t influence the state of the system and hence, this assumption was made.

Unfortunately, since the data is known only for the last couple of years, it wasn’t possible to use actual yearly data as scenarios, but a distribution had to be assumed. All the variables were presented with a normal distribution,

\[ X \sim N(\mu, \sigma^2) \quad (6.1.) \]

where \( \mu \) is the mean value and \( \sigma \) is the standard deviation on an hourly basis. That basically means that for each hour, values from different years were taken and their mean and standard deviation were calculated.

Having in mind that the variables are presented with a normal distribution, simplified Monte Carlo simulation method is used to generate testing scenarios. As development of the testing scenarios is not the core topic of this thesis, the Monte Carlo method won’t be explained in
Monte Carlo method is a technique that deals with random numbers and probability distributions to obtain meaningful results. Normally, given the set of input parameters and the accompanying equations, one gets a certain output. The idea behind the Monte Carlo simulation is to evaluate this model using a set of random parameters as inputs. These parameters are generated from the probability functions of the variables (inflows, prices), thus mimicking the sampling procedure of the actual population[16]. This means that in each hour, some number of random values were chosen from the normal distribution of the variable.

In this thesis, 10 inflow and 10 price scenarios were defined and some of them can be seen in Fig. 13 and Fig. 14.

As it can be seen, the scenarios are good representations of inflow data shown in Fig.1. and it’s seasonality and should, therefore account for meaningful results.

The price scenarios can be seen in figure 14.
6.2.2. Problem formulation

It has already been mentioned that MILP is used to find an “ideal” solution to the optimization problem, taking into consideration known values of inflows and prices. Price and inflow scenarios from the previous chapter are used to test the model.

There are several assumptions that need to be taken into account:

- All turbining, pumping, spilling and reservoir variables are continuous
- The secondary control on offer is either 0 MW or 40 MW which is the maximum possible amount
- There is one binary variable – the one that states whether the secondary control is on offer or not
- When secondary control is on offer, the turbining amount has to be at least the amount of technical minimum
- No pumping is possible when secondary control is offered

All the calculations were done in MATLAB R2011b with CPLEX optimization toolbox.

**Time frame**

The chosen time frame for the simulation is 8832 hours, which is equivalent to a year and four extra days. The simulation is done in hourly intervals.
**Objective function**

The objective of the optimization is to maximize the profit:

\[
\pi = \max \sum_{t \in T} c_{p,t} \cdot u_t - c_{p,t} \cdot b_t - c_{s,t} \cdot s_{u,t} - c_{s,t} \cdot s_{l,t} + c_{2nd,t} \cdot bid_t + c_{p,t} \cdot bid_t \tag{6.2}
\]

where \(c_{2nd,t}\) is the price of the secondary control for an hour \(t\) and \(bid_t\) is the amount of secondary control offered. Other expressions stand for turbining, pumping and spilling profits/loses.

The objective function is subject to a set of equality and inequality constraints:

**Equality constraints**

Equality constraints evolve around the upper and lower reservoir levels.

\[
res_{\text{level}}_{t}^{\text{up}} = res_{\text{level}}_{t-1}^{\text{up}} + \text{inflow}_{t} - u_{t} - s_{u,t} + b_{t} - \text{bin}(bid_{t} + \text{techmin}) \tag{6.3}
\]

\[
res_{\text{level}}_{t}^{\text{low}} = res_{\text{level}}_{t-1}^{\text{low}} + u_{t} - b_{t} - s_{l,t} + \text{bin}(bid_{t} + \text{techmin}) \tag{6.4}
\]

The variable \(\text{bin}\) is a binary variable stating whether the secondary control is on offer or not. The time variable \(t\) is modeled so it accounts for the state of the period at the end of time \(t\). So for instance \(res_{\text{level}}_{t}^{\text{up}}\) is the reservoir level at the end of time \(t\), which means after all the turbining/pumping occurred and inflows were taken into account.

**Inequality constraints**

The non-equality constraints set the limits for turbining and pumping. For instance (6.5.) enables the combined turbining and secondary control offer to exceed the maximum turbining value. (6.6.) shows that in case the secondary control is offered, the turbining has to be at least the amount of the technical minimum. Finally, (6.7.) sets the constraint that no pumping is possible when secondary control is offered.

\[
u_{t} + \text{bin}(bid_{t} + \text{techmin}) \leq u_{max} \tag{6.5}
\]

\[
-u_{t} + \text{bin}(\text{techmin}) \leq 0 \tag{6.6}
\]

\[
b_{t} + \text{bin}(b_{max}) \leq b_{max} \tag{6.7}
\]
Boundary conditions

\[ 0 \leq \text{res\_level}_t^{up} \leq \text{res\_level}_\text{max}^{up} \] \hspace{1cm} (6.8.)

\[ 0 \leq \text{res\_level}_t^{low} \leq \text{res\_level}_\text{max}^{low} \] \hspace{1cm} (6.9.)

\[ 0 \leq s_{ut} \leq \infty \] \hspace{1cm} (6.10.)

\[ 0 \leq s_{lt} \leq \infty \] \hspace{1cm} (6.11.)

Boundary conditions assign upper and lower values to variables that have not yet been considered, which are reservoir levels and spillage possibilities.

6.3. Stochastic price modeling for SDP

Optimal operation of hydropower plants depends heavily on spot price forecasts. There are several ways of building these forecasts and the most popular are based on the historical prices or on a forward simulation of some kind. These forecasts will be given as a certain number of future price scenarios, within a yearly horizon and a weekly time step. Since the historic data is available from only 5 last years, this would provide a limited insight, thus a forward simulation is chosen.

6.3.1. Geometric Brownian motion

Geometric Brownian motion is an easy to use but nevertheless sensible price model that takes into account the stochasticity of the price and determines possible evolutions of the price over a predetermined price period. It takes into account the fact that the prices become more insecure with time.

\[ P_{t+1} = P_t(1 + \hat{\mu}dt + \hat{\sigma} \cdot \sqrt{dt} \cdot \xi_t) \] \hspace{1cm} (6.12)

Equation (6.12) represents the discrete time version of the Geometric Brownian motion. \( P \) is the random variable representing the price, \( \hat{\mu} \) is the drift term, \( \hat{\sigma} \) is the volatility and the distribution \( \xi_t \) is standard normal.

As seen, in order to simulate possible future prices, the current price and the volatility need to be known. The volatility used in (6.12) is obtained from historic data and normalized to an appropriate time frame if necessary. This is done as follows:

\[ \hat{\sigma} = \frac{\sigma}{\sqrt{t_{app}}} \] \hspace{1cm} (6.13)
where \( t_{app} \) is the time frame to which the prices want to be normalized[24].

The volatility was calculated in the following manner:

- A time series of a couple of days was chosen as a base for the calculation
- Logarithmic price changes between the prices were calculated
- The volatility within the price series was calculated
- Using (6.13), the volatility was normalized to an appropriate time frame

For this thesis, 50 scenarios are generated by the Geometric Brownian motion, with a horizon of one year and weekly time steps.

![Price scenarios](image)

**Figure 15. Price scenarios**

Figure 15. shows 50 scenarios generated by the Geometric Brownian motion that are used by SDP to determine the optimal policy of the hydropower plant.

### 6.3.2. Stochastic representations

Now that 50 scenarios have been created, a discrete Markov chain model is made. The model is shown in Fig.16. According to [25], the prices in one week are correlated with prices in the previous week, which justifies the use of Markov chain model.
It consists of a given number of price nodes $P_i(t)$ in each time step and transition probabilities $p_{ij}(t)$ that determine the probability of a price $P_j(t+1)$ if at time step $t$, the price was $P_i(t)$. It should be noted that the number of nodes and thus transition probabilities can vary from one time step to another. In Fig. 16, the first time step has 4 price nodes, while the second one has 3 price nodes. In this thesis, the minimum number of nodes in a time step is 1, while the maximum is 4. This has been chosen, since having more price representations in a single time step would significantly prolong the simulation time.

\[
\begin{align*}
P_1(t) & \rightarrow p_{1,2} \rightarrow P_2(t) \rightarrow p_{1,3} \rightarrow P_3(t) \rightarrow p_{3,1} \rightarrow P_1(t+1) \\
P_2(t) & \rightarrow p_{1,2} \rightarrow P_1(t) \rightarrow p_{1,3} \rightarrow P_3(t) \\
P_3(t) & \rightarrow p_{3,1} \rightarrow P_1(t+1) \\
P_4(t) & \rightarrow \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \qua
Those intervals may be consisted of different number of price scenarios. The node value is determined by calculating the mean value of all the prices within an interval.

### 6.3.4. Determining transition probabilities

The second step in the process is determining the transition probabilities between time steps $t$ and $t+1$, for every $t$.

How this is calculated is shown in (6.14)

$$ p_{ij}(t) = \frac{N_{i,j}^P(t)}{N_i(t)} \quad (6.14) $$

Where $N_i(t)$ is the number of scenarios belonging to the node $i$ in time step $t$ and $N_{i,j}^P(t)$ is the number of scenarios from the node $i$ in time step $t$ that belong to the node $j$ in time step $t+1$.

### 6.4. Genetic algorithm in the SDP schema

**Fitness function**

The fitness function is the difference between the profits obtained by MILP and by SDP. This means that the fitness function is trying to find a stochastic setup that will bring the profit obtained by SDP as close as possible to the “ideal” one of MILP. This can be seen in (6.15)

$$ ff = \frac{\sum_{m \in M} (\pi_{m,\text{MILP}} - \pi_{m,\text{SDP}})}{M} \quad (6.15) $$

where $ff$ is the value of the fitness function and $M$ is the number of scenarios used.
**Time frame**

The time frame of this part is the same as with MILP part, which equals 8832 hours, however the arrangement is different. Here, the time is divided into 56 stages, where the 4 four stages represent the 4 days, and other 52 stages each represent one week of the year.

**Set up and stopping criteria**

The built-in MATLAB function for genetic algorithm is used in this thesis. Since the program is used on an integer problem, special crossover and mutation functions are used, that force the variables to stay integers.

Also, it should be noted that, when dealing with integers, the GA function is minimizing the penalty function, not the fitness function. The difference between the two is the fact that the penalty function includes the infeasibility factor. Since the setup of the GA in this thesis is straight forward, the penalty function is equal to the fitness function.

The setup of the GA is:

- Number of individuals per generation – 7
- Number of generations – 100
- Stopping criteria – 30 generations without significant improvement

A relatively low number of generations was chosen based on Fig. 9 and the fact that the problem is computationally elaborate. Test have also been run with more generations, but the results didn't change, so this set up was chosen as a final one.

Finally, the complete implementation can be seen in Fig. 18.
Generate initial population

Determine optimal policy

Perform forward step in SDP to determine the profit

Evaluate fitness function

Profit from MILP

Stochastic representations (from Geometric Brownian motion)

Scenarios from MILP

Generating next generation

Crossover

Selection

Evaluating fitness function

Store best stochasticity setup

Stopping criteria reached

End

Figure 18. Implementing GA into SDP framework
Case study 2

6.5. General introduction – case study 2

The assumptions used in this model are as follows:

- Power plant owner acts as a price taker – has no market power
- Head dependency is disregarded
- Pumping and turbining at Zermeggern can’t happen simultaneously
- Pump has to on/off for a minimum of two hours in a row
- Pumping power is either 0 or 23 MW

The reason that the head dependency is disregarded is the fact that the upper reservoir has large volume and with the known schedule and estimated inflows, the change in the head over a period of a day is very small and wouldn’t account for a significant change in the results. To see the proof of this, the results chapter should be consulted.

Load curves and price curves used in the simulation can be seen in Fig.19. The prices have been taken for a specific day from EEX, in order to give as realistic approach as possible. The schedule for the same day has been built from confidential data and it’s representation is shown with the blue line. Also the reservoir levels were set up accordingly.

Figure 19. Schedule and price for the short-term planning
6.6. MILP approach – case study 2

6.6.1. Unit curve linearization

As it was mentioned previously, the nonlinearity and nonconcavity of the STHS problem comes from the dependency of power output on the discharge values. Disregarding this dependency would lead to big discrepancy of results and therefore needs to be modeled properly. This relationship is normally represented by the so called unit performance curve. Each plant has multiple performance curves, one for each value of the head[26]. An example of a unit performance curve for three different head values is shown in Fig. 20.

![Figure 20. Unit performance curves for different head values](image)

Since the head dependency is disregarded in this case, only one performance curve per unit is needed. An example of the performance curve can be seen in Fig. 21.
In order to approximate these curves, piecewise linearization method is used. Nonconcavities have been modeled through the use of binary variables. Figure 22. shows the performance curve linearized into three pieces. By using binary variables, the whole range of the curve can be used, unlike some cases where only the local best efficiency points are used[22].

Since the linearization was done in three pieces, three discharge blocks were formed and the green vertical lines show their borders. Each block has its own limits as well as the curve slope that will be used in the simulation.
6.6.2. Problem formulation

The goal of any participant in a market is to maximize its own profit. Since the power plant in question is a hydropower plant, the production cost can be neglected, so the revenue obtained by selling the electricity needs to maximized. Also, the value of stored water in the reservoir should be taken into account.

Knowing this, the problem is presented in a form of an objective function with associated constraints.

**Objective function**

\[
\text{max } \sum_{t \in T} \sum_{i \in I} \pi_i(t) * c_{p,t} + v_t * r
\]  

(6.16)

In (6.13) the first term is related to the revenue of each of the plants, while the second term determines the value of stored water at the end of the planning period.

**Constraints**

1. **Load balance**

\[
\sum_{i \in G} p_i(t) - \sum_{i \in P} p_i(t) = D_t
\]  

(6.17)

The load balance constraints says that the sum of hydro power must satisfy the predetermined system load demand at each time step \( t \).

2. **Reservoir water balance**

\[
v_t = v_{t-1} + q_t + \sum_{i \in U} (s_i(t) + u_i(t)) - u_i(t) - s_i(t) \pm b_t
\]  

(6.18)

Since there is a possibility of pumping between two reservoirs, the final term in (6.15) should be altered depending on which reservoir is discussed.

3. **Pumping and turbining exclusivity constraint**

\[
x_{i,Zerm}(t) + x_{i,Zerm,pump}(t) \leq 1
\]  

(6.19)

This constraint is only valid for Zermeiggern turbines and pumps, as it is allowed to pump, while Stalden turbine is operating.
4. Unit availability constraint

\[
L^q_{\text{min}} \leq \sum_{i \in G} x_i(t) + \sum_{i \in P} x_i(t) \leq L^q_{\text{max}} \tag{6.20}
\]

Unit availability constraint is associated with maintenance or similar events, as it states the minimum and maximum number of units online at a certain time step. \(L^q_{\text{max}}\) and \(L^q_{\text{min}}\) represent the maximum and minimum number of units online respectively.

5. Minimum up time for the pump is 2 hours

\[
x_i(t) - x_i(t-1) = n_i(t) - a_i(t) \tag{6.21}
\]

\[
n_i(t) + a_i(t) \leq 1 \tag{6.22}
\]

\[
n_i(t) - a_i(t+1) \leq 1 \tag{6.23}
\]

Equation (6.21) sets the connection between the on/off status of the pump and start-up and shut-down variables. Equation (6.22) states that shut-down and start-up can’t occur at the same period. Finally, equation (6.23), sets the constraint that if the pump was started up in hour \(t\), it can not be shut down in hour \(t + 1\), which automatically accounts for at least two consecutive pumping hours.

6. Hydrological constraints

Hydrological constraints deal with linearization of performance curves.

\[
p_i(t) - p_{01}(i) \times x_i(t) - \sum_{i \in L} u_i(i, t) \times \rho_i(i) = 0 \tag{6.24}
\]

\[
u_i(t) = \sum_{i \in L} u_i(i, t) + U_i \times x_i(t) \tag{6.25}
\]

\[
u_1(i, t) \leq U_1(i) \times x_i(t) \tag{6.26}
\]

\[
u_1(i, t) \leq U_1(i) \times w_1(i, t) \tag{6.27}
\]

\[
u_1(i, t) \leq U_1(i) \times w_{l-1}(i, t) \tag{6.28}
\]

\[
u_1(i, t) \geq U_1(i) \times w_1(i, t) \tag{6.29}
\]

Constraint (6.24) forces the output to be equal to the minimum power output plus the blocks of the piecewise linear curve. Constraint (6.25) states that the water discharge is equal to the minimum discharge plus sum of water discharged in all blocks. Constraints (6.26)-(6.29) set the limits of water discharged in each of the blocks.

Important data related to 5 turbines and 2 pumps can be found in table 2. Actual data is confidential so an altered version is offered.
Table 2. Data concerning different units

<table>
<thead>
<tr>
<th>Plant</th>
<th>$U_i$</th>
<th>$\bar{U}_i$</th>
<th>$P_{01}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stalden 1</td>
<td>10</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>Stalden 2</td>
<td>10</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>Zerm 1</td>
<td>10</td>
<td>19</td>
<td>10</td>
</tr>
<tr>
<td>Zerm 2</td>
<td>10</td>
<td>19</td>
<td>10</td>
</tr>
<tr>
<td>Saas Fee</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Zerm_pump1</td>
<td>0</td>
<td>12</td>
<td>/</td>
</tr>
<tr>
<td>Zerm_pump2</td>
<td>0</td>
<td>12</td>
<td>/</td>
</tr>
</tbody>
</table>

6.6.3. Water values

Water value normally indicates the value of additional unit of water available. The water value curves are obtained from the mid-term hydro scheduling. There, the reservoir was discretized into 12 levels and the water value curve was made for each of the time steps (1 week interval). It is pretty logical that the water has more value when the reservoir is emptier, rather than when it is full.

For the day of the optimization, the weekly curve can be seen in Fig.23.

![Water value curve](image)

Here, a water value (EUR/MWh) is shown for every reservoir level. Since the initial reservoir level for the short-term scheduling is known, a water value has been chosen accordingly and is:

\[ \gamma = 76 \text{ EUR/MWh} \]
7. Results

7.1. Case study 1

The results of case study 1 will be divided into two sub-sections, one showing the results from the MILP solution, while the other from the genetic algorithm part that will include the optimal stochasticity setup.

7.1.1. MILP solution

The program was executed in hourly increments over a period of the whole year and included seven variables, it would be rather difficult to observe all the results in one figure, so a brief overview of turbining and pumping over this period is given in Fig.24.

![Figure 24. Yearly schedule of the power plant](image)

In Fig.24., we can see that the reservoir level is the same at the beginning of the period, as well as at the end, which represents the real world case as well. Also, at periods of low inflow at the beginning of the year, the turbining amounts are generally lower, while they increase with high inflows and higher reservoir levels.

In order to better understand the processes happening in the optimization, weekly representation will be shown as well. Two figures will be given, one that shows the variables connected to the upper reservoir and one to the lower reservoir. The input data to the model was one set of price and inflow scenario that were later used in GA.

Figure 25., shows the variables connected to the upper reservoir.
As seen from Fig. 25, the results seem meaningful in a sense that one can notice that the reservoir level falls when turbining is occurring, while it is raising when pumping is occurring. Also, on occasions, marked with a X sign, secondary control is on offer and the condition that at that time, turbining amount has to be at least of the amount of technical minimum is also satisfied. Since the upper reservoir is large in size, it never reaches its maximum level of water and no there is no need for spilling to occur and it has therefore been omitted from the figure. It should be noted that the secondary control, when on offer, is in the same range as the pumping power (40MW), which would make the figure a little harder to examine, so it has been raised to 100 MW in Fig. 25 for better visibility.

Figure 26., shows the outputs for the same time period, with regards to the lower reservoir.
Here, results are also proven to be meaningful. In Fig. 26., opposite to Fig. 27., when turbining occurs, the reservoir level increases, while it decreases when pumping occurs. Even though the reservoir is much smaller in size, in this particular week it doesn't reach the maximum level so no spillage needs to occur.

The final profit differs depending on the scenario and the results are shown in the table below:

Table 3. Profits over different scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Profit (monetary value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.6356*10^7</td>
</tr>
<tr>
<td>2</td>
<td>3.6297*10^7</td>
</tr>
<tr>
<td>3</td>
<td>3.5508*10^7</td>
</tr>
<tr>
<td>4</td>
<td>3.5902*10^7</td>
</tr>
<tr>
<td>5</td>
<td>3.5684*10^7</td>
</tr>
<tr>
<td>6</td>
<td>3.6034*10^7</td>
</tr>
<tr>
<td>7</td>
<td>3.5821*10^7</td>
</tr>
<tr>
<td>8</td>
<td>3.5983*10^7</td>
</tr>
<tr>
<td>9</td>
<td>3.5560*10^7</td>
</tr>
<tr>
<td>10</td>
<td>3.5795*10^7</td>
</tr>
</tbody>
</table>
7.1.2. Genetic algorithm

In order to assess the influence of price stochasticity in SDP for mid-term hydro scheduling, the genetic algorithm was implemented and the stochasticity setup was tested with a number of testing scenarios. Even though the model is simple, it represents the main point of mid-term hydro scheduling, which is maximizing the profit by efficient use of resources. This method and model can be expanded to suit more complex models such as hydrothermal problem or a multiple reservoir problem.

Prior to the start of this simulation, there were some expectations considering the results:

- Stochasticity should get higher with progression of time
- Times with higher volatility should have higher stochasticity

The first expectation is rather simple, as it is easier to predict stochastic variables in short-term and as time goes by it gets harder to obtain quality assumptions. Therefore, more representations should be taken into account later on, while at the beginning, it might be enough to take only deterministic solutions.

The term of volatility is usually connected to the financial world and can be defined as a statistical measure of the dispersion of returns for a given security or market index [16].

What this means is that volatility gives an idea of level of uncertainty in a variable and the changes in value that might occur. So, the higher the volatility, the less certain one is about its expected value and thus, higher stochasticity setup should be used. Since the prices were modeled with GBM, where the volatility is constant, the only expectation left is the progression through time.

The results can be seen in Fig.27. As it can be seen, the results don’t accurately fulfill the expectations, as undesired peaks occur. This may have happened because of an inaccurate price model.

This stochasticity setup has, nevertheless, been tested in SDP according to time and profit, with other setups and the results can be seen in Table 4. The profit was calculated by performing the forward step of SDP with testing scenarios, so basically, the optimal policy was put to test with actual scenarios.
Table 4. Analysis of different stochasticity setups

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Deterministic – one representation</th>
<th>Stochastic – 4 representations</th>
<th>Optimal stochasticity setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>25.98</td>
<td>2.99*10^7</td>
<td>65.93</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>26.97</td>
<td>2.92*10^7</td>
<td>65.99</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>26.01</td>
<td>3*10^7</td>
<td>64.44</td>
</tr>
<tr>
<td>Scenario 4</td>
<td>25.80</td>
<td>2.96*10^7</td>
<td>64.84</td>
</tr>
<tr>
<td>Scenario 5</td>
<td>25.32</td>
<td>2.98*10^7</td>
<td>64.58</td>
</tr>
<tr>
<td>Scenario 6</td>
<td>26.15</td>
<td>2.94*10^7</td>
<td>66.13</td>
</tr>
<tr>
<td>Scenario 7</td>
<td>25.01</td>
<td>3.02*10^7</td>
<td>64.69</td>
</tr>
<tr>
<td>Scenario 8</td>
<td>25.62</td>
<td>2.90*10^7</td>
<td>65.83</td>
</tr>
<tr>
<td>Scenario 9</td>
<td>25.48</td>
<td>2.97*10^7</td>
<td>65.16</td>
</tr>
<tr>
<td>Scenario 10</td>
<td>26.35</td>
<td>2.67*10^7</td>
<td>64.15</td>
</tr>
<tr>
<td>Average values</td>
<td>25.869</td>
<td>2.935*10^7</td>
<td>65.174</td>
</tr>
</tbody>
</table>

The first conclusion drawn from this simulation is that both deterministic and fully stochastic approaches give almost the same results. This means that using the fully stochastic approach, doesn’t represent a significant improvement, compared to the deterministic approach. This information is important in case when large system with multiple reservoirs want to be examined, as it justifies the use of deterministic approach that doesn’t require significant model alterations and its computational time is much shorter. Similar results have been already been discussed in [28].

On the other hand, using the optimal stochasticity setup gives a relatively big improvement over other two approaches. It is seen that the execution time compared to the stochastic setup with 4 representations is 30% quicker and gives around 6% better result. When compared to the deterministic case, it is expectedly slower by around 42%, but the result it gives is again around 6% better. This can be seen in Fig. 28.
Figure 27. Optimal stochasticity setup over 56 phases

Figure 28. Analysis of different stochasticity setups
In Fig. 28., scenarios grouped around 25\textsuperscript{th} second are the ones representing the deterministic approach, while the ones grouped around the 45\textsuperscript{th} second are the ones ran with the optimal stochasticity setup. Finally, the ones grouped around 65\textsuperscript{th} second represent the stochasticity setup with 4 variables.

The evolution of the fitness function over generations is showed in Fig. 29.
7.2. Case study II

As known from the model description, there are 5 turbines and 2 pumps in the system and the data obtained from piecewise linearization of their performance curves can be seen in table below. It should be noted that, since the pumps only operate on full power or don’t operate at all, it was not necessary to piecewise linearize their performance curves.

Table 5. Piecewiese linearization data

<table>
<thead>
<tr>
<th>Plant</th>
<th>$\rho_1$</th>
<th>$\rho_2$</th>
<th>$\rho_3$</th>
<th>$\bar{U}_1$</th>
<th>$\bar{U}_2$</th>
<th>$\bar{U}_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stalden 1</td>
<td>12.13</td>
<td>12.5</td>
<td>13.1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Stalden 2</td>
<td>12.13</td>
<td>12.5</td>
<td>13.1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Zerm 1</td>
<td>2.63</td>
<td>3.61</td>
<td>3.42</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>Zerm 2</td>
<td>2.63</td>
<td>3.61</td>
<td>3.42</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>Saas Fee</td>
<td>1.8</td>
<td>2.6</td>
<td>3.3</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 6. Daily inflows into the reservoirs

<table>
<thead>
<tr>
<th>Reservoir</th>
<th>Daily inflow (m$^3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mattmark</td>
<td>130 000</td>
</tr>
<tr>
<td>Zermeiggern</td>
<td>190 000</td>
</tr>
</tbody>
</table>

Table 6. presents some interesting data concerning two reservoirs. Even though Mattmark reservoir is much larger in size, the inflow to Zermeiggern reservoir is almost 50% bigger. This information will greatly influence the results of the optimization.

Figure 30. shows the calculated 24 hour schedule of these power plants. Turbines Stalden 1 and Stalden 2 are covering most of the load, which is reasonable since the inflow to Zermeiggern reservoir is larger than the inflow to Mattmark. Since Mattmark serves as a seasonal reservoir, it is reasonable to take advantage of the high inflows and save the water in Mattmark for later. Also, as seen from Table 5., Stalden turbines have much higher slope values, which means that for the equal amount of discharge, more output power is obtained. This is due to the higher head in Mattmark reservoir.
Figure 30. 24 h schedule of designated power plants

Figure 31, Fig. 32 and Fig. 33, show the daily schedule of turbines Stalden 2, Zermeiggern 1 and pump Zermeiggern 2 respectively. These figures are presented for better understanding of the optimization problem.
As seen from Fig. 33, the pump is operating during the night, when the prices are lower. It also pumps for two hours in row, which was a requirement for a pump. During these two hours, the whole load is covered by Stalden 1 and Stalden 2 turbines, because the Zermeiggern turbines were not allowed to operate.

In order to understand the results more clearly and to prove the optimization setup was indeed meaningful, one unit is taken and shown in more detail. Figure 34., shows one of Zermeiggern turbines. All other turbines behave in a similar way.
As seen, in 2\textsuperscript{nd} and 10\textsuperscript{th} hour, the turbine is working at its maximum power, since the discharge is maximum. In 19\textsuperscript{th} and 23\textsuperscript{rd} hour the plant is operating close to its technical minimum as it is using only the first discharge block.

Figure 34. Discharge data for Zermeiggern1 turbine

Figure 35. shows the profit obtained in this 24 hour period, in hourly increments. The total profit was 103 410 EUR.
Figure 35. Hourly profit with associated prices

Figure 36. shows the reservoir fluctuations within a day, where it is seen that, under the assumption of full reservoir at the beginning of planning period, the daily change in the reservoir content is less the 0.1%, thus supporting the assumption of disregarding head dependency. Also, full use of high inflows to Zermeiggern reservoir can be seen, as the reservoir level in Mattmark at the end of the optimization period is full, thus maximizing the value of stored water.

The content of the lower reservoir is seen in Fig. 37. Daily fluctuations are higher than in Mattmark reservoir, as it is smaller and Stalden turbines are used more. However, due to high inflows, the reservoir content at the end is very close to the starting one.
All the calculations were done using CPLEX and were executed in 10 seconds of CPU time. The solution not only gives the schedules but also unit commitment statuses, which may be used in case start-up costs are considered. This model can also be used, with minor alterations, as a part of a more complex hydro-thermal optimization model. Model extensions like introducing multiple reservoirs can be implemented into the model easily.
8. Conclusion and future work

8.1. Conclusion

Two time intervals were used for hydropower planning optimization in this thesis – mid-term planning which is done in a yearly time span and the short-term planning where a 24 hour period is taken. When dealing with mid-term planning, stochastic variables need to considered and modeled.

Stochastic dynamic programming is one of more common optimization techniques used when there are uncertainties, like prices or inflows, included in the model. However, when dealing with complex models, it may fall under the curse of dimensionality.

By implementing the genetic algorithm into the existing SDP schema, an optimal stochasticity setup has been found. The determining factor in the optimization process was the profit generated by the power plant within a year. In order to obtain meaningful results, the prices needed to be modeled properly. This was done by creating price scenarios and modeling them according to Markov chain principle, which accounted for correlation between different time steps.

The optimal stochasticity setup was compared to deterministic case and the case using 4 stochastic representations in each stage and the results confirmed the quality of the solution. The profit was higher in comparison to other approaches and the time saved compared to using 4 representations was significant. Also, it has been shown that there is no difference between using the deterministic case and the fully stochastic case, as they yield approximately same results.

Mixed integer linear programming approach was used in the short-term scheduling, where demand satisfaction was the main goal. Since the turbine unit performance curves are nonlinear and nonconcave, the introduction of binary variables was necessary. This approach is a tool for a hydropower company to maximize their profit, while satisfying the demand. It is shown that the pump operation is meaningful during low price periods.

The computations shown that the approach gives good results in good computational time.
8.2. Future work

As mentioned, the concepts presented in this thesis need more work in order for them to give more complete and accurate results. The possible extensions are listed below:

Stochastic modeling

The model presented in the case study 1 can be tried out with using deterministic prices, but stochastic inflows, which would account for strong seasonality and could give more meaningful results. A possibility is to include both price and inflows as stochastic variables for a system that would strongly resemble the real world scenario. Also, trying out with more than 4 stochastic representations might yield better results. Modeling possibilities other than GBM can be tried out as well.

Model extension

By including profit as the only determining factor in case study 1, the model was quite limited. In order to try and achieve perfect stochastic dependency over time, some other criteria might be used as fitness function. One example is to account for volatility in the stochastic modeling or to have execution time as a factor as well.

Also, in case study 2, some model extensions could be used in order to get more accurate models. Inclusion of startup costs or head dependencies are at the top of the list.

Heuristics

Genetic algorithm was used as a search heuristics to find the optimal stochasticity setup. In order to test the efficiency of the genetic algorithm, some other search heuristics like greedy algorithm might be implemented into the existing model.
Appendix 1.  Power plant data

A.1. Case study 1 data

Table 7. Case study 1 power plant data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Power [MW]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turbine</td>
<td>120</td>
</tr>
<tr>
<td>Pump</td>
<td>23</td>
</tr>
<tr>
<td>Technical minimum</td>
<td>10</td>
</tr>
<tr>
<td>Max secondary control offer</td>
<td>13</td>
</tr>
</tbody>
</table>

A.2. Case study 2 data

Table 8. Case study 2 power plant data

<table>
<thead>
<tr>
<th>Turbine</th>
<th>Maximum power [MW]</th>
<th>Technical minimum [MW]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stalden 1</td>
<td>45</td>
<td>4</td>
</tr>
<tr>
<td>Stalden 2</td>
<td>45</td>
<td>4</td>
</tr>
<tr>
<td>Zermeiggern 1</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Zermeiggern 2</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Saas Fee</td>
<td>1</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Table 9. Case study 2 reservoir data

<table>
<thead>
<tr>
<th>Reservoir</th>
<th>Maximum content [m$^3$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mattmark</td>
<td>100 000</td>
</tr>
<tr>
<td>Zermeiggern</td>
<td>1000</td>
</tr>
</tbody>
</table>
Appendix 2. Short-term hydropower planning using the genetic algorithm – basic concept and implementation

A2.1. Implementation of the solution

This proposed approach is a part of the Case study 2 and the model description found in Ch. 3.2. is valid for this case as well. As with the MILP approach presented in earlier chapters, the aim of this approach is to satisfy the demand, while trying to maximize the profit and the value of stored water in the reservoir.

The assumptions used in this model are as follows:

- Power plant owner acts as a price taker – has no market power
- Pumping and turbining at Zermeiggern can’t happen simultaneously
- Pump has to be on/off for a minimum of two hours in a row
- Pumping power is either 0 or 23 MW

Since the on/off status has to be implemented as a binary (integer) variable, for easier implementation, all the pumping and turbining values will be considered integer. The data from Table 8 and Table 9 are valid for this case as well.

The genetic algorithm was chosen for the short-term scheduling problem, since it is a relatively new approach and it would be interesting to compare the results to some, more often used, approaches like MILP. Also, since STHS is a nonconvex, nonlinear problem, the genetic algorithm should give high quality results.

The proposed optimization technique, could also be, with some modifications, used a part of a more complex, hydrothermal scheduling problem, as shown in [29].

Objective function

\[
\max \sum_{t \in T} \sum_{i \in I} p_i(t) * c_{p,t} + \gamma
\]  

(A2.1)

Where the first term represents the revenue obtained from power plant operation and the second term accounts for water value in the reservoir. The water value in this case is modeled differently from the MILP approach, because with the approach used in the MILP, the value of stored water would overpower the profit, which might lead to inaccuracy in genetic algorithm operation. This would happen since the reservoir is extremely big and assuming its full of water, the value of stored water would be several times bigger than the revenue from selling the electricity on the market.

In order to get values in the same range, the difference in water values between stages is accounted for. For example, if at stage \( t \) the level of reservoir is \( v_t \) and at stage \( t+1 \) the reservoir content is \( v_{t+1} \), where \( v_{t+1} < v_t \), the difference between the reservoir levels is accounted for.
In general, equations (6.17)-(6.23) are valid for this optimization as well. Since the lack of time didn’t allow for all the constraints to be implemented, only the ones that have been implemented are going to be mentioned below:

\[ \sum_{i \in G} x_i(t) \cdot p_i(t) - \sum_{i \in E} x_i(t) \cdot p_i(t) = D_t \quad (A2.2) \]

The load demand constraint (A2.2) has been satisfied, where \( x_i(t) \) is the on/off status of the turbine/pump. So has the equation (A2.3), which accounts for the reservoir level in different time stages.

\[ v_t = v_{t-1} + q_t + \sum_{i \in UP} (s_i(t) + u_i(t)) - u_i(t) - s_i(t) \mp b_t \quad (A2.3) \]
\[ x_{i, zero}(t) + x_{i, zero, pump}(t) \leq 1 \quad (A2.4) \]
\[ l_{min} \leq \sum_{i \in G} x_i(t) + \sum_{i \in E} x_i(t) \leq l_{max} \quad (A2.5) \]

Equations (A2.4) and (A2.5) represent the no turbining and pumping at the same time constraint and unit availability constraint respectively.

It should be noted that the technical minimum of power plants hasn’t been implemented, which means the plants can produce as low value as 1 MW. The pump constraint concerning pumping either 0 MW or 23 MW has been implemented.

The implementation of the genetic algorithm can be seen below.

**Representations of variables**

As mentioned, the control variables in this optimization problem are the turbine’s and pump’s on/off status, as well as the power output from each of the turbines and pumps. The on/off status of the variables is modeled as binary variable (integer with a minimum of 0 and maximum value of 1), where zero value represents the off status, while the 1 represents the on status. Power outputs are modeled as integer variables with bounds according to Table 8.

The initial population is created randomly. Results from MILP can be used in case a meaningful first population wants to be created.
**Fitness function**

To evaluate the quality of different solutions, the fitness evaluation of each population needs to be done. The iterative steps behind the fitness function evaluation are as follows:

1. For each turbine, the power output is assigned by the GA. With respect to head difference in the reservoir, the discharge in m$^3$/s is calculated for each turbine and pump.
2. The discharge, alongside with the inflows to the reservoirs, is used to calculate the new reservoir level. Potential spillage might occur.
3. Reservoir levels from stages $t$ and $t+1$ are compared and the difference is evaluated and assigned a value.
4. The revenue from selling the electricity on the market is added to the value of stored water.
5. The fitness function (objective function in this case) is evaluated and the best individuals are saved.

The built-on MATLAB function uses special crossover and mutation functions that force the variables to stay integers.

**Stopping criteria**

The maximum number of generations allowed is 300, but the optimization stops if the fitness function hasn’t improved in 150 generations. The number of generations is relatively high since nonlinear constraints are introduced and the genetic algorithm function takes approximately 50 generations to satisfy these constraints, after which it starts finding the optimal solution.
A2.2. Optimization results

The result from this optimization can be seen in Fig. 38. There are some similarities with the results obtained from the MILP approach. This is particularly valid for the fact that turbines Stalden 1 and Stalden 2 cover most of the load by taking advantage of high inflows to Zermeiggern reservoir.

In 7th, 9th, 15th, 19th, 20th, 21st and 22nd hour, the turbines are producing less than what would be their respective technical minimum and thus, the results can’t be considered completely accurate.

It should also be noted that the GA didn’t use the pumps at all, where MILP approach pumped for two hours during the night.

Figure 39. shows the content in Mattmark reservoir over the period of 24 hours. As seen, this approach also tries to maximize the reservoir content and the daily change in the reservoir is less than 1% of the content. Unlike MILP, the reservoir here isn’t full at the end of the optimization period, so a conclusion can be drawn that MILP was more effective in that perspective.
The content of Zermeiggern reservoir can be seen in Fig.40. As it can be seen, the reservoir is kept constant during the whole optimization period, except for the 12th hour, where it drops briefly. This is due to the fact that at that period, only Stalden turbines are active, thus turbining more water than the inflow in the hour. Unlike MILP approach, GA approach optimized the lower reservoir optimally.

All the calculations were done using the built-in genetic algorithm MATLAB function and the simulation time was 5 minutes. If comparing time, the MILP approach is much less time consuming and yields better result concerning optimal use of seasonal reservoir.
References


