Comparison of different optimization approaches on short-term dispatch of a thermal power plant unit

Dominik Putz
Technische Universität Wien, Energy Economics group
Vienna, Austria
dominik_putz@hotmail.com

Abstract: The expansion of renewable energy sources increases the volatility of energy generation. Of fundamental importance for security of supply is the balance between demand and generation at all times. The thermal power plant units which are used for this purpose are subject to restrictions that must be observed to ensure smooth and accident-free operation. The energy supply companies use software solutions developed for this purpose to generate optimal timetables for the electric power plant blocks for the electrical retrieval services. The focus of these applications is optimization. Due to increasing demands and complexity of the models, it is a growing challenge for software development to guarantee optimal timetables. The aim of this research is to explore the differences between two solving approaches namely Mixed-Integer Linear Programming and Backward Dynamic Programming. The focus of the analysis will be on the duration of the optimization, the storage requirements and the operability.

Keywords: Optimization, Portfoliomanagement, Power dispatch, Unit Commitment Model, Mixed Integer Linear Programming, Dynamic Programming,

1 Introduction

The progressive liberalization of the energy markets does not only yield new possibilities and chances. It also raises new risks. The long-term planning of business activities has become difficult due to volatile market prices for fuel procurement and electricity sales. Moreover, with the increasing development of renewable energy production the uncertainty in the generation from water, wind and solar resources rises continuously.

In energy supply, it is important that demand equals production. Since the demand for energy is basically fixed and generally predictable, production must be a variable. In the process, regenerative energy generators are treated preferentially and the production of thermal power plants is purposefully reduced in the event of a possible generation surplus. The thermal power plants are subject to technical constraints that must be met to ensure smooth and accident-free operation. From this situation, at certain times and environmental influences, the control and regulation of thermal power plants is a major challenge for the generation companies.

Portfolio optimization has the task of ensuring the optimal use of power plants for the respective market situation. The generation companies use developed and adapted software solutions with a focus on optimization. Due to the increasing demands and complexity, it is a growing challenge for software development to offer optimal solutions.

Section 2 describes MILP and DP in general. Section 3 explains the power plant to be optimized with the parameters to be met. It mentions Unit Commitment Model on which the optimization is based. Section 4 explains the realization and modeling of the two approaches in detail. The results are then summarized in Section 5 and a conclusion in Section 6 is made.

2 State of the Art

2.1 Mixed-Integer Linear Programming (MILP)

The determination of an optimal roadmap for the electrical demand performance of one or more power plant blocks is based on mixed-integer linear programming. As standard in the energy industry, a 7-day forecast is made in a 15-minute time grid using mixed-integer linear programming. Depending on the data situation, the calculation can take seconds to several hours. This aspect should be considered in all further investigations. Often simplifications and linearization are made to make the processing faster. Long calculation times arise for periods of more than 28 days or better time granularity of 15 minutes for example.

2.2 Backward Dynamic Programming (DP)

A power plant deployment park with many different generators that have large volatilities often results in relatively long computation times. Therefore, this leads to an alternative solution method. The method of dynamic programming has been studied closely, as this represents a promising solution method. This method has generally been known for a long time.

Dynamic programming can be implemented in several ways. The ever-increasing level of complexity of the models to be solved raises the question of whether alternative programming languages are more suitable for implementing dynamic programming. Julia is chosen as the language to study because of her great potential in terms of performance in solving optimization problems compared to other programming languages [1] [2].

3 Model Description

This section is devoted to the description of the developed model. This includes general properties that describe the model and which framework conditions or restrictions it must fulfill. The mathematical description is limited to basic definitions only.

3.1 Thermal Power Plant Unit

The thermal power plant unit to be modeled is a gas and steam combined cycle power plant (CCGT) Due to its technical nature, a combined cycle power plant can be used very flexibly and can be excellently used to cover both medium and peak load [3]. The plant block is a so-called price taker, so its operating condition does not affect the price of electricity unlike that of a market maker, which is of fundamental importance as the market price can be considered independent. The data for the electricity price will be sourced from the Austrian energy exchange EXAA (Energy Exchange Austria), which is a marketplace for energy trading.

3.2 Unit Commitment Model

It is clear from the above-described aspects that the control of thermal power plants must be very flexible at certain times and that this must be carried out by corresponding software applications to ensure an optimal timetable. Due to the lack of storage capacity or technology, production must always be equal to consumption. One model that has proven itself as a description is the so-called unit commitment model. The Unit Commitment Model is used to calculate power plant on-demand performance, especially in the day-ahead schedule. It is used by generation companies and has the task of creating efficient schedules for fossil and renewable power plants under various technical and economic constraints. The focus of the unit commitment model is to minimize the costs of production or to maximize the output of production [4], subject to special restrictions.

The framework conditions to which the unit commitment model is subject include large, complex problems caused by different types of power plants, a high number of energy producers and geographical factors based on existing network infrastructure. Certainly, every unit has different technical and commercial qualities which further increase the complexity of the problem. In addition, the unit commitment model includes other elements, such as forecast uncertainties specific to wind energy and photovoltaics, reserves that need to be available, and policy drivers, such as the EU ETS (European Union Emissions Trading System), which determine the duration of the optimization. From this it can be deduced that the solution of the unit commitment model can take a relatively long time. For generation companies it is important that the optimization takes up as little time as is necessary. To minimize the duration of the optimization, simplifications are made in the model, which significantly reduce the complexity.

3.3 Technical and economical Parameters

The time step interval corresponds to a resolution of 15 minutes. The selected fine granularity results in a relatively high number of steps, which corresponds to 672 steps per week and 35 040 steps per year. The large number of time steps increases the complexity of the problem, which has a direct impact on the calculation time.

The technical restrictions for the power plant unit can be taken from Table 1.

Table 1 Technical restrictions of the power plant unit.

Parameter	Dimension	Value
power max	MW	420
power min	MW	250
min. down time	h	4
min. run time	h	6
max. power jump between two steps	MW/15min	45
coefficient of production a	MW th.	194,4
coefficient of production b	MW th./MW	1,027
coefficient of production c	MW th./MW²	0,000782
down time for hot start	h	4
down time for warm start	h	12
down time for cold start	h	48

The installed net power must be regarded as variable for each time step. In standard mode, this standard has the given value. For certain situations during operation, it may be necessary to set the maximum technically allowed power lower than the installed net power. This is to be considered in the model at best. The minimum downtime describes the time interval that must be met after the power plant block is shut down before it can be restarted. Likewise, the minimum operating time determines that time interval that the power plant block must comply after reaching the minimum technical capacity before it may be shut down again. The maximum power change gradient specifies the maximum allowable power jump between two consecutive time intervals. The maximum power change gradient must be adhered to under all circumstances.

The efficiency of the power plant block is not constant and depends on the electrical demand performance. Figure 1 shows that the relationship between electrical demand performance and thermal fuel thermal power is approximately linear.

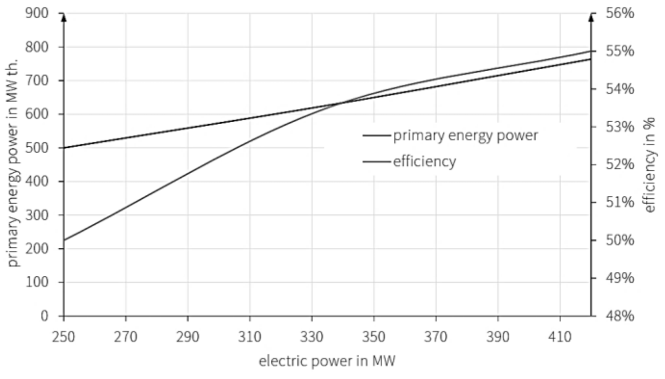

Figure 1 Thermal Power and efficiency in relation to electrical power.

By linear approximation [5], the quadratic inverse production function can thus be simplified to a linear function as seen in Eq. (1).

$$P_{thermisch} = a + b \cdot P + c \cdot P^2 \cong a_{lin} \cdot b_{lin} \cdot P \qquad (1)$$

The data source for the electricity price for 2017 will be the values from the EXAA 15-minute product. These are averaged for each hour to reduce volatility within an hour. Additional economic parameters can be seen in Table 2.

Table 2 Economic parameters.

Parameter	Dimension	Value
CO_2 emission factor	t/MWh th.	0,2
cost for hot start	EUR	30000
cost for warm start	EUR	40000
cost for cold start	EUR	50000
operation costs	EUR/h	200
market price	EUR/MWh	variabel
fuel price	EUR/MWh	variabel
CO_2 price	EUR/t	variabel
additional costs	EUR/MWh	variabel

The cost function which is showed in Eq. (2) consists of fuel costs, operation and maintenance costs, additional costs and start up costs.

$$C_t = p_{fuel,t} \cdot P_{thermisch,t} + p_{O\&M,t} + p_{additional,t} \cdot P + C_{Start,t} \qquad (2)$$

A power plant must be powered up from a standstill with a predetermined ramp to get online or to reach the minimum technical capacity which is shown in Figure 2. Once the starting process of the power plant has been completed, the power may be varied within the technically permissible limits over the maximum power change gradient.

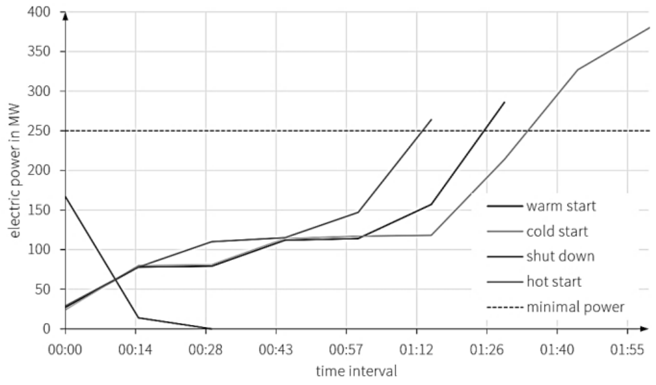

Figure 2 Ramping progress of different starting and shutdown types.

3.4 Objective Function

The goal of the model is to maximize the yield of Eq. (3). The yield corresponds to the sum of the contribution margin at each time step. The contribution margin consists of the difference between the profit from the electricity marketing, the purchase of fuel and other costs.

$$\Pi = \sum_{t=1}^{T} p_{market,t} \cdot P_t - C_t \qquad (3)$$

4 Methodology

The following section is dedicated to the analytical method for implementing the software application.

4.1 Mixed Integer Linear Programming (MILP)

Mixed-integer linear optimization is a practical modeling solution for problems for which there are no explicit algorithms. Basically, it consists of an objective function to be optimized and a set number of equations and inequalities that limit the mathematical problem. The solution must satisfy all equations and inequalities to exist as a valid solution, otherwise the objective function cannot be optimized or there is only a trivial zero solution. Strictly speaking, the variable to be optimized is the so-called decision variable, the solution space is the multiplicity of permissible solutions, and the objective function assigns a value to each solution. The solution of mixed-integer linear problems is usually carried out in practice by so-called solvers, such as GUROBI [6], XPRESS or CPLEX. Undoubtedly, identifying an optimal solution is often a difficult task and takes a significant amount of time, depending on size and complexity.

In the realization of the model with MatLab the solver GUROBI is used. It uses a combination of branch-and-bound, presolving, cut-plane methods, heuristics and. As an objective, GUROBI understands the objective function, which must be minimized. The constraints, called constraints, must be followed by the model. The shape of the Mixed Integer Programming Problem can be determined by the standard form using Eq. (4) - Eq. (6).

$$\min_{x_1, x_2, \dots} f(x_1, x_2, \dots) \tag{4}$$

$$g_i(x_1, x_2, \dots) \leq b_i \quad i = 1, \dots, p \tag{5}$$

$$h_j(x_1, x_2, \dots) = 0 \quad j = 1, \dots, q \tag{6}$$

The calculation is performed on an Intel Core i5-4690K (4 CPUs, 3.50 GHz), Windows 10 Enterprise 64-bit and 8 GB memory.

Maximizing the yield as a target function is given by Eq. (7).

$$\Pi = \max_{P_n} \sum_{n=1}^{N} \Bigg[\sum_{k=1}^{K} (p_k \cdot P_{n,k} \cdot \Delta T) - \big[p_{n,k,fuel} \cdot \big(a_n \cdot u_{n,k} + b_n \cdot P_{n,k} + c_n \cdot P_{n,k}^2\big) + p_{n,misc} \cdot P_{n,k} + \big(p_{n,O\&M} \cdot u_{n,k}\big)\big] \cdot \Delta T - \big(C_{n,cold}^{SU} \cdot z_{n,k,cold} + C_{n,warm}^{SU} \cdot z_{n,k,warm} + C_{n,hoz}^{SU} \cdot z_{n,k,hot}\big) \Bigg] \tag{7}$$

Table 3 contains the description of the symbols used and decision variables of the objective function.

Table 3 Variable description for the objective function.

Symbol	Dimension	Description
n	p.u.	unit $n \dots N$
k	p.u.	time step $k \dots K$
p_k	EUR/MWh	market price
$P_{n,k}$	MW	scheduled power
T	h	time grid
$p_{n,k,fuel}$	EUR /MWh	fuel price
$p_{n,misc}$	EUR/h	miscellaneous costs
$p_{n,O\&M}$	EUR/MWh	operation costs
$u_{n,k}$	1	on/off status
$z_{n,k,\dots}$	1	type of start (hot/warm/cold)
$C_{n,\dots}^{SU}$	EUR	starting costs

The objective function consists of the profit generated by marketing the electricity price less the loss caused by the purchase of fuel, start-up costs, operating costs and additional costs.

4.2 Backward Dynamic Programming (DP)

Dynamic programming can be used to solve optimization problems when this problem can be broken down into subproblems [7]. The so-called optimality principle of Bellman [8] describes the connection that the optimal solution of the problem consists of the optimal solutions of the sub-problems. The sum of the local optima corresponds to the global optimum. The sub-problems are easier to solve or optimize and can thus be used as the optimal solution to the overall problem. Furthermore, once calculated solutions of sub-problems are stored and for similar sub-problems, the previously calculated intermediate solution is used instead of being recalculated. This method, on which the model is based, is called memoization. This has a direct effect on the calculation time of the optimization. Dynamic programming uses a bottom-up solution strategy. Thanks to the many partial solutions, you can get to the solution of the overall problem faster because they are easier and faster to solve. The CPU is relieved thereby. On the other hand, the main memory is loaded more, because in return the partial solutions must be stored. In the implemented model in Julia, which uses backward dynamic programming, the memory is relieved by an extension of the algorithm. The reason for this is that not every part problem is stored. Once small part problems have been resolved into a larger part problem, only the result of the larger part problem is saved, as shown in Figure 3. The previously calculated results are discarded because their information is redundant. As a result, the memory required for larger problems is reduced enormously.

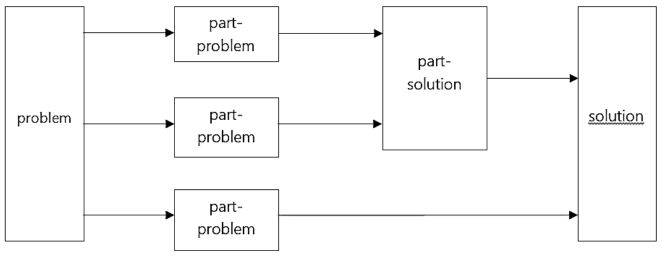

Figure 3 Solving approach of dynamic programming.

In principle, the basic process can be broken down into four steps:

1. Characterization of the structure of an optimal solution
2. Recursive definition of the value of an optimal solution
3. Calculation of the value of an optimal solution (recursive)
4. Construction of the optimal solution of calculated information.

The simulation runs on a system with Intel Core i5-6500 (4 CPUs, 3.20 GHz), Windows 10 Enterprise 64-bit, and 16 GB of memory.

Backward dynamic programming describes the direction in which the optimization problem first goes through. It is started at time T and iterates "backwards" until time T_0. After reaching the starting point T_0, the sub-problems are resolved to "forward" until the time T is reached. Mathematically, this approach can be explained by Eq. (8).

$$\Pi(T) = \max[g(t) + \Pi(T-t)] \quad t = 1, \dots, T \qquad (8)$$

In principle, one can think of the solution method as a directed graph from the network theory. Each node has a certain number of neighbors. There is a so-called adjacency list for each node. The adjacency list contains the information for the transition from a temporally following node. A transition may be allowed or not allowed. It is thus spanned a node array in which only certain transitions are allowed. Figure 4 illustrates the structure and operation of the directed graph.

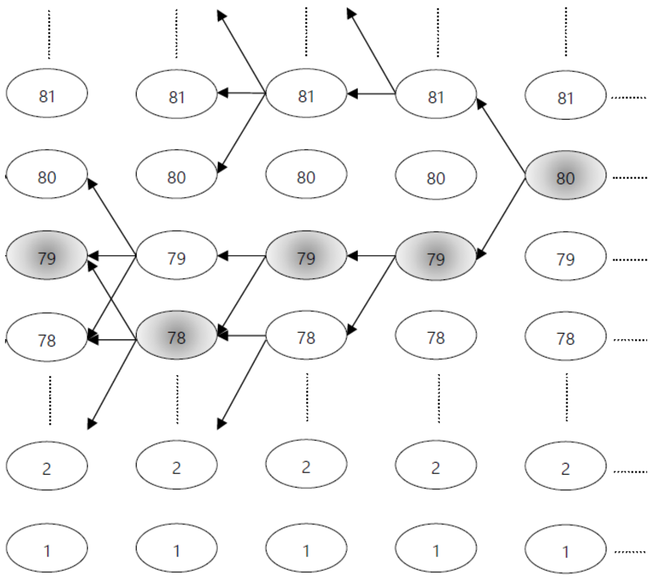

Figure 4 The directed graph: All nodes correspond to a particular state. Allowed and not allowed transitions are calculated. The time axis lies in horizontal level and the states are referred to the vertical axis.

The goal of backward dynamic programming is to find one or more solution paths with the optimal solution. It is clear that the number of possible paths increases rapidly with increasing time steps. Due to a high number of allowed neighbors from one node, there are many possible paths. From this one can conclude that the number of possible transitions from one node to the next is to be kept as small as possible in order to keep the calculation time low.

An extension can reduce the number of paths. The reverse dynamic programming is extended by a modified priority list known from integer-mixed linear programming. In general, this extension can be understood in such a way that, when determining the adjacency list at runtime, it is also possible to determine directly which transitions are most-likely. This results in priorities which transitions are most likely to lead to the optimal solution path. The big advantage is that many transitions that are possible, but not optimal, are calculated only when all previous possible transitions do not provide a clear solution. The introduced modified priority list significantly shortens the calculation time of the optimization, since many paths that are suboptimal are not calculated.

The algorithm works through the following steps:

1. At the beginning of the program, all relevant data is read. The data are suitably prepared for further processing.
2. The recursive function GetOptimalSolution(...) determines the optimal solution path based on reverse dynamic programming.
3. GetPathStates(...) ¨ checks whether the solution path found provides a permissible and permitted solution and whether the roadmap can actually be implemented in reality.
4. GetPowerDispatch(...) provides the relevant roadmap for the power plant block based on the solution path.
5. GetPathProfit(...) provides the optimal profit.

Figure 5 illustrates the schematic flow of the program.

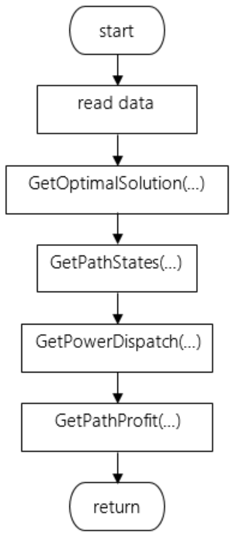

Figure 5 Schematic flow of the application.

5 Results

The following chapter presents the results of the previously described solution methods. The focus is on the calculated optimal schedules, which are obtained as a. Another essential factor is the assessment of performance. Attention is paid to the relationship between the number of steps and the duration of the calculation.

Figure 6 shows the timetable with the relevant boundary data.

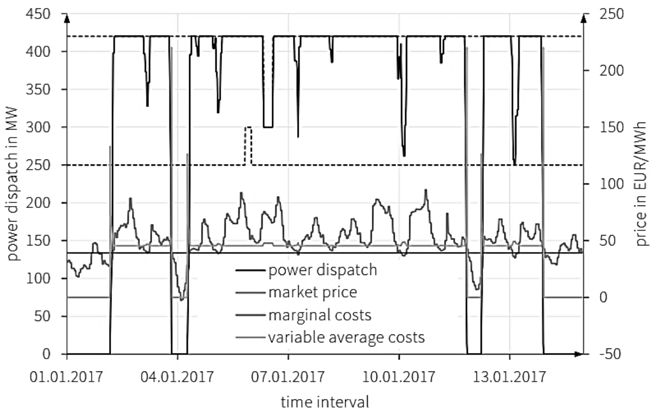

Figure 6 Power schedule as solution of the optimization.

5.1 Mixed Integer Linear Programming (MILP) – Calculation time

The calculation time for optimization by the mixed-integer linear method can vary widely. A major influence on the calculation period is the so-called maximum permitted relative gap. The gap describes the percentage difference between the upper and lower bounds of optimization due to the branch-and-bound approximation method. Since it is an approximation method, the result is closer to the actual optimum, the smaller the relative gap. As a rule, a relative gap of 0.1 % is used. For comparison purposes and to keep the calculation time low, the relative gap is set at 2 %. This results in the calculation periods for the respective number of steps shown in Table 4.

Table 4 Calculation duration of the optimization by using MILP.

Time interval	steps	duration in sec.
1 day	96	4,666
1 week	672	43,932
2 weeks	1 344	254,4
3 weeks	2 016	257
1 month	2 688	350
5 weeks	3 360	557
6 weeks	4 032	988
7 weeks	4 704	1 027
2 months	5 376	1 122
3 months	8 064	2 581

Figure 7 shows the relationship between the number of steps and the duration of the calculation. The trend line illustrates the non-linear relationship, which is a well-known fact in mixed-integer linear programming.

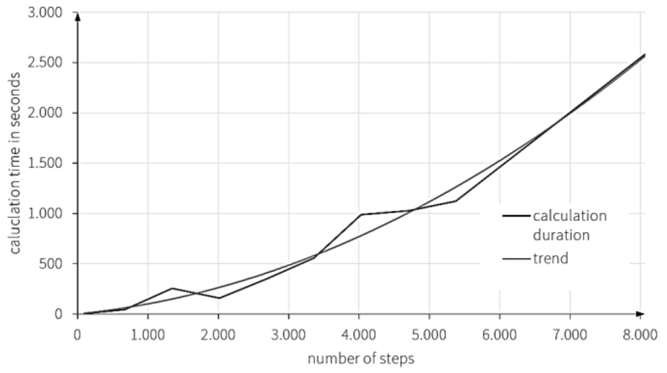

Figure 7 Non-linear relationship between number of steps and calculation duration by using MILP.

5.2 Backward Dynamic Programming (DP) – Calculation time and main memory

The calculation duration of the model with backward dynamic programming always behaves linearly depending on the number of steps. The data situation has no influence on the calculation time of the optimization problem. Furthermore, there is no duality gap, as is the case in mixed-integer linear programming. Table 5 shows all important results of backward dynamic programming.

Table 5 Calculation duration of the optimization by using DP.

Time interval	steps	duration in sec.	memory in MB
1 day	96	0,18	13
1 week	672	0,25	27
2 weeks	1 344	0,29	45
3 weeks	2 016	0,30	49
1 month	2 688	0,35	61
5 weeks	3 360	0,40	83
6 weeks	4 032	0,44	101
7 weeks	4 704	0,48	118
2 months	5 376	0,59	160
3 months	8 064	0,86	284
4 months	10 752	1,14	422
6 months	16 128	1,67	664
8 months	21 504	2,15	850
9 months	24 192	2,33	953
1 year	35 040	3,72	1 515

Dynamic programming has a linear relationship between computation time and number of steps. Figure 8 illustrates this behavior.

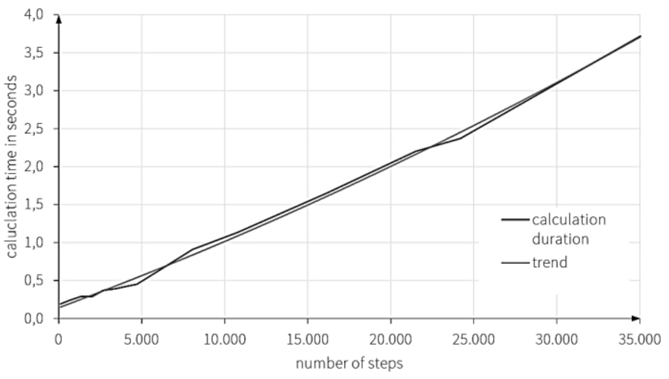

Figure 8 Linear relationship between number of steps and calculation duration by using DP.

Another aspect of interest in studying the results is the amount of memory needed for optimization. There are no measurement results for the mixed-integer linear programming. Figure 9 shows the linear relationship between memory requirements and number of steps for backward dynamic programming.

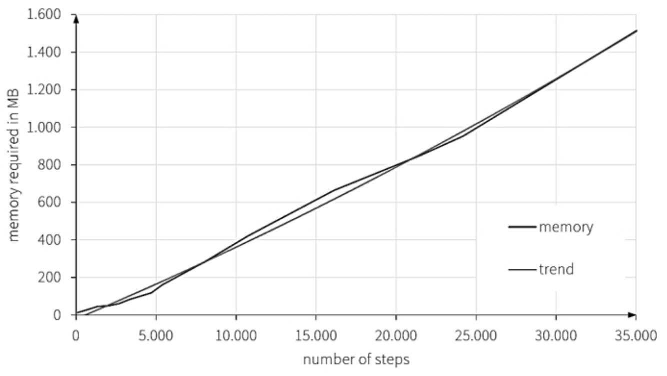

Figure 9 Linear relationship between number of steps and memory needed by using DP.

5.3 Deductions

There is no doubt that backward dynamic programming has a significant advantage in computation time over mixed-integer linear programming. Especially with increasing number of steps, the time difference is significantly greater. In addition, the calculation time for mixed-integer linear programming scales disproportionately with increasing number of steps. As a result, the size of the problem must be kept small or the granularity of time as large as possible.

Table 6 summarizes the measured calculation durations.

Table 6 Comparison of MILP and DP in terms of calculation duration.

Time interval	steps	MILP	DP
		in seconds	
1 day	96	4,67	0,18
1 week	672	43,93	0,25
2 weeks	1 344	254,4	0,29
1 month	2 688	350	0,35
3 months	8 064	2 581	0,86
1 year	35 040	n.v.	3,72

The data situation has a significant influence on the duration of the calculation. During this work, it was observed that, depending on the market price, the calculation period can vary considerably. In times of extreme market situations, caused for example by very low or very high market prices, the model can be solved relatively quickly. The reason is that it is clear for the algorithm that the optimal schedule dictates either standstill or maximum technical performance. In times when the market price is in the range of marginal costs, the optimization takes more time to decide which power step would be the best.

6 Conclusio

In this chapter a final statement of the work is made and subsequently dealt with topics that are of interest in the further course of the research, but which could not be discussed in the context of the research.

The developed application always tries to calculate the best schedule. However, due to various influences, it may happen that the predetermined power value cannot be driven. The only information that is classified as relevant for a ramp process for optimization, for example, is the previously-initiated downtime. The algorithm has no detailed information about the current availability of the power plant block.

As the work has shown, prefer a recursive solution using backward dynamic programming. In the work, it has been proven that when calculating schedules for longer periods of finer time granularity, backward dynamic programming of mixed-integer linear programming is clearly superior. On the other hand, however, it must be considered that the development of a model that uses the dynamic programming method turns out to be much more complex than a model that works with a solver. The big advantage of using a solver like GUROBI lies in the simplicity of the model implementation. The decision of which solution method to use depends on the type of application and the problem. In the case of the present problem, a dynamic method with a minimum computation time is to be preferred, since the optimization is called very frequently. In addition, their result is time-critical, since it is used for decisions in the real market and it is important that an optimization process has the shortest possible calculation time.

As the results have shown, the linearization of the quadratic inverse production function brings a relatively large time saving. The reason for this lies in the reduction of the quadratic optimization problem to a linear one. GUROBI can solve a quadratic problem. For an exact solution, a quadratic formulation would be necessary. Here it should be asked whether such a mathematically more elaborate formulation provides far better results in terms of accuracy, although the circumstance of the calculation duration plays a role.

Although the limitations on the start and end points have very little effect on the result, they are further simplifications that adversely affect the accuracy of the result.

One question that could not be resolved by this research is the inclusion of storage. Storages often come in resource planning, which were not included in the course of this work. It is undisputed that the implementation of memory increases the level of complexity even further. Thus, the integration of memories is a worthwhile task for future investigations.

When developing the algorithm, special cases were detected that could provide further performance enhancements through intelligent enhancements. For example, the priority list that estimates the optimal predecessor could still be optimized. It would be fundamentally possible to estimate not just the next predecessor, but several, based on one step, so that complete path elements can be processed faster. In a market situation of very high or very low market prices, the preferred electrical retrieval performance would already be known and thus several time steps could be skipped. Another interesting aspect is the introduction of neural networks. For certain sections of the program, substitution by neural networks would be fundamental. To be able to answer this question unambiguously, further investigations are needed.

7 References

[1] S. B. Aruoba and J. Fernandez-Villaverde, “sas,” [Online]. Available: https://www.sas.upenn.edu/ jesusfv/comparison languages.pdf. [Accessed 22 November 2018].

[2] “Julia The Programming Language,” [Online]. Available: https://julialang.org. [Accessed 22 November 2018].

[3] G. W., Energieversorgung, TU Wien, 2015.

[4] D. Putz and M. Gumhalter, “Different approaches on power-based Unit Commitment formulation,” Wien, 2018.

[5] W. E., “Taylor Series,” [Online]. Available: http://mathworld.wolfram.com/TaylorSeries.html/. [Accessed 13 Dezember 2018].

[6] GUROBI, “GUROBI,” [Online]. Available: http://www.gurobi.com/resources/gettingstarted/mipbasics. [Accessed 13 Dezember 2018].

[7] A. Martin, “Effiziente Algorithmen,” [Online]. Available: https://www.tuilmenau.de/fileadmin/public/iti/Lehre/Eff.Alg/SS12/EA-SS12-Kapitel5.pdf/. [Accessed 3 Oktober 2018].

[8] A. H., “Energiemodelle und Analysen - Lineare Optimierung,” 2017.