Diese Arbeit wurde vorgelegt am
Lehr- und Forschungsgebiet Theorie der hybriden Systeme

Modellbasierte Optimierung der Gebäudehülle zur Steigerung der Energieeffizienz

Model-Based Optimization of the Building Envelope for Energy Efficiency

Bachelorarbeit
Informatik

September 2025

Vorgelegt von Presented by	Till Determann Matrikelnummer: 444940 till.determann@rwth-aachen.de
Erstprüfer First examiner	Prof. Dr. Dr. h.c. Erika Ábrahám Lehr- und Forschungsgebiet: Theorie der hybriden Systeme RWTH Aachen University
Zweitprüfer Second examiner	Prof. Dr. rer. nat. Jürgen Giesl Lehr- und Forschungsgebiet: Programmiersprachen und Verifikation RWTH Aachen University
Betreuer Supervisor	Dr. rer. nat. Pascal Richter Lehr- und Forschungsgebiet: Theorie der hybriden Systeme RWTH Aachen University

Contents

1. Introduction 1
1.1. Motivation . . . 1
1.2. Related Work . . . 1
1.3. Scope and Limitations . . . 4
1.4. Contribution . . . 4
1.5. Outline . . . 5

2. Thermal Model of Buildings 5
2.1. Model Implementation and Modifications . . . 6
2.2. Performance . . . 6
2.2.1. Quality . . . 6
2.2.2. Time to Energy Efficiency Class . . . 7

3. Objective and Constraints: Model and Analysis 8
3.1. Model of Construction Measures . . . 9
3.1.1. Wall Measures . . . 10
3.1.2. Roof Measures . . . 12
3.1.3. Basement Ceiling/Base Plate Measures . . . 14
3.1.4. Window Measures . . . 15
3.1.5. Door Measures . . . 16
3.1.6. Cost Adjustment to 2024 . . . 17
3.2. Integration of Measures into the Thermal Model . . . 19
3.2.1. Examination of Additivity . . . 20
3.2.2. Sensitivity Analysis . . . 23

4. Optimization Methodology 26
4.1. Problem Formulation . . . 26
4.1.1. Complexity and NP-Hardness . . . 28
4.2. Exhaustive Search . . . 32
4.3. Branch-and-Bound with Additive Relaxation . . . 32
4.3.1. Fundamentals of Branch-and-Bound . . . 34
4.3.2. Problem-Specific Search Strategy and Pruning Rules . . . 35
4.3.3. Validation-Driven Branching . . . 37
4.3.4. Iterative Evaluation . . . 38
4.4. Genetic Optimization Algorithm . . . 41

5. Benchmarking and Evaluation 42
5.1. Benchmark Setup . . . 42

5.2. Evaluation Metrics 44
5.2.1. Number of Model Evaluations 44
5.2.2. Solution Quality 44
5.3. Results 45
5.3.1. Number of Model Evaluations per Algorithm 45
5.3.2. Solution Quality per Algorithm 47

6. Conclusion and Outlook **51**
6.1. Future Work 52

A. Abbreviations of Insulation Materials **53**

References **54**

1. Introduction

1.1. Motivation

Due to climate change, improving the energy efficiency of buildings has become increasingly important in reducing greenhouse gas emissions. Energy efficiency projects are subsidized in Germany, and the more efficiency gained in a project, the more fundings it receives. The fundings, along with the possibility of lowering heating and cooling expenses, raising the property's resale value, and promoting environmentally friendly living, give building owners strong incentives to initiate an energy-efficient renovation.

Such a renovation includes energy-efficient construction measures and has a maximum target energy demand that must be met. Considering economic factors, the cost of the measures should be as low as possible.

There are numerous software tools available on the German market for evaluating a building's energy efficiency. The calculation methodology is regulated by national standards, with DIN V 18599 serving as the legally recognized framework for evaluating building energy performance. However, none of the existing tools provides optimization approaches. As a result, the selection of appropriate construction measures must be performed manually by the user. This can be achieved through professional consultation, which incurs additional costs, or by independently researching the market. Because of the wide variety of different construction measures that can be applied, each with varying costs and potential energy savings, it is very difficult to choose a combination that meets the required target efficiency and is also economically convenient. In order to meet the predetermined energy target, an algorithmic optimization approach is necessary that uses the thermal model provided by DIN V 18599 and suggests a set of cost-effective measures.

This work shows how the above requirements can be translated into a model that maintains realistic costs and benefits while ensuring generalizability and reasonable computation times, even under high problem complexity. Various approaches will be compared and benchmarked in terms of quality and runtime.

1.2. Related Work

The thermal model used for optimization is based on the framework for evaluating the energy efficiency of buildings described in DIN V 18599 [10]. The method can be exerted to non-residential buildings and complex residential buildings. As an alternative for evaluating simple residential buildings, DIN V 4108-6 [8] in combination with DIN V 4701-10 [9] can also be applied. However, Himburg [19] showed that DIN V 18599 provides a much more precise representation of the building services engineering and thus also more realistic values. Owing to the more complex database and calculations, it was also shown that results are more prone to errors, and the calculations are time-consuming.

The implementation of the thermal model in this work is based on the *EnerEval*-algorithm presented by Feiereisen [14]. The author describes the algorithm as compliant with DIN V 18599 and validates this statement with various test cases and runtime analysis. Consequently, it is possible after all for an implementation to be error-robust and yet runtime-efficient.

Such implementations of the DIN V 18599 procedure are common, and a variety of software products provide such functionality. The *18599 Gütegemeinschaft*[1] brings together software products that implement DIN V 18599 and meet a certain quality standard. For this purpose, the organization provides test cases that define reference buildings with intermediate and final results for energy demand. These example projects must be included as test cases in the software so that users can validate the results. However, the complete test cases are not publicly available, which means that only members of the organization can fully validate their products. For example, the test cases for the 2018 version of the standard are not freely accessible. Ricken [33] shows in his work how test cases for 2018 can be generated. This work utilizes the test cases generated in this manner.

The optimization approach, on the other hand, is not adequately represented. Some of the tools included in the *18599 Gütegemeinschaft* also take economic considerations into account, see Table 1. As a rule, an individual renovation roadmap (*individueller Sanierungsfahrplan*, iSFP) workflow is used, in which the user enters several measures and then groups them together into variants. The savings

Tool	**iSFP**	**Cost Calculation**	**Optimization**
EVEBI (Envisys)	✓	✓	X
ZUB Helena	✓	✓	X
Solar-Computer	X	✓ (VDI 2067)	X
Hottgenroth	✓	/	X
BKI Energieplaner	✓	✓ (BKI data)	X
ROWA-Soft	✓	/	X
DÄMMWERK (Kern)	✓	/	X
GEG-PRO (Visionworld)	✓	/	X

Table 1: Overview of software tools certified by the *18599 Gütegemeinschaft*, indicating their support for iSFP workflows, cost calculation, and optimization.

and costs are then calculated for these variants. Typically, entering critical values

[1] *18599 Gütegemeinschaft* – `https://www.18599siegel.de/`

(e.g., U-values, g-values, thicknesses) and defining the corresponding groups is left to the user, which can be both time-consuming and error-prone when managing a large set of measures.

Since climate change is a global phenomenon, Germany is not the only country interested in evaluating and improving the energy efficiency of buildings. Looking at the international landscape, EnergyPlus is one of the most widely used building energy simulation engines, developed by the *U.S. Department of Energy*[2] in collaboration with several research institutions [6]. EnergyPlus uses a heat balance model to calculate the building thermal zones. For more diverse application cases, this model is linked to other modules such as COMIS and WINDOW 5, which provide calculations for specific housing components. It is designed for dynamic simulation at minute-hour intervals and generates highly detailed simulation output, and hence is frequently used as a core model in energy-efficiency-related topics for buildings, both in theory and in practice. Even though the use case appears to be the same, EnergyPlus and DIN V 18599 differ fundamentally in the manner of calculation. EnergyPlus requires more modeling work and calculation time to deliver high-resolution simulations. DIN V 18599 is more efficient to calculate, but is less detailed and meets conservative assumptions.

An example of optimization software that uses EnergyPlus as its core model is BEopt, elaborated by Christensen et al. [5]. This tool was developed by the *National Renewable Energy Laboratory*[3] to minimize the total cost of a renovation while maximizing energy savings. The main focus of the application is on optimization towards residential zero net energy (ZNE) buildings. These buildings are defined as producing at least as much energy from renewable sources as they consume. Consequently, they are emission-free in use and independent of the utility grid. For optimization, a path based on the current input, towards ZNE is created. BEopt searches along this path for the optimal building design and the most cost-effective measures to achieve it. At each step, EnergyPlus is used to calculate and evaluate the energy demand. This sequential search approach not only finds the best solution but also identifies near-optimal solutions. Such a multi-stage simulation via EnergyPlus appears to be prone to errors. The procedure is more efficient than exhaustive search, but it still can take several hours of computation time.

For this reason, there are alternative approaches for optimizing building energy in the literature. Especially the rapid advancements in artificial intelligence (AI) and computational intelligence in recent years, have also influenced the field of building energy optimization. Within this context, metaheuristic algorithms

[2] *U.S. Department of Energy* – `https://www.energy.gov/`

[3] *National Renewable Energy Laboratory* – `https://www.energy.gov/eere/buildings/building-energy-optimization-beopt-software`

(MAs) such as genetic algorithms, particle swarm optimization, and ant colony optimization are often considered part of the broader AI landscape due to their bio-inspired, adaptive, and stochastic nature.

Karbasforoushha et al. [23] benchmarked MAs performance applied in the building energy optimization sector. These algorithms have been applied to optimize construction measures for reducing energy demand, integrating renewable energy systems, and improving occupant comfort. All algorithms were based on the EnergyPlus model. However, while metaheuristic approaches demonstrate strong exploratory capabilities in large and complex search spaces, the study shows that they typically involve higher computational costs and do not always guarantee repeatable results due to their stochastic nature. In contrast, the optimization techniques described in this work achieve comparable performance for the problem setup under consideration while offering deterministic solutions with noticeably shorter computation times. To ensure a wider variety of algorithms, a genetic algorithm was nevertheless included in the benchmark for comparison.

1.3. Scope and Limitations

The scope of this thesis is subject to several limitations.

In practice a distinction is made between residential and non-residential buildings. This work refers only to residential buildings. Nevertheless, these are only the limitations of the thermal model and not the entire optimization.

Structural measures can be carried out on various components of a building. This work only considers measures that address the thermal envelope of buildings. This term covers components such as walls, roofs, doors, and windows.

1.4. Contribution

This work contributes to the optimization aspect of assessing the energy efficiency of buildings.

As shown above, software tools compliant with DIN V 18599 already exist on the German market, which not only evaluate energy efficiency but also offer options for planning energy-efficient renovations. Some of these tools also provide integrated cost and efficiency estimation. As a consequence, the user must select suitable measures themselves and transfer them to the program. They can then find a suitable solution set either through “keen observation” or by trying out several variants. If there are more than a few measures, it quickly becomes apparent that this method does not provide the optimal solution in most cases. Even selecting and transferring certain measures causes a bias in the solution.

The algorithm presented in this work bridges the gap between the thermal model, compliant with DIN V 18599, the cost/benefit modeling of measures, and the actual optimization. Currently, there is no official tool available to the user that can perform the subsequent optimization.

The work also contributes to the broader field of optimization by formulating two new NP-hard optimization problems. The complexity of these problems necessitates benchmarking of the proposed algorithms. This allows the optimization methods to be classified and evaluated in the context of energy efficiency.

1.5. Outline

The work is divided into five main sections. Section 2 deals with the thermal model presented by DIN V 18599 and algorithmically captured as the *EnerEval*-algorithm. A conceptual explanation and benchmark of performance, especially in comparison to older implementations, are provided. The following sections focus on various aspects of the optimization. The model of the construction measure is presented in Section 3, which provides properties such as objective values and constraint values. The integration of this model into the thermal model is also explained and various side effects are highlighted, such as additivity and sensitivity to certain input variables. Section 4 defines the optimization problem and classifies it according to its complexity. Possible solution algorithms are explained conceptually and their differences are highlighted. Section 5 evaluates the presented algorithms in terms of solution quality and calculation speed. The final section 6 highlights the main findings of this work and suggests possible directions for future research.

2. Thermal Model of Buildings

In Germany, the *Energieausweis* (energy performance certificate) is used to describe the energy efficiency of buildings, and, according to § 108 of the *Gebäudeenergiegesetz* (German Building Energy Act), must be presented when a building is newly constructed, sold, or re-let. The primary energy demand denotes the central value of the *Energieausweis*, and is used to determine the energy efficiency class of the building. To prevent invalid calculations, the DIN V 18599 provides a legally recognized framework for calculating this value [10].

The *EnerEval*-algorithm presented by Feiereisen [14] represents such a model based on DIN V 18599. Ricken [33] provides an extension for the current versions of the DIN V 18599 from 2018. The following section presents a new implementation of the model, in which performance is improved and errors are reduced. This results in higher accuracy and faster runtime.

2.1. Model Implementation and Modifications

The thermal model calculates the primary energy demand based on many different input parameters that together describe a building. Although the conceptual model does not differ from the one presented in [33], there are major differences in the implementation. The implementation used here is written in Swift rather than in Python and, unlike in [33], does not use a classic class structure but is based on structs with computed properties. Instead of storing values, computing properties recalculate them when called. This enables a single source of truth structure where the respective calculation of the individual variables is encapsulated. The clear advantage is that if the basic model changes, for example, as a result of a measure being implemented, there is no need to implement a complex update logic to update the values stored in the classes that are outdated. The decisive disadvantage is that multiple uses of a property imply necessary recalculations. To guarantee faster runtime, five central values are stored as a compromise. This requires a relatively simple update logic that deletes and recalculates these five values.

Another major difference is that the input parameters are not stored in a JSON format and transferred to a class [33]. They are stored directly in a constructor, which does not require time-consuming reading. In addition, many input parameters can be typed by using enums.

In conclusion, the implementations differ in various aspects. This will result in differences in performance.

2.2. Performance

The performance of the implementation presented here can be analysed from two perspectives. Firstly, in terms of quality, i.e., the accuracy of the model. The second focus of the analysis is the calculation time for the energy efficiency class. The implementation in [33] is compared with these two metrics against industry standards. Consequently, if the new implementation demonstrates better performance compared to that in [33], the results can be transferred.

2.2.1. Quality

The *18599 Gütegemeinschaft* covers software products that meet a certain quality standard, which is ensured by the provision of test cases and reference values. If a tool deviates by no more than 1% from these reference values, it can be described as DIN-compliant [33]. Since these test cases/reference values are not freely accessible for every version of DIN V 18599, Ricken [33] presents a method for generating these for new versions of the norm. The test cases are divided into single-family buildings (SFB) and apartment buildings (AB). The values generated in this way are used in the following as test cases or reference values, and the deviation of the values, calculated by the implementations, from the reference values is considered. Table 2 lists the deviation of the primary energy demand of the individual test cases from the reference values. It is evident that the new implementation provides greater accuracy, as the deviation is smaller in almost every test case. This is mainly due to the corrected error in Ricken's [33] implementation. As each deviation is less than 1%, the implementation is considered DIN-compliant.

Test Case	**Deviation [%] (New Impl.)**	**Deviation [%] (Impl. According to [33])**
AB 5 (2018)	0.47	0.48
AB 6 (2018)	0.80	1.55
AB 7 (2018)	0.93	3.58
AB 8 (2018)	0.76	1.46
AB 11 (2018)	0.99	51.68
SFB 5 (2018)	0.43	0.38
SFB 6 (2018)	0.61	1.13
SFB 7 (2018)	0.45	0.46
SFB 8 (2018)	0.08	1.46
SFB 11 (2018)	0.03	0.99

Table 2: Deviation of the calculated primary energy demand per square meter with the respective implementation from the reference values of the test cases provided by Ricken [33], distinguishing between apartment buildings (AB) and single-family buildings (SFB).

2.2.2. Time to Energy Efficiency Class

The time required to calculate the energy efficiency class describes the time component of performance. In [33], the implementation used there is benchmarked in terms of its runtime. However, if these values are used to compare the runtime of the new implementation, two problems arise. First, both implementations are executed with different computing power, which invalidates the comparison of the two runtimes. Second, the runtime in [33] also includes the generation of various output files, etc. Therefore, the time in milliseconds (ms) of both implementations executed on the same workstation is measured to calculate the energy efficiency class of the test case buildings.

The workstation has the following hardware configuration: AMD Ryzen 7 7700X CPU (8 cores, 16 threads, base clock 4.5 GHz), 32 GB DDR5 RAM, and an NVMe

SSD. The operating system is Windows 11 (64-bit). Each calculation is performed thousands of times on the workstation, and the mean is taken to avoid fluctuation effects.

Table 3 shows the two calculation times for the energy efficiency classes of all test buildings in both implementations. The new implementation is significantly faster (up to 150 times). Partly, this can be ascribed to the implementation language Swift, which is compiled and generally faster, but also to more efficient calculation implementations. Reasonable optimization requires repeated evaluation and is the most time-intensive operation within the optimization process. Minimizing the runtime is a strikingly important requirement for this thesis and will be elaborated later on.

Test Case	**Calculation Time [ms] (New Impl.)**	**Calculation Time [ms] (Impl. According to [33])**
AB 5 (2018)	0.07	5.35
AB 6 (2018)	0.18	6
AB 7 (2018)	0.96	6.05
AB 8 (2018)	0.94	7
AB 11 (2018)	0.05	5
SFB 5 (2018)	0.07	10.51
SFB 6 (2018)	0.19	7.51
SFB 7 (2018)	1.22	7.51
SFB 8 (2018)	1.21	7
SFB 11 (2018)	1	10.34

Table 3: Calculation times for the energy efficiency classes of all test buildings in both implementations. The test cases are separated between apartment buildings (AB) and single-family buildings (SFB).

3. Objective and Constraints: Model and Analysis

The thermal model provides the basis for the calculation of the optimization, but the structuring of the construction measures as optimization objects is not included yet. This model adds the essential variables that are used for optimization, which in turn can also represent conflicts or optimization goals. These aspects subsequently give rise to constraints and objectives.

3.1. Model of Construction Measures

It is crucial for the model of construction measures to reflect reality as accurately as possible. The scheme must therefore be able to model measures that operate on the building envelope.

The thermal building envelope is defined as the set of components that separate the heated from the unheated zones [34]. In its simplest form, the concept refers to external components that connect heated indoor areas with the unheated air outside. In a broader sense, the definition also includes components that connect two rooms, one heated and one unheated (e.g., a hallway and a garage). Hence, it consists of walls, roofs, a basement ceiling/base plate, as well as windows and doors. Its primary function is to regulate the flows of heat, air, and moisture between heated and unheated zones. Mathematically, the envelope can be represented as the boundary surface

$$\partial V = \{A_i \mid A_i \in \text{walls, roof, basement ceiling/base plate, windows, doors}\}.$$

The energy performance of the envelope is characterized by the overall heat transfer coefficient H_T, which results, accordingly to [10], from the sum of all thermal transmittances U_i of the components multiplied by their respective areas A_i:

$$H_{\mathrm{T}} = \sum_i U_i \cdot A_i. \tag{1}$$

This coefficient has a direct influence on the building's energy demand via Q_{T}, which is represented according to [10] by

$$Q_{\mathrm{T}} = H_{\mathrm{T}} \cdot |\theta_i - \theta_e| \cdot t, \tag{2}$$

where H_{T} denotes the heat transfer coefficient according to (1), θ_i is the internal temperature, θ_e is the average external temperature for the month, and t describes the duration of the calculation step (24h). Consequently, the degree of the thermal insulation has a major impact on the energy efficiency of the building. The better heat, i.e., energy, is retained in the building, the more efficiently it is used.

Each of the components, which belong to the building envelope, can only be optimized through specific construction measures. Even within a single component, there may be different ways of renovating it. In order to include all components in the optimization, it is necessary to generalise the individual components and procedures. Defining general characteristics that apply to all measures results in both comparability and parallel applicability. Such properties that characterise a measure are:

1. A name

2. A function that maps the old U-value of the component to the new one
3. The cost per square meter of the component in euros
4. A group of conflicting measures which, when applied, prevent the application of the measure (e.g., simultaneous insulation of a wall with different insulation materials)

The renovation process and characteristics are explained for each component below.

3.1.1. Wall Measures

The wall component is one of the most important factors in energy-efficient renovation, as it accounts for the largest percentage of the building envelope among all components. The study by Hinz [20] used to calculate the costs of construction measures in this work shows that only 9.76% of all measures are carried out on the exterior walls [20]. In contrast, the cost volume represents 22.6% of the total costs. Consequently, the energy-efficient renovation of the exterior walls has a major impact on energy efficiency, but also on the budget.

Such a renovation always involves insulating the walls, and there are two different approaches in practice: the external thermal insulation composite system (ETICS) [7] and the core insulation [27].

The most common method in Germany, according to [20], is ETICS, as it can be used on many types of walls. Here, the insulation material is attached to the outside of the wall and bonded and plastered in a special way [7]. Figure 1 shows a wall cross-section insulated with ETICS. The unique characteristic of ETICS is the

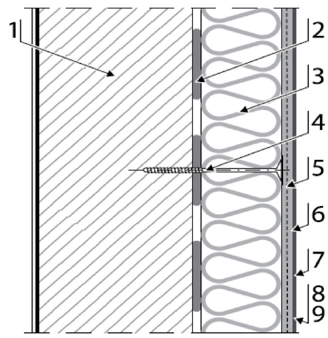


Figure 1: Layer structure of an ETICS system: (1) wall structure (substrate), (2) adhesive (basic fixing), (3) thermal insulation, (4) anchor (if necessary), (5) reinforcement layer (base coat with glass fiber mesh), (6) key coating, (7) finishing coat (render), (8) primer (optional), (9) decorative coat (optional). Source: Own illustration based on [7].

anchoring or plastering of the insulation. Layers 5 to 9 represent this feature and thus form the protective and finishing part of the ETICS. They provide crack resistance and mechanical stability (reinforcement layer), ensure proper adhesion (key coat and primer), and deliver weather protection as well as the desired aesthetic appearance through finishing and decorative coats [7].

If a component in the model is insulated with ETICS, the above definition requires a function that maps the old U-value to a new one. This function is described by the equation

$$U_{\mathrm{new}} = \frac{1}{\frac{1}{U_{\mathrm{old}}} + \frac{d_{\mathrm{ins}}}{\lambda_{\mathrm{ins}}}}, \tag{3}$$

where U_{old} is the old U-value, d_{ins} is the insulation thickness and λ_{ins} is the thermal conductivity of the insulation material used. This is a specialization of the formula from [34], assuming that U_{old} already accounts for the internal and external heat transfer resistances as well as the remaining insulation resistances.

To ensure the highest possible accuracy of the model, it initially appears reasonable to allow for arbitrary insulation materials and insulation thicknesses. Allowing arbitrary insulation materials would result in an unmanageable number of possible variants, each with distinct costs and thermal properties. To ensure both computational tractability and practical relevance, the model restricts the choice to a representative set of commonly used materials. It is concretely tailored to the insulation materials expanded polystyrene (EPS), extruded polystyrene (XPS), mineral wool, polyurethane (PUR) rigid foam, and wood fibre. The thermal conductivity of all materials was taken from product data sheets of leading German manufacturers. Allowing arbitrary insulation thicknesses is also not appropriate, since in reality, insulation materials are only available in certain thicknesses, and optimization with continuous insulation thicknesses may provide solutions that cannot be applied to the building. The common insulation thicknesses of 12, 14, 16, and 18 cm are included in the model. This results in 20 different ETICS-based insulation measures, which remain computationally manageable for optimization.

Another requirement is cost. Cost functions are used to describe the cost per centimetre of insulation thickness per square meter of a component [20]. The problem is that different insulation materials would also require different cost functions due to their different thermal conductivities. In order to make different insulation materials comparable and thus obtain a single cost function, the equivalent insulation thickness is introduced by [20] as

$$d_{\mathrm{eq}} = d_{\mathrm{ins}} \cdot \frac{\lambda_{\mathrm{ins}}}{\lambda_{\mathrm{ref}}}, \tag{4}$$

where d_{ins} describes the insulation thickness and λ_{ins} describes the thermal conductivity of the insulation material used. The reference value λ_{ref} is set to 0.035, which corresponds approximately to EPS. The equivalent insulation thickness d_{eq} represents the insulation thickness required to provide the same thermal performance as the reference insulation material [20]. Consequently, $d_{\text{eq}} < d_{\text{ins}}$ if $\lambda_{\text{ins}} < 0.035$, which implies that the thermal conductivity of the material is lower (i.e. better) than 0.035. d_{eq} can then be inserted into the cost equation, which for ETICS is described according to [20] as

$$C_{\text{ETICS}}(d_{\text{eq}}) \;=\; 2.8102 \cdot d_{\text{eq}} + 96.882. \tag{5}$$

The other insulation method, core insulation, is not as popular due to its limited application. In this method, the insulation material is sprayed into the space between the outer and inner walls [27], assuming such a space exists. The calculation of the new U-value is done in line with (3). Permitted insulation materials are EPS granulate, perlite, mineral wool, and aerogel. The permitted insulation thickness is a value that cannot be freely chosen due to the nature of the process. Nevertheless, for the sake of simplicity, thicknesses of 6, 8, and 9 centimetres were assumed.

[20] sets the cost function to

$$C(d_{\text{eq}}) \;=\; 1.6459 \cdot d_{\text{eq}} + 10.369. \tag{6}$$

The calculation method is identical to that used for ETICS.

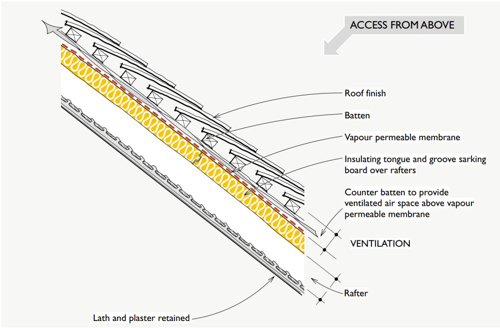
It is striking that core insulation is a much cheaper method than ETICS. However, due to its limited application and the small number of insulation materials available, it is rarely used in practice [20]. Nevertheless, the model retains this procedure for the sake of completeness.

Each wall measure is assigned to the conflict group 'Wall', as two wall measures can never be applied in parallel.

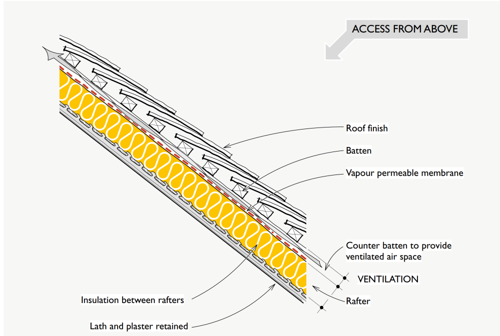
3.1.2. Roof Measures

The roof component plays an important role in energy loss from buildings. On the one hand, it accounts for a relatively large proportion of the total surface area of the building envelope, even if not as large as the walls. More importantly, however, the rising heat causes the roof to become a barrier between the outside air. Anyway, the evaluation in [20] shows that only 6.6% of the measures relate to the roof. One possible reason is the 25.18% share of the total cost of the evaluation.

Like the wall, the roof is one of the components that is being renovated for energy efficiency through insulation. As shown in [31], there are also two insulation methods that differ in terms of where the insulation is installed. Figure 2 shows the difference between the two installation methods. In the first, insulation is applied over the rafters in a continuous layer that wraps over the structural members and



(a) Above-rafter insulation method



(b) Between-rafter insulation method

Figure 2: Comparison of the two roof insulation methods. Source: Own illustration based on [31].

therefore reduces the potential thermal bridges. In the second, insulation is placed between the rafters with the timber structure exposed to the thermal envelope. The distinction is whether the rafters are inside or outside the insulated layer. Even though the two methods sound similar, they differ in practice. If the insulation is installed above the rafters, the entire roof usually has to be uncovered. This provides better insulation, as the rafters themselves do not form thermal bridges. If the insulation material is installed between the rafters, the rafters form thermal bridges, which leads to poorer insulation. Nevertheless, the insulation is usually easier to install.

Both processes are modeled similarly, whereby the new U-value is calculated using the formula outlined in (3). In the model, possible insulation materials for the above rafters are PUR, EPS, and wood fiber. The thicknesses used are 13, 17, 20, and 23 centimeters. Insulation installed between rafters can be modeled using mineral wool, wood fiber, and glass wool. The insulation thickness poses a similar problem to core insulation, because the width of the cavity is often simply used as the insulation thickness. Nevertheless, for reasons of simplification, various insulation thicknesses are assumed, namely 13, 17, 20, and 23 centimeters.

Although the two methods differ in reality, [20] assumes the same underlying cost function

$$C_{\text{Roof}}(d_{\text{eq}}) \;=\; 2.7738 \cdot d_{\text{eq}} + 151.01 \; . \tag{7}$$

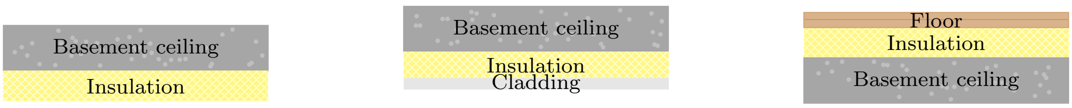
This significant simplification results in a low explanatory power of the cost model, as evidenced by a coefficient of determination of 0.09 (with 1 being the best possible value) [20].

Each roof measure is assigned to the conflict group ‘Roof’, as two roof measures can never be applied in parallel.

3.1.3. Basement Ceiling/Base Plate Measures

The building envelope includes not only components that form the barrier between heated rooms and the outside air, but also those that seal the building to the ground or basement. To minimize energy loss here, the basement ceiling or base plate can also be insulated. The evaluation in [20] shows that around 4% of the measures involve such renovations. The total cost amounts to only 1.56%, indicating a relatively low cost share. In practice, the energy-efficient renovation of this component usually achieves only small efficiency gains, which can be explained by the small surface area and the insulation effect of the ground. In any case, it is a cheaper way to insulate the building envelope to some extent.

After [20], there are two ways to insulate the component, which differ in whether the insulation is attached to the plate from above or from below. In practice, when insulating from below, it is also possible to cover the insulation with a cladding, see Figure 3. Hence, the model accounts for three insulation methods: insulation from below, insulation from below with cladding, and insulation from above.



(a) Insulation below (b) Insulation below with cladding (c) Insulation above

Figure 3: Schematic cross-sections of the basement ceiling with different insulation and finishing options (not to scale).

The U-value is calculated as in (3), where materials such as EPS, extruded polystyrene (XPS), mineral wool, PUR, and wood fibre are possible.

In principle, the cost calculation does not differ from other insulation methods, but there is a difference in the calculation of the equivalent insulation thickness. Here, a proportion of 20% wood is assumed by [20] to be installed in the basement ceiling. This results in the new formula for the equivalent insulation thickness as

$$d^*_{\text{eq}} = d_{\text{ins}} \cdot (0.8 + 0.2 \cdot \frac{\lambda_{\text{ins}}}{\lambda_{\text{ref,wood}}}) \cdot \frac{\lambda_{\text{ins}}}{\lambda_{\text{ref}}}. \tag{8}$$

The linear cost functions are provided by [20] with

$$C_{\text{U}}(d^*_{\text{eq}}) = 1.2463 \cdot d^*_{\text{eq}} + 30.754, \tag{9}$$

$$C_{\text{U,Cladding}}(d^*_{\text{eq}}) = 1.547 \cdot d^*_{\text{eq}} + 54.25, \tag{10}$$

$$C_{\text{T}}(d^*_{\text{eq}}) = 1.6207 \cdot d^*_{\text{eq}} + 8.9629. \tag{11}$$

Installing insulation above the floor is much cheaper than other methods. However, it should be noted that this is not possible in all floor situations. There must be enough space to install the material.

Each base plate measure is assigned to the conflict group 'Base Plate', as two base plate measures can never be applied in parallel in reality.

3.1.4. Window Measures

Windows are a critical component of a building's envelope. Although they only make up a small proportion of the total envelope, they are naturally susceptible to energy loss. That is why it is common practice to replace windows during energy-efficient renovations. This is also confirmed by the study [20], where window measures account for 31.7% of the total, with a cost share of only 15.9%.

Until now, only components that could be renovated for energy efficiency through insulation were considered. Windows are not insulated, but the glazing is changed. In the model, the types of glazing used are double glazing (2G), triple glazing (3G), and glazing in accordance with the Passive House standard (PHG) [1] [15]. The Passive Institute [4] (PHI) is an independent research institute based in Darmstadt, Germany, that develops and promotes highly energy-efficient building standards. PHI's criteria have become internationally recognized benchmarks for minimizing building energy demand. The standard defines PHG as an optimized version of normal triple glazing, see Table 4. The difference between 3G and 2G is trivial.

Property	**3G**	**PHG**
Number of panes	3	3
Gas filling	Argon (90%)	Argon/Krypton (high concentration)
Low-E coatings	2 layers	up to 3 layers
Frame	standard insulated	highly insulated
Edge spacer	warm edge	optimized warm edge

Table 4: Comparison of standard triple glazing and Passive glazing

Since windows are being completely replaced, the old U-value is no longer needed to calculate the new one. The ISO 10077-1 standard [22] provides a weighted sum for calculating the U-value for a window:

$$U_{\text{new}} = \frac{A_g \cdot U_g + A_f \cdot U_f + l_g \cdot \psi_g}{A_g + A_f}. \tag{12}$$

The terms A_g and U_g represent the area and thermal transmittance of the glazing, while A_f and U_f correspond to the area and thermal transmittance of the frame.

[4] *Passive Institute* – https://passiv.de/

The parameter l_g is the length of the glazing edge, and ψ_g denotes the linear thermal transmittance associated with this edge region.

Since only the total area of a window exists as data in the thermal model, assumptions must be made in order to use (12). For simplicity, this work assumes a ratio of 30 % frame to 70 % glass, and the windows are assumed to be square in order to determine their length and width. Table 5 shows typical U_g, U_f and ψ_g values for the three glazing systems, taken from [20].

	2G	**3G**	**PHG**
U_g	1.1	0.7	0.6
U_f	1.35	1.35	0.95
ψ_g	0.08	0.06	0.03

Table 5: U_g, U_f and ψ_g values of different glazing systems, according to [20].

The costs of the three systems are modeled using three different cost functions [20]. Each of these cost functions is exponential by nature. Furthermore, they do not represent the costs per equivalent insulation thickness per square meter, but rather the costs per average area per unit. The following specific cost functions are defined as

$$C_{\mathrm{2G}}(A_{w,avg}) = A_{w,avg}^{-0.231} + 413.45, \tag{13}$$

$$C_{\mathrm{3G}}(A_{w,avg}) = A_{w,avg}^{-0.222} + 472.33, \tag{14}$$

$$C_{\mathrm{PHG}}(A_{w,avg}) = A_{w,avg}^{-0.257} + 658.86, \tag{15}$$

where $A_{w,avg}$ is the average area of a window. However, the exponential approach seems to fit well, as the average certainty measure of 0.28 is relatively high compared to other estimates [20].

Each window measure is assigned to the conflict group 'Window', as two window measures can never be applied in parallel.

3.1.5. Door Measures

The doors of a building do not have a major impact on its energy efficiency. Nevertheless, like basement ceilings, they are a cost-effective option to achieve small efficiency improvements in energy-efficient renovations. This allows the target efficiency to be better approximated, as expensive and powerful measures tend to fall far short of the target or overshoot it.

As with windows, doors are also simply a matter of replacement. The model describes a door replacement using the replacement material. Liem [25] points out that calculating door U-values is complex; hence, the model relies on flat-rate U-values for certain materials. The considered materials are steel (hollow), steel (insulated), wood, fiberglass, half-lite steel insulated, half-lite wood, and half-lite fiberglass. A fixed price per square meter of door describes the cost [20]. Written as a function

$$C_{\text{Door}} \;=\; 1313.5. \tag{16}$$

Each door measure is assigned to the conflict group 'Door', as two door measures can never be applied in parallel.

3.1.6. Cost Adjustment to 2024

When looking at [20], the study from which all cost functions originate, it is noticeable that it was published in 2015. Due to inflation, the costs no longer correspond to the current ones, and all of the cost functions become invalid. Consequently, a procedure is needed to adjust the costs.

When prices need to be adjusted accordingly to inflation, a common method in literature is the Fixed Basket Laspeyres' method [29]. Put simply, a consumer basket consisting of a set of products and their prices is defined in the first period of the survey (base period). In all subsequent periods, the prices of the consumer basket are compared with the prices of the base period, and an index for that period is calculated. The period is usually a quarter or a year. With the help of the indices, costs can be converted by taking the index of the period from which the costs originate and the index of the period to which the prices are to be converted, and plugging these values according to [29] in

$$C^{*}(x) \;=\; C(x) \cdot \frac{i_{\text{current}}}{i_{\text{cost}}}, \tag{17}$$

where $C(x)$ represents the original cost function and i_{current} and i_{cost} are the corresponding indices.

The procedure described above is used to create the *Verbraucherpreisindex* (Consumer Price Index, CPI) that represents general inflation in Germany. The *Statistisches Bundesamt*[5] compiles and publishes the CPI on a regular basis. A naive approach would be to adjust the costs from [20] using the difference of CPI between 2015 and 2024 and applying the formula (17). However, since inflation does not affect all sectors equally, this approach is not accurate. For this reason, a sector-specific inflation differential in the extrapolation is used.

In Germany, the *Baupreisindex* (Construction Price Index, BPI) reflects the temporal development of prices in the entire construction industry. The calculation is similar to the CPI, except that the consumer basket consists only of

[5] *Statistisches Bundesamt* — `https://www.destatis.de/DE/Home/_inhalt.html`

products/services in the construction industry. The *Statistisches Bundesamt*[5] also compiles and publishes the BPI on a regular basis. It provides the general index, but also various indices for the sub-sectors of construction work.

Cost extrapolation based on the general construction price index provides more accurate values than those based on general inflation, but they are even more accurate when using the different indices from the sub-sectors. In the previous chapters, procedures for the individual components were explained. In order to adjust the costs with the sub-sector indices, the individual procedures must be assigned to the individual sub-sectors. If, for example, triple-glazed windows are used, the index for glazing work would seem appropriate. Even if this describes the majority of the work, roller shutter work is also often required, which has its own index. It is therefore necessary to weight the individual sub-sectors and thus the indices, through

$$C^*_{\text{sub}}(x) \;=\; C(x) \cdot \sum_{j \in \mathcal{S}} p_j \cdot \frac{i_{j,2024}}{i_{j,2015}}. \tag{18}$$

Here, $\mathcal{S}$ represents the sub-sectors covered by the measure, p_j defines the share of sub-sector j in the measure, and $i_{j,2024}$ and $i_{j,2015}$ define the indices of sub-sector j for 2024 and 2015, respectively.

Koch et al. [24] have carried out the weighting p_j for all measures described in Section 3.1. For example, triple glazing of a window is defined as 90% glazing work and 10% shutter work [24]. Table 6 shows a comparison of the costs in euros of three different measures, calculated using each of the three methods on AB 5 (2018). It can be observed that the prices differ considerably from one another.

Measure	**Inflation [€]**	**BPI [€]**	**Diverse Indices [€]**
Triple-glazed windows	25229.57	32961.06	32485.26
ETICS (EPS, 16 cm)	72340.77	94509.28	85953.94
Fiberglass Door	12237.57	15987.72	15912.98

Table 6: Cost comparison of selected measures under various price extrapolation until 2024 on AB 5 (2018).

There is a large difference between the price calculated with normal inflation and the BPI. In most cases, the improvement between the BPI and the use of different indices is not significant. This is explained by the comparison of the rates of change for the recent years relative to 2015 in Figure 4. However, both figures provide evidence that BPI overestimates and therefore generally calculates higher costs.

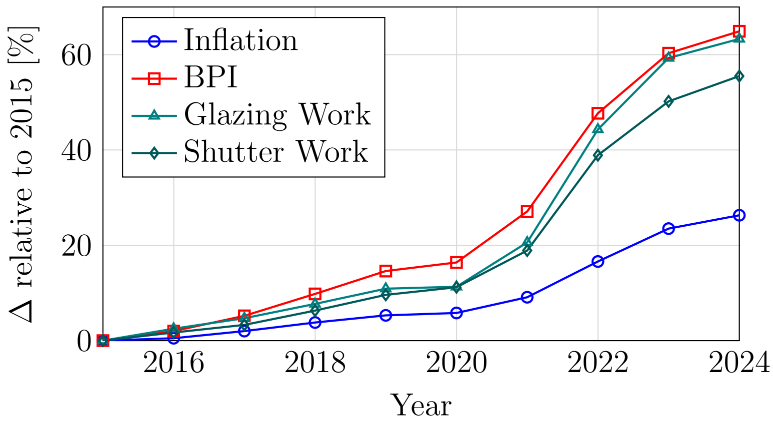


Figure 4: Relative change of the indices for inflation, BPI, glazing, and shutters work compared to the base year 2015.

3.2. Integration of Measures into the Thermal Model

The thermal model and the measures model form the basis for optimization. However, there is currently no interface between the two models, which is necessary in order to simulate the energy renovation of the thermal model using different measures. This abstract interface is described by the function

$$\mathcal{G} : \mathcal{B} \times 2^{\mathcal{M}} \to \mathbb{R}, \tag{19}$$

which, for any configuration of the thermal model (a specific residential building) and a combination of measures, returns the corresponding primary energy demand. Here, $\mathcal{B}$ denotes all possible thermal model configurations (i.e., all possible residential buildings), and $\mathcal{M}$ the set of all available measures.

In the following, $\mathcal{F}$ will denote the concrete implementation of this interface used in this work. Since, according to the definition in Section 3.1, every measure must have a function that maps the old value of a component to the new value, applying measures in $\mathcal{F}$ simply proves to be assigning the new U-value by mapping the old one. When measures are chosen that reduce (improve) the U-value of the components, the primary energy demand also decreases. This improvement in the demand is expressed below by the term benefit.

An important aspect to note is that $\mathcal{F}$ also applies the respective measures to all components of the appropriate type. For example, if there are four walls in the building envelope and the chosen measure is core insulation with aerogel at a thickness of 8 cm, the measure is applied to all four walls. This is a simplification, but it is close to practical renovation practice, as most renovations involve renovating all components in the same way, since uniform materials and tools simplify the process and lower costs.

What happens more often is that individual components are not renovated, for example, two windows. However, this is not specified in most existing tools either. It should also be noted that the function does not check for conflict-free status or the costs of the measures.

3.2.1. Examination of Additivity

Now that the application of measures can be simulated on the thermal model, the question of the additivity of the implementation $\mathcal{F}$ arises. This question proves to be one of the most important in this work, as it significantly influences the runtime of the optimization, as will be shown in the following section.

Additivity, together with homogeneity, forms the superposition property, which defines a system as linear [3]. Specifically, the two principles are expressed as

$$\text{Additivity:} \quad f(u_1 + u_2) = f(u_1) + f(u_2), \tag{20}$$

$$\text{Homogeneity:} \quad f(\alpha u) = \alpha f(u), \quad \alpha \in \mathbb{R}. \tag{21}$$

However, only additivity is of interest in this work. The definition in (20) cannot be applied, since the addition of two measures is not meaningfully defined. Therefore, reformulation is necessary with respect to the thermal model and the measures. A better formulation is

$$\triangle\mathcal{F}(B, X \cup Y) = \triangle\mathcal{F}(B, X) + \triangle\mathcal{F}(B, Y), \quad B \in \mathcal{B}, \tag{22}$$

where $\mathcal{F}$ corresponds to the definition of the interface function above, B denotes an arbitrary configuration of the thermal model (arbitrary residential building), and X and Y represent different measure sets. $\triangle\mathcal{F}(B, M)$ defines the percentage benefit of measure set M on B as

$$\triangle\mathcal{F}(B, M) := \frac{|\mathcal{F}(B, \emptyset) - \mathcal{F}(B, M)|}{\mathcal{F}(B, \emptyset)} \cdot 100 \; . \tag{23}$$

Here, M denotes an arbitrary set of measures and $\mathcal{F}(B, \emptyset)$ represents the initial primary energy demand of the specific building B. Consequently, additivity with this formulation means that the sum of the benefits of a set of measures, applied independently of each other, is equal to the benefit of the set of measures applied to the thermal model.

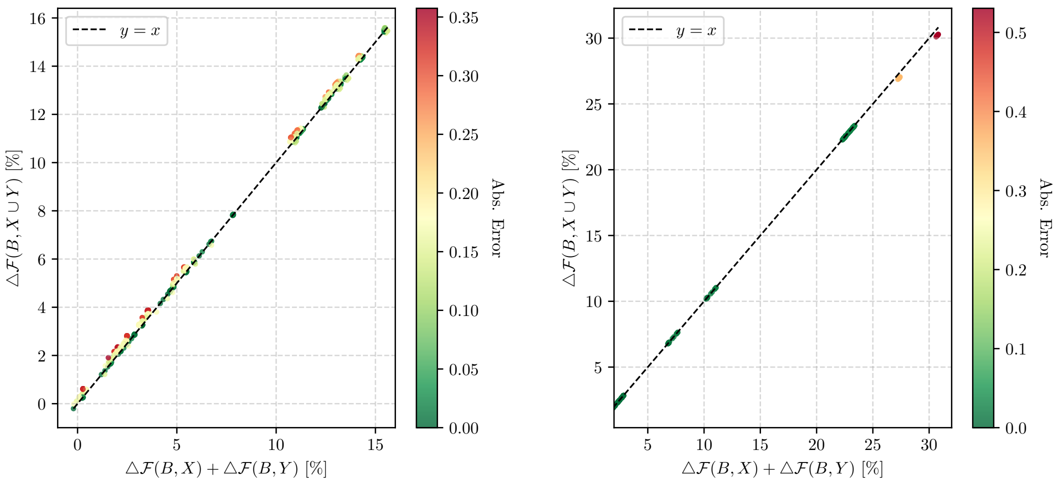
As shown in Section 2.2.2, the evaluation of the thermal model takes a long time. It will be the most expensive operation in the entire optimization process. Therefore, it is important to minimize the number of model evaluations. If $\mathcal{F}$ were additive for arbitrary residential buildings, the benefit of each combination of measures could be calculated with a simple sum by knowing the individual benefit of each measure. Resulting, the model would have to be evaluated as many times as there are measures, and the number of model evaluations would grow linearly with the increase in measures.

So, the aim is to show that $\mathcal{F}$ is additive for arbitrary buildings $B \in \mathcal{B}$. Such proof can be achieved mathematically by decomposing $\mathcal{F}$ and demonstrating additivity. In this case, however, this is not practical due to the high degree of complexity and the many possible configurations [33]. Another approach is simply to take all measures defined so far and test all possible combinations on additivity with (22) for all test case buildings defined in 2.2.1. Since only the additivity of the measures that are conflict-free is relevant, only conflict-free combinations are investigated, which strongly restricts the number of all combinations. As proof of additivity, this procedure is sufficient in this case. Nevertheless, the number of combinations of arbitrary length grows exponentially with the number of measures, despite the conflict-free status. Thus, this procedure is not computable in reasonable time.

This work tests only combinations of two non-conflicting measures. Firstly, the number of model evaluations of such combinations can be calculated efficiently, and if non-additivity is determined, this also applies to larger combinations. Specifically, the benefit of each measure used is determined for each conflict-free pair, and the benefit of the two measures together is calculated. Then, it is checked whether the sum of the individual benefits is equal to the joint benefit and, if not, how large the deviation is. Figure 5 illustrates such a graphical analysis for test case buildings SFB 11 (2018) and AB 7 (2018). The x-axis shows the total sum of the two benefits of the two measures, and the y-axis shows the actual benefit of the two measures applied in parallel. The intensity of the red color of a dot indicates the magnitude of the deviation between the sum of the individual benefits and the actual benefit.

Figure 5 conveys a clear additive character of $\mathcal{F}$. However, Table 7 show that true equality, i.e., strict additivity, does not hold in any single case. Nevertheless, there are stronger non-additive test cases than others. SFB 11 (2018) is by far the most non-additive, followed by SFB 8 and 7 (2018). In order to find a reason for this strong non-additivity, the unique characteristics of these test cases must be specified. Such an analysis demonstrates that the heat pump is one decisive factor for significant non-additivity. Other factors were not apparent from the analysis. This means that when optimising buildings with heat pumps, the additivity character of $\mathcal{F}$ can be used even less than usual.

A further point of interest is the proportion of all positive (the actual benefit is greater than the estimate) or negative (the actual benefit is less than the estimate) deviations. If, in a test case such as AB 5 (2018) (see Table 7), the sum of the benefits is greater than the actual benefit every time, then an algorithm that only works with the sums of the benefits will always produce values that are greater



(a) SFB 11 (2018)

(b) AB 7 (2018)

Figure 5: Comparison between the estimated benefit (sum of individual benefits) and the actual benefit obtained when applying both measures simultaneously. The dotted line indicates perfect additivity (y = x), while deviations show non-additive effects.

Test Case	**Pairs (Total)**	**Exact Add. Pairs**	**MAE**	**Share Pos. [%]**	**Share Neg. [%]**
SFB 11 (2018)	2414	0	0.14	93.54	6.46
SFB 8 (2018)	2414	0	0.08	55.84	44.16
SFB 7 (2018)	2414	0	0.06	41.30	58.70
AB 5 (2018)	911	0	-0.04	0	100
AB 7 (2018)	911	0	-0.04	0	100
SFB 5 (2018)	2414	0	-0.03	5.43	94.57
AB 8 (2018)	911	0	-0.02	1.43	98.57
AB 11 (2018)	911	0	-0.02	8.78	91.22
AB 6 (2018)	911	0	0	44.24	55.76
SFB 6 (2018)	2414	0	0	24.73	75.27

Table 7: Analysis of the distribution of non-additivity per test case building, showing the number of exact additive pairs, the mean absolute error (MAE), and the share of positive or negative deviations. Test cases are grouped into single-family buildings (SFB) and apartment buildings (AB).

than they would be in reality. The same holds in reverse; however, with respect to a target benefit, exceeding the benefit is less critical than falling short.

Finally, it can be concluded that the strict additivity of $\mathcal{F}$ never holds. It must be examined to what extent the slight non-additivity restricts algorithms that make use of this property. Variants that are aware of the slight deviations and can deal with them must also be considered.

3.2.2. Sensitivity Analysis

The introduction of various measures that can be implemented on the thermal model raises the question of the thermal model's sensitivity to changes in different components, regardless of the configuration. In other words, how much do measures applied to the roof, walls, or windows actually contribute to energy savings, and to what extent are these components responsible for overall energy losses?

This question can be answered by a sensitivity analysis. Here, real measures are abstracted, and only the improvement of the U-value in the respective component (in %) is taken as a characteristic of a measure.

A first approach is to implement a measure that improves the U-value from a certain component by x percent and then observe the benefit. Hamby [18] defines this procedure as One-at-a-Time sensitivity analysis (OAT). OAT allows the sensitivity of different components to be compared by checking whether the same changes to the U-value also result in the same improvement in benefits. Although OAT delivers faster results due to its simplicity, it is not meaningful according to [18]. OAT is referred to as local sensitivity analysis because it does not take into account the entire distribution of the test points.

After [21] and [18], the standardised regression coefficients (SRCs) are a more robust method for testing sensitivity. The standardised regression is based on linear regression and aims to extract and compare the vector of the regression coefficients [21]. In comparison, SRCs are more reliable than OAT because they assess the effects of several measures on the thermal model rather than just one. Figure 6 shows a typical workflow when analyzing with SRCs.

Since regression requires samples, a method for generating these is needed. The literature often refers to Latin Hypercube Sampling (LHS) in this context, which is treated in [26]. LHS generates samples that reflect a balanced representation of the multidimensional input space. This work analyzes the sensitivity of the wall component, roof component, window component, and door component. For that purpose, the input space is defined as a four-dimensional space where each axis represents the improvement of the U-value of the respective component by a measure in %, resulting in a four-dimensional hypercube $[0, 100]^4$.

To generate n samples, the input space is divided along each of its dimensions into n equally probable intervals, as shown in Figure 7. Samples are generated such that each interval of every dimension is represented exactly once across the

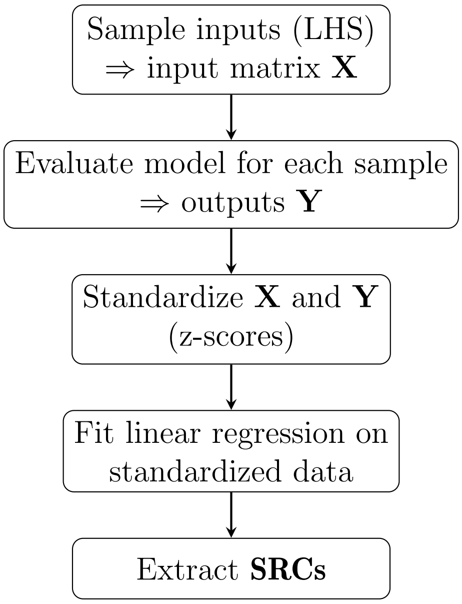


Figure 6: Workflow of standardised regression coefficients [21].

sample set, while no two samples contain values from the same interval within a given dimension. Specific values are selected at random from the corresponding interval, resulting in n samples consisting of four values, each of which represents a measure that improves the U-value of the respective component by this percentage. These samples define the input matrix $\mathbf{X}$.

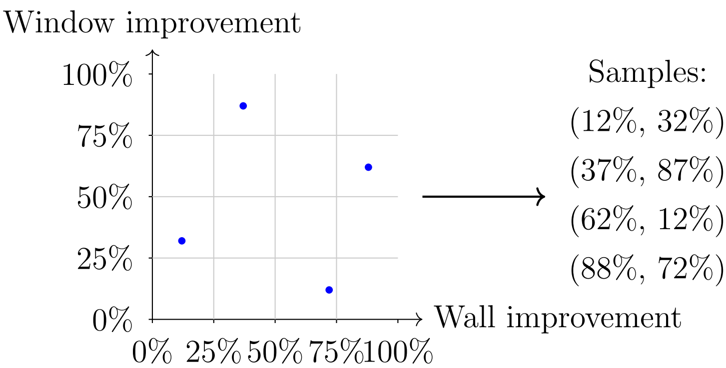


Figure 7: Example of valid sampling according to LHS with $n = 4$ in two-dimensional space.

$\mathbf{X}$ is then evaluated with $\mathcal{F}$ under a test case building, and the benefit is defined as the output vector $\mathbf{Y}$. Since later, the different regression coefficients of the individual variables $\mathbf{X}_i$ will be compared, units and scales must be removed to ensure comparability. In SRCs, this is done using the z-transformation. Each component $\mathbf{X}_i$ and the output $\mathbf{Y}$ is standardised by

$$\mathbf{X}_i^* = \frac{\mathbf{X}_i - \mu_{\mathbf{X}_i}}{\sigma_{\mathbf{X}_i}}, \qquad \mathbf{Y}^* = \frac{\mathbf{Y} - \mu_{\mathbf{Y}}}{\sigma_{\mathbf{Y}}},$$

where μ denotes the mean and σ the standard deviation of the respective variable. Then, simple linear regression is used on the standardised data to determine the regression coefficients of the variables $\mathbf{X}_i$. The regression coefficient of variable or component i can then be interpreted in relation to sensitivity, according to Table 8.

SRC β_i	**Interpretation in Sensitivity Analysis**
$\beta_i > 0$	Increasing $\mathbf{X}_i$ improves the output; parameter has a positive contribution to sensitivity.
$\beta_i < 0$	Increasing $\mathbf{X}_i$ deteriorates the output; parameter has a negative contribution to sensitivity.
$\|\beta_i\| \approx 0$	Variation of $\mathbf{X}_i$ varies output by a fraction; Component i can be considered non-influential.
$\|\beta_i\| >> 0$	Variation of $\mathbf{X}_i$ noticeably varies the output; Component i can be considered sensitive.

Table 8: Interpretation guidelines for standardized regression coefficients (SRCs) β_i in sensitivity analysis, where β_i denotes the regression coefficient of component i.

The SRCs for the window, wall, roof, and door components were calculated as representative for the test case SFB 8 (2018) using 10,000 samples and are presented in Table 9. The findings are consistent across all other test cases, which are omitted here for the sake of clarity. Many of the assumptions made in Section 3.1 are confirmed. Although windows do not account for a large proportion of the building envelope, they are very susceptible to energy loss and represent a major adjustment factor. The roof actually has less influence than expected. Consequently, measures for wall and window components that change the U-value by a large percentage also have a large positive effect on the output of $\mathcal{F}$. The optimization should therefore place greater focus on these components.

Component	SRCs	Interpretation
Window	0.696	Strong positive influence, most sensitive component.
Wall	0.674	Strong positive influence, nearly as important as windows.
Roof	0.210	Weak to moderate sensitivity, limited contribution to output variance.
Door	0.155	Weak sensitivity, least influential component.

Table 9: Estimated SRCs for building components of SFB 8 (2018) and their interpretation in terms of sensitivity.

4. Optimization Methodology

There are currently software solutions on the German market that offer the definition and simulation of various measures, see Table 1. The definition and simulation of the measures were described in Section 3. Even though the user can simulate various measures, finding a set that is conflict-free and, when applied, generate less or the same amount of primary energy demand as the target demand is very time-consuming and complex. When searching for the optimal selection, the individual cannot be expected to choose the economically most convenient set of measures that meets the requirements mentioned above, without any support.

Consequently, an algorithmic solution is required to solve the optimization problem. In order to construct such a solution, the optimization problem must be formalized and examined for its complexity. This allows a worst-case runtime to be defined, which must then be improved upon in the average case. For this purpose, various algorithms are presented, which make different assumptions and therefore differ in terms of runtime and solution quality.

4.1. Problem Formulation

The user defines a target primary energy demand that the building should have at most after the energy-efficient renovation. In most cases, this target value is chosen corresponding to the highest value of the desired energy efficiency class. $\mathcal{M}$ represents all available measures, which are produced by all the different procedures with the various materials and thicknesses described in Section 3.1. In total, $\mathcal{M}$ contains in the worst case 104 specific measures, but can also be filtered by the user to exclude certain measures, for example, the replacement of windows. A set of measures must be selected from $\mathcal{M}$ so that applies:

1. The set is conflict-free, meaning it can be applied.

2. If the set is applied to the building, the building has a primary energy demand $\leq$ the target demand.

3. The total cost of the individual measures in the set is minimal among all sets that satisfy requirements (1) and (2).

An important observation regarding these requirements is that (1) and (2) are requirements that must be met for a solution to be valid. Omitting (3) results in a decision problem, while including it yields an optimization problem. This optimization problem is defined under the name *Cost Minimization Problem with Conflicts* (CMP-C).

In the literature, such problems as CMP-C are also referred to as constrained optimization problems. Dong [12] provides a precise definition and characteristics of this optimization category. Formally, a constrained optimization problem can be defined as

$$\begin{aligned} \text{minimize} \quad & f_0(x_1, \ldots, x_l) \\ \text{subject to} \quad & f_1(x_1, \ldots, x_l) \leq 0, \\ & \vdots \\ & f_k(x_1, \ldots, x_l) \leq 0, \end{aligned}$$

where f_0 is called the objective function and f_i, $i = 1, \ldots, k$ are the constraint functions. In order to define the objective function and the constraint functions in the context of work, the selection of measures must be modeled mathematically. In this scenario, each measure can only be selected or not. So the selection from $\mathcal{M}$ is modeled with binary decision variables

$$x_j \in \{0, 1\}, \quad j = 1, \ldots, |\mathcal{M}|,$$

where $x_j = 1$ denotes that measure j is selected and $x_j = 0$ means that it is not. Thus, the selection of a set is represented by setting the corresponding x_j to one.

Requirement (3) represents the objective of optimization. This objective function or cost function is described with binary decision variables by

$$\sum_{j \in \mathcal{M}} c_j x_j, \tag{24}$$

where c_j denotes the cost of measure j. Since the sum yields a single value (is one-dimensional), CMP-C can be characterized as a single-objective constraint optimization problem.

Requirements (1) and (2) define the constraints and are therefore represented by constraint functions. (2) represents the energy demand constraint and is modeled by

$$\mathcal{G}(B, M) \leq b, \quad B \in \mathcal{B}, \tag{25}$$

where $M := \{ j \in \mathcal{M} \mid x_j = 1 \}$ denotes the set of selected measures and b represents the target demand.

(1) ensures that the set is conflict-free. Let L be the set of all conflict groups, for example, 'Window', 'Wall', etc. The measures belonging to conflict group $i \in L$ are formally given as

$$S_i = \{ j \in \mathcal{M} \mid j \text{ in conflict group } i \}.$$

The conflict freedom is formulated as a set of constraint functions:

$$\sum_{j \in S_i} x_j \leq 1, \quad \forall i \in L. \tag{26}$$

In summary, the single-objective constrained optimization problem *Cost Minimization Problem with Conflicts* (CMP-C) is described through (24), (25), (26) by

$$\begin{aligned} \text{(CMP-C)}\min \quad & \sum_{j \in \mathcal{M}} c_j x_j \\ \text{s.t.} \quad & \mathcal{G}\big(B, \{ j \in \mathcal{M} \mid x_j = 1 \}\big) \leq b, \quad B \in \mathcal{B} \\ & \sum_{j \in S_i} x_j \leq 1, \quad \forall i \in L \\ & x_j \in \{0, 1\}, \quad \forall j \in \mathcal{M}. \end{aligned} \tag{27}$$

Remark. It should be noted that CMP-C is defined with the abstract interface $\mathcal{G}$. Whenever CMP-C is referred to in a practical context in this work, the abstract function $\mathcal{G}$ is instantiated by its concrete implementation $\mathcal{F}$. Correspondingly, all empirical analyses are carried out with $\mathcal{F}$, while $\mathcal{G}$ is used solely to keep the theoretical problem definition and the complexity analysis, independent of specific implementations.

4.1.1. Complexity and NP-Hardness

In computer science, computational problems are classified according to their difficulty. This allows the resources required for solving the problem, such as computing time or memory space, to be estimated. There are two popular classes where problems can lie, namely P (polynomial time) and NP (nondeterministic polynomial time). Both are covered in [16]. P includes problems that can be solved efficiently (in polynomial time), and NP contains problems for which solutions can be verified in polynomial time. It is generally assumed that P $\neq$ NP applies, which implies that many NP problems cannot be solved efficiently and may require super-polynomial time in the worst case. This classification lays the foundation for understanding the theoretical or worst-case performance of algorithms and optimizers. It is important to clarify that even though a problem lies in NP, i.e., it supposedly requires non-polynomial computation time, there may still be efficient solvers that solve the problem in polynomial time in most cases, i.e., in pseudo-polynomial time [32]. However, there will always be a case where those solvers require non-polynomial computation time.

According to [17], the NP-hardness property of a problem describes that there is a polynomial reduction from any problem in NP. Such a reduction is described by a function that converts instances of one problem into another problem and must be executable in polynomial time. If such a reduction exists, every efficient algorithm that solves an NP problem can also be used to solve an NP-hard problem by applying the reduction. NP-hardness, therefore, indicates that a problem is at least as difficult to solve as any problem in NP. It is often proven by specifying a polynomial reduction from an already known NP-hard problem.

To show NP-hardness of CMP-C (27), the problem is restricted to a special subclass of instances where $\mathcal{G}$ has an additive structure. Proving NP-hardness for this variant of CMP-C immediately implies NP-hardness for CMP-C. The variant is defined as *Cost Minimization Problem with Conflicts under Additivity*

$$\begin{aligned}
\text{(CMP-CA)}\min \quad & \sum_{j\in\mathcal{M}} c_j x_j \\
\text{s.t.} \quad & \sum_{j\in\mathcal{M}} b_j x_j \geq b \\
& \sum_{j\in S_i} x_j \leq 1, \quad \forall i \in L \\
& x_j \in \{0,1\}, \quad \forall j \in \mathcal{M},
\end{aligned} \tag{28}$$

where all variables are defined as in (27), and $b_j := \triangle\mathcal{G}(B, M_j)$ with $B \in \mathcal{B}$, while $\triangle\mathcal{G}$ is given by (23). b represents the target benefit, i.e. the percentage by which the current primary energy demand must be improved in order to reach the target demand.

The NP-hardness of CMP-CA is shown with polynomial reduction from the multiple-choice knapsack problem (MCKP), defined by Cacchiani et al. [4]. MCKP is a general variant of knapsack and is also known as the knapsack problem with generalized upper bound constraints. The central idea is that the item set of the normal knapsack problem is divided into ℓ classes $N_1, \ldots, N_\ell$ so that the classes form a partition of the original item set. At most one item may be selected from each class. The definition is therefore formally given as

$$
\begin{aligned}
\text{(MCKP)}\max \quad & \sum_{j=1}^{n} p_j x_j \\
\text{s.t.} \quad & \sum_{j=1}^{n} w_j x_j \le c \\
& \sum_{j \in N_i} x_j \le 1, \quad (i = 1, \dots, \ell) \\
& x_j \in \{0,1\}, \quad (j = 1, \dots, n),
\end{aligned}
\tag{29}
$$

where n is the number of items in the item set, p_j the non-negative profit and w_j the non-negative weight of item j. According to [32], MCKP is NP-hard.

Theorem 1. *CMP-CA is NP-hard.*

Reduction from MCKP. Given an instance of MCKP with classes $N_1, \dots, N_\ell$, profits p_j, weights w_j, and capacity c, construct an instance of CMP-CA by

Ground set	$\mathcal{M} := \bigcup_{i=1}^{\ell} N_i$, it follows that $\lvert\mathcal{M}\rvert = n$
Costs	$c_j := -p_j, \quad j \in \{1, \dots, n\}$
Weights	$b_j := -w_j, \quad j \in \{1, \dots, n\}$
Capacity	$b := -c$
Conflict set	$L := \{1, \dots, \ell\}$
Conflicts	$S_i := N_i$ for all $i = 1, \dots, \ell$.

This mapping is computable in $\mathcal{O}(n)$.
Constraint equivalence. Consider any vector $x \in \{0,1\}^n$. The capacity constraint in MCKP or CMP-CA is realized by

$$
\begin{aligned}
\text{MCKP:} \quad & \sum_{j=1}^{n} w_j x_j \ \le \ c, \\
\text{CMP-CA:} \quad & \sum_{j \in \mathcal{M}} b_j x_j \ \ge \ b.
\end{aligned}
$$

By construction $b_j = -w_j$ and $b = -c$, so

$$
\sum_{j=1}^{n} w_j x_j \le c \iff -\sum_{j=1}^{n} w_j x_j \ge -c \iff \sum_{j=1}^{n} (-w_j) x_j \ge -c \iff \sum_{j \in \mathcal{M}} b_j x_j \ge b.
$$

Therefore, x satisfies the MCKP capacity constraint if and only if x satisfies the CMP-CA capacity constraint.

The conflict constraints of MCKP or CMP-CA are given by

$$\begin{aligned} \text{MCKP:} \quad & \sum_{j \in N_i} x_j \leq 1, \quad (i = 1, \ldots, \ell), \\ \text{CMP-CA:} \quad & \sum_{j \in S_i} x_j \leq 1, \forall i \in L. \end{aligned}$$

Since $S_i := N_i$ and $L = \{1, \ldots, \ell\}$, the conflict constraints are identical. Thus, x satisfies all MCKP conflict constraints if and only if x satisfies all CMP-CA conflict constraints.

Objective equivalence. Consider any vector $x \in \{0,1\}^n$. The objective for MCKP or CMP-CA is

$$\begin{aligned} \text{MCKP:} \quad & \max \quad \sum_{j=1}^{n} p_j x_j, \\ \text{CMP-CA:} \quad & \min \quad \sum_{j \in \mathcal{M}} c_j x_j. \end{aligned}$$

By construction $c_j := -p_j$ so

$$\min \sum_{j \in \mathcal{M}} c_j x_j = \min \sum_{j=1}^{n} (-p_j) x_j = \min \left(-\sum_{j=1}^{n} p_j x_j \right) = -\max \sum_{j=1}^{n} p_j x_j.$$

Accordingly, maximizing the MCKP objective is equivalent to minimizing the CMP-CA objective.

Finally, both problems impose the domain restrictions $x_j \in \{0,1\}$. Therefore, every feasible solution of MCKP corresponds to a feasible solution of CMP-CA, and every optimal solution of MCKP corresponds to an optimal solution of CMP-CA. Since this reduction is polynomial and preserves optimality, CMP-CA is NP-hard because MCKP is NP-hard [32].

□

Since CMP-CA is by definition a restricted special case of CMP-C, every instance of CMP-CA can be converted into an instance of CMP-C. Thus, when considered as a problem class, CMP-CA $\subseteq$ CMP-C applies, and the NP-hardness is transferred.

Under the assumption P $\neq$ NP, it follows that CMP-C and CMP-CA lie in NP. Nevertheless, for MCKP, and thus through the reduction also for CMP-CA, efficient algorithms that solve in pseudo-polynomial time are known [32]. Although CMP-C and CMP-CA share the NP-hardness, in practice, CMP-C will be harder, since fewer mathematical properties of $\mathcal{G}$, such as additivity, can be exploited.

4.2. Exhaustive Search

Exhaustive search is the simplest and also the most straightforward approach to solving optimization problems, like CMP-C (27). The approach consists of trying all possible combinations and selecting the best one [30]. Specifically, this can be done by recursively traversing the entire search tree. In context of CMP-C, all possible combinations of measures are evaluated by $\mathcal{F}$ and then checked for their costs, benefits, and absence of conflicts. Since each measure can either be taken or not, the number of different combinations grows exponentially with the number of available measures, i.e., $2^{|\mathcal{M}|}$. The maximum total number of measures is 104, which results in a number of 2^{104} combinations and, above all, 2^{104} models to evaluate. With an evaluation time in milliseconds for the thermal model (see Section 2.2.2), this would result in a computationally infeasible runtime.

Nevertheless, exhaustive search is used in this work. The decisive advantage is that considering all combinations guarantees finding the optimal solution. Thus, exhaustive search is used to obtain reference values for benchmarking the quality of other algorithms.

In order to be able to use the algorithm, it must be made practicable by reducing the number of possible combinations. For this purpose, combinations that are either not conflict-free or are already more expensive than the cheapest valid combinations found so far are filtered out. Correctness is nevertheless guaranteed, as only conflict-free combinations are valid and negative costs are excluded by definition. This pruning drastically reduces the number of combinations to 3801599 $\approx 2^{22}$, but calculating this would still take around one hour under the above conditions. So, options, like reducing the measure set, must be considered for generating reference values.

4.3. Branch-and-Bound with Additive Relaxation

Since exhaustive search has proven to be computationally expensive, different approaches are needed that evaluate the model significantly less. This section introduces a whole class of algorithms that use the partial additivity of $\mathcal{F}$ (see Section 3.2.1). According to 4.1.1, CMP-C (27) and its additive variant CMP-CA (28) are NP-hard, but CMP-CA can be solved in pseudo-polynomial time. The algorithmic strategy is therefore to:

1. Assume additivity in CMP-C to transform it into CMP-CA.
2. Solve CMP-CA.
3. Adapt the solution for the CMP-C context.

This section presents branch-and-bound (B&B) algorithms for solving CMP-CA and discusses variants for transferring these solutions to CMP-C.

Due to the additive relaxation in the following, the individual benefit of each measure, i.e., the percentage by which the measure reduces primary energy demand, is focused, and no longer the evaluation with $\mathcal{F}$ of various combinations of measures. If additivity applies, the benefit of each combination of measures can be calculated by summing them. For this purpose, from now on, a target benefit is considered and no longer the target primary energy demand, which must be achieved.

However, all presented variants have the following mandatory steps in common, shown in Figure 8. To embed CMP-CA, the CMP-C instance is converted into a

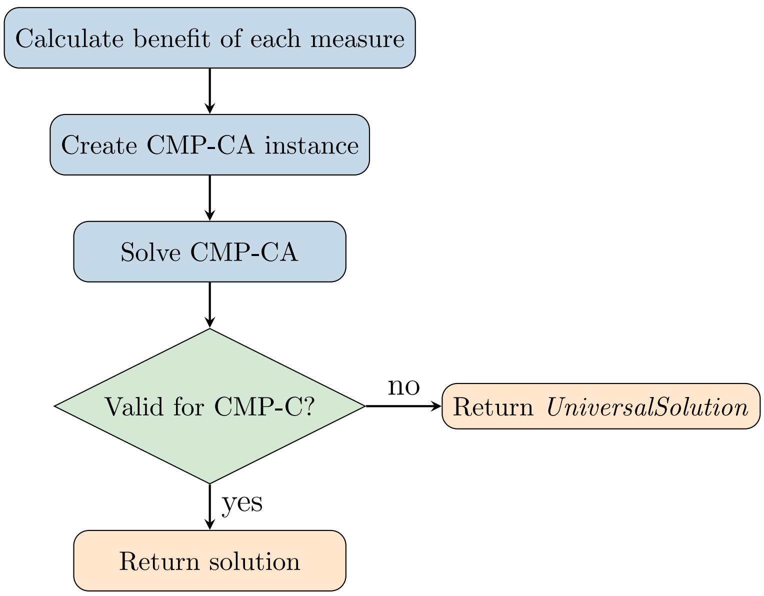


Figure 8: Basic workflow of B&B with an assumed additivity of CMP-C and validation in CMP-C.

CMP-CA instance by defining a mapping of all measures in $\mathcal{M}$ to optimization objects of CMP-CA. The individual benefit of each measure of $\mathcal{M}$, if it is applied to the thermal model, is calculated according to the formula (23). This benefit, cost, and conflict group defines this mapping. The instance is then solved with B&B, and the solution is validated, i.e., checked to see whether the selected measures, when applied simultaneously, achieve a benefit greater than or equal to the target benefit. If the solution is not valid, a fallback solution can still be returned. There is a set of conflict-free measures that provide the greatest possible benefit, called *UniversalSolution*. This set is predefined and is selected by choosing the best insulation materials and the thickest insulation thicknesses, etc. This solution is very expensive, but it fulfills every user request, unless the user asks for a value that cannot be achieved. However, these requests are filtered out.

4.3.1. Fundamentals of Branch-and-Bound

B&B provides a framework for systematically traversing the search tree, whereby branches that cannot produce valid solutions are discarded as early as possible. The following section refers to the work of Morrison et al. [28] for definitions and procedures. In the literature, B&B is often referred to as an algorithm. This is not entirely correct, as only various basic strategies are provided, but the individual elements remain configurable. B&B thus consists of three components: the search strategy, the branching strategy, and the pruning rules. There are standard procedures for all three components, but it is up to the user to decide which one to choose.

The search strategy describes how the search tree will be traversed. The variants depth-first, breadth-first, and best-first search are frequently used here. Depth-first search is defined as exploring to the first leaf of the tree and then backtracking to the last split and exploring it. This is very easy to implement and memory-saving. Nevertheless, if the search tree is unbalanced, i.e., has long paths without the optimal solution, depth-first search can explore subtrees in detail without a solution. To prevent this, breadth-first search can be used. Here, each level of the search tree is completely explored, starting from the root before moving on to the next level. Yet for most problems, complete solutions are found at great depths and are therefore not found quickly by breadth-first search, unless pruning is very good. Best-first search always examines the subproblem with the most promising bound or heuristic. This allows unpromising subtrees to be excluded directly. However, best-first search consumes a lot of memory, as all open nodes must be kept in a priority structure.

The branching strategy describes how subproblems or subtrees are generated. Here, only binary branching and wide branching are of interest. In most cases, nevertheless, the type of branching is determined by the nature of the problem. Binary branching describes the division of the problem into two independent subproblems, with each node having two children. Problems with binary decisions such as the 0–1 knapsack problem, where each item can be taken or not, require binary branching. Wide branching, on the other hand, describes generating nodes with many children, often required by problems such as maximum cliques or independent sets, according to [28]. Branching affects the shape of the search tree and, as a result, the efficiency of the pruning rules or the search strategy.

Pruning rules are a characteristic feature of B&B and determine when a subproblem is no longer pursued, because it can no longer produce a valid or optimal solution. This is also a very critical part of B&B, as a poor choice of pruning rules can significantly increase the runtime or cause incorrectness. A frequently chosen strategy is lower bounds. Here, the smallest (best) function value of the subtree is calculated that it can achieve in the best case. If this value is worse than the current incumbent, the subtree no longer needs to be pursued. Sometimes, however, this best value cannot be calculated efficiently. Then an estimate is needed that is always greater than or equal to the actual maximum achievable value. If this is not the case, B&B is no longer correct, as valid optimal solutions are not examined. This pruning rule therefore applies that the faster an incumbent is found, the sooner this pruning rule can take effect and exclude subtrees.

B&B is deterministic and complete for a finite search space, assuming the pruning rules are correct. This follows from the non-use of heuristic or random-based selection and the finiteness of the search tree.

4.3.2. Problem-Specific Search Strategy and Pruning Rules

The following section describes the concrete implementation of the three components.

Since binary decision variables are required for both CMP-C and CMP-CA, binary branching is fixed as the branching scheme.

A classic search tree, as shown in Figure 9, is considered to select a suitable search strategy and pruning rules. The cheapest conflict-free combination is sought that has a benefit greater than or equal to the target benefit. If an order is defined on $\mathcal{M}$, then level j always decides whether measure j should be added to the existing combination or not. When it is added (always the left child), the benefit and cost are summed, and the conflict group is joined to a set of conflict groups of the combination. If not, these parameters remain unchanged. The graph shows that with each level of depth, the benefit always increases or remains the same, but never decreases, with the benefit being greatest in the leftmost branch. Therefore, it can be useful to explore deeper levels of the search tree first in order to quickly find valid combinations with a benefit above the target benefit. Of course, this only applies to paths that frequently go into the left subtree, i.e., take a measure. The path on the far right, for example, leads to a benefit of zero. It can also be seen that the tree is completely balanced before pruning.

All of this suggests a depth-first search strategy. However, there are a few details to note. First, the traversal must always take the left child first and then the right child. This ensures that a valid combination is found as quickly as possible, even if the costs are high. In contrast, it was observed in this work that going first into the right subtree deteriorates the runtime by up to 25%. Furthermore, it is noticeable that the order of the measures in $\mathcal{M}$ can influence the traversing. The more cost-efficient measures there are at the beginning, the faster valid cost-efficient combinations are found. For this reason, the measures are sorted according

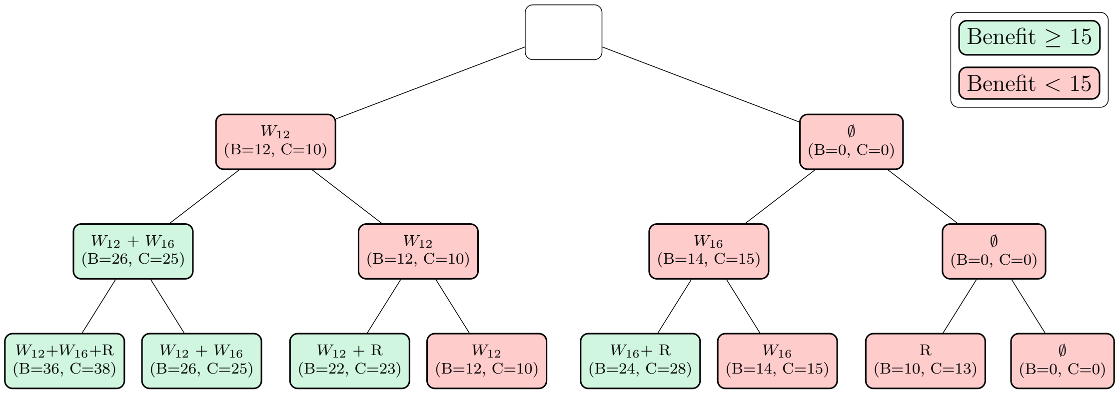


Figure 9: Complete search tree generated by B &B with binary branching, including benefit (B) and cost (C) values for each node, with order W_{12} (wall insulation with 12 cm), W_{16} (wall insulation with 16 cm) and R (roof insulation) under target benefit 15.

to their benefit-cost ratio, i.e., the benefit per euro invested. Consequently, as in a greedy approach, the most cost-efficient measures are selected first.

As already explained above, the pruning rules are decisive for the runtime of B&B. Three such rules are presented below, and their correctness is proven. The first and most important pruning rule is to exclude combinations with conflicts. An example of the rule is shown in Figure 10 by pruning rule 1. In Section 4.2, it was shown that the number of possible combinations decreases dramatically when only conflict-free combinations are considered. This rule can be classified as feasibility-based pruning, since a combination that already has conflicts can never become conflict-free, and this branch therefore does not contain a valid solution.

Since costs cannot be negative and are always additive, a combination that is more expensive than the cheapest valid one found so far can be pruned, as it will always remain more expensive. This constitutes a feasibility-based pruning rule, as it detects early that no feasible solution can be found in this subtree. An example of the rule is shown in Figure 10 by pruning rule 2, highlighting the importance of finding an early solution. The earlier a valid solution is found, the earlier it can be pruned based on the costs of this solution.

The last pruning is introduced as a benefit-based upper bound. Since the measures are in order, all measures that are still selectable can be determined based on the level in the search tree. If the conflict groups used so far are also known, the maximum sum of the benefits of the measures that are conflict-free with the current combination at the specified level can be determined. This sum, added to the current benefit, yields the maximum benefit still achievable with conflict-free measures. If this maximum value is less than the target benefit, pruning can be performed. An example of the rule is shown in Figure 10 by pruning rule 3. Here, measure R with a benefit of 10 is not sufficient to exceed the target benefit of 15, and the entire tree no longer needs to be pursued.

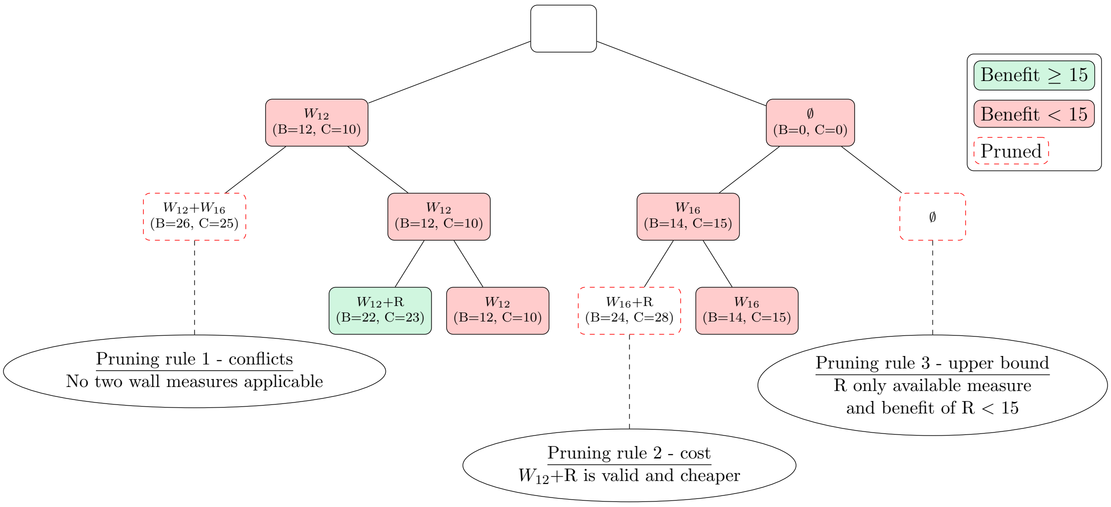


Figure 10: Pruned search tree generated by B &B with binary branching, including benefit (B) and cost (C) values for each node, with order W_{12} (wall insulation with 12 cm), W_{16} (wall insulation with 16 cm), and R (roof insulation) under target benefit 15.

4.3.3. Validation-Driven Branching

The partial additivity of $\mathcal{F}$ for some residential buildings has often been emphasized. The classical B&B algorithm described above assumes complete additivity, as the benefit of a new measure is simply added when generating the left subproblem. However, the tests in Section 3.2.1 show that this additivity is not strictly given. Especially at greater depths, the effects of non-additivity become more pronounced, as shown in Figure 11. Even if a valid solution is found during traversal by summing the benefits, it is not validated with the thermal model, but only after termination. If the solution is invalid, the *UniversalSolution* is returned as a fallback. Even with a valid solution, the algorithm may exclude some minimal solutions because of non-additive effects, which compromises optimality guarantees.

This chapter describes a slightly modified variant of the B&B algorithm presented in Section 4.3.2. This variant attempts to utilise basic additivity, but still performs genuine model evaluation where necessary in order to compensate the non-additivity effects.

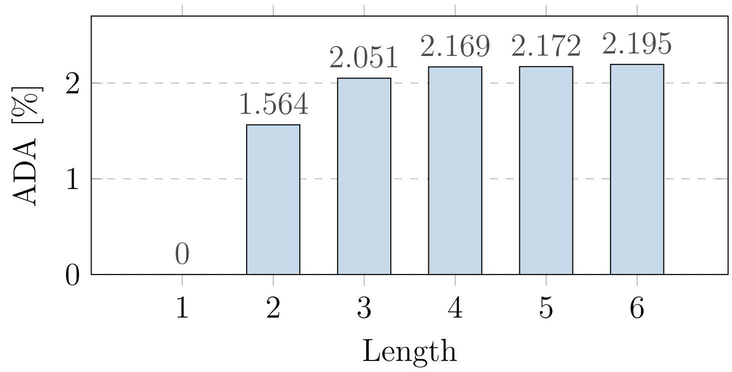


Figure 11: Average deviation of the sum of benefits from the actual benefit (ADA) for length of combination of measures evaluated on SFB 11 (2018) in %.

Since the biggest error occurs during branching, the branching must be adjusted. The simplest approach is to evaluate the model every time a measure is added. Yet this causes the number of model evaluations to explode, and thus also the runtime, since the model must be evaluated on every second branching step (i.e., whenever a measure is added). Ultimately, this results in an exhaustive search. Therefore, rules are needed to determine when to evaluate during branching and when not to. However, evaluation should only be carried out at important points to guarantee a feasible runtime, so the algorithm uses the real model for branching when:

(i) The measure is part of a pair with a relative deviation $\geq 1\%$.

(ii) The size of the combination is greater than 5.

In Section 3.2.1, the procedure for analysing additivity was presented. It determines the deviation of the sum of benefits of all measure pairs on the thermal model. This allows these pairs with a deviation greater than 1 % to be identified. If the size of the combination of measures exceeds 5, the real model is also evaluated, as according to Figure 11, non-additive effects do occur here often.

In order to find only valid solutions, validation-driven branching checks whether a solution is valid on the real thermal model each time a provisional solution is found after summing the benefits. Consequently, the algorithm only determines combinations of measures that are actually valid.

4.3.4. Iterative Evaluation

Validation-driven branching is a promising way to compensate for non-additive effects, but it incurs a correspondingly higher runtime due to it´s frequent model evaluation. The aim of the algorithm presented in this section is to use an iterative approach that uses B&B, but still finds valid and cheap solutions very quickly. The underlying idea relies on the assumption that $\mathcal{F}$ is so additive for arbitrary residential buildings that if B&B found an invalid solution, the invalid solution very narrowly misses the target benefit. According to this assumption, the optimal solution is not very 'far' from the one found. Table 10 supports that, as there are generally only a few cases in which an invalid solution is returned. Moreover, these invalid solutions are relatively close to the target value, with an average never greater than 0.3 %. Therefore, the B&B algorithm can simply be called up again, but with modified parameters. Figure 12 shows the typical sequence of a process. Clearly the critical step is adjusting the input parameters of the B&B algorithm.

Test Case	**Success Rate** [%]	**Mean Invalid Distance** [%]
AB 5 (2018)	87.5	0.219
AB 6 (2018)	92.5	0.134
AB 7 (2018)	85	0.232
AB 8 (2018)	85	0.254
AB 11 (2018)	90	0.188
SFB 5 (2018)	95	0.187
SFB 6 (2018)	100	–
SFB 7 (2018)	100	–
SFB 11 (2018)	100	–

Table 10: Evaluation of B&B results after one iteration with 40 target values. The success rate denotes the share of valid solutions, while the mean invalid distance gives the average deviation from the target value in %.

In each iteration step, there are three updated variables. The central variable is M_{used}, which contains the iteratively developed measure selection. b describes the benefit to be achieved and F_i the current primary energy demand. As an initialization, no measures have been selected yet, and the target benefit must be achieved under the initial primary energy demand $\mathcal{F}(B, \emptyset)$. Then the normal B&B algorithm is executed. If the solution is not valid, the found solution is stored in M_{used} and the new target benefit is calculated using the solution. Similarly, F_i is set.

To generate the new input for B&B, a distinction is made for each measure j where either j does not conflict with M_{used} or j conflicts with M_{used}. If j does not conflict with M_{used}, the primary energy demand of the building with M_{used} in combination with j is calculated and used to determine the benefit under F_i,

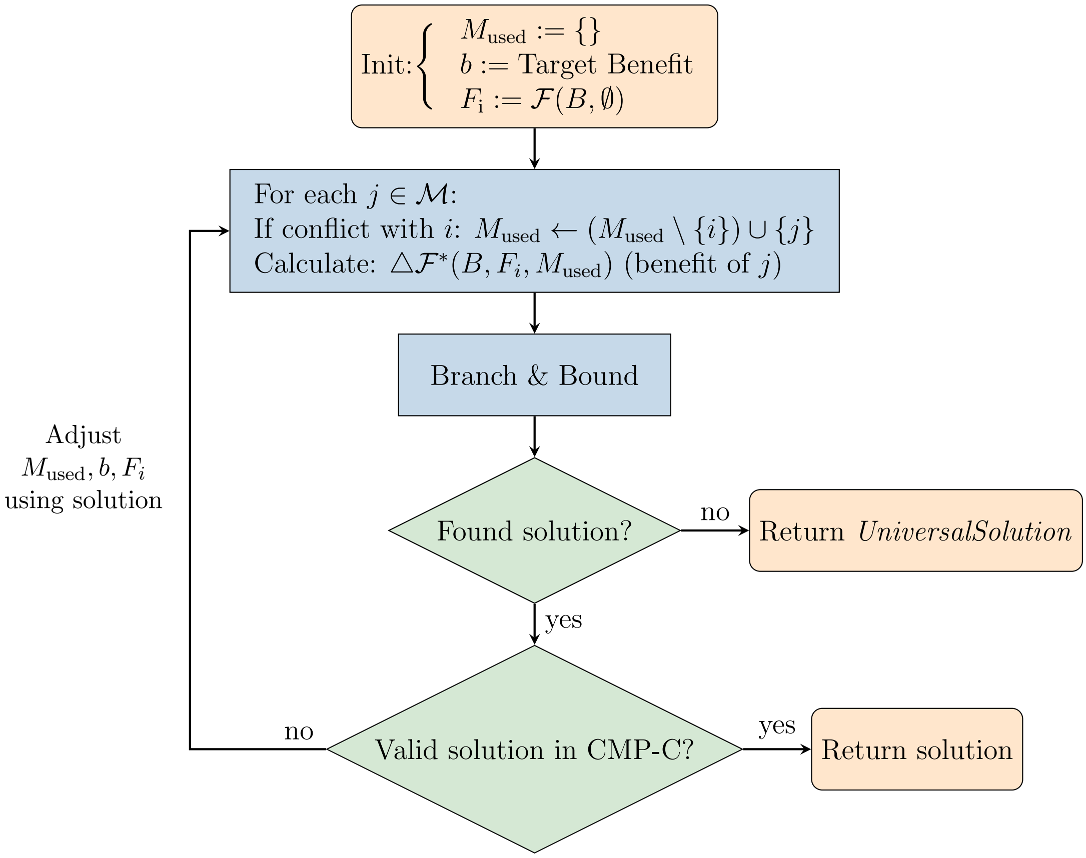


Figure 12: Iterative Evaluation B&B flowchart.

defined as

$$\triangle\mathcal{F}^*(B, F_i, M) := \frac{F_i - \mathcal{F}(B, M)}{F_i} \cdot 100 \ . \tag{30}$$

This defines the new benefit of j. The costs are defined as the costs of M_{used} added to the costs of j.

As established in Section 3.1, each measure belongs to exactly one conflict group, a measure can only conflict with a single measure in M_{used}. If this is the case, the conflicting measure in M_{used} is replaced by j and the benefit is calculated under F_i with (30). The costs are calculated as the current costs of M_{used} minus the original costs of M_{used}. If the replaced measure in M_{used} was better and cheaper than j, this can result in negative benefits and also negative costs. This condition implies the incorrectness of some pruning rules defined in Section 4.3.2. For example, the cost bound becomes invalid, as it is possible to reduce the cost of a combination by taking measures with negative cost. The benefit upper bound must be adjusted, but can still be used. Meanwhile, the termination condition and the conflict pruning rule remain unchanged. With the presented procedure, valid solutions can be found efficiently through iteration with the help of B&B.

4.4. Genetic Optimization Algorithm

This work focuses on deterministic algorithms. However, non-deterministic algorithms play an important role in optimization. For the sake of completeness, a genetic optimization algorithm (GA) was used to test its performance on CMP-C (27). Because of the limited scope the basic idea is outlined roughly, as theoretical details and the implementation of the GA cannot be discussed in depth here.

A major advantage of GAs is that they have been developed for black box optimization, meaning that no mathematical properties or formulation of the object-/constraint functions are necessary, as is the case with $\mathcal{F}$. If future amendments to DIN V 18599 change the additivity of $\mathcal{F}$, GAs will provide a good replacement. GAs use the Darwinian theory of evolution to find the approximated smallest solution [2]. While optimizing, the algorithm runs through several iterations in which it evolves a set of solutions, called population, until a termination criterion, such as a maximum number of generations, is reached.

This procedure is realized in six characteristic steps, as outlined in [2]. In the first step, the initial population is generated, while the size and diversity of the initial solution set are hyperparameterized. In the GA used in this work, the initial population is selected as a mixture of various solutions found by B&B, the *UniversalSolution*, and individual measures, with a total population size of 80. In the second step, the fitness value is determined for each solution, calculated via the fitness function. Specifically, this function discards combinations that have conflicts or that are below the target primary energy demand due to very low function values. In the case of a conflict-free combination that exceeds the target value, the negative costs are simply taken into account in order to minimize the costs of valid solutions. Continuing with step 3, the best solutions are selected (where the fitness function is high), and the rest is discarded. In step 4, features of the selected solutions are recombined to create new solutions. In step 5, new aspects of solutions are introduced by modifying random features of some solutions. The last step is to check whether the solutions have the required quality. If this is the case, the current solutions set is outputted. Otherwise, the algorithm jumps back to step 2. Since there is no known optimum for CMP-C, there is also no required quality, and the algorithm terminates after a certain number of iterations (also called generations), here 50, and returns the best solution found. Since *UniversalSolution* is contained in the initial population, it is output unless a better valid solution is found.

5. Benchmarking and Evaluation

In Section 4, five algorithms were presented conceptually, namely exhaustive search, standard B&B, B&B with validation-driven branching, B&B with iterative evaluation, and a genetic optimization algorithm. Possible advantages and disadvantages were highlighted. The aim of this section is to benchmark the presented algorithms based on their performance in solving CMP-C (27) and, if possible, to determine the best algorithm. To do this, the general test environment and the performance metrics must be clarified first. Then the tests are performed and evaluated.

5.1. Benchmark Setup

In order to make the performance values of algorithms on CMP-C comparable, a unified basis is created. Consequently, the same instances of CMP-C are taken, and the comparison of performance only takes place between algorithms on the same instance. Instances of CMP-C are described by the specific configuration of the thermal model $B \in \mathcal{B}$, the target value b, the total set of available measures $\mathcal{M}$, and the resulting conflict groups L. Strictly speaking, the instances are also described by the implementation of $\mathcal{G}$, but here, as before, the implementation is always assumed by $\mathcal{F}$. Even if one algorithm appears to be superior to all others, it is necessary to examine how close this algorithm is to the optimum. For this reason, reference values are generated using exhaustive search. The instantiations of all these variables are described below.

For this work, two different sets of measures are generated. The measure catalog $\mathcal{M}_{\text{all}}$ consists of all available measures shown in Table 11. $\mathcal{M}_{\text{small}}$ consists of a slimmed-down version with only a few measures, see Table 12. It was created by randomly omitting various groups of measures from $\mathcal{M}_{\text{all}}$. For example, many insulation materials or procedures are no longer available, and the entire roof insulation was omitted. The reason for the distinction is that the number of possible measures has a big influence on the number of possible combinations and thus the length of the runtime. Since the reference values are generated using exhaustive search and, according to Section 4.2, this becomes computationally very expensive on $\mathcal{M}_{\text{all}}$, the smaller measure set $\mathcal{M}_{\text{small}}$ is needed to generate these values efficiently, as it is only one-third the size of $\mathcal{M}_{\text{all}}$. It should be noted that some of these measures require certain building conditions in order to be applied, such as the presence of a roof for roof insulation. Consequently, the number may differ across test cases but remains fixed within a given case.

Another important differentiation is in which building the primary energy demand is optimized, i.e., which configuration of the thermal model is used. There are a total of ten test cases from 2018, described by Ricken [33]. In order to obtain data that allows a strong conclusion about the performance of the algorithms,

Component	Type	Materials	Thickness [cm]	No. of Measures
Base Plate	Ins. Below	EPS, MW, PUR, WF, XPS	8, 10, 12	15
	Ins. Below With Cladding	GB, WFC	8, 10, 12, 14	8
	Ins. Above	BMW, BWF, CEL, FGG	5, 6, 9	12
Roof	Above Rafters	EPS, PUR, WF	13, 17, 20, 23	12
	Between Rafters	GW, MW, WF	13, 17, 20, 23	12
Wall	Core Insulation	AER, EPSG, MW, PER	6, 8, 9	12
	ETICS	EPS, MW, PUR, WF, XPS	12, 14, 16, 18	20
Door	–	FG, SH, SI, WO, HFG, HSI, HW	–	7
Roof Window	–	2G, 3G, PHG	–	3
Window	–	2G, 3G, PHG	–	3
			Total:	**104**

Table 11: Overview of measure catalogue $\mathcal{M}_{\text{all}}$. For the meaning of the abbreviation of insulation materials, see Table 14.

Component	Type	Materials	Thickness [cm]	Measures
Base Plate	Ins. Below	EPS, PUR	8, 10, 12	6
	Ins. Below With Cladding	GB	8, 10, 12, 14	4
Wall	ETICS	PUR, XPS	12, 14, 16, 18	8
Door	–	FG, HFG, HSI, HW, SH, SI, WO	–	7
Roof Window	–	2G, 3G, PHG	–	3
Window	–	2G, 3G, PHG3	–	3
			Total:	**31**

Table 12: Overview of reduced measure catalogue $\mathcal{M}_{\text{small}}$. For the meaning of the abbreviation of insulation materials, see Table 14.

more than ten instances are needed (one target value per test case). To generate multiple target values for the different test cases, the *UniversalSolution* is used. With this set, the best possible primary energy demand can be determined by $\mathcal{F}(B, \textit{UniversalSolution})$. The initial demand $\mathcal{F}(B, \emptyset)$ and $\mathcal{F}(B, \textit{UniversalSolution})$ span an interval within optimization can be performed. From this interval, n values with equal distance to the neighbors are taken, which are considered as the target demands. For this work, $n = 40$ is set, which leads to 400 instances of CMP-C regardless of the two different sets of measures.

5.2. Evaluation Metrics

Just as there are quality features for products, there are also quality features for algorithms. Consequently, the term 'performance' of an algorithm can be interpreted in many ways.

5.2.1. Number of Model Evaluations

The first association most people have with performance is the runtime of the algorithm. The runtime of an algorithm is defined as the time it takes for the algorithm to terminate. The deterministic algorithms have fixed worst-case runtimes, and the runtime does not change when repeatedly applied to the same instance. Although the GA is not deterministic, under the assumption described in Section 4.4, it always runs for the same length of time.

The problem is the unit of measurement for time. If the time is measured from the start of the algorithm to its termination, environmental variables such as computationally intensive processes can lead to insufficient CPU time and prolong the runtime. Even the time on the CPU alone is not very meaningful, as the values obtained in this way are difficult to reproduce. If, in the future, another algorithm solves CMP-C faster than the algorithms presented here, it is not clear whether it is really faster or was simply executed with more computing power.

A suitable metric for the runtime is the number of model evaluations (NME), evaluated in [13]. As described in Section 2.2.2, the calculation of the new primary energy demand takes a long time compared to other typical operations. It is the most expensive operation in the entire optimization. This calculation is presented by $\mathcal{F}$. The number of evaluations of $\mathcal{F}$ therefore describes a kind of structural runtime and correlates with the actual runtime. So if the number of evaluations of $\mathcal{F}$ is minimized, the runtime is also minimized. The advantage of this metric over runtime in milliseconds, etc., is that it is easily reproducible regardless of the hardware. Consequently, for each of the 40 target values for each test case, the number of evaluations $\mathcal{F}$ is recorded.

5.2.2. Solution Quality

All algorithms return by definition a conflict-free set of measures that meet the target demand. This behavior ensures the correctness of all solutions that are produced and thus also the correctness of all algorithms. The minimal cost of the solutions reflects the quality of the solution and thus also of the algorithms on the specific instance. The quality of an algorithm is considered more important than the runtime because a fast algorithm that rarely finds the optimum is less useful in practice than a slower one that, however, finds the optimum with a high degree of accuracy.

When comparing two algorithms, a direct statement about quality can be made by comparing the costs of the solutions, whereby lower costs naturally mean better quality. This simple comparison finds the algorithm with the best quality, but does not make any statement about its quality in the context of the optimum. Even the best-performing algorithm determined in this manner may produce solutions with excessively high costs. The optimum is needed to assess this quality aspect. The difference between the costs and the minimum possible costs of a valid solution is the quality measure used in this work. Since it is a minimum, this value is always greater than or equal to zero. It is easy to see that algorithms evaluated with this measure are also comparable with a normal inequality operation, since the minimum cancels out. Consequently, the difference to the optimum is recorded for each of the 40 target values for each test case. In the case of the GA, to compensate for the non-deterministic nature, the algorithm was executed 50 times, and the best solution was taken.

5.3. Results

In the following, the presented algorithms will be compared with each other using the presented metrics. The following abbreviations will be used for the different algorithms: *B&B* refers to the standard B&B algorithm, *B&B Validation* refers to B&B with validation-driven branching, *B&B Iterative* refers to B&B with iterative evaluation, and *Genetic* refers to the genetic optimization algorithm.

5.3.1. Number of Model Evaluations per Algorithm

This section presents and classifies the results of the number of model evaluation of the individual algorithms. Since exhaustive search can only be used on $\mathcal{M}_{\text{small}}$, the runtime of the algorithms including exhaustive search on the 400 instances under $\mathcal{M}_{\text{small}}$ is first analyzed. Then, the runtime of the algorithms without exhaustive search on $\mathcal{M}_{\text{all}}$ is presented.

Figure 13 shows the means of the number of function evaluations for each building for each algorithm. The index denotes the position within the interval between the current and the best primary energy demand, with higher indices corresponding to more challenging targets. The standard B&B algorithm, with an average of approximately 32 model evaluations, is by far the fastest and shows consistent performance across all indices. This is because it only has to evaluate the benefits of all measures once. This is followed by B&B with iterative evaluation, which requires more evaluation on average in only a few cases, as one iteration is not sufficient here. B&B with validation-driven branching requires significantly more model evaluation, which also increases with increasing complexity. Nevertheless, this is significantly more efficient than exhaustive search, which requires an extremely large number of evaluations, especially in cases of higher complexity. The genetic algorithm represents a special case here, as it has a fixed population size by definition and runs a fixed number of generations (iterations). The model evaluations are therefore independent of the number of measures or the solutions developed during the iteration.

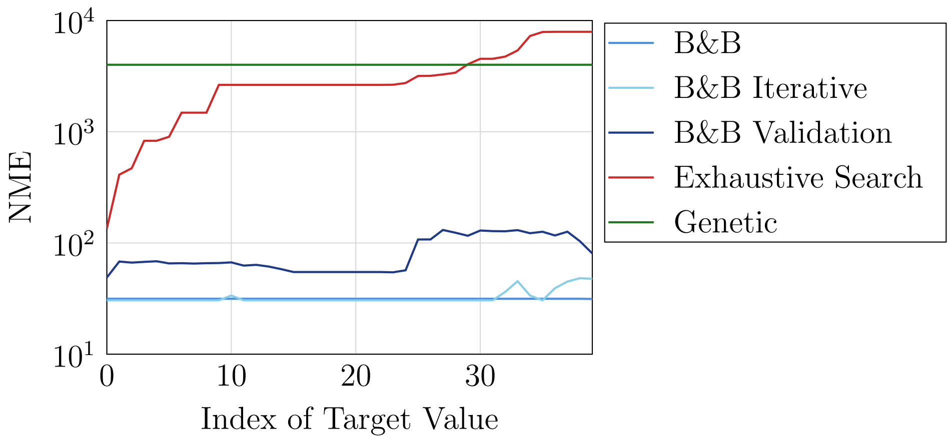


Figure 13: Mean of NME for each index for each algorithm on each test case on a logarithmic scale. The results are based on the 400 problem instances generated according to Section 5.1 with $\mathcal{M}_{\text{small}}$.

The results from Figure 13 can also be applied to the runtimes under $\mathcal{M}_{\text{all}}$. Figure 14 shows these runtimes. It can be observed that B&B with validation-driven branching requires significantly more model evaluation. Standard B&B and B&B with iterative evaluation have scaled linearly with the increased number of measures. According to the principle described above, the GA has the same number of model evaluations. The results of the dependence on additivity can be transferred in a trivial manner.

Since three of the algorithms analysed above are based on the partial additivity of $\mathcal{F}$, the question arises as to what extent non-additivity influences the runtime of the algorithms. By definition, B&B with validation-driven branching is the only algorithm in which model evaluation increases when non-additivity is present. Therefore, with this algorithm, evaluation increases rapidly with increasing non-additivity. All other algorithms do not correlate with the non-additivity of $\mathcal{F}$.

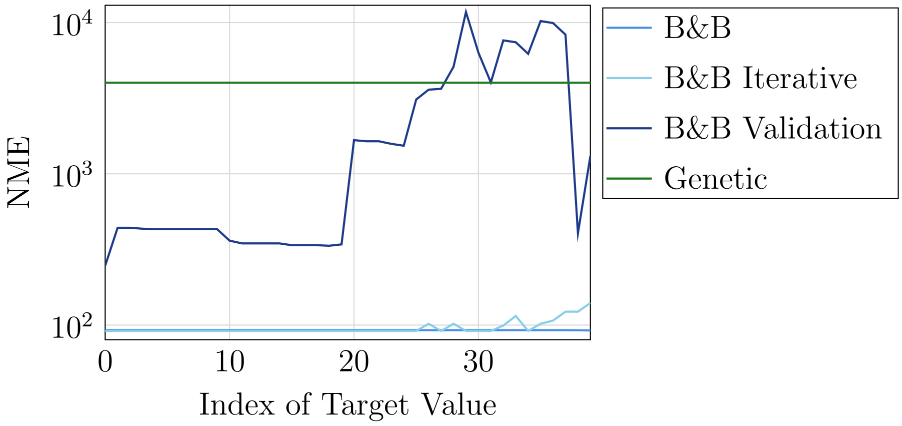


Figure 14: Mean of NME for each index for each algorithm on each test case on a logarithmic scale. The results are based on the 400 problem instances generated according to Section 5.1 with $\mathcal{M}_{\text{all}}$.

In conclusion, it can be said that only the runtime of exhaustive search and B&B with validation-driven branching suffers from the increase in complexity and number of measures. These show significantly higher runtimes in all tests. B&B with iterative evaluation and especially standard B&B remain robust under this increase and can be considered as most efficient in terms of runtime. The efficiency of the GA, in relation to the others, depends on the number of measures. While it performs relative poorly for small measure sets, the algorithm shows good performance for larger ones.

5.3.2. Solution Quality per Algorithm

In order to evaluate and compare the quality of the presented algorithms, the same approach as in Section 5.3.1 is followed. First, the 400 instances of CMP-C are generated over the measure set $\mathcal{M}_{\text{small}}$, and then the solution of the algorithms is evaluated in terms of quality using the reference values from exhaustive search. After that, a quality analysis based on $\mathcal{M}_{\text{all}}$ is evaluated to observe the behavior in real-world environments.

Table 13 shows the average deviation from the reference value across all instances under $\mathcal{M}_{\text{small}}$. It is noticeable that B&B with validation-driven branching does not achieve the optimum in only 2.75% of cases, with a maximum deviation of approximately 5245€. This is clearly the best result compared to the other algorithms. Although GA more often achieves the optimum than B&B with iterative evaluation, it deviates by larger amounts and also has a higher maximum deviation. Standard B&B is by far the worst in all categories.

Figure 15 provides a detailed breakdown of the cost deviations. This figure, com-

Algorithm	**Max. Deviation [€]**	**Mean Deviation [€]**	**Suboptimal Cases [%]**
B&B	52146.27	1929.14	11
Genetic	19853.76	397.88	5.50
B&B Iterative	15629.02	317.86	6.75
B&B Validation	5244.60	50.58	2.75

Table 13: Comparison of algorithm deviations against the reference values. The results are based on the 400 problem instances generated according to Section 5.1 with $\mathcal{M}_{\text{small}}$.

bined with Table 7, provides an explanation for the poor quality of the solution in specific test cases for the respective algorithms. For example, it can be seen that standard B&B does not find high-quality solutions when there is a very high proportion of negative deviations in this test case, see Table 7. This can be explained by the fact that in these test cases, the sums of benefits are always smaller than the real benefit. As a result, invalid solutions are found more frequently, and the valid but very expensive solution *UniversalSolution* must be returned. A very surprising result is the average solution quality in the SFB 11 (2018) test case. This appears to be the least additive test case, yet B&B seems to perform relatively well here. This is because the sum of the benefits is worse than the actual benefit. Hence, the algorithm always finds valid, not significantly optimal, solutions, but *UniversalSolution* is never returned. In Table 7, there are also test cases where the distribution of the deviation is very even. This partially cancels out the effect described above, and B&B delivers relatively high-quality results.

Since the GA is partially initialized with solutions found by B&B, these phenomena are transferred. B&B with iterative evaluation can then still compensate for these effects through iterations, but as can be seen, not completely. Only B&B with validation-driven branching can almost completely eliminate these effects, although it also has problems with very large non-additivity.

However, when considering the GA and the B&B with iterative evaluation algorithm, Table 13 and Figure 15 do not provide a clear quality order. Therefore, a framework is needed that provides a clear classification of the quality of the algorithms without making it dependent on user observations. Performance profiles provide such a method.

Performance profiles, introduced by Dolan and Moré [11], are a standard tool for benchmarking optimization algorithms. For each problem instance p and algorithm a, the performance ratio is computed by

$$r_{p,a} = \frac{c_{p,a}}{\min_j c_{p,j}},$$

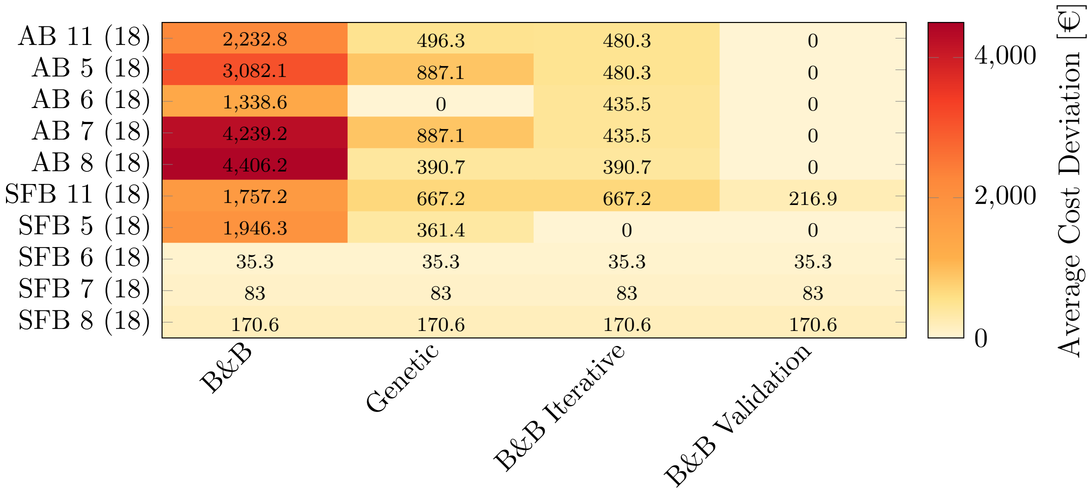


Figure 15: The average cost deviation from the reference costs, generated with exhaustive search, per test case, which are split into single-family buildings (SFB) and apartment buildings (AB). The results are based on the 400 problem instances generated according to Section 5.1 with $\mathcal{M}_{\text{small}}$.

where $c_{p,a}$ denotes the cost obtained by algorithm a on problem instance $p \in P$ and the denominator is the best cost achieved across all algorithms for the same problem instance. The performance profile of an algorithm a is then defined as the cumulative distribution function

$$\rho_a(\tau) = \frac{1}{|P|} \left|\{p \in P : r_{p,a} \leq \tau\}\right|,$$

which gives the fraction of problems for which algorithm a achieves a cost within a factor τ of the best cost. Plotting $\rho_a(\tau)$ against τ yields a step function that summarizes solver performance: curves that rise more quickly and reach higher values indicate more competitive and reliable algorithms.

This defines clear rules for classifying one algorithm as more qualitative than another. The algorithm whose profile is always greater than or equal to the profiles below it dominate these and can be considered to be of better quality in every case.

Figure 16 shows such performance profiles for all algorithms except exhaustive search, as this always finds the best solution. As expected, B&B with validation-driven branching dominates all other algorithms. It is clear that B&B with iterative evaluation dominates the GA. Standard B&B is the worst in terms of quality.

These results can also be transferred to the evaluation with $\mathcal{M}_{\text{all}}$. The performance profiles in Figure 17 show the same quality hierarchy using $\mathcal{M}_{\text{all}}$. Therefore, the generated costs of B&B with validation-driven branching can be used as reference values. Figure 18 shows the evaluation with these reference values. Here, the

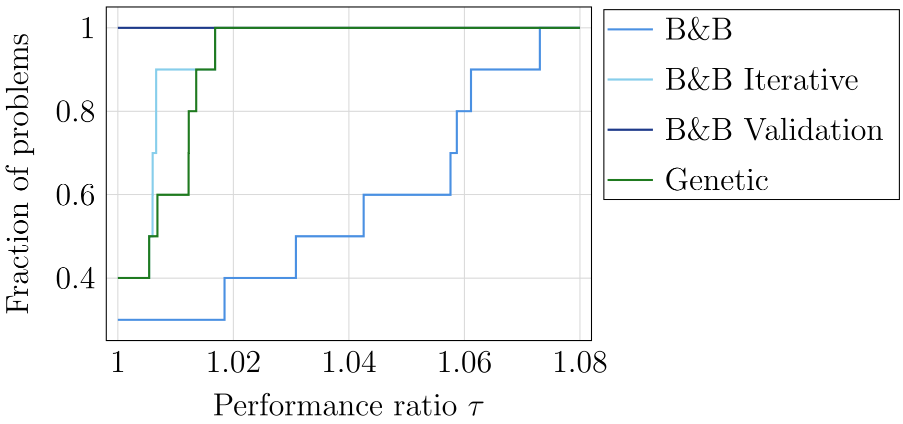


Figure 16: Performance profile of the algorithms for $\tau \in [1, 1.08]$. The results are based on the 400 problem instances generated according to Section 5.1 with $\mathcal{M}_{\text{small}}$.

clear dominance of B&B with iterative evaluation over all other algorithms, except B&B with validation-driven branching, is clearly evident, as it is always closer to the results of B&B with validation-driven branching.

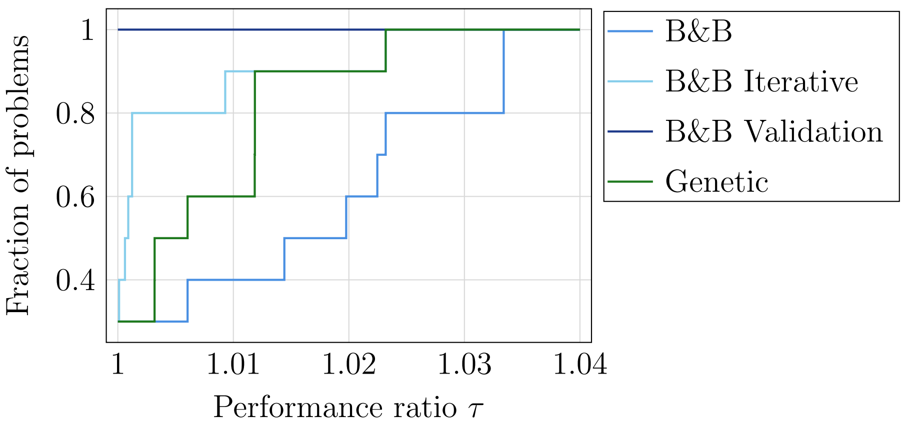


Figure 17: Performance profile of the algorithms for $\tau \in [1, 1.04]$. The results are based on the 400 problem instances generated according to Section 5.1 with $\mathcal{M}_{\text{all}}$.

In conclusion, it can be said that in terms of quality, B&B with validation-driven branching is by far the best algorithm, apart from exhaustive search. The second-best algorithm is B&B with iterative evaluation. Together with the results from Section 5.3.1, the algorithm provides a reasonable trade-off between computational efficiency and solution quality. Standard B&B and the GA should not be used, as they offer no significant advantages in the current scenario.

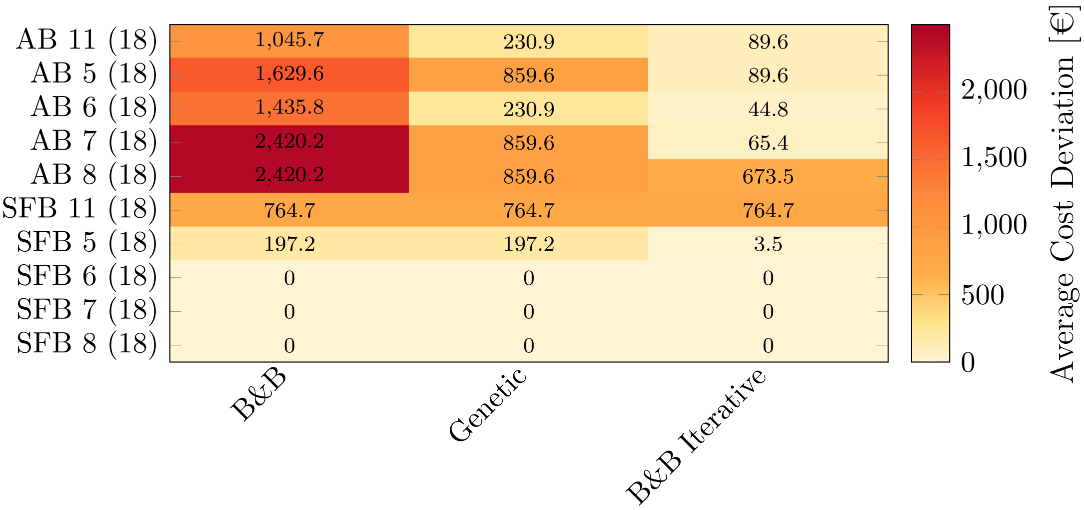


Figure 18: The average cost deviation from the reference costs, generated with B&B with validation-driven branching, per test case. The test cases are separated into single-family buildings (SFB) and apartment buildings (AB). The results are based on the 400 problem instances generated according to Section 5.1 with $\mathcal{M}_{\text{all}}$.

6. Conclusion and Outlook

This thesis investigated model-based optimization of the building envelope with the goal of improving the energy efficiency of residential buildings while keeping energy-efficient renovation costs as low as possible. To this end, a thermal simulation model compliant with DIN V 18599 was implemented, extended with a catalogue of construction measures, and coupled with different optimization algorithms.

Results showed that it is possible to represent the relevant measures (walls, roofs, base plates, windows, and doors) in a unified optimization model by abstracting their characteristics into cost and benefit functions as well as conflict groups. This allowed a formal problem formulation as *Cost Minimization Problem with Conflicts* (CMP-C). The analysis confirmed that CMP-C is NP-hard, which implies that no polynomial-time algorithm is known for solving it in general. However, in most cases, practical instances can still be solved efficiently using dedicated optimization algorithms.

In this work, five such algorithms were presented and compared: exhaustive search, three variants of Branch-and-Bound that use additive relaxation, and a genetic algorithm. The benchmark study across multiple building instances highlighted clear differences:

- Exhaustive search serves as a baseline because it guarantees to find the global optimum, but is computationally infeasible except for very small problem sizes.

- B&B variants showed a remarkable balance between runtime and solution quality. In particular, the B&B with validation-driven branching frequently achieved solutions that matched the exhaustive search optimum, while the B&B with iterative evaluation produced near-optimal results with consistently low runtime.

- The genetic algorithm confirmed its exploratory capabilities, occasionally produced competitive solutions, but exhibited larger variance and higher computational cost due to its preset runtime.

Overall, the evaluation via performance profiles demonstrated that the B&B variants clearly dominated in terms of quality and runtime, providing deterministic and reproducible results with practical efficiency.

Another insight is the additivity property of $\mathcal{F}$ directly influencing both the runtime and the quality of the algorithms that utilise additive relaxation. The distribution and the strength of the deviations play an important role here.

In summary, this work provides a formal optimization framework and demonstrates that exact algorithms, especially tailored Branch-and-Bound variants, can offer practically efficient and reproducible solutions for cost-optimal energy-efficient building renovation.

6.1. Future Work

Several avenues for future research emerge from this work.

First, deviations can be seen in the evaluation of the accuracy of the thermal model. Although these are within one percent and thus meet the requirements of the *18599 Gütegemeinschaft*, they can be made even more accurate in order to obtain more precise results during optimization. This can be achieved through more diverse test cases with many configuration variants, such as not only square floor areas, but also more differences in the heating system configuration.

A natural extension is to broaden the scope of measures. Including heating and cooling systems, more insulation materials, and procedures would yield a more holistic optimization. This requires finding new lambda values and permissible thicknesses, as well as new cost functions, with minimal need for extrapolation methods.

This work describes single-solution algorithms that solve CMP-C. Consequently, the economically optimal solution found by the algorithm is returned to the user.

Since the construction measures also cause external/internal changes to the building, some measures may not be acceptable to the user. Therefore, a further requirement is to find as many diverse solutions as possible, which are not all optimal but are close to the optimum. The user can then choose the solution that best suits their needs. A similar approach is to exclude from the beginning any measures that alter unwanted components.

Finally, optimization approaches other than B&B can be considered, although these would primarily only improve runtime and not quality.

A. Abbreviations of Insulation Materials

Abbr.	**Material**	**Abbr.**	**Material**
AER	Aerogel	HSI	Half Lite Steel Insulated
BMW	Blown Mineral Wool	HW	Half Lite Wood
BWF	Blown Wood Fiber	MW	Mineral Wool
CEL	Cellulose	PER	Perlite
EPS	Expanded Polystyrene	PH	Passive House
EPSG	EPS Granulate	PUR	Polyurethane
FG	Fiberglass	SH	Steel Hollow
FGG	Foam Glass Gravel	SI	Steel Insulated
GB	Gypsum Board	WF	Wood Fiber
GW	Glass Wool	WFC	Wood Fiber Cellar
HFG	Half Lite Fiberglass	WO	Wood
		XPS	Extruded Polystyrene

Table 14: Insulation material abbreviations

References

[1] J. L. Aguilar-Santana, H. Jarimi, M. Velasco-Carrasco, and S. Riffat. Review on window-glazing technologies and future prospects. *International Journal of Low-Carbon Technologies*, 15(1):112–120, December 2019. ISSN 1748-1325. doi: 10.1093/ijlct/ctz032.

[2] T. Back and H.-P. Schwefel. Evolutionary computation: an overview. In *Proceedings of IEEE International Conference on Evolutionary Computation*, ICEC-96, pages 20–29. IEEE, 1996. doi: 10.1109/icec.1996.542329.

[3] P. Benner, L. Feng, and E. B. Rudnyi. Using the superposition property for model reduction of linear systems with a large number of inputs. In *Proceedings of the 18th International Symposium on Mathematical Theory of Networks & Systems*, Blacksburg, VA, USA, 2008.

[4] V. Cacchiani, M. Iori, A. Locatelli, and S. Martello. Knapsack problems — an overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems. *Computers & Operations Research*, 143:105693, July 2022. ISSN 0305-0548. doi: 10.1016/j.cor.2021.105693.

[5] C. Christensen, S. Horowitz, T. Givler, A. Courtney, and G. Barker. BEopt: Software for identifying optimal building designs on the path to zero net energy. In *Proceedings of ISES 2005 Solar World Congress*, Orlando, FL, USA, August 2005.

[6] D. B. Crawley, L. K. Lawrie, F. C. Winkelmann, W. F. Buhl, Y. J. Huang, C. O. Pedersen, R. K. Strand, R. J. Liesen, D. E. Fisher, M. J. Witte, and J. Glazer. EnergyPlus: Creating a new-generation building energy simulation program. *Energy and Buildings*, 33(4):319–331, April 2001. ISSN 0378-7788. doi: 10.1016/s0378-7788(00)00114-6.

[7] S. Czernik, M. Marcinek, B. Michałowski, M. Piasecki, J. Tomaszewska, and J. Michalak. Environmental footprint of cementitious adhesives—components of ETICS. *Sustainability*, 12(21):8998, October 2020. ISSN 2071-1050. doi: 10.3390/su12218998.

[8] Deutsches Institut für Normung e.V. *DIN V 4108-6: Wärmeschutz und Energie-Einsparung in Gebäuden – Berechnungsverfahren zur Bestimmung des Jahresheizwärme- und des Jahresheizenergiebedarfs*. Beuth Verlag GmbH, June 2003. doi: 10.31030/9493310.

[9] Deutsches Institut für Normung e.V. *DIN V 4701-10: Energetische Bewertung heiz- und raumlufttechnischer Anlagen – Teil 10: Heizung, Trinkwassererwärmung und Lüftung von Wohngebäuden*. Beuth Verlag GmbH, August 2003. doi: 10.31030/9496133.

[10] Deutsches Institut für Normung e.V. *DIN V 18599: Energetische Bewertung von Gebäuden - Berechnung des Nutz-, End- und Primärenergiebedarfs für Heizung, Kühlung, Lüftung, Trinkwarmwasser und Beleuchtung*. Beuth Verlag GmbH, September 2019. doi: 10.31030/2874317.

[11] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, January 2002. ISSN 1436-4646. doi: 10.1007/s101070100263.

[12] S. Dong. Methods for constrained optimization. Master's thesis, Massachusetts Institute of Technology, August 2006.

[13] A.E. Eiben and M. Jelasity. A critical note on experimental research methodology in ec. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 1 of *CEC-02*, page 582–587. IEEE, 2002. doi: 10.1109/cec.2002.1006991.

[14] N. Feiereisen. Energetische Bilanzierung von Gebäuden. Bachelor thesis, RWTH Aachen University, January 2023.

[15] K.-H. Fingerling, W. Feist, J. Otte, and R. Pfluger. Das kostengünstige mehrgeschossige Passivhaus in verdichteter Bauweise. Teil 1 des Abschlußberichtes: Konstruktionshandbuch für Passivhäuser. Final report, Passivhaus Institut, Darmstadt, Germany, 2000.

[16] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, volume 29, chapter 1, pages 1–17. W. H. Freeman & Co., 2002. ISBN 0716710447.

[17] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, volume 29, chapter 2, pages 17–45. W. H. Freeman & Co., 2002. ISBN 0716710447.

[18] D. M. Hamby. A review of techniques for parameter sensitivity analysis of environmental models. *Environmental Monitoring and Assessment*, 32(2): 135–154, September 1994. ISSN 1573-2959. doi: 10.1007/bf00547132.

[19] S. Himburg. Energetische Bilanzierung von Wohngebäuden nach DIN V 18599. *Bauphysik*, 33(2):99–110, 2011. doi: 10.1002/bapi.201110012.

[20] E. Hinz. Kosten energierelevanter Bau- und Anlagenteile bei der energetischen Modernisierung von Altbauten. Final report, Bundesministerium für Umwelt, Naturschutz, Bau und Reaktorsicherheit, Berlin, Germany, 2015.

[21] R. L. Iman and J. C. Helton. An investigation of uncertainty and sensitivity analysis techniques for computer models. *Risk Analysis*, 8(1):71–90, March 1988. ISSN 1539-6924. doi: 10.1111/j.1539-6924.1988.tb01155.x.

[22] International Organization for Standardization. *ISO 10077-1:2017. Thermal performance of windows, doors and shutters – Calculation of thermal transmittance – Part 1: General*. ISO, 2017. doi: 10.31030/3178547.

[23] M. A. Karbasforoushha, M. Khajehzadeh, T. Jearsiripongkul, S. Keawsawasvong, and M. Eslami. A comprehensive review of building energy optimization using metaheuristic algorithms. *Journal of Building Engineering*, 98:111377, December 2024. ISSN 2352-7102. doi: 10.1016/j.jobe.2024.111377.

[24] T. Koch, S. Achenbach, and A. Müller. Anpassung der kostenfunktionen energierelevanter bau- und anlagenteile bei der energetischen modernisierung von altbauten auf das preisniveau 2020. Working paper, Institut Wohnen und Umwelt GmbH (IWU), Darmstadt, Germany, March 2021.

[25] M. Liem. Energy efficient door design. Bachelor's thesis, University of British Columbia, April 2009.

[26] W. L. Loh. On latin hypercube sampling. *Annals of Statistics*, 24(5):2058–2080, 1996. ISSN 0090-5364. doi: 10.1214/aos/1069362310.

[27] A. Magrini, L. Marenco, V. Leoni, and R. Gamba. Air cavity building walls: A discussion on the opportunity of filling insulation to support energy performance improvement strategies. *Energies*, 15(23):8916, November 2022. ISSN 1996-1073. doi: 10.3390/en15238916.

[28] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, February 2016. ISSN 1572-5286. doi: 10.1016/j.disopt.2016.01.005.

[29] Z. E. Msokwa. Fixed basket Laspeyres' method compared to modified Laspeyres' method in computing consumer price indices. 2013.

[30] J. Nievergelt. *Exhaustive Search, Combinatorial Optimization and Enumeration: Exploring the Potential of Raw Computing Power*, page 18–35. Springer Berlin Heidelberg, 2000. ISBN 9783540444114. doi: 10.1007/3-540-44411-4_2.

[31] D. Pickles. Energy efficiency and historic buildings: Insulating pitched roofs at rafter level. Guidance Note HEAG070, Historic England, London, UK, 2016.

[32] D. Pisinger. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research*, 83(2):394–410, June 1995. ISSN 0377-2217. doi: 10.1016/0377-2217(95)00015-i.

[33] B. Ricken. Relaxierte energetische Simulation des Energiebedarfs in Gebäuden. Bachelor thesis, RWTH Aachen University, September 2023.

[34] B. Rodríguez-Soria, J. Domínguez-Hernández, J. M. Pérez-Bella, and J. J. del Coz-Díaz. Review of international regulations governing the thermal insulation requirements of residential buildings and the harmonization of envelope energy loss. *Renewable and Sustainable Energy Reviews*, 34:78–90, June 2014. ISSN 1364-0321. doi: 10.1016/j.rser.2014.03.009.