A Data-Driven Decision Support Tool for a priori Multi-Objective Optimization of Building Portfolio Assets

Preference-Based Decision Support for Sustainable Rooftop Strategies

Master Thesis - CME5200

Fedor Pallandt

A Data-Driven Decision Support Tool for a priori Multi-Objective Optimization of Building Portfolio Assets

Preference-Based Decision Support for Sustainable Rooftop Strategies

by

Fedor Pallandt

Student number:	4904702	
Project duration:	February 2025 – September 2025	
Document type:	Master thesis	
Thesis committee:	Dr.ir. S. van Nederveen,	TU Delft, Chair
	Dr.ir. R. Binnekamp,	TU Delft, First supervisor
	Dr. T. Chatzivasileiadis,	TU Delft, Second supervisor
	G. van den Engel,	Arcadis, Company supervisor

Preface

To fully understand the context of this master thesis project within the Construction Management program, it is important to distinguish between two possible approaches. One option is to conduct pure research and write a full thesis report, while the other is to develop a tool or model that addresses a specific need. This project follows the second approach, focusing on the development of a model rather than extensive scientific research.

However, developing a model still requires identifying a scientific research gap and conducting research to justify its design. While scientific studies have been conducted to ensure the model is well-founded and fit for purpose, the emphasis remains on the creation and implementation of the model, rather than providing an elaborate theoretical explanation of every underlying principle.

Writing this master thesis and developing the decision support tool has been an extremely learning full experience. I am very proud of how this work brings together everything I have learned throughout my academic career. Looking back, I realize how much the process has taught me. A master thesis is essentially a project you manage on your own. As Ruud once told me, you are the project manager of your own project.

Of course, there are academic and company supervisors who bring in their own perspectives and expectations about how the project should be carried out. But in the end, it is up to you to make decisions and steer the project in a direction that not only meets the minimum requirements of all stakeholders involved but also remains meaningful and interesting to yourself.

At the beginning, I was faced with many possibilities for how this tool could be developed. My ideas were still vague, and the abundance of data and options made it complex to determine where to start. I felt that I first needed to do a lot of reading before I could design a concept, and I also believed the modeling had to be correct right from the beginning, which made me cautious in taking the first steps. With no clear case study in the beginning, it was difficult to see what the actual possibilities were. Selecting The Hague as the case municipality, and looking at its portfolio and neighborhoods, was a first start in creating a tangible test case where results and data could be verified. Starting with this case study gave the project a clear direction and made the outcomes more concrete.

The real turning point was realizing the importance of starting very small and simple. A portfolio problem involving hundreds of buildings and neighborhoods can quickly become extremely complex. However, if you reduce the problem to just three buildings and three neighborhoods, one can more easily develop a tool. Starting small enabled to test what kind of output the tool could produce, what could be measured, and which parts required further research.

Another important shift was moving away from a strictly linear process, first research, then tool development, towards a more parallel and iterative approach. I started developing small parts of the tool and different prototypes early on, and whenever knowledge gaps appeared, I conducted research to address them. This back-and-forth process allowed me to progress more effectively and learn along the way. These two lessons, start small rather than waiting until everything is clear, and treat research and development as parallel rather than sequential, were key in helping me converge, accelerate, and bring this project to completion.

And to you, the reader: I hope you enjoy reading this thesis. I am always happy to answer questions or to have a conversation about this work or any related topic.

Fedor Pallandt
Delft, September 2025

Acknowledgements

I would like to express my gratitude to everyone who supported me during this master thesis.

First, I would like to thank *Sander van Nederveen* for taking on the role of chair, ensuring that my graduation committee was formally complete, and for your flexibility in this informal role.

I am very grateful to *Theodoros Chatzivasileiadis* for your continuous enthusiasm and flexibility from the very beginning. Your course in Spatial Data Science increased my curiosity in data science and inspired me to pursue this direction in my thesis.

I owe special thanks to *Ruud Binnekamp* for always being quick to respond, open to answering my questions, and guiding me whenever I felt stuck. Your encouragement to start with the smallest possible concept and gradually build on it was key in moving this project forward and completing it within the scope of a master thesis.

I would also like to thank *Stefan van de Schootbrugge* for participating in the workshops with me, sharing your expertise in decision support software and data, and introducing me to the world of software development at your company.

Many thanks to *Gerard van den Engel* for giving me the opportunity to carry out my graduation internship within your team at Arcadis. Your enthusiasm for sustainable portfolio management and strategy, your involvement in meetings, and your constant reflections on how sustainability themes could be integrated in decision-making helped me broaden my perspective on measuring and interpreting sustainability.

Thanks to *Anne-marije Scheffe* for involving me in your projects and introducing me to practical cases where portfolio asset optimization strategies were applied, such as the project for the municipality of Groningen and the UWC project. These experiences showed me the importance of setting clear goals and KPIs, and how stakeholder sessions are organized to gather input and define strategies in practice.

Finally, I want to thank my family and friends for their support throughout this process.

Summary

Due to climate change, rising temperatures, extreme rainfall, biodiversity loss, and urban densification, the pressure on the built environment within the urban landscape becomes more visible. Cities must respond to these challenges by becoming climate adaptive, lively, safe, and socially connected. These challenges are not evenly distributed, and each neighborhood faces its own unique issues. Therefore, adaptation or mitigation strategies should be applied locally. Buildings can play a role in this response, as they shape and directly interact with the built environment. They can contribute to neighborhood resilience and, eventually, be part of creating a healthy city. Public buildings, often owned and managed by municipalities, can play a key role in this. Municipalities can use buildings within their portfolio as instruments to create resilient, livable neighborhoods and, ultimately, livable cities.

With the pressing challenges in the time to come, municipalities will need to improve their assets. However, with often hundreds of buildings within the portfolio, spread across the entire city, the number of possible interventions and the varying local challenges make the decision-making process for an effective real estate strategy highly complex. Data and decision support tools such as Multi-Objective Optimization (MOO) and Multi-Criteria Decision Analysis (MCDA) become key components in building an effective real estate strategy. However, within classical MOO and MCDA approaches, several flaws exist. Namely: 1) stakeholders are not integrated a priori into the decision-making process, which results in solutions that are ineffective or unusable in practice; 2) a set of predefined design solutions is evaluated, but this does not guarantee that the optimal design is included.3) aggregation and mathematical modeling errors occur; 4) and many MOO approaches conclude with a Pareto front, a set of non-dominated possible solutions, leaving the decision-maker without a clear final design.

As a response to these shortcomings, Open System Design (ODESYS) introduces Preferendus as a decision support software. Preferendus addresses these common flaws in optimization: it integrates stakeholders early, avoids fixed solution spaces, uses mathematically sound preference modeling, and converges on a single optimal design.

This thesis develops and tests a multi-objective optimization (MOO) decision support tool, based on the Preferendus framework, for municipal portfolio owners. The tool helps them manage their real estate portfolio by identifying the most effective interventions, matched to specific buildings, targeted in the neighborhoods where they are needed most, and aligned with the municipality's own objectives.

To develop and test the model, a demonstrator case was used. Fictive sample data based on data collected for neighborhoods in The Hague and buildings from the real estate portfolio of The Municipality of The Hague. The data set contained roof surface data for 25 buildings, spread over 5 neighborhoods. To model neighborhood challenges in these five areas, additional spatial data was used on biodiversity, social cohesion, and surface water overload.

To make impact and adapt on these challenges through portfolio asset management, the optimization focused on assigning rooftop interventions, such as green roofs, solar panels, water retention roofs, and social-commercial roofs, to address challenges like biodiversity, social cohesion, and surface water overload, while also minimizing investment costs or maximizing annual return to earn back the investment over time.

The tool was validated in a live workshop with Stefan van de Schootbrugge, owner of Bress, a software company providing software for municipal portfolio owners to query open-source data and link it to the assets in their portfolio. Multiple iterations were run, in which Stefan took the role of portfolio owner and introduced new objectives over time, adjusted weights, and changed preference curves in real-time to observe how final design configurations changed. The results demonstrated that the tool is able to effectively guide decisions, explore trade-offs between objectives, and converge on feasible intervention strategies that address spatial, data-driven neighborhood needs. It allowed portfolio owners to develop and design a strategic portfolio-wide asset management plan.

Arcadis' real estate portfolio sustainability team reviewed the workshop results, and both Arcadis and Bress agreed that the core strength of Preferendus lies in its ability to model objectives, weights, preferences, and constraints in real time, enabling portfolio owners to directly observe how optimal designs evolve during iterative workshops. At the same time, the use of preference functions forces decision-makers to make their objective valuations explicit, therefore tangible and comparable, a feature highly valued by professionals from both organizations.

While the tool's results appear promising, there are always points for further development such as an improvement of spatial impact modeling, data quality, a more user-friendly interface, and the prioritization of assets over time.

In conclusion, the decision support tool showed that rooftop allocation, when approached through a preference-based MOO framework, can serve as a powerful demonstrator for supporting sustainable asset management at the portfolio scale. Preferendus proves capable of structuring complex decision-making problems and translating stakeholder preferences and objectives into an optimized, spatial data-driven intervention strategy, offering real potential within portfolio asset management.

Contents

1 Introduction

1.1. Urban challenges and the role of buildings in local adaptation

Dutch cities are becoming denser while pressures on the urban environment due to climate change intensify more and more (College van Rijksadviseurs, 2022). Cities face urbanization, rising temperatures, extreme rainfall, and biodiversity loss. These challenges call cities to adapt and rethink how space can be used. For the Netherlands, which has a dense urban setup and a long history of strategic spatial planning, these challenges will take on a unique form. For cities, it is important that they are able to adapt to the challenges of climate change, but for a city to be vibrant, it should also be lively, safe, and socially connected (United Nations, 2015)(Arcadis, 2024). The challenges above call for both adaptation and mitigation strategies in urban planning (Hurlimann et al., 2021).

To apply effective mitigation and adaptation strategies, geographical, or spatial, data analysis is crucial to map where specific challenges arise (Roest et al., 2023). Namely, challenges are not evenly distributed over neighborhoods. In some neighborhoods there is more heat stress, biodiversity loss, water overload during peak rainfall, or a lack of access to green space and social infrastructure (Roest et al., 2023). Therefore, it becomes inevitable that adaptation strategies must be tailored locally. Not surprisingly, to understand the distribution of those challenges, geographical data analysis is already being used in the Netherlands. Tools like *Leefbaarheidsbarometer* (Ministerie van Binnenlandse Zaken en Koninkrijksrelaties, 2024) and the *Sustainable Cities Index 2024* (Arcadis, 2024) help to visualize which neighborhoods are under performing. Not only hard climate data is assessed, but also softer social aspects such as safety, livability, and social cohesion are expressed in neighborhood-level data.

Buildings shape the livable environment of cities and play an important role in creating lively and resilient neighborhoods (Urban Land Institute, 2022). This also means that buildings can be part of the solution. They provide means to address the challenges previously mentioned and support both mitigation and adaptation efforts. Buildings do not exist in isolation but constantly interact with their surroundings. Buildings and neighborhoods can therefore be seen as components within a bigger, interconnected system. By applying targeted interventions that improve buildings and address specific local needs, it becomes possible to reduce environmental and social pressures in the surrounding area. Thus, upgrading buildings is not just about improving individual structures, but also about strengthening neighborhood resilience and ultimately contributing to the resilience of the entire city.

1.2. Complex decision-making in public real estate portfolio strategies

For a single building, applying interventions to mitigate environmental or social challenges can already be complex, due to the technical, financial, architectural, and operational factors involved. However, making decisions over an entire portfolio of buildings increases decision complexity significantly. Especially municipalities, which often own a substantial amount of buildings. For instance, the portfolios of the municipalities of Amsterdam, The Hague, and Rotterdam contain approximately 1015, 800, and 2600 public real estate assets respectively(Bouwstenen voor Sociaal, 2024; Gemeente Rotterdam, 2025). As public actors, municipalities are responsible to create healthy and resilient cities and therefore neighborhoods. Their building, within their portfolios, are not just physical assets, but rather instruments through which they can address climate, social, and spatial challenges. But as stated, the complexity of decision making in asset management for municipalities grows significantly due to the number of available interventions, the variety of spatial neighborhood challenges, and the diversity and amount of buildings within the portfolio. Not surprisingly, making decisions that align real estate strategies with broader goals is one of the most complex challenges in portfolio management (Arkesteijn, 2019).

1.3. Multi-Criteria Decision Analysis (MCDA) and Multi-Objective Optimization (MOO) as response for complexity

As the number of buildings, interventions, and strategic goals increases, decision-making becomes too complex to do manually. This is especially true in portfolio-level asset management. To create structure and transparency within the decision-making processes, decision-makers increasingly rely on decision-support tools. The most common used decision support methods are Multi-Criteria Decision Analysis (MCDA) and Multi-Objective Optimization (MOO) (Wolfert, 2023). MCDA is used to rank a predefined set of alternatives based on multiple criteria. By assigning a weight to each criterion, it becomes possible to score and rank alternatives. These scores are combined into a single score for each alternative, often by using weighted sums. This makes comparison and selection of the most preferred option possible. MOO, on the other hand is best for situations with many or even infinite possible solutions. Instead of evaluating predefined options, decision-makers define objective functions. Then an algorithm searches the design space and generates a large range of alternatives. MOO helps to explore trade-offs between objectives and identifies and generates a set of optimal solutions. MOO allows decision-makers to see and understand how their input affects the generated set of solutions, often referred to as final design configurations (Wolfert, 2023). Both MCDA and MOO are often used in engineering, urban development, and design.

1.4. Methodological shortcomings in MOO and MCDA

Recent literature by van Heukelum et al. (2023) and Zhilyaev et al. (2022) challenges key assumptions in conventional MOO and MCDA approaches. These authors point out four systematic flaws that can hinder or misguide the decision-making process.

1. Lack of a priori stakeholder integration: technically feasible but socially infeasible designs

In many MOO applications, stakeholders are either not integrated into the process at all or are only involved afterwards, *postiori*, in a post hoc manner (van Heukelum et al., 2023). As a result, design configurations might technically satisfy the conditions, but once they need to be implemented in practice, they turn out to be politically or socially unfeasible. By not including stakeholders a priori of the optimization, contextual factors or important objectives or constraints of key stakeholders might be ignored. As a result, 'optimal' design configurations prove to be ineffective or even unusable in practice (Zhilyaev et al., 2022).

2. Predefined and limited alternatives: not exploring the full design space

Another shortcoming in current optimization methods is the evaluation of a set of predefined design configurations. This implies that the solution space already is defined in advance, and that not all possibilities for potentially more optimal configurations are explored. As van Heukelum et al. (2023) pointed out, this approach excludes potentially better solutions before evaluation begins, increasing the risk of suboptimal results.

3. Aggregation and preference modeling errors

In multi-objective optimization (MOO), all objectives are often converted into a single common score, scale, or domain (van Heukelum et al., 2023). But this can be problematic if it's not done in the right way. Many objectives are measured on different scales, like euros (money) or kilograms of CO2. These scales aren't mathematically compatible, so adding or averaging them assumes they have the same meaning, which they don't (Barzilai, 2005, 2010). This could therefore go against the basic rules of measurement theory and can lead to mathematically incorrect results(Barzilai, 2005; van Heukelum et al., 2023).

To avoid invalid mathematical aggregation, objectives should be translated to a common scale that allows mathematical operations. In practice, this is often done by expressing the value of objectives with different scales in a monetary scale. However, this highly oversimplifies context. The monetary scale doesn't reflect how much people care about something or value different objectives(Barzilai, 2010). According to utility measurement theory, the common scale of measuring how much we value something should be measured in preference, not just cost (van Heukelum et al., 2023). Additionally, by not aggregating to a common nonlinear scale, one implies that preferences are linear and therefore an increase and decrease are always valued the same. But real preferences are often nonlinear and personal. Without a valid and shared preference scale, decisions can become unclear, unfair, or misleading (Binnekamp, 2010).

4. Lack of decision-readiness: no single optimal configuration

With MOO, the final output often concludes with a Pareto front. A Pareto front is a set of potential trade-off solutions that reflect competing objectives, where choosing a different solution cannot improve one objective without worsening the other. The problem of concluding with a Pareto front is that it doesn't propose a single most optimal design configuration (van Heukelum et al., 2023). Instead, the final decision is left to the decision-maker, who needs to navigate between trade-offs without a structured decision-making framework (van Heukelum et al., 2023) (Wolfert, 2023).

1.5. Odesys and Preferendus as MOO framework

To address the four systemic flaws from the previous section, Wolfert (2023) introduce ODESYS (Open Systems Design) as a new way of thinking and approaching engineering design problems. A part of ODESYS is Integrative Maximized Aggregated Preference (IMAP). IMAP is a MOO method that maximizes the overall aggregated objectives measured in preference. IMAP forms the core of a new software decision support tool introduced in ODESYS, the Preferendus. By using the Preferendus, earlier-mentioned flaws can be addressed Wolfert (2023). Key components of the preferendus are shown below:

---

1. Integrates stakeholders a priori

The Preferendus integrates stakeholders a priori in the decision-making process. Therefore, stakeholder and preference are used as input, functioning like a decision compass where trade-offs are explored interactively. The model becomes open-ended and human-interactive.

2. Proactive synthesis instead of evaluation

Unlike traditional methods that evaluate predefined (potentially suboptimal) alternatives, the Preferendus proactively generates optimal configurations from the full solution space. This expands the range of outcomes and supports creative and adaptive solutions.

3. Mathematically correct Preference Function Modeling (PFM)

Decision behavior is mathematically modeled based on the Preference Function Modeling (PFM) theory, as described by (Barzilai, 2022). Objectives on a different scale are aggregated on a properly defined scale that measures the value of the objective expressed in preference on a scale from 0 to 100. This workflow allows mathematically correct aggregation and comparison.

4. Convergence to one final optimal design solution

Instead of concluding with a Pareto front and leaving decision makers without one optimal solution, Preferendus iteratively generates one final optimal design configuration.

Figure 1.1 shows a summary figure of how the Preferendus overcomes shortcomings often committed in other MOO or MCDA studies.

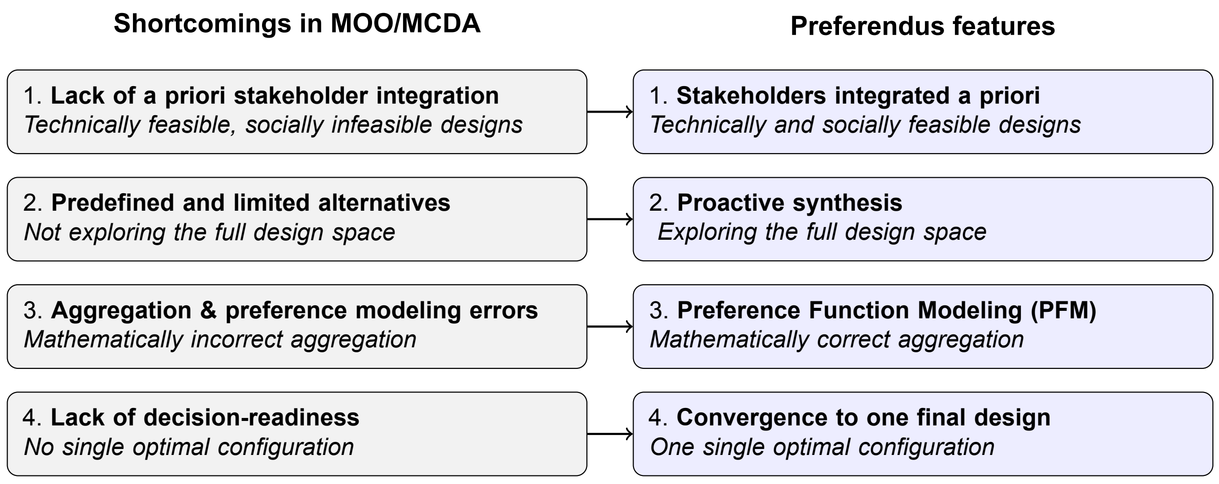

Shortcomings in MOO/MCDA
Preferendus features
1. Lack of a priori stakeholder integration
Technically feasible, socially infeasible designs
1. Stakeholders integrated a priori
Technically and socially feasible designs
2. Predefined and limited alternatives
Not exploring the full design space
2. Proactive synthesis
Exploring the full design space
3. Aggregation & preference modeling errors
Mathematically incorrect aggregation
3. Preference Function Modeling (PFM)
Mathematically correct aggregation
4. Lack of decision-readiness
No single optimal configuration
4. Convergence to one final design
One single optimal configuration

Figure 1.1: From MOO/MCDA shortcomings to Preferendus features.

Following features of the Preferendus (see Figure 1.1), Arkesteijn (2019) has created a framework for portfolio real estate alignment. Within this self-developed framework, the Preference-based Accommodation Strategy (PAS), Arkesteijn (2019) showcased that portfolio-level problems can be resolved on several case studies. The PAS was specifically developed to integrate stakeholders a priori within the decision-making process and align stakeholders preferences with organizational goals in portfolio strategies. The PAS model proved effective in integrating strategic, spatial, and functional considerations, ultimately delivering the design of an optimal and aligned real estate portfolio.

Several case studies in the fields of urban development, architecture, construction management, and corporate real estate portfolio alignment have successfully used the Preferendus, or PFM, to support decision making (Binnekamp, 2006; Binnekamp, 2010; De Visser et al., 2017; Arkesteijn, 2019; van Eijck & Nannes, 2022; Raaphorst, 2024; Zhilyaev et al., 2022; van Heukelum et al., 2023). Recently, Zhilyaev et al. (2022) published a Preferndus framework which engineers and designers can use for real world design and construction decision-making problems. As a next step in the development of this methodology, it Zhilyaev et al. (2022) highlighted the need to explore and test this framework in specific domains, such as supply chain management or asset management.

1.6. Rooftops as a demonstrator case for portfolio MOO

An interesting demonstrator case for portfolio optimization in sustainable asset management could be the optimization of rooftops of buildings within the portfolio. As exterior parts of the building, they directly interact with the surrounding environment. Within big municipalities such as Amsterdam, Rotterdam, and The Hague, there is an increasing focus on using roof space effectively to tackle urban challenges(Dakenplan, 2025). Not only in practice but also in the academic field, recent studies have shown growing interest in rooftop optimization, especially within cities facing climate challenges and the need to adapt existing buildings (Brenner et al., 2023; Dong et al., 2022, 2024; Kumar et al., 2022; Langemeyer et al., 2020; Liu et al., 2022; Xiong et al., 2023; Yuan et al., 2025; Zhang et al., 2024). However, many of these approaches still suffer from the earlier addressed four key methodological flaws identified by (van Heukelum et al., 2023; Zhilyaev et al., 2022):

(1) *A lack of early stakeholder integration*: In Dong et al. (2024) stakeholder and policy factors are mentioned only briefly at the end (Section 4.4.3). By stating that issues such as governance and funding fall “beyond the scope,” the study places them outside the core of the design process.

(2) *Reliance on a predefined set of alternatives*: Xiong et al. (2023) evaluate only 64 fixed design strategies, 32 green and 32 gray infrastructure scenarios, while Liu et al. (2022) only assess four predefined static retrofitting scenarios with 10, 30, 50, and 100 percent green roof coverage.

(3) *Flawed aggregation of incomparable indicators*: This occurs in almost every study. Yuan et al. (2025), for example, normalize CO2 emissions, economic cost, and food production, and combine them into one score, treating them as commensurate. Dong et al. (2024) follow a similar approach by aggregating cooling effect, runoff reduction, and investment cost. Xiong et al. (2023) go further by converting ecological and environmental benefits into monetary values, assuming that all value can be expressed in financial terms. Langemeyer et al. (2020), for instance, combine urban heat island intensity, social cohesion, and income differentials into a single utility score using weighted sums. This not only assumes scale compatibility but also imposes fixed, linear preferences.

(4) *Limited decision-readiness of the results*: This flaw occurs, for instance, in Zhang et al. (2024), who also emphasize that their multi-objective optimization results in a set of Pareto-optimal solutions.

1.7. Academic contribution and development statement

In the previous paragraphs, it becomes clear that cities face climate and societal challenges, which differ per neighborhood, and which call for neighborhood-specific adaptation strategies. Buildings can be a part of those strategies, and building owners can improve their buildings to tackle those challenges and, eventually, contribute to addressing the issues cities face. For municipalities, who are responsible for creating a livable environment in those cities and who often own a large number of buildings within their portfolio, coming up with a real estate strategy that responds to local needs, by selecting the most suitable interventions within limited resources and across many assets, becomes a complex decision-making problem. This asks for decision support tools such as Multi-Objective Optimization (MOO) and Multi-Criteria Decision Analysis (MCDA). Arkesteijn (2019) showed that MOO with Preferendus principles is possible in a portfolio context. Preferendus has shown to offer many advantages and address flaws present in other MOO methods. A recently developed step-by-step Preferendus methodology by Zhilyaev et al. (2022) calls for testing in other fields of the construction industry, such as asset management. Rooftops can serve as a relevant demonstrator case: in practice, there is an increasing focus on rooftop interventions, and in academic fields, many MOO studies, despite their limitations addressed by Preferendus, have been performed on the strategic allocation of roofs in urban cities, but never in a portfolio context. This thesis aims to answer the following questions, which respond to core challenges for portfolio owners, specifically for municipalities:

- In what way can building-level interventions be selected and spatially allocated to buildings in neighborhoods in a way that reflects both local needs and organizational portfolio goals?
- Which buildings should be prioritized when financial resources are limited and several conflicting objectives are relevant to the decision-maker?

In short, with limited resources and many buildings to manage, municipalities are faced with the urgent question: *How do I choose the right interventions, for the right buildings, in the right places, to achieve the greatest impact while satisfying my own objectives?*

To answer these questions, this thesis responds by developing a decision support tool that follows the Preferendus methodology step by step, as proposed by Zhilyaev et al. (2022), and testing its applicability in a portfolio asset management context.

See the development statement below:

> *"There is a need for a decision support tool in portfolio asset management that selects interventions for buildings, in alignment with organizational objectives and neighborhood needs, and results in a final optimal design configuration"*

1.8. Reading guide

Chapter 2: Analysis presents the literature review on how to build the decision support tool. Section 2.1 reviews current MOO and MCDA studies to understand the workflow of rooftop optimization studies, and to identify what types of challenges can be addressed by different roof types. Section 2.2 analyzes the methodology of Zhilyaev et al. (2022) for the synthesis of the decision support tool.

Chapter 3: Synthesis sets up the entire mathematical structure of the model.

Chapter 4: Operationalization describes, on a conceptual level, how the mathematical problem can be transposed into an operationalized decision support tool.

Chapter 5: Demonstration introduces a demonstrator case, which is used to validate the developed decision support tool in a workshop. The chapter also discusses the results of the workshop and concludes with a reflection on the validation of both the workshop and the tool.

Chapter 6: Discussion outlines the limitations of the decision support tool and proposes steps for further development, as well as suggestions for future research.

Chapter 7: Conclusion reflects on how the decision support tool contributes to addressing complexity in spatial problems and in portfolio asset management strategies for portfolio owners.

2 Analysis

2.1. Review of MOO and MCDA frameworks studies in urban rooftop planning

Since MOO and MCDA often appear in rooftop planning studies, a brief analysis is conducted on what kind of MOO was done and how spatial problems are integrated within the decision-making process of urban strategies. All the studies discussed aim to identify optimal configurations of urban roof layouts. However, the types of roof interventions and the objectives differ per study. Below, a selection of studies is reviewed to highlight how multi-objective and MCDA frameworks are used to generate and assess rooftop intervention strategies over an urban layout. First, the MOO studies are reviewed, followed by the MCDA studies.

Previous MOO-based studies

Dong et al. (2024) use MOO to allocate green roofs in Xiamen, China. Using an algorithm, they explore trade-offs between three objectives: storm water retention, urban cooling, and investment costs. An optimal design configuration is selected from the Pareto front using Entropy-TOPSIS, a ranking methodology. Yuan et al. (2025) optimize objectives such as life cycle impact potential, operational benefits, and costs by applying three rooftop types: Bare Roof (PV + water), Green Roof (vegetation + PV + water), and Open-Air Farming. Zhang et al. (2024) optimize the allocation of PV panels, maximizing three objectives: performance cost, visual impact, and spatial compactness. Kumar et al. (2022) optimize storm-water management by allocating green roofs and infiltration trenches. The objectives that were minimized and maximized included the volume of storm-water runoff, benefits, and costs.

Previous MCDA-based studies

Beyond MOO, several studies also use MCDA frameworks to allocate roof types.

Langemeyer et al. (2020) use MCDA to select an allocation of five possible green roof types: extensive, semi-intensive, intensive, naturalized, and allotment. These types are evaluated against six ecosystem service goals to support a healthy city: thermal regulation, runoff control, habitat provision and pollination, food production, recreation, and social cohesion. First, spatial data on each ecosystem service demand is plotted in choropleths to map where neighborhood needs are most visible. Second, the green roof types receive scores based on how much they contribute to ecosystem service delivery. Eventually, through MCDA, roof types are allocated where they are most needed.

Brenner et al. (2023) aim to mitigate the urban heat island effect by allocating green roofs where they are most impactful. Two maps are overlaid based on spatial data: one vulnerability map combining data on heat stress and sensitive groups such as the elderly, children, and low-income residents; and one map based on satellite imagery of rooftops. This leads to a selection of roofs categorized as high (high vulnerability + high/medium potential) and medium (high vulnerability + low potential), guiding selection to areas with the most impact.

Dong et al. (2022) classify roof types and assign them suitability scores, while also including landscape connectivity. This makes it possible to find the best combination of roof surfaces to upgrade to green roofs. The goal is to allocate rooftops in a way that supports the creation of a biodiversity corridor through the city.

Liu et al. (2022) aim to allocate extensive green roofs in Beijing. Existing rooftops are ranked using an index that combines green space demand, runoff risk, and roof suitability. Rooftops are selected based on the highest overall priority scores.

2.2. Methodological framework Preferendus

This thesis adapts the methodological framework for Preferendus modeling developed by (Zhilyaev et al., 2022). The authors present a flowchart (see Figure 2.1) which represent 10 steps of this methodology. The 10 steps of this methodology are briefly summarized to give the reader an understanding of how a design problem can be addressed using the Preferendus. For an in depth understanding and mathematical explanation refer to (Zhilyaev et al., 2022). The mathematical problem will be discussed less in this section, as it is elaborately presented in chapter 3.

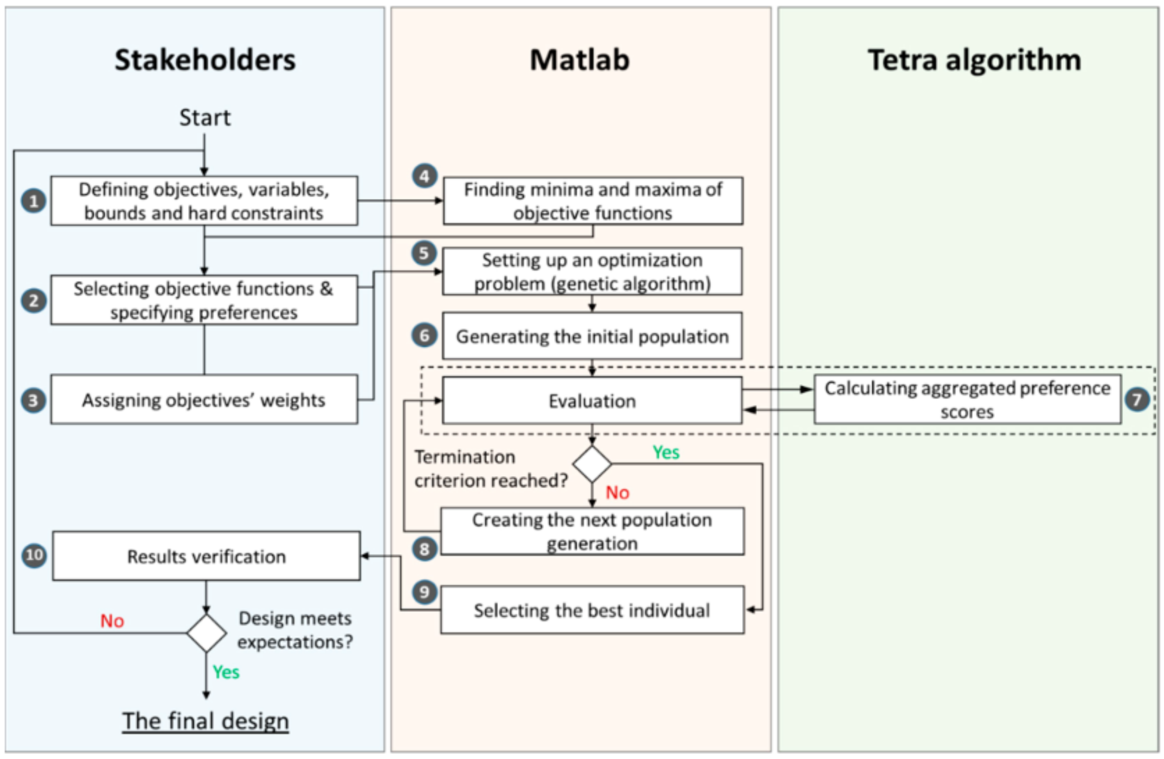

Stakeholders
Matlab
Tetra algorithm
Start
1 Defining objectives, variables, bounds and hard constraints
2 Selecting objective functions & specifying preferences
3 Assigning objectives' weights
4 Finding minima and maxima of objective functions
5 Setting up an optimization problem (genetic algorithm)
6 Generating the initial population
Evaluation
7 Calculating aggregated preference scores
Termination criterion reached?
Yes
No
8 Creating the next population generation
9 Selecting the best individual
10 Results verification
No
Design meets expectations?
Yes
The final design

Figure 2.1: Methodology flowchart by (Zhilyaev et al., 2022)

1. **Defining the design problem**
   Stakeholders define the design problem by defining design variables, which are design parameters that can be varied, the bounds of these design variables, and any equality or inequality constraints.
2. **Finding the minima and maxima (ranges) of the objective functions**
   To understand the range and therefore boundaries of the stakeholder preferences, it is necessary to run optimizations and understand the minimum and maximum raw values of the objectives.
3. **Defining preferences**
   Preferences are measured on a preference scale from 0 to 100 for each objective. Stakeholders need to at least select 3 points (the worst = 0, values in between, and the best=100) for each objective. As a result preference curves can be constructed that translate the performance of

each objective into a subjective desirability score (see Figure 2.2).

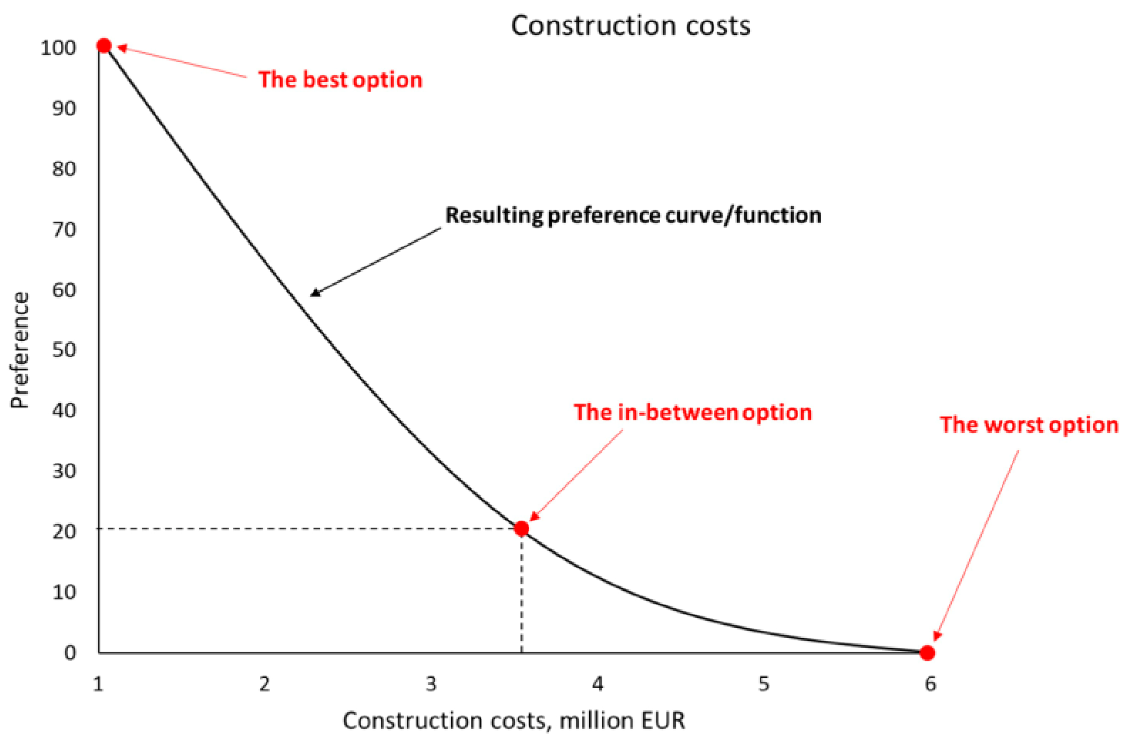


Figure 2.2: Preference curve by (Zhilyaev et al., 2022)

4. **Specifying weights**
   Each objective needs to get assigned a weight, of which the total sum of the weights of objectives must be 1. The weights express the relative importance of each objective.
5. **Model setup**
   All the input data, variables, objectives, bounds, constraints, preferences, and weights can be used to set up the optimization problem in Matlab, an environment for progamming. Within this thesis, the problem is set up in Jupyter Notebook, an environment that supports Python scripts.
6. **Generating the initial population**
   A genetic algorithm (GA) generates an initial population, where each individual is a possible design solution. For each individual, the raw value of the objective is calculated on the preference scale (0-100) of the defined preference curve.
7. **Preference aggregation and evaluation**
   Each individual of the population now has a preference score per objective (from step 6). Those preferences need to be aggregated into a final score, which eventually reflects the individuals values. The aggregated preference score is the score that the genetic algorithm will try to optimize within one final objective function. Calculating this aggregated preference score is done on a web server software tool called Tetra, after which the algorithm evaluates the results and their feasibility.
8. **Evolving the population over a next generation**
   Through mutation, and crossover, the algorithm generates new configurations in each generation.
9. **Selecting the optimal configuration (best individual)**
   Once the stopping criterion is met, and the algorithm finds no further improvement, the GA selects the configuration (individual) with the highest aggregated preference score.
10. **Stakeholder verification and iteration**
    The selected solution is presented to stakeholders. If the outcome is satisfactory, the final configu-

ration is selected. Otherwise, stakeholders can change their preferences, weights, and objectives, add constraints and the steps are repeated until the final design solution satisfies the stakeholder.

3 Synthesis: mathematical structure model

3.1. Mathematical formulation of the optimization problem

This model assigns exactly one rooftop intervention to each roof surface in a building portfolio. It is formulated as a discrete, multi-objective, nonlinear optimization problem with integer decision variables. Each decision variable x_n represents the choice of a roof type t (e.g., green roof, solar panels, or none) from a predefined set of feasible options. Since roof types are indexed as integers and each surface receives only one option, the problem is fully discrete.

The model optimizes six objectives O_i: *investment cost*, *financial return*, *CO_2 reduction*, *biodiversity*, *social cohesion*, and *water retention*. All these objectives have different measurement scales. These are converted by using preference functions $p_i(O_i(x))$ which are measured on a common 'preference' scale from (0–100).

$$O_1^*(x) = p_1(O_1(x)), O_2^*(x) = p_2(O_2(x)), ..., O_I^*(x) = p_I(O_I(x)) \tag{3.1}$$

Where:

- $O_i(x)$: Objective function i, calculated over the full design vector x. Examples include investment cost, annual return, CO_2 savings, biodiversity, social cohesion, and water retention.
- $p_i(O_i(x)) \in [0, 100]$: Preference function i, translating the measured value from $O_i(x)$ into a preference score.
- $O_i^*(x) \in [0, 100]$: Rewritten preference function for $O_i(x)$.

A more detailed explanation and the final objective function of the algorithm are provided on the next page.

The objective function for the algorithm, Equation (3.2), searches for the configuration with the maximized aggregated preference score (IMAP). Refer to van Heukelum et al. (2023) and Zhilyaev et al. (2022) for more detailed explanation.

$$\begin{aligned} \max_{x} \quad & P^{*}\left(O_1^{*}(x), O_2^{*}(x), ..., O_I^{*}(x); w_1, w_2, ..., w_I\right) \\ & i = 1, 2, \dots, I \\ & x = [x_1, x_2, \dots, x_N] \end{aligned} \tag{3.2}$$

subject to:

$$g_j(x) \leq 0, \quad j = 1, 2, \dots, J \tag{3.3}$$

$$h_k(x) = 0, \quad k = 1, 2, \dots, K \tag{3.4}$$

$$x_n \in \{0, 1, 2, 3, 4, 5\}, \quad n = 1, 2, \dots, N \tag{3.5}$$

Where:

- $\max_x$: The optimization algorithm searches for the decision vector x that maximizes the aggregated preference score.
- $P^* \in [0, 100]$: Aggregated preference score calculated by the Tetra web-server using Preference Function Modeling (PFM) theory.
- $x = [x_1, x_2, \dots, x_N]$: Design vector of decision variables, where each x_n represents the selected rooftop intervention type t for rooftop surface n.
- N: Total number of rooftop surfaces across all buildings in the portfolio.
- x_n: Decision variable for rooftop surface n. It takes on a discrete value corresponding to one of the six available rooftop types t:

$$x_n = t \quad \text{for some } t \in \{0, 1, 2, 3, 4, 5\} \tag{3.6}$$

- $t \in \{0, 1, 2, 3, 4, 5\}$: Index for the six predefined rooftop intervention types. Each t corresponds to a rooftype with each a unique set of technical, financial, and environmental parameters used in the objectives.
- $O_i(x)$: Objective function i, calculated over the full design vector x. Examples include investment cost, annual return, CO_2 savings, biodiversity, social cohesion, and water retention.
- $w_i \in [0, 1]$: Weight for objective i, representing its importance. All weights sum to 1:

$$\sum_{i=1}^{6} w_i = 1 \tag{3.7}$$

- $g_j(x) \leq 0$ for $j = 1, 2, \dots, J$: Inequality constraints (not used in validation).
- $h_k(x) = 0$ for $k = 1, 2, \dots, K$: Equality constraints (not used in validation).
- $x_n \in \{0, 1, 2, 3, 4, 5\}$ for $n = 1, 2, \dots, N$: Bounds: Each decision variable x_n represents the intervention type t assigned to rooftop surface n and can take one of six discrete values $0 \dots 5$.

3.2. Roof types as design variables

Each decision variable x_n in the optimization model represents the selection of one rooftop intervention from a discrete set of six alternatives. Each intervention is denoted by an index $t \in \{0, 1, 2, 3, 4, 5\}$, where each t represents a specific rooftop type (see Figure 3.1) with unique technical, environmental, and financial characteristics (see Table 3.1). The decision variable x_n selects one of these types for each rooftop surface n. The interventions differ in their contribution to the six objectives $O_i(x)$. Table 3.1 lists the parameter values defined for each rooftop type t, with data derived and adapted from (Municipality of Rotterdam & B.V., 2024). These values are used in the objective functions $O_i(x)$ discussed later. The coefficients α_t, β_t, and γ_t represent the assumed effects of each rooftop type t on biodiversity, social cohesion, and water retention, respectively, and are used in Equation 3.14, Equation 3.16, and Equation 3.18. These coefficients are not derived from literature but can be defined by the decision-maker, similar to the approach of Langemeyer et al. (2020).

The six rooftop types t are:

- 0 — No intervention
- 1 — Sedum roof
- 2 — Biodiversity roof
- 3 — Commercial social roof
- 4 — Water storage roof
- 5 — Solar panels

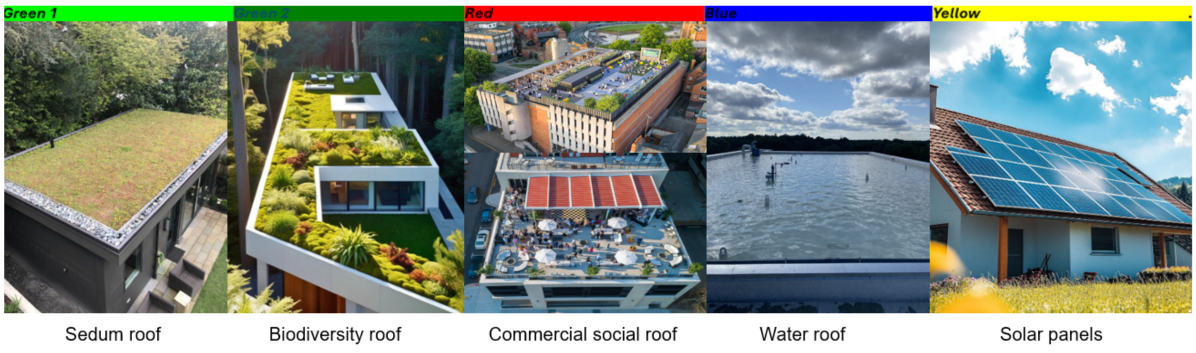


Figure 3.1: Overview of rooftop intervention types used in the optimization model

t	**Roof type**	**Description**	$\text{Slope}_t^{\max}$ (°)	c_t^{repl} (€/m²)	c_t^{inst} (€/m²)	c_t^{maint} (€/m²/yr)	r_t (€/m²/yr)	α_t (-)	β_t (-)	γ_t (-)	κ_t (kgCO$_2$/m²/yr)	**Investment cost (€/m²)**	**Annual return (€/m²/yr)**
0	None	No intervention	90	0	0	0.00	0.00	0	0	0	0	0	0.00
1	Green 1	Sedum roof	45	45	50	1.20	0.00	30	0	30	0	95	-1.20
2	Green 2	Biodiversity roof	4	45	75	1.80	0.00	100	0	40	0	120	-1.80
3	Red	Commercial social roof	0	45	500	10.00	70	10	100	20	0	545	60.00
4	Blue	Water roof	0	45	150	2.00	5.00	0	0	100	0	195	3.00
5	Yellow	Solar panels	70	0	235	0.35	27.60	0	0	0	85.28	235	27.25

Table 3.1: Combined coefficient, slope constraint, and objective values per roof type

3.3. Objective functions

These six objective functions $O_i(x)$ form the basis of the aggregated preference score in Equation 3.2. They are calculated using three key inputs: the surface area A_n of each rooftop surface n, the selected rooftop type t, and the neighborhood-specific needs. The first three objectives; investment cost, annual financial return, and annual CO_2 reduction mainly scale with total assigned area and are defined in Equation 3.8, Equation 3.9, and Equation 3.10. The latter three; biodiversity, social cohesion, and water retention, are spatially sensitive and combine roof-specific effect coefficients with neighborhood-level needs, as shown in Equation 3.14, Equation 3.16, and Equation 3.18.

Objective 1: Investment cost

This objective calculates the total Investment cost needed for all rooftop interventions, summing the installation and replacement costs for each selected roof type over all rooftop surfaces of the portfolio.

$$O_1(x) = \sum_{n=1}^{N} A_n \cdot \left(c_t^{\text{repl}} + c_t^{\text{inst}}\right) \tag{3.8}$$

- $O_1(x)$: Total investment cost (€) required for implementing all rooftop interventions across the portfolio.
- A_n: Area (m²) of rooftop surface n.
- c_t^{repl}: Replacement cost (€/m²) for the chosen roof type t.
- c_t^{inst}: Installation cost (€/m²) for the chosen roof type t.

Objective 2: Annual financial return

This objective calculates the net annual return by subtracting annual maintenance costs from the annual financial returns associated with each selected roof type over all rooftop surfaces of the portfolio.

$$O_2(x) = \sum_{n=1}^{N} A_n \cdot \left(r_t - c_t^{\text{maint}}\right) \tag{3.9}$$

- $O_2(x)$: Total annual financial return (€/yr) gained for implementing all rooftop interventions across the portfolio.
- r_t: Annual financial return (€/m²/yr) for the roof type t.
- c_t^{maint}: Maintenance cost (€/m²/yr) for the roof type t.

Objective 3: Annual CO_2 reduction

This objective calculates the total annual CO_2 savings achieved by assigning solar panel roofs (roof type 5) across the building portfolio. The result sums the CO_2 reduction potential across all rooftop surfaces.

$$O_3(x) = \sum_{n=1}^{N} A_n \cdot \kappa_t \tag{3.10}$$

- $O_3(x)$: Total annual kg CO_2 ($kgCO_2$/yr) saved gained for implementing all rooftop interventions across the portfolio.
- κ_t: Annual CO_2 ($kgCO_2$/m²/yr) savings for the chosen roof type t.

General structure for objectives 4, 5 and 6

Objectives $O_4(x)$, $O_5(x)$, and $O_6(x)$ represent the impact objectives on neighborhoods for biodiversity, social cohesion, and water retention, respectively. Each of these objectives is calculated using the same general structure: first, a raw impact score is computed across all rooftop surfaces and neighborhoods; second, this score is normalized by dividing it by a calculated maximum value. This ensures that the final objective value lies within the range $[0, 1]$, where a value of 1 represents the maximum possible impact if the most effective rooftop type t were applied to every surface n of the portfolio.

Calculate the raw impact score:

$$\mathrm{Raw}_{\mathrm{impact}} = \frac{\sum_{n=1}^{N} A_n \cdot \left(\frac{\mathrm{effect}_t}{100}\right)^{\gamma} \cdot \left(\frac{\mathrm{need}_t}{100}\right)^{\delta}}{\sum_{n=1}^{N} A_n} \tag{3.11}$$

- $\mathrm{Raw}_{\mathrm{impact}}$: Raw (unnormalized) impact score, calculated across all rooftop surfaces.
- A_n: Area (m²) of rooftop surface n.
- $\mathrm{effect}_t \in [0, 100]$: Effectiveness score of the roof type t
- $\mathrm{need}_p \in [0, 100]$: Relative urgency or need score of the neighborhood p
- γ, δ: Exponents used to adjust the sensitivity of the score to performance and need, respectively. These components introduce non-linearity, so that rooftop types with a greater effect_{x_n} or neighborhoods with higher need_n are prioritized more strongly. In this way, the model assigns higher scores (closer to 1) to high-impact rooftop types implemented in high-need areas.

The final normalized impact score becomes:

$$O_i(x) = \frac{\mathrm{Raw}_{\mathrm{impact}}}{\mathrm{Max}_{\mathrm{impact}}} \quad \text{for } i \in \{4, 5, 6\}, \quad O_i(x) \in [0, 1] \tag{3.12}$$

- $\mathrm{Max}_{\mathrm{impact}}$: The maximum unnormalized impact score when all rooftops are assigned the type that maximally contributes to the neighborhood need.
- $O_i(x)$ for $i \in \{4, 5, 6\}$: The normalized impact score, taking values in $[0, 1]$.

Objective 4: Biodiversity impact

Following the structure explained above, this objective calculates how well the selected roof types align with the biodiversity needs of each rooftop location, taking into account both roof performance and local neighborhood need. The raw biodiversity impact score is:

$$\mathrm{Raw}_{\text{bio}} = \frac{\sum_{n=1}^{N} A_n \cdot \left(\frac{\alpha_t}{100}\right)^{\gamma} \cdot \left(\frac{N_p^{\text{bio}}}{100}\right)^{\delta}}{\sum_{n=1}^{N} A_n} \tag{3.13}$$

- $\alpha_t \in [0, 100]$: Biodiversity effect coefficient of the roof type t.
- $N_p^{\text{bio}} \in [0, 100]$: Biodiversity need score of the neighborhood p.

The final, normalized biodiversity objective is:

$$O_4(x) = \frac{\mathrm{Raw}_{\text{bio}}}{\mathrm{Max}_{\text{bio}}} \quad \text{where } O_4(x) \in [0, 1] \tag{3.14}$$

- $O_4(x) \in [0, 1]$: Biodiversity impact score for implementing all rooftop interventions across the portfolio.

Objective 5: Social cohesion impact

This objective calculates how effectively the selected roof interventions support social cohesion, again using the general structure described earlier. The raw social cohesion score is:

$$\text{Raw}_{\text{soc}} = \frac{\sum_{n=1}^{N} A_n \cdot \left(\frac{\beta_t}{100}\right)^{\gamma} \cdot \left(\frac{N_p^{\text{soc}}}{100}\right)^{\delta}}{\sum_{n=1}^{N} A_n} \tag{3.15}$$

- $\beta_t \in [0, 100]$: Social cohesion effect coefficient of the roof type t.
- $N_p^{\text{soc}} \in [0, 100]$: Social cohesion need score of the neighborhood p

The final, normalized social cohesion objective is:

$$O_5(x) = \frac{\text{Raw}_{\text{soc}}}{\text{Max}_{\text{soc}}} \quad \text{where } O_5(x) \in [0, 1] \tag{3.16}$$

- $O_5(x) \in [0, 1]$: Social cohesion impact score for implementing all rooftop interventions across the portfolio.

Objective 6: Water retention impact

This objective calculates the capacity of selected roof types to enhance water retention taking into account both roof performance and local neighborhood need. As with the other objectives, it follows the same structure:

$$\text{Raw}_{\text{wat}} = \frac{\sum_{n=1}^{N} A_n \cdot \left(\frac{\gamma_t}{100}\right)^{\gamma} \cdot \left(\frac{N_p^{\text{wat}}}{100}\right)^{\delta}}{\sum_{n=1}^{N} A_n} \tag{3.17}$$

- $\gamma_t \in [0, 100]$: Water retention effect coefficient of the roof type t.
- $N_p^{\text{wat}} \in [0, 100]$: Water retention need score of the neighborhood p.

The final, normalized water retention objective is:

$$O_6(x) = \frac{\text{Raw}_{\text{wat}}}{\text{Max}_{\text{wat}}} \quad \text{where } O_6(x) \in [0, 1] \tag{3.18}$$

- $O_6(x) \in [0, 1]$: Water retention impact score for implementing all rooftop interventions across the portfolio.

3.4. Preference functions

In the main objective function, Equation 3.2, the aggregated preference score P^* depends on how each objective value $O_i(x)$ is translated into a preference score. This requires the decision-maker to define a separate preference function, $p_i(O_i(x))$, for each of the six objectives. These functions reflect how desirable different outcome levels are and shape the overall optimization result (see Figure 3.2). The blue dots can be set by the decision maker and determine the shape of the preference curve.

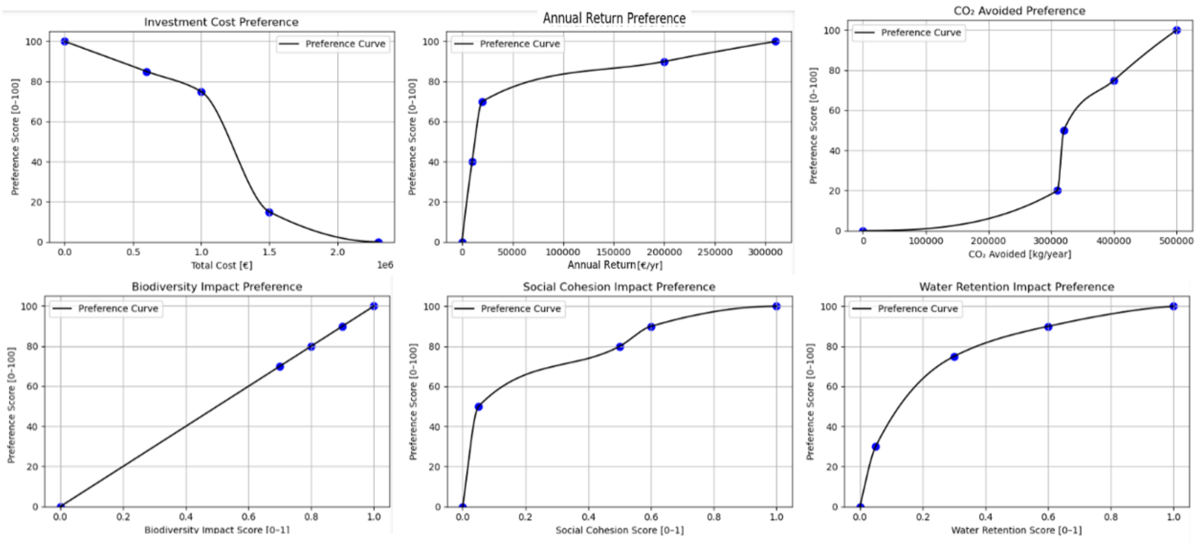


Figure 3.2: Example of preference functions, $p_i(O_i(x))$, for the six objectives O_1 to O_6

3.5. Constraints and bounds

The optimization model includes various types of constraints that define the solution space: bounds on decision variables, equality and inequality constraints, and feasibility constraints based on rooftop typology.

1. Discrete bounds

A bound restricts a variable to a range of values. For this case the decision variable x_n represents the intervention type t assigned to rooftop surface n and can take one of six discrete values:

$$x_n \in \{0, 1, 2, 3, 4, 5\}, \quad n = 1, 2, \ldots, N \tag{3.19}$$

2. Equality and inequality constraints

Additional constraints ensure technical, financial, or policy-related requirements are met. These can be formulated generically as:

$$g_j(x) \leq 0, \quad j = 1, 2, \ldots, J \tag{3.20}$$

$$h_k(x) = 0, \quad k = 1, 2, \ldots, K \tag{3.21}$$

Examples include total budget limits, technical balance rules, or required assignments of specific interventions.

3. Feasibility domain constraints

In addition to the general bounds in Equation 3.19, each rooftop surface n is restricted to a subset of feasible intervention types based on the surface slope and the maximum allowable slope for each roof type. A roof type is only considered feasible if:

$$\text{Slope}_n \leq \text{Slope}_t^{\max}, \qquad n = 1, 2, \ldots, N, \qquad i = 1, 2, \ldots, I \tag{3.22}$$

- Slope_n: Actual slope (in degrees) of rooftop surface n.
- $\mathrm{Slope}_t^{\max}$: Maximum slope (in degrees) allowed for the roof type t, as listed in Table 3.1.

3.6. Nomenclature

Symbol	**Description**	**Unit**
x_n	Decision variable: intervention type assigned to rooftop surface n (integer index from 0–5)	–
x	Design vector containing all x_n values for all rooftop surfaces	–
max_x	The optimization algorithm searches for the decision vector x that maximizes the aggregated preference score	–
N	Total number of rooftop surfaces in the portfolio	–
n	Index of rooftop surface (used to iterate over N surfaces)	–
t	Index representing one of the 6 predefined rooftop types	–
p	Index of neighborhood (used in need-based impact objectives)	–
$O_i(x)$	Objective function i, calculated over the full design vector x. Examples include investment cost, annual return, CO_2 savings, biodiversity, social cohesion, and water retention.	–
$p_i(O_i(x))$	Preference function i, translating the measured value from $O_i(x)$ into a preference score.	$\in [0, 100]$
$O_i^*(x)$	Rewritten preference function for $O_i(x)$.	$\in [0, 100]$
P^*	Aggregated preference score, computed using PFM	$\in [0, 100]$
$O_i(x)$	Objective function value for criterion i given design x	varies
w_i	Weight assigned to objective i (sum of all w_i equals 1)	–
$g_j(x)$	Inequality constraint j (e.g., budget, max area)	–
$h_k(x)$	Equality constraint k (e.g., fixed assignments)	–
Slope_n	Actual slope of rooftop surface n	degrees (°)
$\text{Slope}_\text{max t}$	Maximum allowable slope for roof type t	degrees (°)
S_n	Feasible set of intervention types for rooftop surface n based on slope	–
$x_n \in \{0, 1, 2, 3, 4, 5\}$	Bounds: rooftop n must be assigned a valid intervention type t within a particular set of options	–
$O_1(x)$	Total investment cost across all rooftop surfaces	€
A_n	Area of rooftop surface n	m²
c_t^repl	Replacement cost for roof type t	€/m²
c_t^inst	Installation cost for roof type t	€/m²
$O_2(x)$	Total annual financial return across all rooftop surfaces	€/year
r_t	Annual financial return for roof type t	€/m²/year
c_t^maint	Annual maintenance cost for roof type t	€/m²/year
$O_3(x)$	Total annual CO_2 savings across the portfolio	kg CO_2/year
κ_t	Annual CO_2 savings per m² for roof type t	kg CO_2/m²/year
Raw_impact	Unnormalized total impact score over all surfaces	–
effect_t	Effectiveness score of roof type t used in Raw Impact Score	$\in [0, 100]$
need_p	Relative need score for neighborhood p in Raw Impact Score	$\in [0, 100]$
γ	Exponent for performance sensitivity scaling in impact objectives	–
δ	Exponent for need sensitivity scaling in impact objectives	–
Max_impact	Maximum possible value for normalization of Raw Impact Score	–

Symbol	**Description**	**Unit**
$O_4(x)$	Normalized biodiversity impact score	$\in [0,1]$
$\mathrm{Raw}_{\text{bio}}$	Unnormalized biodiversity impact score	–
α_t	Biodiversity effect score of roof type t	$\in [0,100]$
N_n^{bio}	Biodiversity need score for the neighborhood of surface n	$\in [0,100]$
$\mathrm{Max}_{\text{bio}}$	Maximum value for normalizing biodiversity score	–
$O_5(x)$	Normalized social cohesion impact score	$\in [0,1]$
$\mathrm{Raw}_{\text{soc}}$	Unnormalized social cohesion impact score	–
β_i	Social cohesion effect score of roof type i	$\in [0,100]$
N_n^{soc}	Social cohesion need score for the neighborhood of surface n	$\in [0,100]$
$\mathrm{Max}_{\text{soc}}$	Maximum value for normalizing social cohesion score	–
$O_6(x)$	Normalized water retention impact score	$\in [0,1]$
$\mathrm{Raw}_{\text{wat}}$	Unnormalized water retention impact score	–
γ_i	Water retention effect score of roof type i	$\in [0,100]$
N_n^{wat}	Water retention need score for the neighborhood of surface n	$\in [0,100]$
$\mathrm{Max}_{\text{wat}}$	Maximum value for normalizing water retention score	–

Table 3.2: Nomenclature of symbols used in the optimization model

4 Operationalizing the decision support tool

4.1. Workflow Preferendus in flowcharts

The mathematical notation of this problem has been formulated in chapter 3. The Python code, which forms the core structure of the decision support model, generally follows this mathematical notation. The Python code (see Appendix B) is attached as a notebook so that the line of reasoning behind the coding can be followed.

To provide a general overview of how this decision support tool is set up for the demonstrator case discussed in chapter 5, and how decision makers can interact with the tool to obtain the results they are interested in, the first flowchart is shown in Figure 4.1. The workflow consists of five steps:

1. **Load and prepare data**
   There are three datasets: intervention roof types, roof data of the portfolio, and neighborhood data. These datasets are exported to Excel and loaded into Jupyter Notebook. The code loads these datasets as pandas DataFrames so that the optimizer can work with these values. Together, the three datasets provide all the necessary input parameters to calculate the six objectives $O_i(x)$.
2. **Define parameters in interface**
   Within the Jupyter Notebook environment, separate cell blocks form the interface. In these cell blocks, decision makers can adjust and express their objectives through preference curves, assign relative importance by selecting weights, and introduce constraints if considered necessary. It is also possible to change the algorithm settings, as described in Table 4.1.
3. **Run model**
   Once everything is set according to the decision maker's preferences (step 2), the following cells can be executed to let the model run.
4. **Model presents output**
   The output of the code provides an optimized plan with the optimized objectives and a layout of how the portfolio should be redesigned.
5. **Review optimized plan**
   As a decision maker, one can review the optimized plan. If the results are satisfactory, it is possible to export them as a PDF. If not, step 2 can be revisited to adjust preferences and re-run the model.

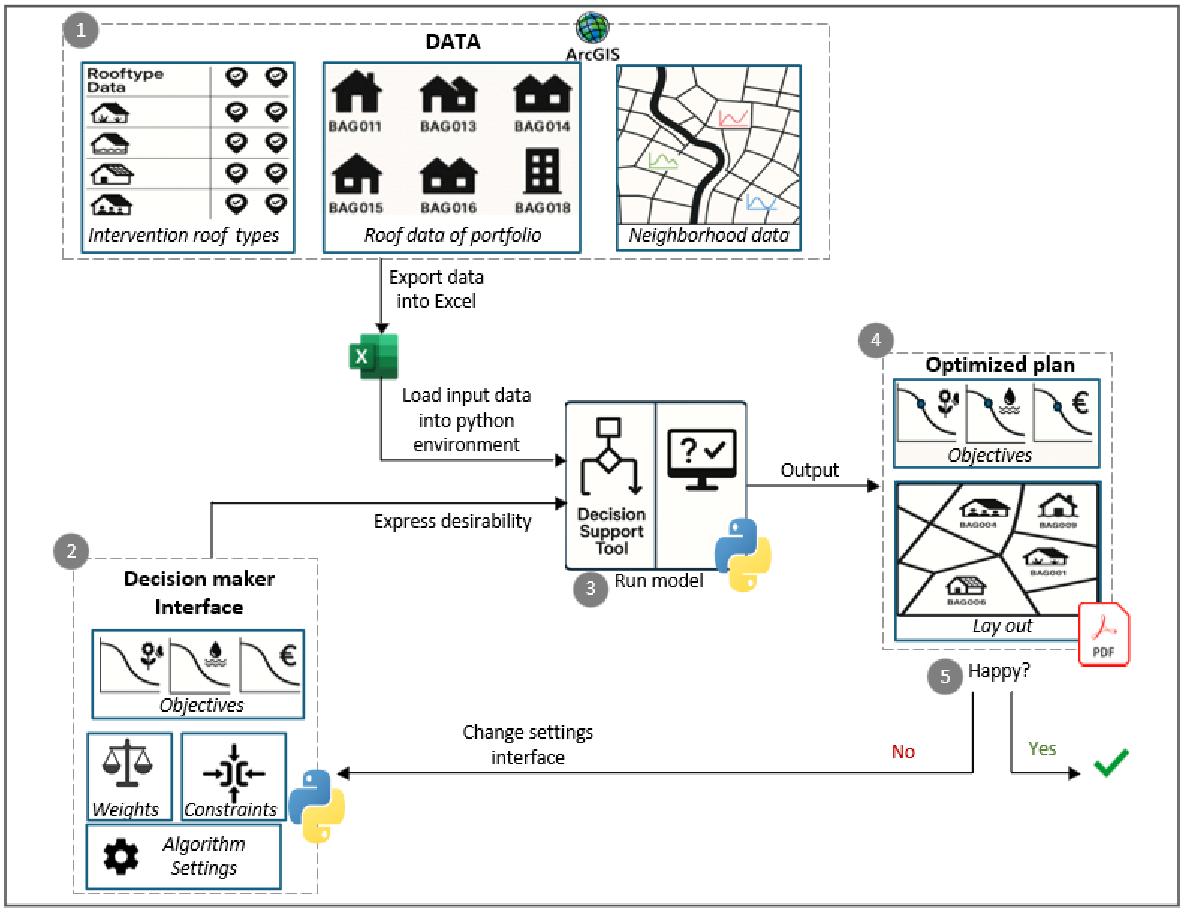
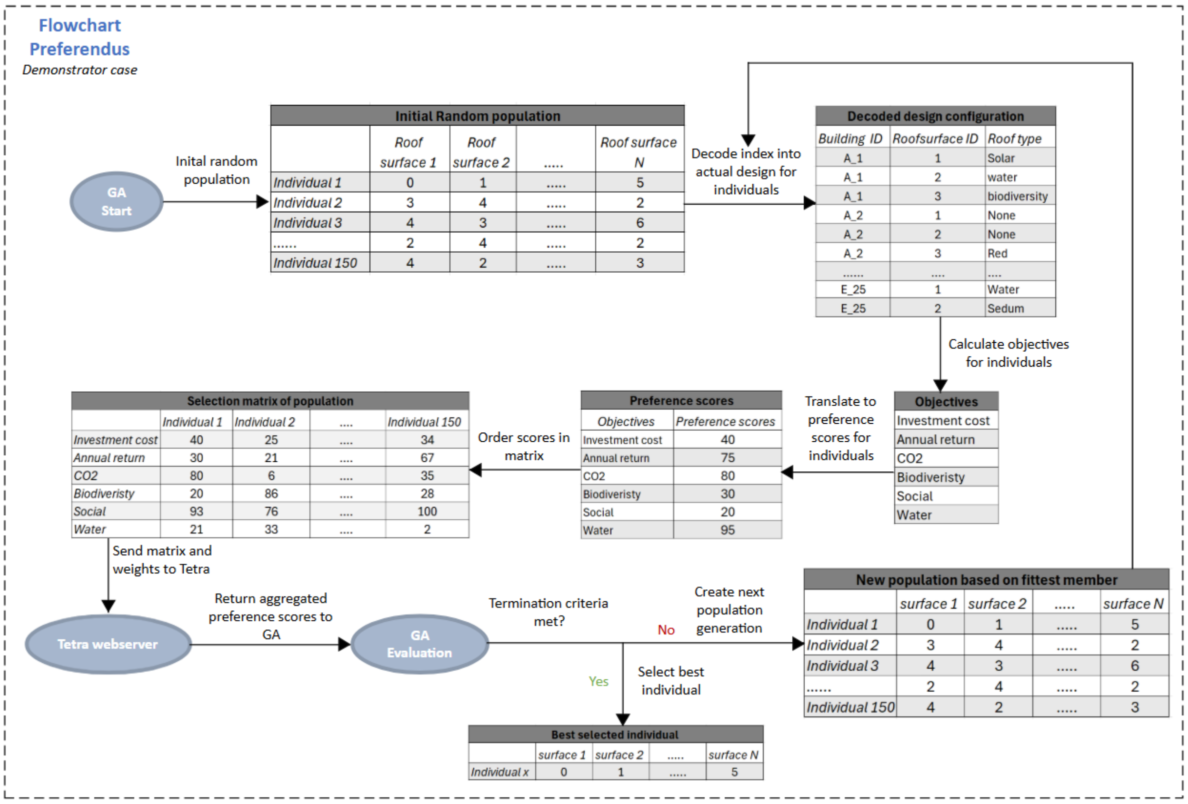


Figure 4.1: Simplified flowchart operationalizing Preferendus demonstrator case

The flowchart below (see Figure 4.2) zooms in on what happens after running the model (step 3 in Figure 4.1) and shows how the algorithm converges towards an optimal individual. It provides a clearer understanding of how the algorithm works and converges to one final optimal design configuration. In this case, the final design specifies which roof types t a portfolio owner should allocate to which buildings, based on the selected objectives, weights, constraints, and preferences.


Flowchart
Preferendus
Demonstrator case
GA Start
Inital random population
Decode index into actual design for individuals
Calculate objectives for individuals
Translate to preference scores for individuals
Order scores in matrix
Send matrix and weights to Tetra
Tetra webserver
Return aggregated preference scores to GA
GA Evaluation
Termination criteria met?
No
Create next population generation
Yes
Select best individual

Initial Random population				
	Roof surface 1	Roof surface 2	.....	Roof surface N
Individual 1	0	1	.....	5
Individual 2	3	4	.....	2
Individual 3	4	3	.....	6
......	2	4	.....	2
Individual 150	4	2	.....	3

Decoded design configuration		
Building ID	Roofsurface ID	Roof type
A_1	1	Solar
A_1	2	water
A_1	3	biodiversity
A_2	1	None
A_2	2	None
A_2	3	Red
......	....	....
E_25	1	Water
E_25	2	Sedum

Objectives
Investment cost
Annual return
CO2
Biodiveristy
Social
Water

Preference scores	
Objectives	Preference scores
Investment cost	40
Annual return	75
CO2	80
Biodiveristy	30
Social	20
Water	95

Selection matrix of population				
	Individual 1	Individual 2	....	Individual 150
Investment cost	40	25	....	34
Annual return	30	21	....	67
CO2	80	6	....	35
Biodiveristy	20	86	....	28
Social	93	76	....	100
Water	21	33	....	2

New population based on fittest member				
	surface 1	surface 2	.....	surface N
Individual 1	0	1	.....	5
Individual 2	3	4	.....	2
Individual 3	4	3	.....	6
......	2	4	.....	2
Individual 150	4	2	.....	3

Best selected individual				
	surface 1	surface 2	.....	surface N
Individual x	0	1	.....	5

Figure 4.2: Simplified flowchart Preferendus algorithm demonstrator case

4.2. Algorithm settings

Algorithm settings that were used to run the model are displayed in Table 4.1. Changing the values of these parameters can heavily effect the final result of the MOO.

Parameter	Value	Description
`n_bits`	4	Number of bits used to encode each variable
`n_iter`	100	Number of iterations (generations)
`n_pop`	150	Population size
`r_cross`	0.8	Crossover rate
`max_stall`	10	Max. generations without improvement before early stopping
`aggregation`	`tetra`	Aggregation method for preference scores
`var_type`	`int`	Variable encoding type

Table 4.1: Genetic algorithm settings used in the optimization model

5 Demonstrator case results and evaluation

5.1. Data collection demonstration case

Selecting the demonstrator case: The Hague

To test the developed decision support tool, a demonstrator case was selected to ensure the proof of concept could be grounded in real-world spatial and asset data. The city of The Hague was chosen for two reasons: (1) its public real estate portfolio data was available, and (2) it includes diverse urban neighborhoods with varied environmental and social challenges, making it a suitable test case for evaluating rooftop interventions at the portfolio level.

Data availability and scope

A critical first step was to assess what data is publicly available and relevant to the types of rooftop challenges and functions explored in previous studies (see chapter Analysis) (e.g., storm water management, urban heat mitigation, biodiversity enhancement, and social cohesion).

Relevant challenge indicators were found to be available at the neighborhood level for all 114 neighborhoods in The Hague. These include:

- Heat stress
- Storm water runoff
- Biodiversity
- Social cohesion

By linking this data spatially to neighborhoods, it became possible to quantify the degree of local urgency and identify areas where rooftop interventions could have the most contextual value. For each challenge indicator, data were normalized on each neighborhood need to a scale from 0 to 100, where 100 indicates the neighborhood with the highest need for intervention and 0 the lowest.

To visualize the spatial variation in challenges, choropleth maps were generated in Python (see Appendix C and Figure 5.1). A choropleth is a thematic map where areas (in this case, neighborhoods) are shaded in proportion to the value of a particular variable, such as heat stress or green space deficiency. These visualizations provide an intuitive overview of where urban interventions are most needed. The data of these visualizations, need_p, is used in objectives $O_4(x)$, $O_5(x)$, and $O_6(x)$.

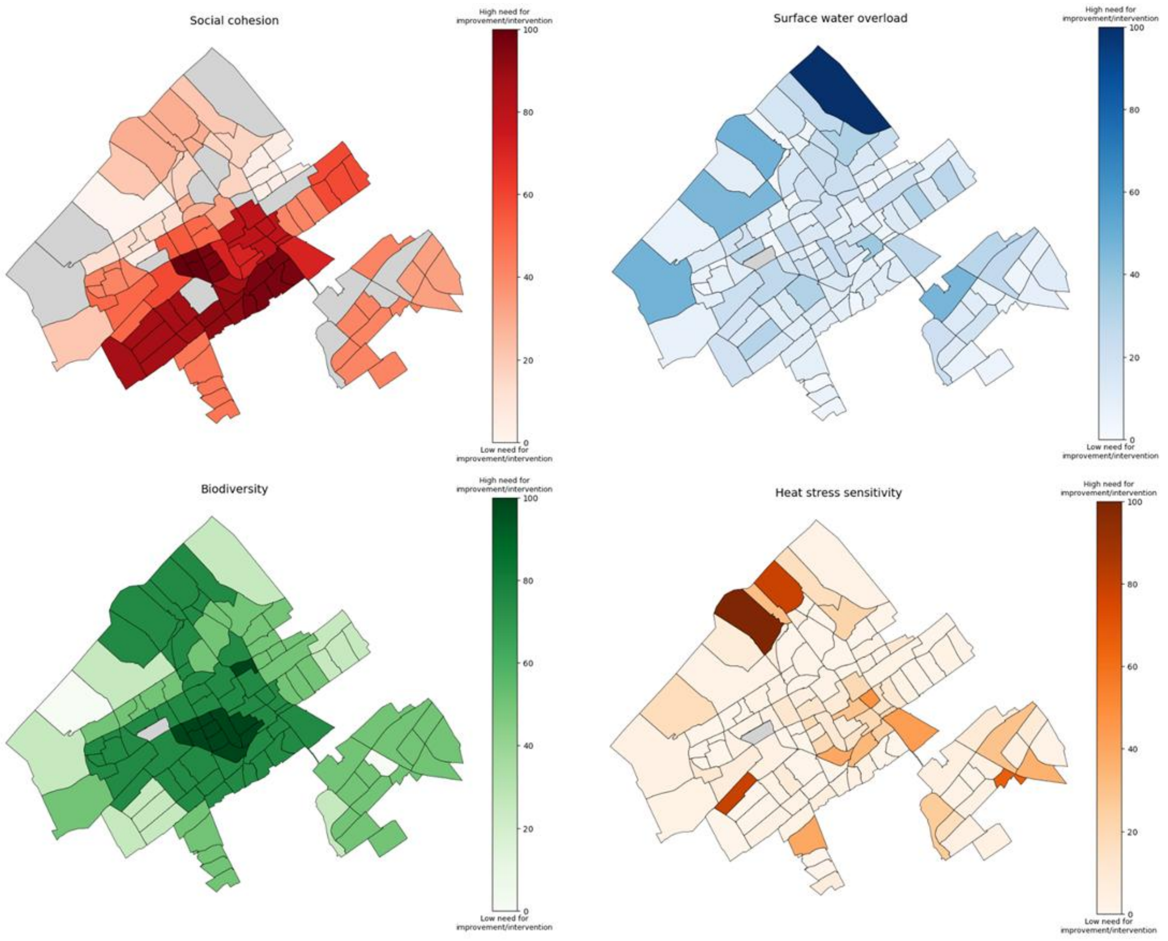


Figure 5.1: Choroplets: Social cohesion, Surface water overload, Biodiversity, Heat stress sensitivity

By overlaying the building footprints of the portfolio (see black dots) onto the challenge-based choropleths, it becomes clear which buildings are located in high-need areas (see Figure 5.2). This spatial intersection is key for portfolio owners: it helps them visualize which assets not only have technical potential but are also situated in contexts where their impact could be maximized.

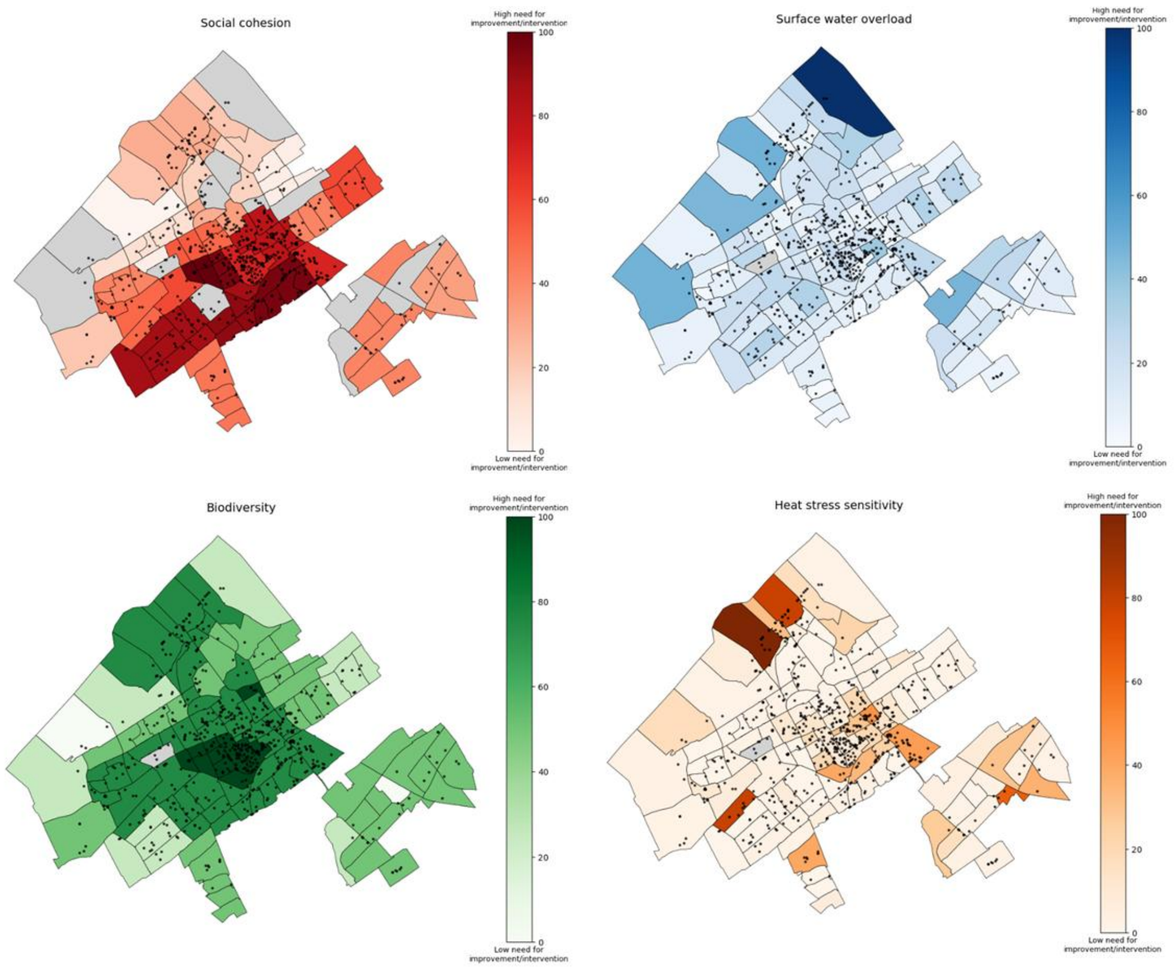


Figure 5.2: Choroplets with overlay of building portfolio : Social cohesion, Surface water overload, Biodiversity, Heat stress sensitivity

Filtering roof data using BAG ID

Unlike previous rooftop optimization studies that evaluate all buildings in a city, this thesis focuses on specific portfolios of buildings, mimicking the decision-making context of a portfolio owner such as a municipality or housing association. To do this, detailed roof data had to be filtered for a specific selection of buildings.

This was achieved using the BAG ID, a unique identifier assigned to each registered building in the Netherlands. With this ID, it became possible to query high-resolution roof datasets and extract all relevant data for only the buildings within the portfolio.

A Python script was used to loop through the dataset and select roof data for each BAG ID in the chosen portfolio. This yielded a portfolio-specific dataset containing:

- Number of roof surfaces per building
- Surface area (m^2)
- Slope (in degrees)
- Polygon geometry (roof outline shape)

Visualizing extracted roof data: different roof surfaces n

The figure below shows one building from the dataset. The image displays each roof surface n with its shape, size, and slope. This information helps decide which roofs are suitable for which interventions, for example, flat roofs for green or blue roofs, and sloped, south-facing roofs for solar panels (see

Figure 5.3). The roof data for the portfolio was derived from the link of this reference (ArcGIS Online, 2025).

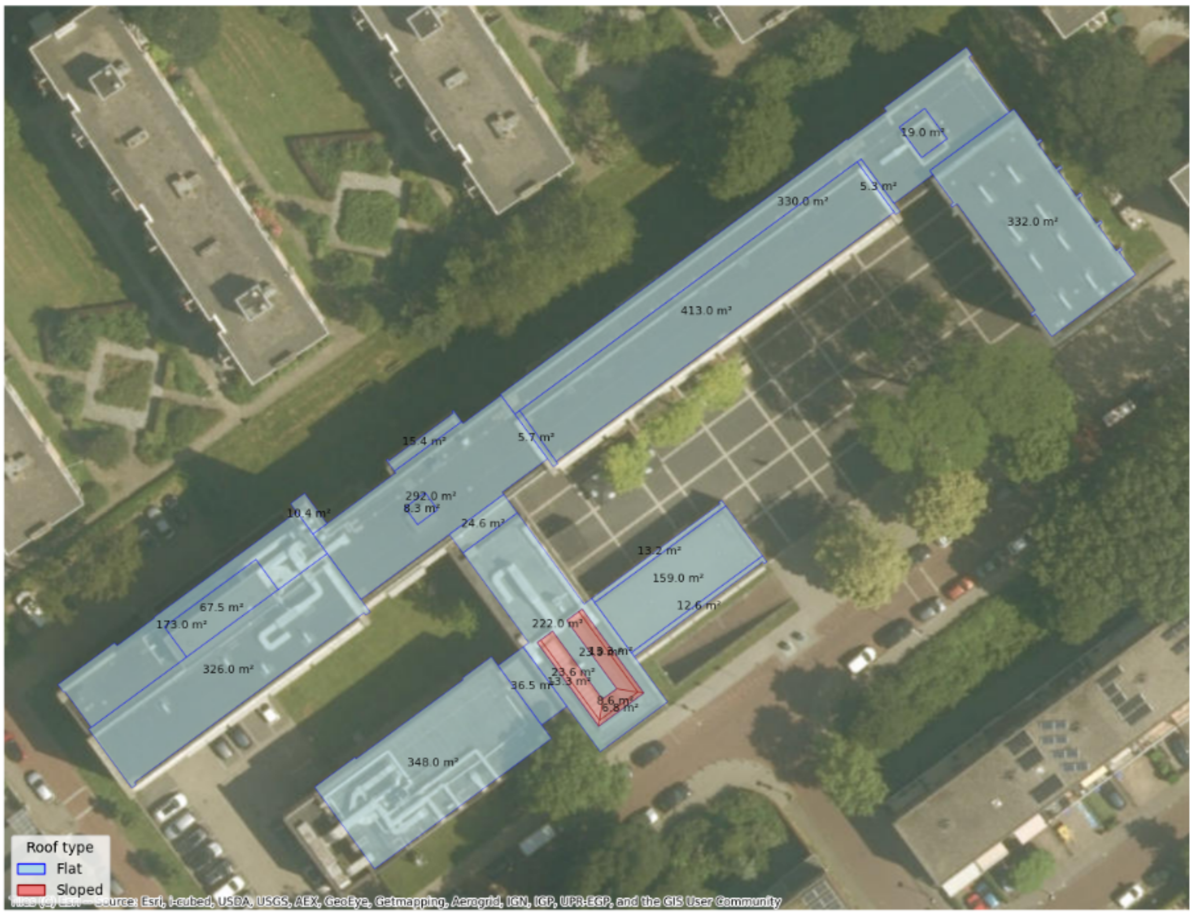


Figure 5.3: Sample plot of roofsurfaces n of one building from portfolio

5.2. Demonstrator case using simplified sample dataset

To support model development and validate its functionality before applying it to real-world data, a simplified sample dataset has been constructed. This fictive dataset replicates the structure of the actual dataset described in Section 2.2, including the same design variables and rooftop characteristics. It offers a controlled, low-complexity environment to verify whether the model behaves as intended, facilitates rapid iteration, and simplifies debugging in case of errors or unexpected outputs.

By working with this simplified dataset, it becomes possible to efficiently test and interpret model behavior, explore how different configurations influence outcomes, and fine-tune both the model logic and the visualization of results.

Portfolio and building setup: roof surfaces n

In the demonstrative scenario, a fictive portfolio owner manages a portfolio of 25 buildings. These buildings are distributed across five neighborhoods, labeled A through E, with each neighborhood containing five buildings (e.g., A_1 refers to Building 1 in Neighborhood A). Each building consists of one or more distinct roof surfaces n, each characterized by different roof surfaces in shape, size and slope (see Figure 5.4).

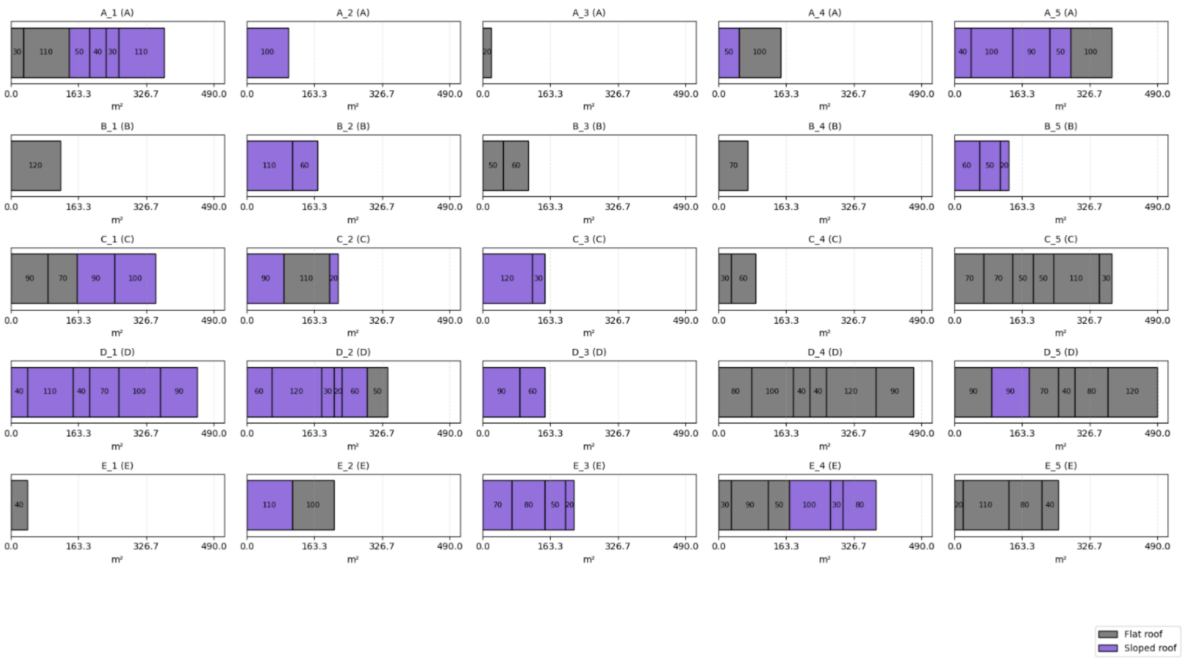


Figure 5.4: Roof surfaces n of the building portfolio of 25 Buildings in Neighborhoods A–E

Neighborhood challenge data: need_p

To reflect varying urban needs, each of the five neighborhoods is assigned normalized scores (0–100) for: biodiversity, storm water overload, and social cohesion. Within each category, the highest-need neighborhood scores 100, and the lowest scores 0. These scores enable comparison across neighborhoods within each domain, but are not aggregated across categories. Combining them would violate measurement theory and lead to flawed conclusions, as discussed previously (Barzilai, 2005, 2010; Barzilai, 2022). Instead, this input data forms the need_p for each of the Objectives $O_4(x)$, $O_5(x)$, and $O_6(x)$.

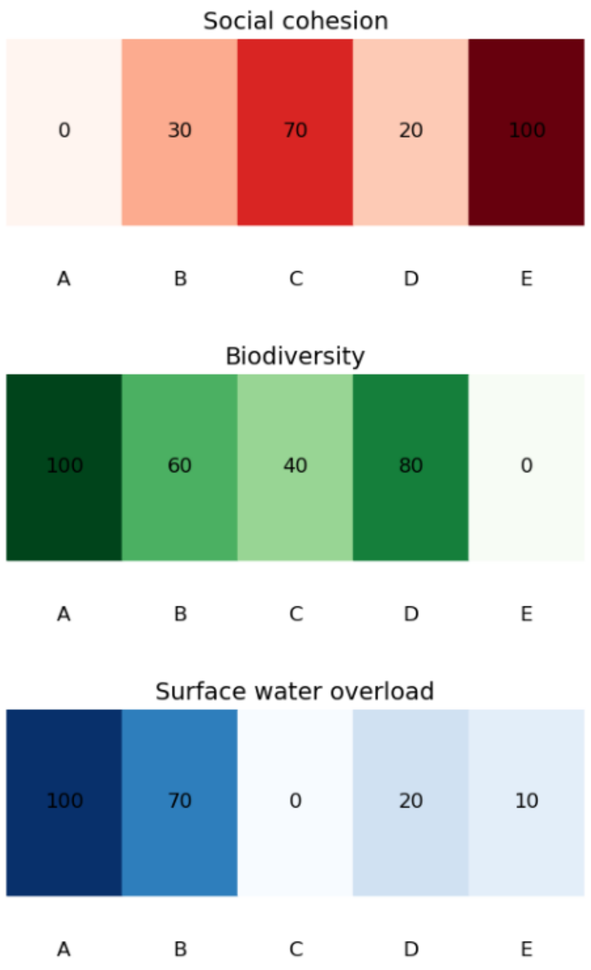


Figure 5.5: Normalized neighborhood needs need_p and visual representation of urban challenges

The bars (see Figure 5.5) display the normalized needs (0–100) for each neighborhood (A–E) across three urban challenges:

- For social cohesion, the highest need is in Neighborhood E (100), and the lowest need is in Neighborhood A (0).
- For biodiversity, the highest need is in Neighborhood A (100), and the lowest need is in Neighborhood E (0).
- For storm water overload, the highest need is in Neighborhood A (100), and the lowest need is in Neighborhood C (0).

Bringing data together

Plotting the portfolio across the neighborhoods, as shown in (see Figure 5.6), reveals how assets are distributed in areas facing the greatest challenges.

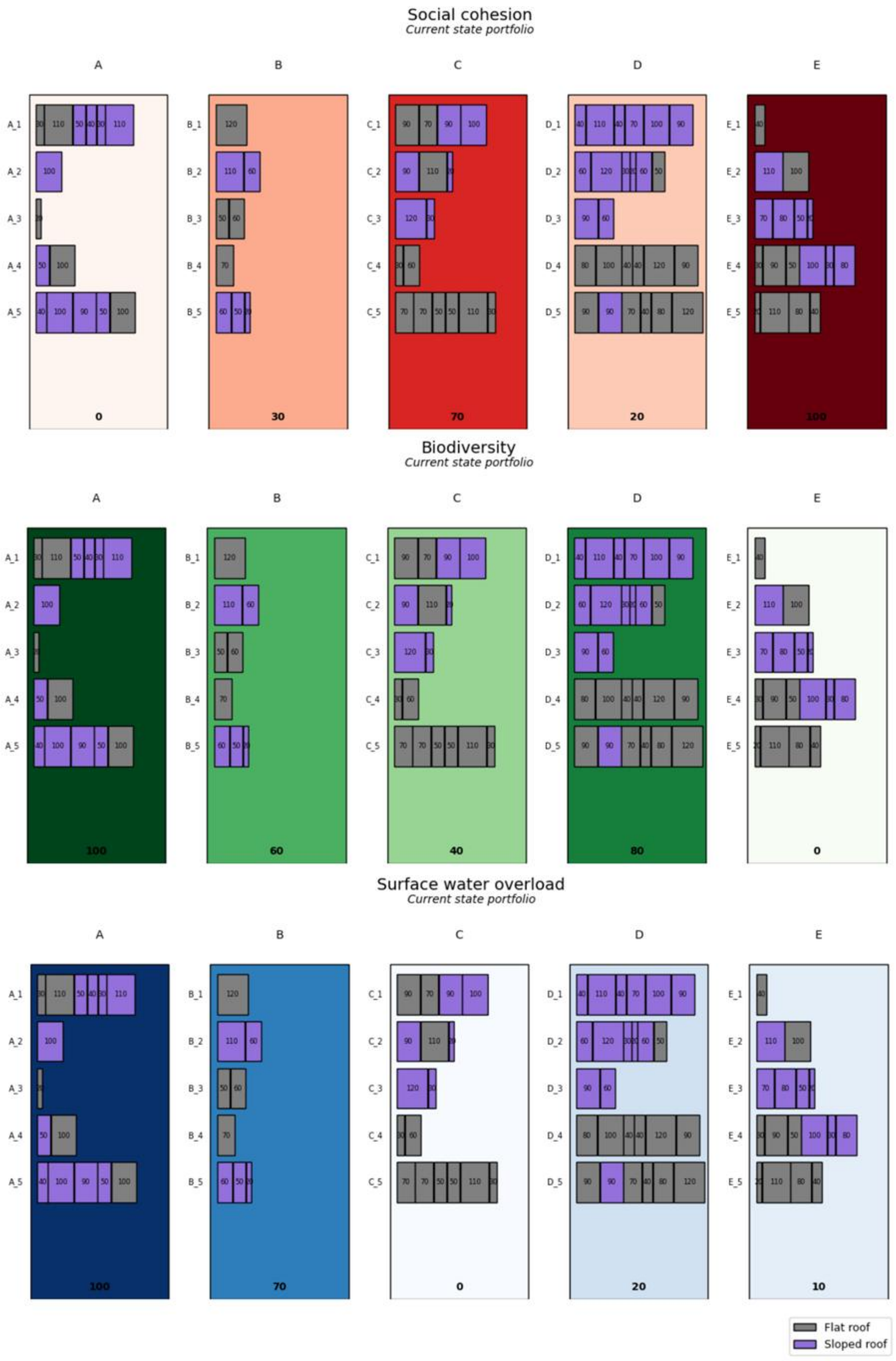


Figure 5.6: Building portfolio of 25 buildings across Neighborhoods A–E, plotted against normalized neighborhood needs.

Design variables

The possible interventions (roof types/design variables) (Figure 3.1) remain unchanged, reflecting the options available in the real-world case.

Design solution space

As shown in constraint (3.22), each roof surface n can only be assigned intervention types whose maximum allowed slope is not exceeded.This constraint excludes roof types that are not physically feasible due to slope limitations. For the dataset used in this model, this results in the following feasible type sets:

$$S_n = \begin{cases} \{0,1,5\} & \text{if surface } n \text{ is sloped (based on dataset)} \\ \{0,1,2,3,4,5\} & \text{if surface } n \text{ is flat} \end{cases} \tag{5.1}$$

These sets are not universal rules, but follow from applying slope-based filtering to the current dataset and result in the possibilities shown in (see Figure 5.7).

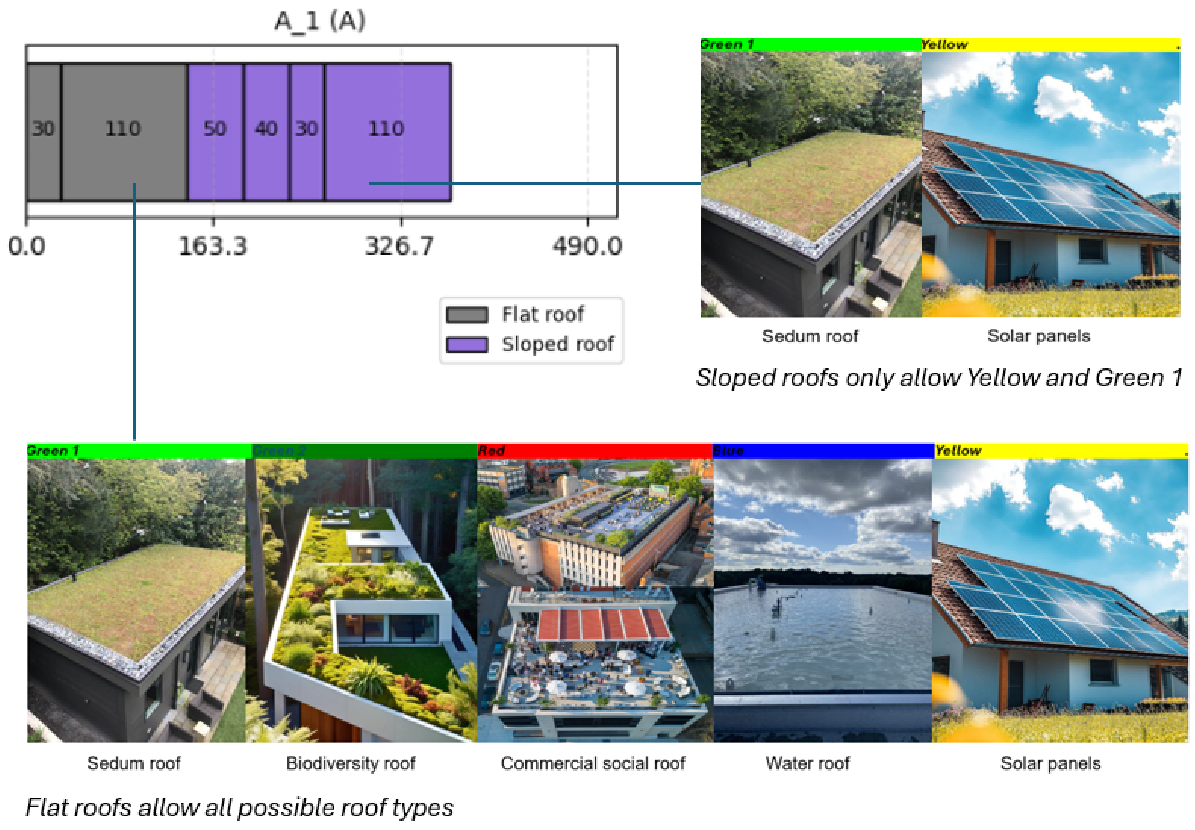


Figure 5.7: Intervention feasibility based on current roof condition

5.3. Results demonstrator case

The demonstrator case was used to validate the model. This was done during a workshop (see the next section for more information on the validation). In this workshop, nine iterations, or runs, were carried out to arrive at a final design configuration that satisfied the portfolio owner's objectives. The results are presented in this section.

Before running the model, all maximum values were calculated (see Table 5.1). These values give the portfolio owner an understanding of the potential range of outcomes that the objectives can take.

Objective function	Minimum value	Maximum value	Unit
Surface water overload	0	1	[-]
Biodiversity	0	1	[-]
Social cohesion	0	1	[-]
Avoided CO_2	0	495476.8	[kg/year]
Investment cost	0	2,258,150	[€]
Annual return	0	310,242.5	[€/year]

Table 5.1: Minimum and maximum values for each objective function used in the optimization model.

Figure 5.8 contains the optimized raw values and weights for the objectives of the final (9th) iteration of the workshop. The iterations and an interpretation of the results are extensively explained within the appendix (see Appendix A).

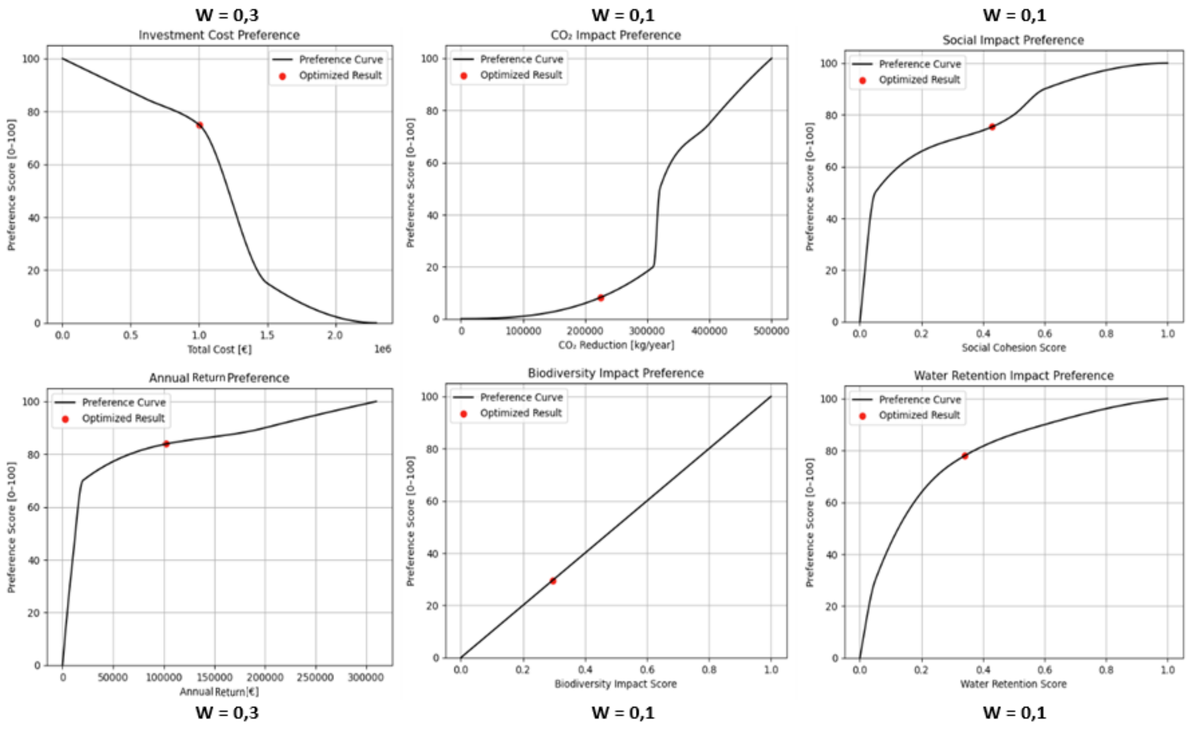


Figure 5.8: Optimized preference functions, $p_i(O_i(x))$, for the six objectives O_1 to O_6

Table 5.2 and Table 5.3 give some summary statistics of the allocated roof types.

Indicator	**Raw value**	**Preference score**	**Weight**
Cost [€]	1,002,200	74.9	0.30
Annual return [€]	102,238	83.9	0.30
CO_2 reduction [kg]	224,286	8.2	0.10
Biodiversity	0.2958	29.6	0.10
Social cohesion	0.4301	75.4	0.10
Water retention	0.3413	78.1	0.10
Overall preference score		**66.75**	

Table 5.2: Summary of optimized objectives and preferences.

Category	**Metric**	**Roof type**	**Value**	**Unit**
Summary	Total roof area assigned	None	5,810.00	m²
Summary	Total roof surfaces	None	84	count
Summary	No intervention (%)	None	31.0	%
Area per type	Area	Water retention	140.00	m²
Area per type	Area	Solar panels	2,630.00	m²
Area per type	Area	No intervention	1,860.00	m²
Area per type	Area	Sedum	230.00	m²
Area per type	Area	Biodiversity	430.00	m²
Area per type	Area	Commerical social	520.00	m²
Count per type	Count	Solar panels	38	surfaces
Count per type	Count	No intervention	26	surfaces
Count per type	Count	Commerical social	9	surfaces
Count per type	Count	Biodiversity	6	surfaces
Count per type	Count	Sedum	3	surfaces
Count per type	Count	Water retention	2	surfaces

Table 5.3: Summary of assigned optimized roofs.

The final configuration of the roof allocation over the neighborhoods is shown below (see Figure 5.9).

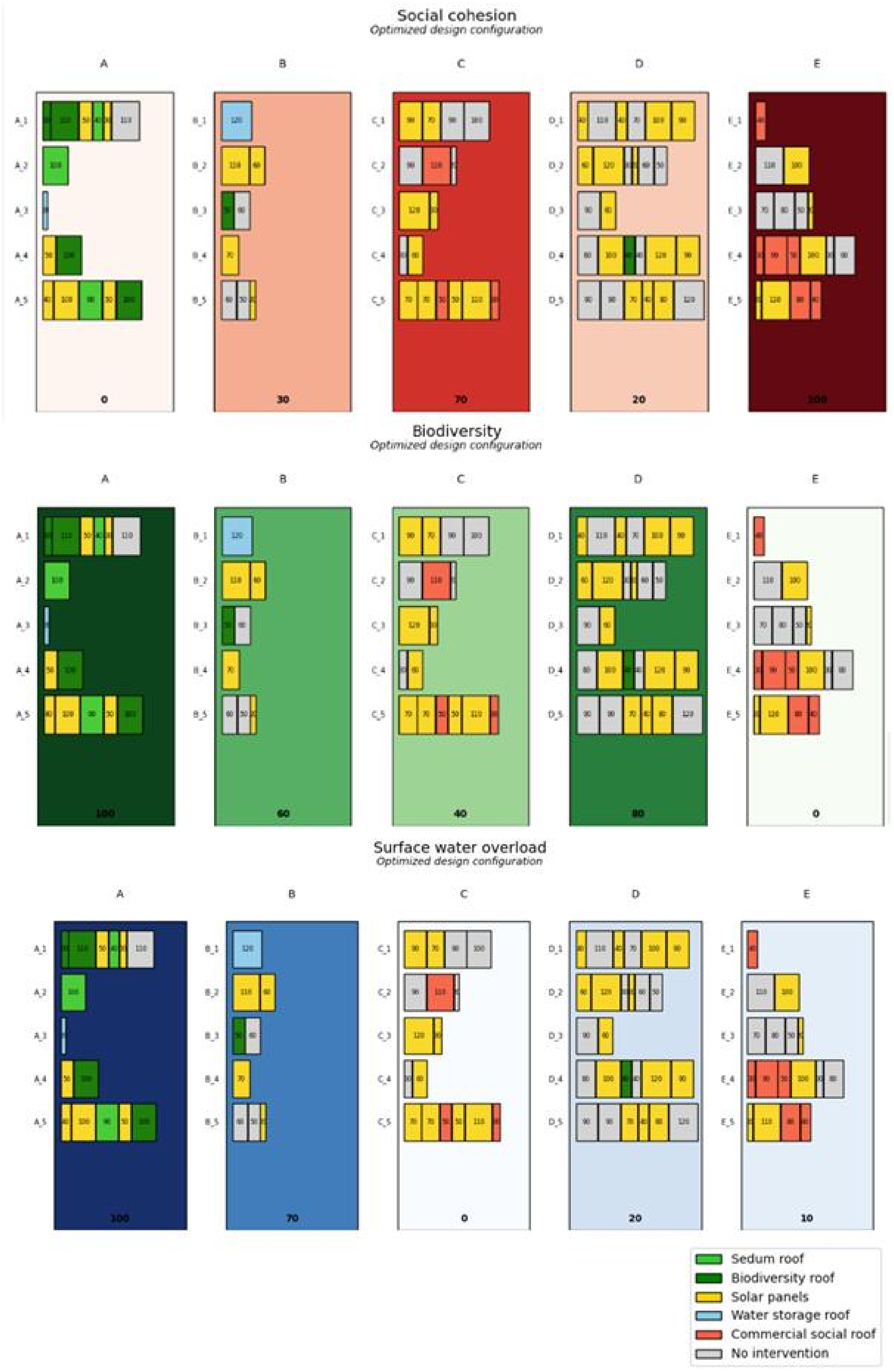


Figure 5.9: Iteration 9: final configuration of the workshop

5.4. Validation of the decision support tool

The model has been validated during a validation workshop with Stefan van de Schootbrugge, founder of the software company Bress, which has developed a software tool for real estate portfolio owners to query and link open-source data directly to buildings within the portfolio. With major clients like the municipalities of The Hague and Amsterdam, and a background in real estate management, Stefan was well-positioned to assess the model.

This validation tested both the iterative nature of the framework proposed by Zhilalyv et al. (2022) and its practical applicability for portfolio owners. The method is designed for flexibility: objectives and constraints can be added, and preference weights adjusted to reflect stakeholder priorities. During the session, Stefan acted as a portfolio owner working with municipal sustainability goals. For an extensive summary refer to Appendix A.

Stefan emphasized two core strengths of the model. First, the ability to iterate input in real time enabled flexible exploration of design solutions during the workshop. Second, the use of preference functions required decision-makers to explicitly articulate how they value outcomes across different objectives, making their priorities tangible and comparable. However, he highlighted that without a simple, user-friendly interface, municipal portfolio owners would struggle to use the tool, as it currently requires programming knowledge. These main findings were agreed upon by Arcadis' real estate portfolio sustainability team.

At present, Arcadis is conducting a portfolio optimization for the municipality of Groningen using a software tool called Enterprise Decision Analytics (EDA). However, one of the main issues is that it is difficult to model input from decision makers in real time. This was mainly due to the absence of an EDA expert at the workshop, but it still highlights how highly relevant real-time iterative modeling, as enabled by Preferendus, can be.

6 Discussion

6.1. Limitations

The limitations of this study are mainly related to how the objective functions are defined and to the quality of the input data. In particular, the impact objectives $O_4(x)$, $O_5(x)$, and $O_6(x)$, representing biodiversity, social cohesion, and water retention, could be modeled in a more realistic manner. These objectives rely on $need_p$ as an input parameter, expressed as a normalized neighborhood need, $\in [0, 100]$. For $O_6(x)$, which represents water retention, this is problematic because normalizing the need of an entire neighborhood does not adequately capture where the actual challenges occur. The effect of roofs on water retention is highly localized, and using an average across the neighborhood therefore oversimplifies the situation. For biodiversity and social cohesion this issue is less pronounced, since the underlying data is expressed as a percentage of area. Still, working with averages always carries risks, and more accurate neighborhood modeling would be possible, as demonstrated in studies such as Dong et al. (2024), Yuan et al. (2025), and Xiong et al. (2023).

For the solar panels, more realistic constraints could have been set. Namely, all energy generated by solar panels should ideally match the energy demand for that building. If solar panels yield more energy, the additional energy surplus should be fed back into the electricity grid. However, in some neighborhoods, due to net congestion, it cannot be fed into the grid. This would restrict feasibility of installing as many solar panels as needed by the model.

6.2. Steps for further development

The current model shows a proof of concept and technical foundation, but there are some steps for further development or further research in the applicability of the Preferendus domain:

- **User-friendly interface**
  One of the key concerns expressed by Stefan and professionals from Arcadis was that the Preferendus should have a user-friendly interface to ensure it can be applied in practice. Since the current decision support tool is written in Python code, someone who understands coding logic always needs to be present during stakeholder sessions to enable iterative modeling. A user-friendly, interactive interface would therefore be highly recommended for real-world application.
- **Introduction of Preferendus to stakeholders**
  The mathematical modeling behind the tool might increase reluctance to adopt it, especially among users without a technical background, because the tool could be perceived as a black box. However, this is not the case: Preferendus functions more as an open glass iterative decision-making compass, guiding users step-by-step toward an optimal design (Wolfert, 2023). Research could be conducted into the most effective ways to introduce Preferendus and preference function modeling, and work with stakeholders with non-technical backgrounds, to ensure the tool is accessible and useful as a decision support system.

- **Function modeling refinements**
  As discussed in the previous paragraph, objective functions and constraints could be modeled more realistically.
- **Algorithm settings**
  Additional research should be done on how one can easily determine optimal algorithm settings in relation to running time and their influence on the final results. Especially the population size and the crossover rate influence how algorithms explore the solution space. A larger population encourages better exploration but results in longer running time. A higher crossover rate promotes more diversity and exploration, while a lower rate focuses more on exploiting current good solutions. However, it remains difficult to understand how the algorithm can most efficiently generate a solution that reaches the global optimum, rather than getting trapped in a local optimum.
- **Roadmap-orientated asset management**
  Asset management in general, but also of buildings take time. Therefore often within asset management maintenance or improvement of assets due to interventions is set out in time. The time component and which asset should be prioritized over time is missing. The current output of this model for asset management shows what the optimal design should look like but doesn't provide information on how to get there over time in a time planning table. Further research could be done on how this time component can be integrated and the Prefrendus doesn't only show an optmal design but also converges towards a roadmap. This feature is already present in Arcadis' Enterprise Decision Analytics (EDA), where it is highly valued in the field of asset maintenance management. Latifi et al. (2021) can be an interesting case studie to look at. They developed a machine learning decision support system which optimizes multiple objectives and provides a 20-year roadmap as output of what maintenance type to do and when to apply it on an asset.
- **Connection to real-time data sources**
  Further research could explore how the tool can be connected to regularly updated data sources such as QGIS or ArcGIS, so that the output remains current without the need for manual data updates. A comparable approach is used in Bress, which queries data directly from the server in real time.
- **Exploring Preferendus in other domains**
  Lastly, research or tool development could also extend beyond rooftops, for example, to public space, transport infrastructure, or energy systems or other fields within engineering. This would broaden the scope of Preferendus and strengthen its position as a general decision support framework.

6.3. Recommendations in practice

The tool proved successful in designing an optimized building portfolio asset management plan. Roofs were used as a demonstrator case, but Preferendus can also be applied to other portfolio problems, as shown by Arkesteijn (2019) and De Visser et al. (2017) in earlier case studies such as office location selection for Oracle or the transformation and upgrading of campus buildings.

From a consultancy perspective, using Preferendus or similar MOO software (e.g., EDA) is especially valuable. In stakeholder sessions, Key Performance Indicators (KPIs) are often defined *a priori* to track current performance and desired improvements. These KPIs can be translated into objectives and therefore preference curves, while interventions, typically requiring investments, serve to improve them. Preferendus can help to steer these KPIs in the desired direction. Preferendus or EDA is recommended when:

1. Large datasets are involved.
2. Objectives conflict and trade-offs must be explored.
3. The design space is complex, making manual scenarios unlikely to yield optimal results.

7 Conclusion

The developed decision support tool proved successful in addressing the methodological gaps identified in the introduction. The main challenge for portfolio owners was to select impactful interventions in neighborhoods most in need and apply them efficiently on their building assets, and by doing so to create an optimal design configuration that satisfies their objectives.

A large part of the complexity in this problem lies in gathering and connecting the right data. An important step is the collection and preparation of data for the design problem. First, data on neighborhood needs must be collected and cleaned. Once prepared, this data should be visually mapped in a choropleth, to identify spatial variation. However, simply mapping neighborhood challenges is not enough; portfolio-building data must also be collected and plotted. Plotting these assets already provides a clearer picture of where potential actions can be taken. Eventually, to make the link between building assets, interventions, and neighborhood needs, one needs to include data of these three components within objective functions.

The tool was tested on a fictive sample dataset mimicking the same structure as the original datasets of the Hague. This allowed the decision support tool to be developed in a controlled way. It's important to note that this was a highly simplified demonstrator case, but that in reality, the Municipality of The Hague owns around 800 buildings, with each having 20–30 roof surfaces of varying sizes across 114 neighborhoods. In such cases, the design space becomes exponentially larger, and designing one's own solutions might lead to suboptimal results. Additionally, complexity also increases once one uses more realistic objective functions, decision variables or constraints which better model 'real world' context.

To test the model's functionality, nine iterations were run in a stakeholder workshop, simulating how the tool could be used interactively. There were two core strengths of the model. First, it can generate and present new configurations in real time, allowing portfolio owners to adjust preferences, introduce new objectives, weights, or constraints during the session and immediately observe how the optimal design evolves over iterations. Second, the use of preference functions requires decision-makers to explicitly articulate how they value outcomes across different objectives, making their priorities both tangible and comparable. Both of these strengths were highly valued by professionals from portfolio strategies within Arcadis and Bress.

Of course, as mentioned in the section on further development, this decision support tool still needs improvements before it can realistically model full-scale urban portfolio scenarios. However, it demonstrates its applicability in portfolio asset management and spatial allocation problems. Even though this test was performed in a portfolio context, the method could theoretically be applied to all buildings in a city. By using ODESYS principles, this tool showcases that the methodological shortcomings in the discussed MOO rooftop studies, could theoretically be addressed.

References

Arcadis. (2024). Sustainable cities index 2024 [Accessed July 3, 2025].

ArcGIS Online. (2025). Dakvlakken2d_plus_1set_gdb featureserver [ArcGIS Map Viewer. Accessed: 2025-09-11]. https://services1.arcgis.com/sbwSZYQOs4GMs7tr/ArcGIS/rest/services/dakvlak ken2d_plus_1set_gdb/FeatureServer

Arkesteijn, M. (2019). *Corporate real estate alignment: A preference-based design and decision approach* [Doctoral dissertation, TU Delft] [Doctoral thesis]. https://journals.open.tudelft.nl/abe/article/view/4125

Barzilai, J. (2005). Measurement and preference function modelling. *International Transactions in Operational Research*, *12*(2), 173–183. https://doi.org/10.1111/j.1475-3995.2005.00496.x

Barzilai, J. (2010). Preference function modelling: The mathematical foundations of decision theory. In M. Ehrgott, J. R. Figueira, & S. Greco (Eds.), *Trends in multiple criteria decision analysis* (pp. 57–86). Springer. https://link-springer-com.tudelft.idm.oclc.org/chapter/10.1007/978-1-4419-5904-1_3

Barzilai, J. (2022, April 5). *Pure economics*. FriesenPress.

Binnekamp, R. (2006). *Open design: A stakeholder-oriented approach in architecture, urban planning, and project management* (Vol. 1). IOS Press. https://research-ebsco-com.tudelft.idm.oclc.org/c/54fx7q/search/details/notyylqy75?db=nlebk

Binnekamp, R. (2010). *Preference-based design in architecture*. IOS Press. https://www.researchgate.net/publication/43796818_Preference-Based_Design_in_Architecture#fullTextFileContent

Bouwstenen voor Sociaal. (2024). *Actualisatie vastgoedstrategie* (Rapport / PDF) (Geraadpleegd via bouwstenen.nl op 3 juli 2025). Bouwstenen voor Sociaal. https://bouwstenen.nl/sites/default/files/uploads/actualisatie_vastgoedstrategie.pdf

Brenner, J., Schmidt, S., & Albert, C. (2023). Localizing and prioritizing roof greening opportunities for urban heat island mitigation: Insights from the city of krefeld, germany. *Landscape Ecology*, *38*(7), 1697–1712. https://doi.org/10.1007/s10980-023-01644-8

College van Rijksadviseurs. (2022). Nederlands verdichtingsverleden in kaart: 15 jaar verdichting, verdunning en krimp [Data-analyse op basis van CBS□buurtcijfers 2005–2020].

Dakenplan, S. N. (2025, July 5). *Nationaal dakenplan* [Accessed 5 July 2025]. Nationaal Dakenplan. https://dakenplan.nl/

De Visser, H., Arkesteijn, M., Binnekamp, R., & de Graaf, R. (2017). Improving cre decision making at oracle: Implementing the pas procedure with a brute force approach. *24th Annual Conference of the European Real Estate Society (ERES 2017)*. https://pure.tudelft.nl/ws/portalfiles/portal/51464195/P_20170114194419_111_1.pdf

Dong, J., Guo, F., Lin, M., Zhang, H., & Zhu, P. (2022). Optimization of green infrastructure networks based on potential green roof integration in a high-density urban area—a case study of beijing, china. *Science of The Total Environment*, *834*, 155307. https://doi.org/10.1016/j.scitotenv.2022.155307

Dong, J., Guo, R., Lin, M., Guo, F., & Zheng, X. (2024). Multi-objective optimization of green roof spatial layout in high-density urban areas—a case study of xiamen island, china. *Sustainable Cities and Society*, *115*, 105827. https://doi.org/10.1016/j.scs.2024.105827

Gemeente Rotterdam. (2025). *Gemeentelijk vastgoed* [Online; accessed 3 July 2025]. https://www.rotterdam.nl/gemeentelijk-vastgoed

Hurlimann, A., Moosavi, S., & Browne, G. R. (2021). Urban planning policy must do more to integrate climate change adaptation and mitigation actions. *Land Use Policy*, *101*, 105188. https://doi.org/10.1016/j.landusepol.2020.105188

Kumar, S., Guntu, R. K., Agarwal, A., Villuri, V. G. K., Pasupuleti, S., Kaushal, D. R., Gosian, A. K., & Bronstert, A. (2022). Multi-objective optimization for stormwater management by green-roofs and infiltration trenches to reduce urban flooding in central delhi. *Journal of Hydrology*, *606*, 127455. https://doi.org/10.1016/j.jhydrol.2022.127455

Langemeyer, J., Wedgwood, D., McPhearson, T., Baró, F., Madsen, A. L., & Barton, D. N. (2020). Creating urban green infrastructure where it is needed – a spatial ecosystem service-based decision analysis of green roofs in barcelona. *Science of The Total Environment*, *707*, 135487. https://doi.org/10.1016/j.scitotenv.2019.135487

Latifi, M., Darvishvand, F. G., Khandel, O., & Nowsoud, M. L. (2021). A deep reinforcement learning model for predictive maintenance planning of road assets: Integrating LCA and LCCA [Version 3 revised 27 Nov 2023]. *arXiv preprint arXiv:2112.12589*. https://arxiv.org/abs/2112.12589

Liu, W., Qian, Y., Yao, L., Feng, Q., Engel, B. A., Chen, W., & Yu, T. (2022). Identifying city-scale potential and priority areas for retrofitting green roofs and assessing their runoff reduction effectiveness in urban functional zones. *Journal of Cleaner Production*, *332*, 130064. https://doi.org/10.1016/j.jclepro.2021.130064

Ministerie van Binnenlandse Zaken en Koninkrijksrelaties. (2024). Leefbaarheidsbarometer [Accessed July 3, 2025].

Municipality of Rotterdam & B.V., A. N. (2024, March). *Background report on key figures: LIFE@urban roofs 3.0* (Technical Report) (Calculation tool version 3.0). LIFE@Urban Roofs / Municipality of Rotterdam. https://dakenplan.nl/storage/docs/Background%20report%20LIFE@Urban%20Roofs%203.0.pdf

Raaphorst, T. B. (2024). *Raising the acceptance for a preference-based design methodology in the context of urban development* [Master's thesis, Delft University of Technology]. https://repository.tudelft.nl/file/File_bf59e7ed-32d0-48db-bdf3-5e355064269b?preview=1

Roest, A. H., Weitkamp, G., van den Brink, M., & Boogaard, F. C. (2023). Mapping spatial opportunities for urban climate adaptation measures in public and private spaces using a gis-based decision support model. *Sustainable Cities and Society*, *96*, 104651. https://doi.org/10.1016/j.scs.2023.104651

United Nations. (2015). The millennium development goals report 2015 [Accessed July 3, 2025]. https://unstats.un.org/unsd/mdg/Resources/Static/Products/Progress2015/MDG_Report_2015.pdf

Urban Land Institute. (2022). *Enhancing resilience through neighborhood-scale strategies* (Accessed July 3, 2025). Urban Land Institute. Washington, DC. https://knowledge.uli.org/en/reports/research-reports/2022/enhancing-resilience-through-neighborhood-scale-strategies

van Eijck, S., & Nannes, R. (2022). *Preference-based decision support system for waelpolder: An a priori design optimization approach (pdoa) as decision support system, applied to the urban development of waelpolder* [Master's thesis, Delft University of Technology]. http://resolver.tudelft.nl/uuid:36146902-4c0c-4d50-8643-59c067008978

van Heukelum, H., Binnekamp, R., & Wolfert, R. (2023). Human preference and asset performance systems design integration. https://www.researchgate.net/publication/370058771_Human_preference_and_asset_performance_systems_design_integration

Wolfert, A. (2023). *Open design systems*. IOS Press BV.

Xiong, L., Lu, S., & Tan, J. (2023). Optimized strategies of green and grey infrastructures for integrated control objectives of runoff, waterlogging and wwdp in old storm drainages. *Science of The Total Environment*, *901*, 165847. https://doi.org/10.1016/j.scitotenv.2023.165847

Yuan, Q., Meng, F., Li, W., Lin, J., Puppim de Oliveira, J. A., & Yang, Z. (2025). Tradeoff optimization of urban roof systems oriented to food-water-energy nexus. *Applied Energy*, *380*, 124987. https://doi.org/10.1016/j.apenergy.2024.124987

Zhang, J., Zheng, H., & Wu, B. (2024). Multi-objective optimization of distributed photovoltaics on building surfaces from visual impact. *Journal of Asian Architecture and Building Engineering*, 1–18. https://doi.org/10.1080/13467581.2024.2397097

Zhilyaev, D., Binnekamp, R., & Wolfert, R. (2022). Best fit for common purpose: A multi-stakeholder design optimization methodology for construction management. *Buildings*, *12*(5), 527. https://doi.org/10.3390/buildings12050527

A
Validation iterations - workshop at Bress

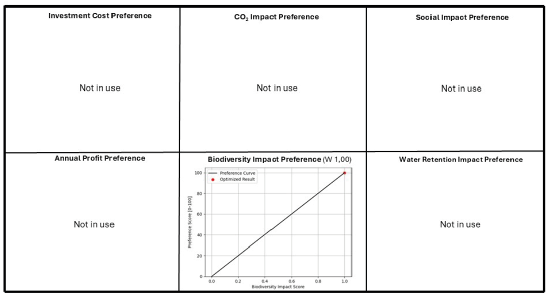
The iterations below stem from a validation workshop with Stefan van de Schootbrugge. Figure A.1 shows, how different objectives, weights and curves where changed during iterations. The summary tables show also the raw values from all the objectives of the iterations. Later, an interpretation of each iteration is given, supported by visual choropleths.

Iteration	Cost (€)	Annual Return (€)	CO_2 Red. (kg)	Biodiversity	Social Cohesion	Water Retention
1	789,650	33,886	12,792	1.0000	0.4677	0.5092
2	353,450	-4,962	0	0.8974	0.0000	0.4594
3	321,900	-4,518	0	0.8650	0.0000	0.4576
4	460,950	-6,462	0	0.9887	0.0000	0.5083
5	344,500	-1,627	1,706	0.6600	0.0000	0.7548
6	1,287,050	143,046	54,579	0.5875	0.9020	0.5831
7	1,010,950	92,277	104,042	0.5639	0.3480	0.6895
8	1,073,100	80,746	110,011	0.6847	0.5748	0.7272
9	1,002,200	102,238	224,286	0.2958	0.4301	0.3413

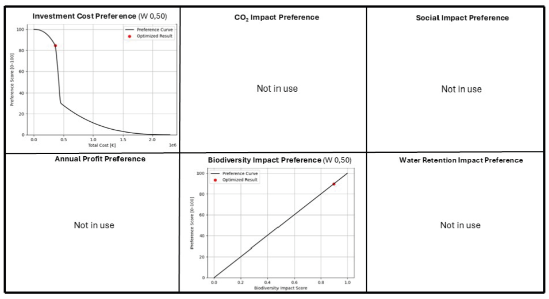
Iteration	Cost Score	Return Score	CO_2 Score	Biodiv. Score	Social Score	Water Score	Pref. Score
1	17.1	73.7	0.0	100.0	77.6	61.6	**100.00**
2	84.5	0.0	0.0	89.7	0.0	55.6	87.11
3	92.0	0.0	0.0	86.5	0.0	55.4	89.23
4	88.5	0.0	0.0	98.9	0.0	61.4	96.79
5	91.4	0.0	0.0	66.0	0.0	94.9	84.83
6	38.7	86.3	0.2	58.7	99.3	89.4	71.88
7	74.5	83.0	1.1	56.4	72.0	92.9	77.14
8	69.7	81.9	1.2	68.5	88.2	94.1	79.13
9	74.9	83.9	8.2	29.6	75.4	78.1	66.75

Table A.1: Performance of indicators and preference scores across 9 iterations.

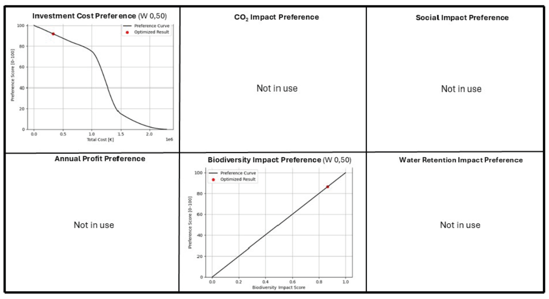
Iteration 1:



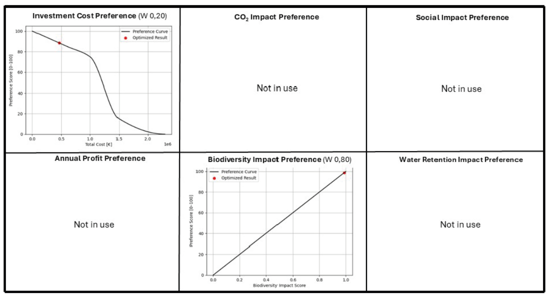
Iteration 2:



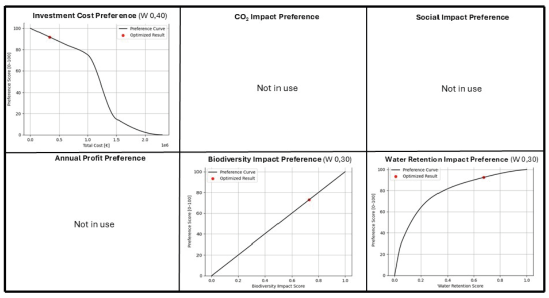
Iteration 3:



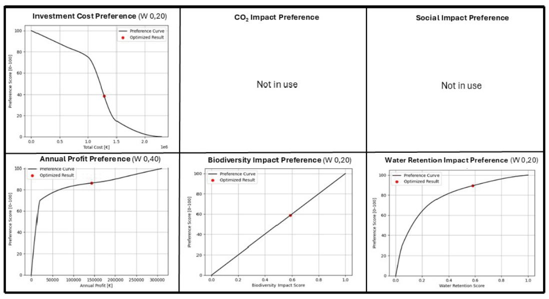
Iteration 4:



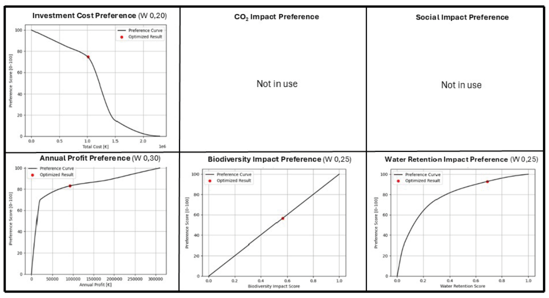
Iteration 5:



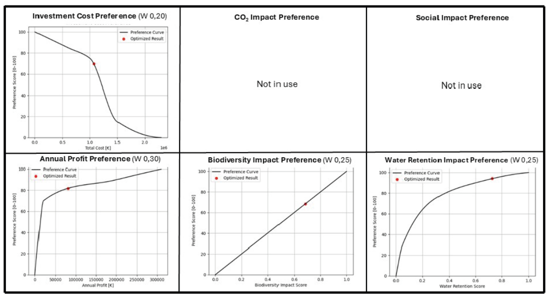
Iteration 6:



Iteration 7:



Iteration 8:



Iteration 9:

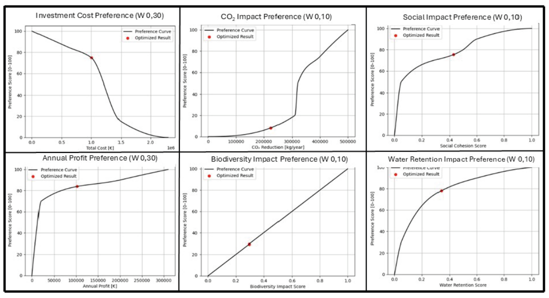


Figure A.1: Optimized preference functions for the 9 iterations of the validation workshop

Iteration 1: Understanding neighborhood needs and biodiversity allocation

We initially focus on *biodiversity impact* (a single objective), to better understand how the model allocates roof types based on neighborhood needs. This objective is represented by an index ranging from 0 to 1. A linear preference curve was used to maximize the biodiversity impact. The model allocated biodiversity roofs to all flat rooftops, since these offer the highest contribution to biodiversity. This was according to the expectation. Sloped roofs received sedum roofs, contributing less to biodiversity, however the only feasible option for sloped surfaces, and therefore the best option.

We noted an exception in **Neighborhood E**, receiving a rather random mix of roof types. This may be explained by the fact that Neighborhood E has a biodiversity need score of zero. Consequently, any rooftop intervention results in a biodiversity impact score of zero, making all options equally valid. As a result the model randomly allocated rooftops in such a situation.

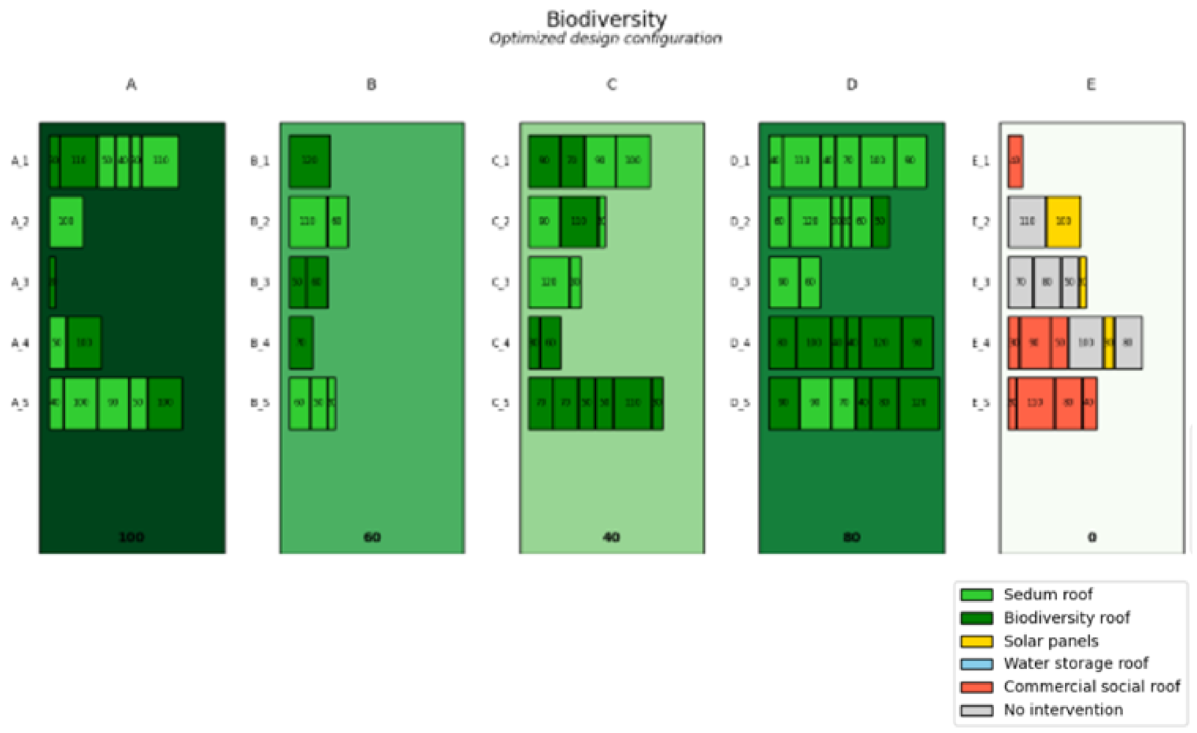


Figure A.2: Iteration 1

Iteration 2: Introducing cost as a conflicting objective

Stefan suggested to test the model with an extra element and we added cost as conflicting objective to see how the model would responds next to the biodiversity needs as a single objective. The cost objective was modeled with a nonlinear preference curve, whereas the biodiversity impact preference curve remained linear. Both objectives were assigned equal weights (50/50).

A more nuanced allocation pattern was demonstrated by the model, at this iteration. The model started to weigh cost against impact; selecting either sedum or intensive biodiversity roofs, based on the balance of the costs and contribution to biodiversity.

- In **Neighborhood A** (with the highest biodiversity need), all roofs were assigned either a biodiversity or sedum roof.
- In **Neighborhood D** (2nd highest need), most rooftops were assigned, except for three surfaces (buildings D_2, D_D_1, and D_3). The three surfaces received no intervention at all, probably due to unfavorable cost-benefit tradeoffs.
- In **Neighborhood B** (3rd highest need), only biodiversity roofs were applied and no sedum roofs were applied. Some rooftops received no intervention, indicating a stricter application of tradeoffs.
- In **Neighborhood C** (low biodiversity need), only a few rooftops received biodiversity roofs. For example, Neighborhood B received 300 m2 of interventions, whereas Neighborhood C received only 240 m2. This demonstrated that the model applies lower-impact interventions in areas with less need.
- **Neighborhood E**, with zero biodiversity need, received no interventions in this iteration. In contrast with Iteration 1, where the absence of a cost objective resulted in random allocation. With this Iteration 2, the model correctly assigns zero-cost interventions in situations where the expected benefit is also zero.

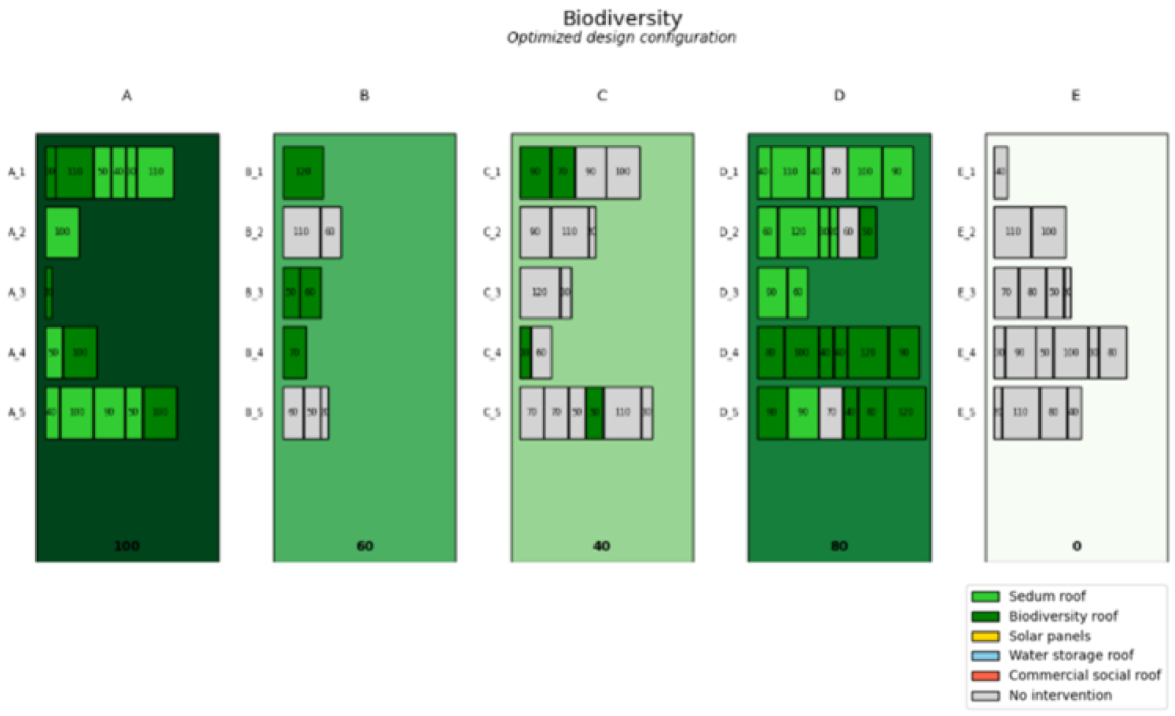


Figure A.3: Iteration 2

Iteration 3: Adjusting the cost preference curve

After carefully reviewing the results of the first two iterations, we revisited the steepness of the cost preference curve. We assumed that the original curve penalized higher costs too strongly. In order to test this, we created a less steep cost curve, and thus allowing for a more gradual drop in preference at increasing costs levels.

After running the model we saw that this adjustment had a rather subtle but noticeable impact. Since the upper end of the curve (representing lower costs) after this adjustment, received higher preference values, the model became somewhat more sensitive to cost again. As a consequence, the overall investment decreased, though the rooftop allocation pattern remained mostly similar.

The most noticeable difference occurred in **Neighborhood C**, characterized by relatively low biodiversity needs. In this neighborhood, a smaller area of biodiversity roofs was allocated in comparison to the previous iteration. This clearly demonstrates how even minor changes in preference curve shapes, can influence allocation decisions, particularly in areas with less pressing needs.

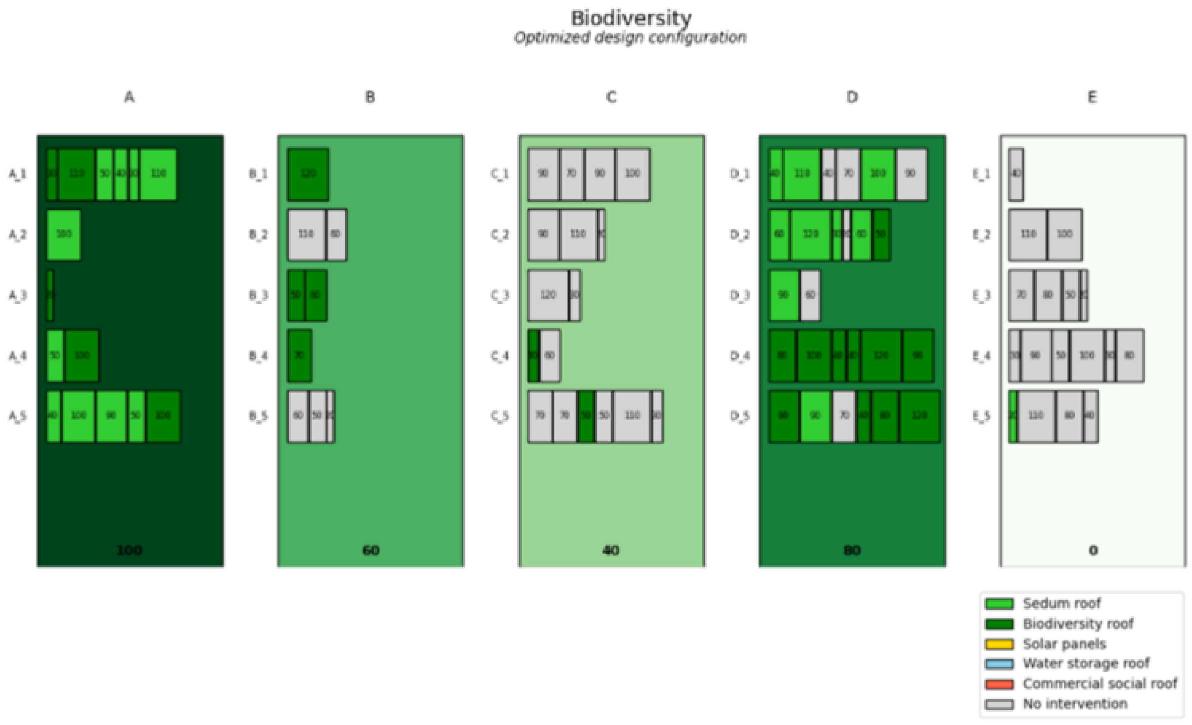


Figure A.4: Iteration 3

Iteration 4: changing the relative importance of objectives

In this iteration we tested how the relative weighting of objectives would have an impact on rooftop allocation. In the first three iterations iterations, both biodiversity impact and investment cost were equally weighted at 0.50 each.

To put more emphasis on biodiversity, the weights were adjusted as follows:

- Biodiversity: 0.80
- Investment cost: 0.20

The impact of the adjustment was immediately shown:

- The total assigned rooftop surface area increased to 460,950 m2.
- A significantly higher number of rooftops received either sedum or biodiversity roofs.
- Also, areas with lower needs for biodiversity (such as the neighborhoods C and B) obtained more interventions, showing the higher priority put on maximizing the impact over minimizing the costs.

This fourth iteration shows how the model responds to strategic priorities selected by the portfolio owner. By manipulating the weights of the model, users may actively manage the outcome towards the sustainability objectives, they are aiming for.

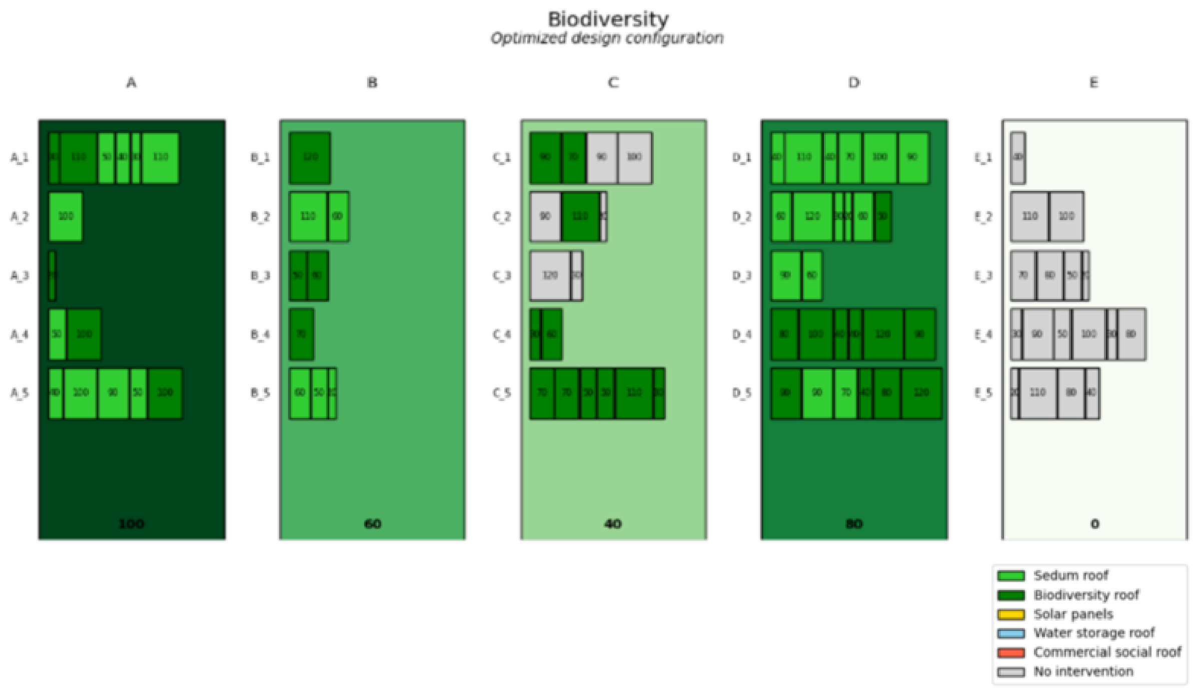


Figure A.5: Iteration 4

Iteration 5: Adding water impact as a third objective

In this scenario the portfolio owner introduced water retention as a third objective and chose a non-linear preference curve. Earlier on we already commented on the fact that sedum and biodiversity roofs offer some water retention, although less so than blue roofs. The latter are specifically designed for that purpose. The weights chosen for the various objectives were: 0.40 for investment cost, 0.30 for biodiversity impact, and 0.30 for water impact. In **Neighborhood A** (high biodiversity and water needs) the model allocated a mix of sedum, biodiversity, and blue roofs. **Neighborhood B** (as neighborhood A also with high water needs) obtained several blue roofs.

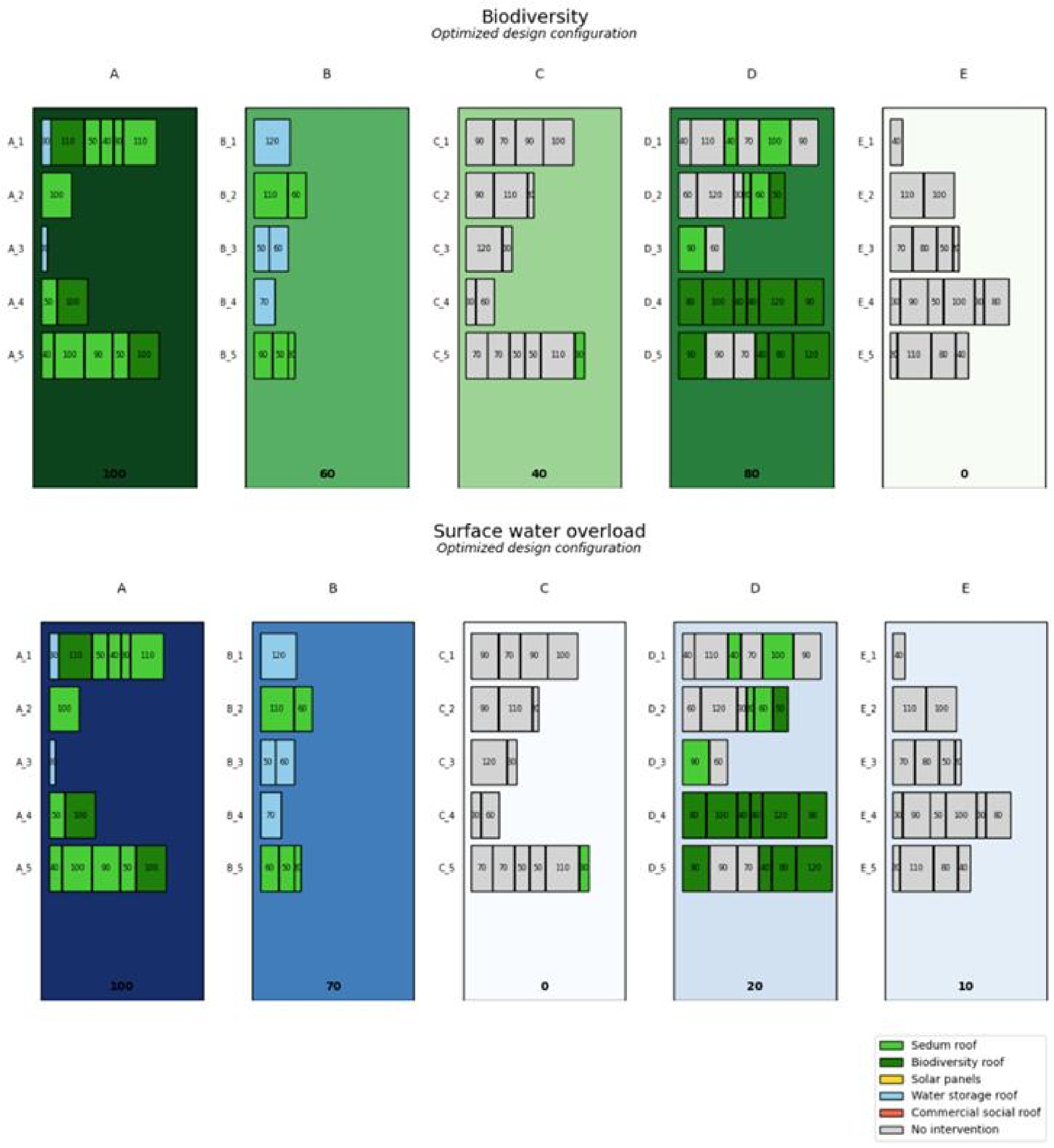


Figure A.6: Iteration 5

Iteration 6: Introducing annual return as objective

This sixth iteration we tested how the model reacts if annual return is added as a dominant objective. Investments in rooftops may provide financial returns; advantages for instance by providing solar energy and savings, as well as for commercial use. Based on Stefan's experience, an attractive investment is expected to have a payback period of about 12 years.

To simulate this, the weights were set as follows: annual return at 0.40, and investment cost, biodiversity, and water impact each at 0.20. The model gave priority to profitable rooftop types, especially in Neighborhoods C and E with low biodiversity and/or water needs. This result is fully aligned with the model's logic: in areas with limited neighborhood needs, financial return is considered as more important.

The result was a total investment of 1,287,050 EUR and an expected annual return of 143,046 EUR, yielding a payback period of 9 years. This is considered to be within an acceptable range. However, the investment exceeded a 1 million EUR budget, asking for a new iteration to adjust weights and/or apply a stricter budget constraint.

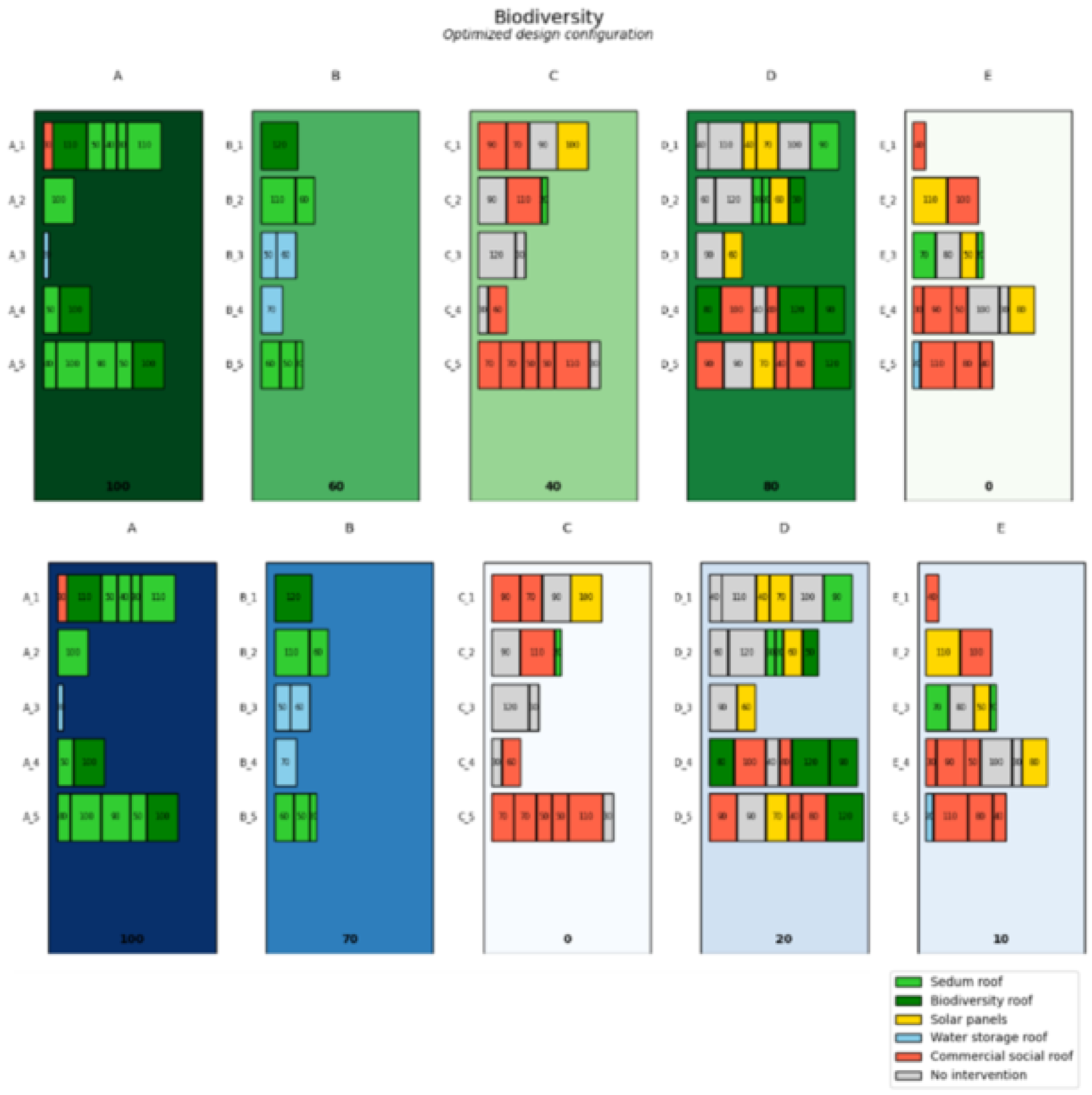


Figure A.7: Iteration 6

Iteration 7: Adjusting weights to target a 1 million investment

To meet the 1 million EUR budget requirement, the weights of the model were adjusted as follows; investment cost was increased to 0.30 and the weight for annual return was lowered to 0.30. Biodiversity and water retention were both set at 0.25. The model then generated a total investment need of 1,010,950 EUR and expected an annual return of 92,277 EUR. The payback period is then approximately 11 years, according to Stefan's within an acceptable range. However, Stefan also shared his concerns about the of the believability of these results. With the rather high performance on sustainability of the selected rooftops, Stefan considered the high financial return as rather optimistic. This may imply that the model overestimates the financial returns and/or underestimates the costs for certain roof types. This seventh iteration emphasizes the need to validate the input data as well as the assumptions when making financial performance part of a sustainability model.

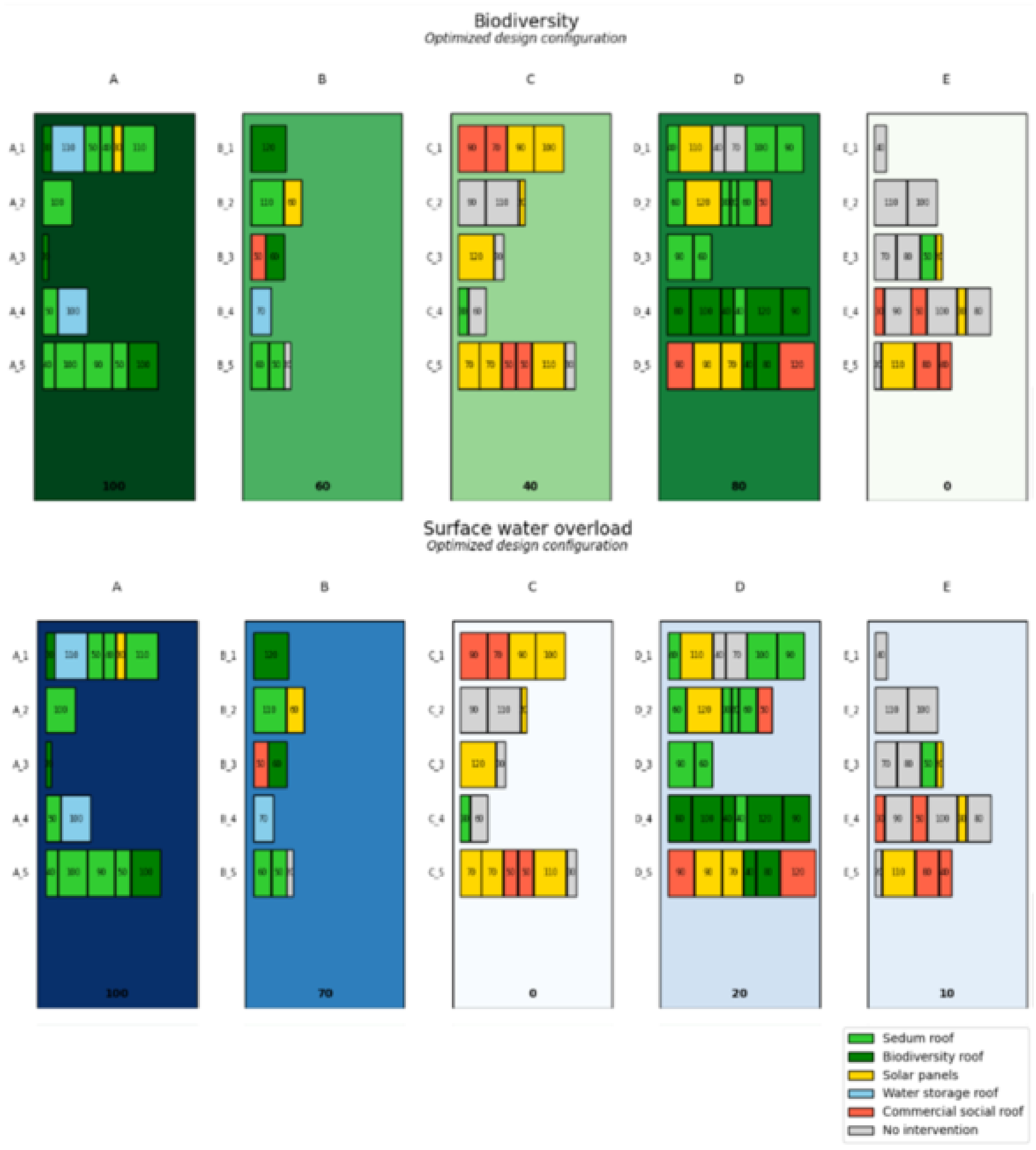


Figure A.8: Iteration 7

Iteration 8: Adjusting parameters annual return for social commercial roof

The annual return for quite a sustainable investment seems a bit to high, this is why Stefan wanted to have a look at the input parameters, which are used for solar panels and social commercial roof. The annual return for social commercial roofs /m2 was 90, this value was changed to 70. Within the output it becomes visible, that solar panels are more often assigned then commercial roofs. This stems from the slightly, almost negligible, better annual return of solar panels.

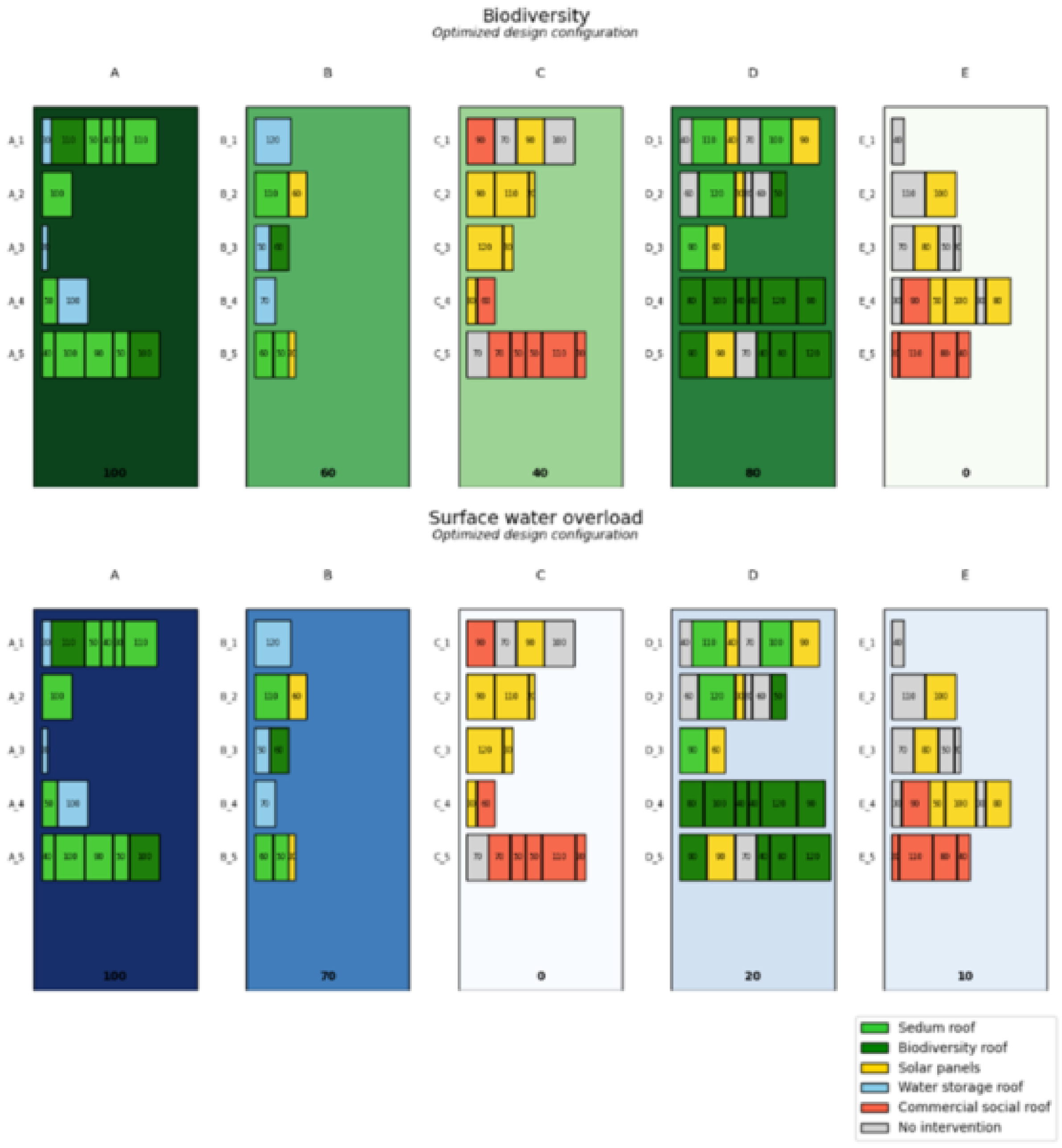


Figure A.9: Iteration 8

Iteration 9: Adding final ojectives social cohesion and CO_2

To efficiently allocate the rooftops and to trust data where social cohesion is low, this portfolio owner wants to strategically allocate the social commercial roofs, also CO_2 was introduced as objective to express the importance of solar panels not only in annual return due to energy savings but also to see how much CO_2 can be avoided. The eventual investment cost where 1,002,200, with an annual return of 102,238 , which is a payback period of around 10 years. This final result was accepted as an interesting final design configuration. When we look how rooftops are allocated it now becomes clear how the model carefully allocates the roof types. The social commercial roofs (red) are only allocated in the neighborhoods C and E which have the highest neighborhood need of 80 and 70. For the water impact roofs, they are only assigned to neighborhoods with the highest water needs A and B. Biodiversity and sedum roofs are only allocated at the 3 neighborhoods with highest need for biodiversity A,B and D. Solar panels are distributed to ensure annual return and the final design configuration was accepted.

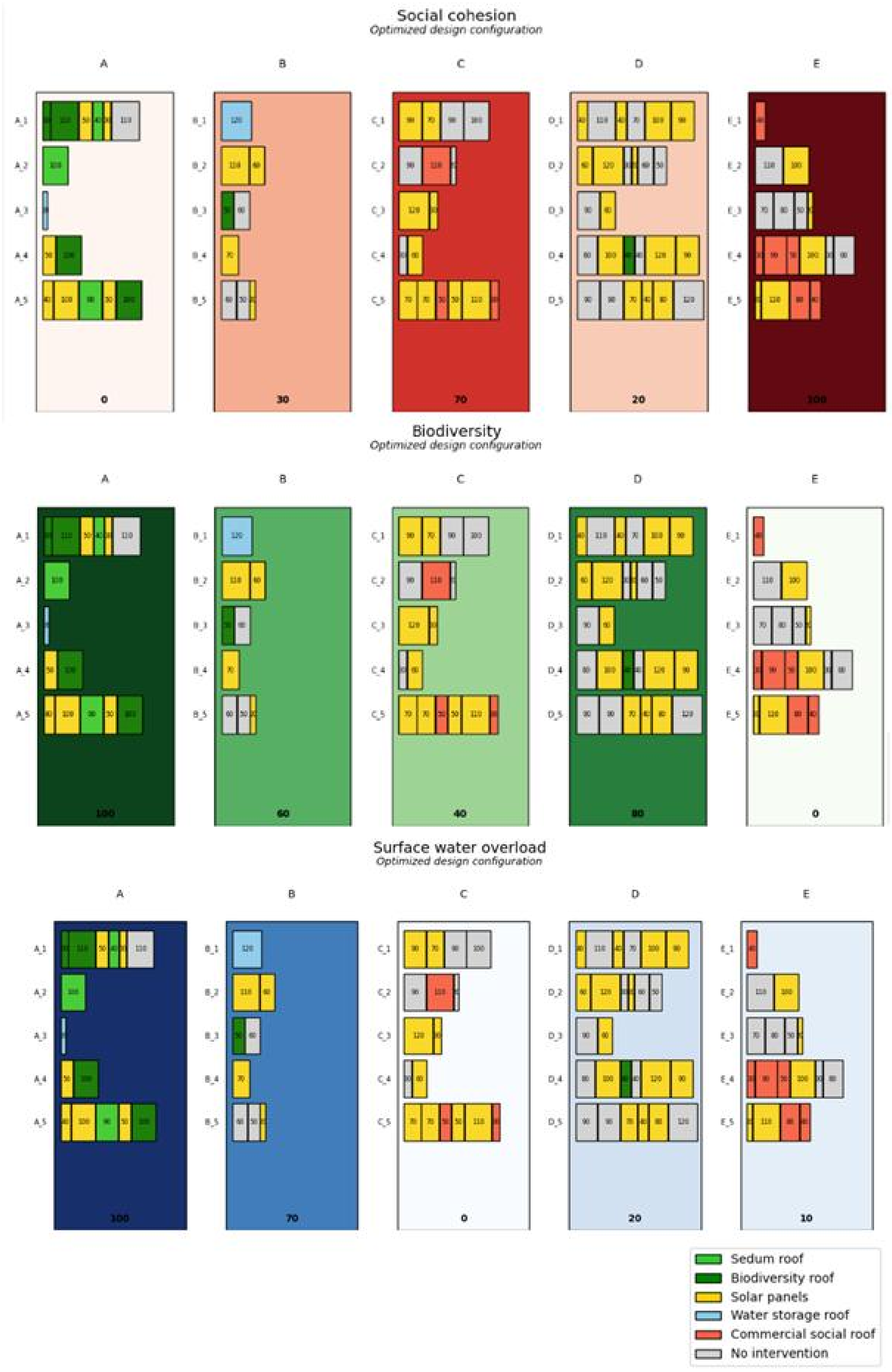


Figure A.10: Iteration 9

B
Python code - optimization

See code next page.

Multi-objective Optimization Notebook

This notebook is structured into three chapters:

1. Plotting and visualizing data sets

- Plot roof surfaces portfolio
- Plot neighborhoods
- Plot portfolio and neighborhoods

2. Optimization model

- Import libraries and genetic algorithm
- Load input data
- Build feasible design space for optimization
- Objective function formula definitions
- Defining and plotting preference curves
- Define bounds
- Define constraints
- Define weights
- Define objective function for algorithm
- Run and adjust algorithm

3. Visualizing and loading results

- Optimized output statistics
- Preference functions with optimized points
- Plot portfolio and optimized roof surfaces
- Optimized portfolio as overlayer over neighborhoods

Remarks

This code follows the general structure, of the deepnote environment from TUDELFT-ODESYS. Therefore it contains similar structure and similar parts of code as provided in the examples. For more insights visit: https://deepnote.com/workspace/tudelft-odesys-e3302e92-17a2-4a90-b40f-45cdf271893a/project/OPEN-DESIGN-SYSTEMS-e410fe0f-e33d-4092-8e69-745b3b09d0eb/notebook/Chapter-8-DA4-South-Korean-floating-wind-farm-ed75b0a5c01d4ab1948c7cc88ba31ea5

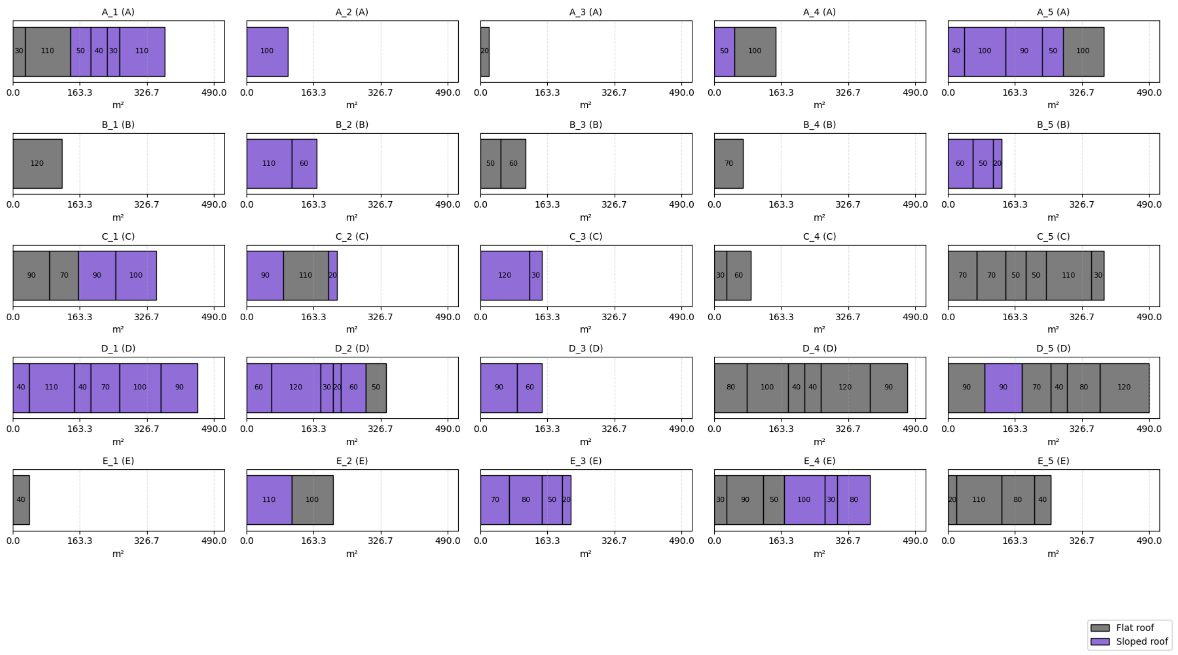
For coding, debugging, style, variable names, and titles, CHATGPT has been used.

1. Plotting and visualizing data sets

Plot roof surfaces portfolio

We start with visualizations of the current data frames, starting with a plot of the roof surfaces of the portfolio.

In [38]:
```python
# Visualize roof surface units per building with scaled width

import matplotlib.pyplot as plt
import pandas as pd
import matplotlib.patches as patches
import numpy as np

# Load data
roof_data_path = "./Data_mockup/Colored_Roof_Surface_Data test.xlsx"
roof_df = pd.read_excel(roof_data_path, sheet_name='Sheet1')

# Clean column names and rename
roof_df.columns = [col.strip() for col in roof_df.columns]
roof_df = roof_df.rename(columns={
    'Bag_ Pand_ Id': 'Building',
    'Object dak unit id': 'Surface_ID',
    'oppervlak': 'Area',
    'Helling graden': 'Slope',
    'Object label (schuin/ plat)': 'Type',
    'Buurt': 'Buurt'
})

# Calculate total area per building
building_totals = roof_df.groupby('Building')['Area'].sum()
global_max_area = building_totals.max()

# Group by building
grouped = roof_df.groupby('Building')

# Create plot
fig, axs = plt.subplots(6, 5, figsize=(18, 10))
axs = axs.flatten()

for i, (building_id, group) in enumerate(grouped):
    ax = axs[i]
    buurt = group['Buurt'].iloc[0]
    ax.set_title(f"{building_id} ({buurt})", fontsize=10)

    x_start = 0
    for _, row in group.iterrows():
        type_str = str(row['Type']).strip().lower()
        color = 'gray' if type_str == 'plat' else 'mediumpurple'
        width = row['Area']
        rect = patches.Rectangle((x_start, 0.1), width, 0.8, edgecolor='black',
        ax.add_patch(rect)
        ax.text(x_start + width / 2, 0.5, f"{int(width)}", ha='center', va='cent
        x_start += width

    # Use global scale for all buildings
    ax.set_xlim(0, global_max_area * 1.05)
    ax.set_ylim(0, 1)
    ax.set_xticks(np.linspace(0, global_max_area, 4))
    ax.set_xlabel("m²")
    ax.set_yticks([])
    ax.grid(True, axis='x', linestyle='--', alpha=0.3)

# Hide unused subplots
for j in range(i + 1, len(axs)):
    axs[j].axis('off')

# Add legend
legend_elements = [
    patches.Patch(facecolor='gray', edgecolor='black', label='Flat roof'),
    patches.Patch(facecolor='mediumpurple', edgecolor='black', label='Sloped roo
]
fig.legend(handles=legend_elements, loc='lower right', fontsize=10)

plt.tight_layout()
plt.show()
```



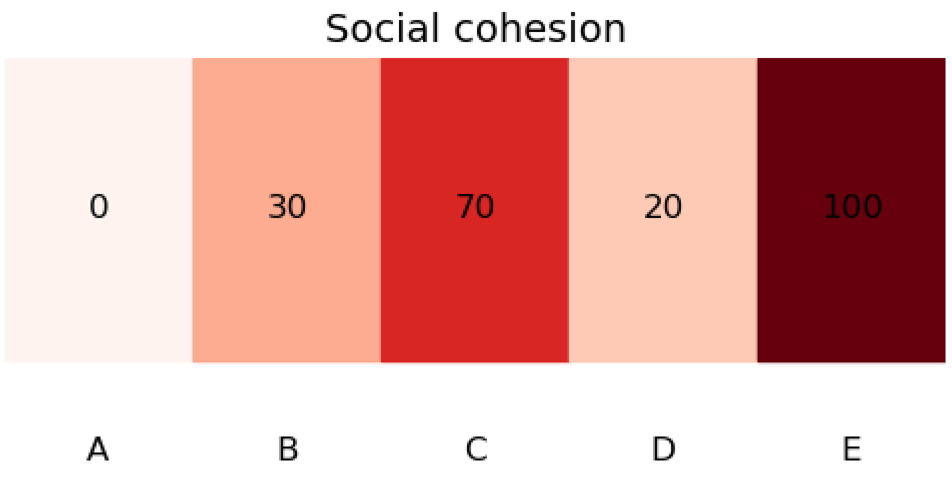
This plot above shows the roof surfaces of the portfolio of 25 buildings. It shows how each rooftop surface is distributed per building, with surface width proportional to its area (in m^2). Each rectangle represents a surface unit, colored by roof type: flat roofs in gray and sloped roofs in purple. Buildings are grouped by their BAG Pand ID, and all plots use the same x-axis scale for comparability. A_1, for example is building 1 in neighborhood A.

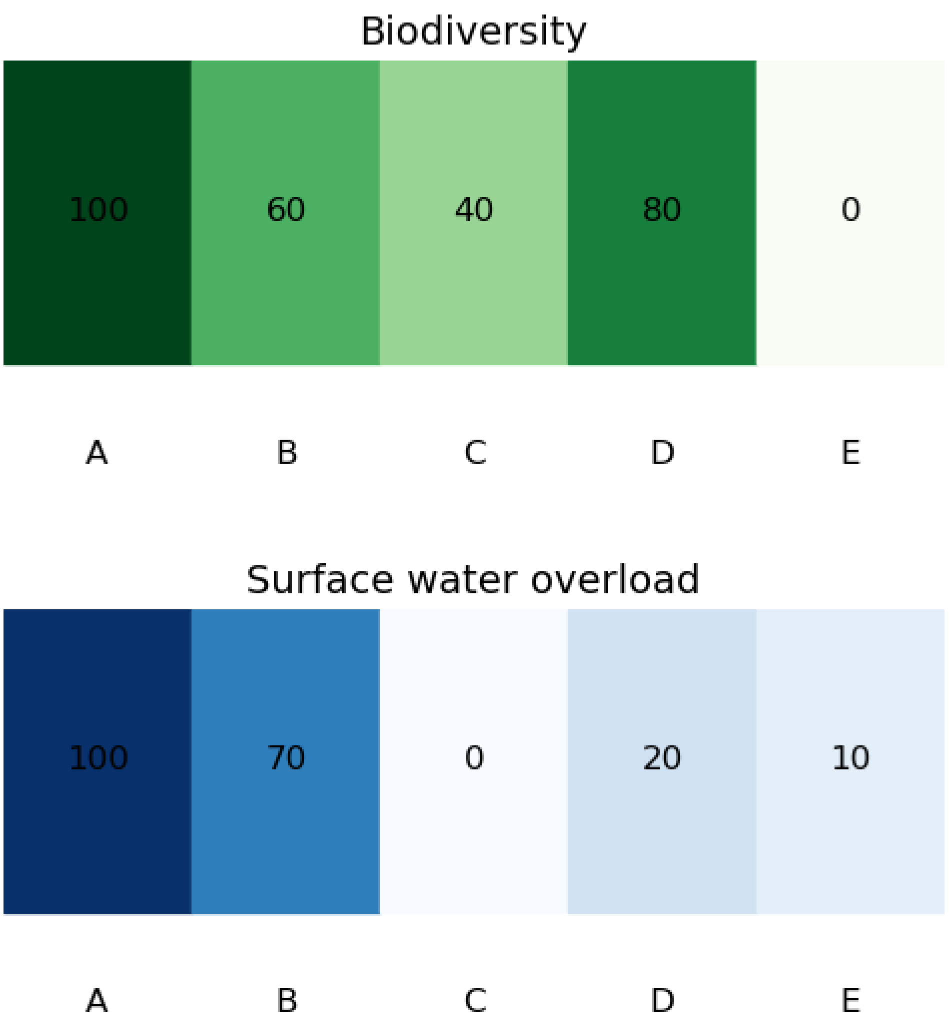
Plot neighborhoods

Below the data is plotted for neighborhoods and their challenges.

In [39]:
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
from matplotlib.patches import Rectangle

# Load Excel
file_path = './Data_mockup/Buurten_need.xlsx'
df = pd.read_excel(file_path, sheet_name='Sheet1')
```

```
# Skip first row and rename columns
df = df.iloc[1:]
df.columns = ['Symbol', 'Nsoc', 'Nbio', 'Nwat', 'Abuurt']
df.set_index('Symbol', inplace=True)

# Convert to numeric
df[['Nsoc', 'Nbio', 'Nwat']] = df[['Nsoc', 'Nbio', 'Nwat']].apply(pd.to_numeric,

# Updated choropleth function with neighborhood labels underneath
def plot_block_choropleth(data, title, cmap_name):
    values = data.values.astype(float)
    labels = data.index
    cmap = plt.colormaps.get_cmap(cmap_name)
    norm = mcolors.Normalize(vmin=0, vmax=100)
    colors = cmap(norm(values))

    fig, ax = plt.subplots(figsize=(len(values), 2.8))
    for i, (val, color, label) in enumerate(zip(values, colors, labels)):
        ax.add_patch(Rectangle((i, 0.5), 1, 1, color=color))
        ax.text(i + 0.5, 1.0, f"{int(val)}", va='center', ha='center', fontsize=
        ax.text(i + 0.5, 0.2, label, va='center', ha='center', fontsize=12, colc

    ax.set_xlim(0, len(values))
    ax.set_ylim(0, 1.5)
    ax.set_xticks([])
    ax.set_yticks([])
    ax.set_title(title, fontsize=14)
    ax.axis('off')
    plt.tight_layout()
    plt.show()

# Run the plots
plot_block_choropleth(df['Nsoc'], 'Social cohesion', 'Reds')
plot_block_choropleth(df['Nbio'], 'Biodiversity', 'Greens')
plot_block_choropleth(df['Nwat'], 'Surface water overload', 'Blues')
```





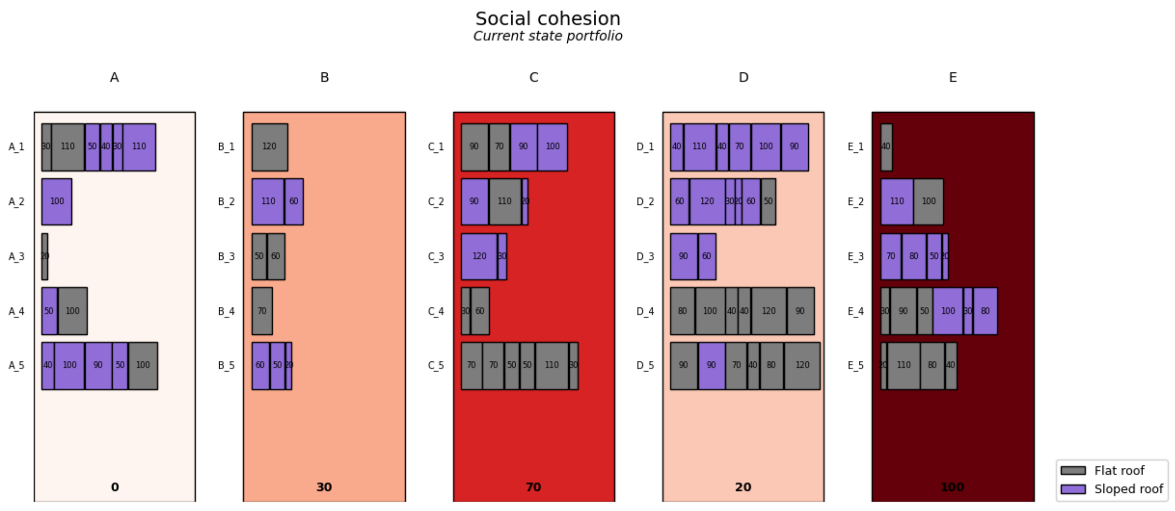
The plot above shows the need scores for social cohesion, biodiversity, and surface water overload across neighborhoods, based on the dataset Buurten_need.xlsx.

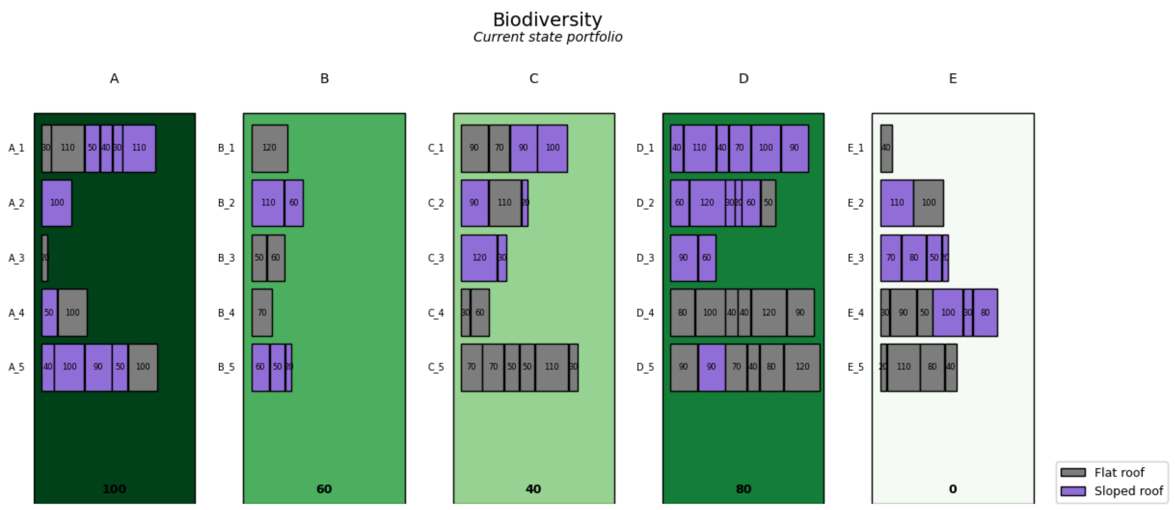
Each colored block represents a single neighborhood (labeled underneath), and the color intensity indicates the urgency or priority level on a scale from 0 to 100. The color mapping within each plot is to highlight relative differences between neighborhoods A-E.

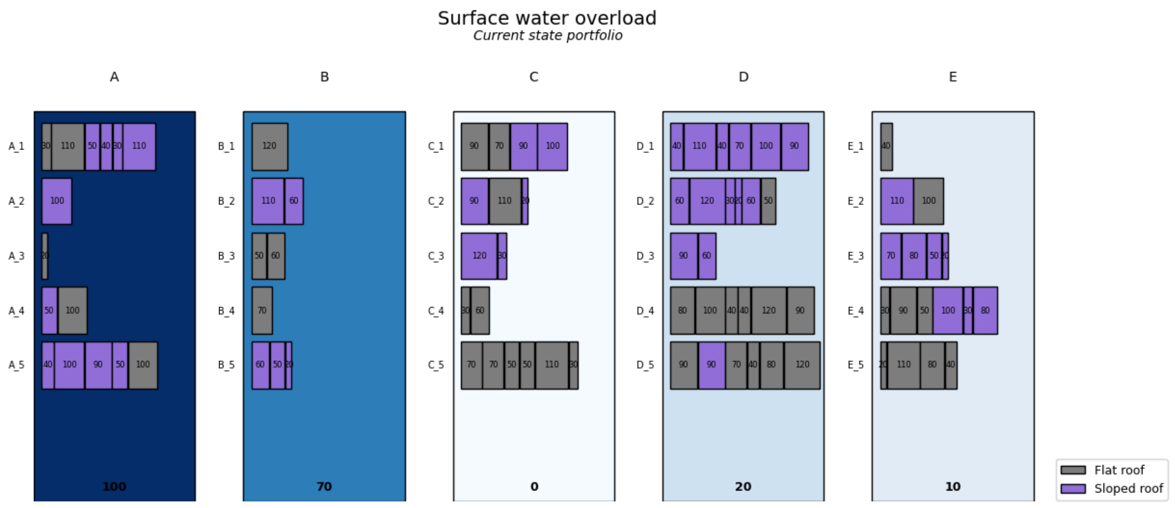
Plot portfolio and neighborhoods

The code below is used to plot an overview of which buildings from the portfolio are located in which neighborhood, the individual building plots have been overlaid.

In [56]:
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import matplotlib.colors as mcolors
import numpy as np

# Load data
neigh_file = './Data_mockup/Buurten_need.xlsx'
df = pd.read_excel(neigh_file, sheet_name='Sheet1')
df = df.iloc[1:]
df.columns = ['Symbol', 'Nsoc', 'Nbio', 'Nwat', 'Abuurt']
df.set_index('Symbol', inplace=True)
df.index = df.index.astype(str).str.strip()

roof_data_path = './Data_mockup/Colored_Roof_Surface_Data test.xlsx'
roof_df = pd.read_excel(roof_data_path, sheet_name='Sheet1')
roof_df.columns = [col.strip() for col in roof_df.columns]
roof_df = roof_df.rename(columns={
    'Bag_ Pand_ Id': 'Building',
    'Object dak unit id': 'Surface_ID',
    'oppervlak': 'Area',
    'Helling graden': 'Slope',
    'Object label (schuin/ plat)': 'Type',
    'Buurt': 'Neighborhood'
})
roof_df['Neighborhood'] = roof_df['Neighborhood'].astype(str).str.strip()

building_totals = roof_df.groupby('Building')['Area'].sum()
global_max_area = building_totals.max()

# Plot function
def plot_neighborhood_roofs_with_challenge(theme_col, title, cmap_name):
    challenge_data = df[theme_col].to_dict()

    neighborhoods = sorted(roof_df['Neighborhood'].unique())
    buildings_per_neigh = {n: roof_df[roof_df['Neighborhood'] == n].groupby('Bui

    fig, ax = plt.subplots(figsize=(14, 6))

    bar_width = 1
    bar_gap = 0.3
    building_height = 0.12
    roof_gap = 0.02

    cmap = plt.colormaps.get_cmap(cmap_name)
    norm = mcolors.Normalize(vmin=0, vmax=100)

    for i, neighborhood in enumerate(neighborhoods):
        challenge_val = challenge_data.get(neighborhood, 0)
        neigh_color = cmap(norm(challenge_val))

        x_pos = i * (bar_width + bar_gap)
        ax.add_patch(patches.Rectangle((x_pos, 0), bar_width, 1, facecolor=neigh

        # Neighborhood label above the bar
        ax.text(x_pos + bar_width / 2, 1.07, f"{neighborhood}",
                ha='center', va='bottom', fontsize=10)

        # Challenge value inside the bar
        ax.text(x_pos + bar_width / 2, 0.02, f"{challenge_val:.0f}",
                ha='center', va='bottom', fontsize=9, color='black', fontweight=

        grouped = buildings_per_neigh[neighborhood]
        for j, (building_id, group) in enumerate(grouped):
            y_base = 0.85 - j * (building_height + roof_gap)
            x_building = x_pos + 0.05
            x_start = x_building
            for _, row in group.iterrows():
                type_str = str(row['Type']).strip().lower()
                color = 'gray' if type_str == 'plat' else 'mediumpurple'
                width = row['Area'] / global_max_area * (bar_width - 0.1)
                rect = patches.Rectangle((x_start, y_base), width, building_heig
                ax.add_patch(rect)

                # Area label
                ax.text(x_start + width / 2, y_base + building_height / 2, f"{in
                        ha='center', va='center', fontsize=6, color='black')
                x_start += width + 0.005

            # Building ID label on the left
            ax.text(x_pos - 0.05, y_base + building_height / 2, building_id,
                    va='center', ha='right', fontsize=7)

    ax.set_xlim(0, len(neighborhoods) * (bar_width + bar_gap))
    ax.set_ylim(0, 1.2)
    ax.axis('off')

        # Legend just next to the last bar
    x_last = (len(neighborhoods) - 1) * (bar_width + bar_gap)
    x_legend = x_last + bar_width + 0.1
    axis_width = len(neighborhoods) * (bar_width + bar_gap)
    legend_x_frac = x_legend / axis_width

    legend_elements = [
        patches.Patch(facecolor='gray', edgecolor='black', label='Flat roof'),
        patches.Patch(facecolor='mediumpurple', edgecolor='black', label='Sloped
    ]
    ax.legend(handles=legend_elements, loc='lower left', bbox_to_anchor=(legend_



    # Title and subtitle
    fig.text(0.41, 0.91, f"{title}", ha='center', va='center', fontsize=14)
    fig.text(0.41, 0.88, 'Current state portfolio', ha='center', va='center', fo

    plt.tight_layout(rect=[0, 0.05, 0.88, 0.91])
    plt.show()

# Run for each theme
plot_neighborhood_roofs_with_challenge('Nsoc', 'Social cohesion', 'Reds')
plot_neighborhood_roofs_with_challenge('Nbio', 'Biodiversity', 'Greens')
plot_neighborhood_roofs_with_challenge('Nwat', 'Surface water overload', 'Blues'
```







This combined plot above helps to quickly see which buildings belong to which neighborhood.

2. Optimization model

Import libraries and genetic algorithm

In [41]:
```python
from genetic_algorithm_pfm import GeneticAlgorithm
```

In [42]:
```python
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import pchip_interpolate
```

Load inputdata

There are three data sets that need to be loaded. First, the data set containing information on roofs of the building portfolio, including their roof surfaces, slope, surface area, and the neighborhoods they belong to. Second, the data set outlining neighborhood challenges, specifically biodiversity, social cohesion, and water surface overload, which have been selected for this example. Lastly, the data set describing possible rooftop interventions and the characteristics of each roof type.

In [43]:
```
# Load and clean roof, buurt, and dak parameter data

import pandas as pd

# File paths
roof_data_path = "./Data_mockup/Colored_Roof_Surface_Data test.xlsx"
buurten_path = "./Data_mockup/Buurten_need.xlsx"
dak_params_path = "./Data_mockup/Dak parameters.xlsx"

# Load roof data
roof_df = pd.read_excel(roof_data_path, sheet_name="Sheet1")
roof_df.columns = [col.strip() for col in roof_df.columns]
roof_df = roof_df.rename(columns={
    'Bag_ Pand_ Id': 'Building',
    'Object dak unit id': 'Surface_ID',
    'oppervlak': 'Area',
    'Helling graden': 'Slope',
    'Object label (schuin/ plat)': 'Type',
    'Buurt': 'Buurt'
})


# Load BUURTEN NEEDS (neighborhood)

buurten_raw = pd.read_excel(buurten_path, sheet_name="Sheet1")
buurten_df = buurten_raw.iloc[1:].copy().reset_index(drop=True)
buurten_df.columns = ['Buurt', 'Nsoc', 'Nbio', 'Nwat', 'Abuurt']
for col in ['Nsoc', 'Nbio', 'Nwat']:
    buurten_df[col] = pd.to_numeric(buurten_df[col], errors='coerce')

# Load DAK PARAMETERS (roof type)

dak_raw = pd.read_excel(dak_params_path, sheet_name="Sheet1")
dak_df = dak_raw.iloc[1:].copy().reset_index(drop=True)

# Rename columns
dak_df.columns = [
    'Dak kleur', 'Omschrijving', 'Max hellingsgraad', 'Crepl', 'Cinst',
    'Gewicht dak', 'Waterberging', 'Cmaint', 'Levensduur', 'WOZ increase',
    'ROI', 'Biodiv effect', 'Sociaal effect', 'Water effect'
]

# Replace commas with dots and convert to numeric
for col in ['Crepl', 'Cinst', 'Gewicht dak', 'Waterberging', 'Cmaint',
            'Levensduur', 'WOZ increase', 'ROI',
            'Biodiv effect', 'Sociaal effect', 'Water effect']:
    dak_df[col] = dak_df[col].astype(str).str.replace(',', '.')
    dak_df[col] = pd.to_numeric(dak_df[col], errors='coerce')

# Print first few rows data sets
```

```
print("Roof data:")
display(roof_df.head())

print("\nBuurten needs:")
display(buurten_df.head())

print("\nDak parameters:")
display(dak_df.head())
```

```
Roof data:
```

	Building	Surface_ID	Area	Slope	Type	Buurt
0	A_1	1	30	0	plat	A
1	A_1	2	110	0	plat	A
2	A_1	3	50	5	schuin	A
3	A_1	4	40	25	schuin	A
4	A_1	5	30	25	schuin	A

```
Buurten needs:
```

	Buurt	Nsoc	Nbio	Nwat	Abuurt
0	A	0	100	100	NaN
1	B	30	60	70	NaN
2	C	70	40	0	NaN
3	D	20	80	20	NaN
4	E	100	0	10	NaN

```
Dak parameters:
```

	Dak kleur	Omschrijving	Max hellingsgraad	Crepl	Cinst	Gewicht dak	Waterberging	Cmaint	Lev
0	Geen	Geen ingreep	90	0	0	0	0	0.0	
1	Groen 1	Sedium dak	45	45	50	80	20	1.2	
2	Groen 2	Natuurdak	4	45	75	300	45	1.8	
3	Rood	Sociaal Tuindak	0	45	500	600	20	10.0	
4	Blauw	Blauw dak	0	45	150	250	150	2.0	

Build feasible design space for optimization

The first two datasets to be merged are roof_df and buurten_df, that is, the roof data of the current portfolio and the data on neighborhood needs. These datasets can be merged using the common column 'neighborhood'. The third dataset, which contains possible roof interventions, is later combined using a Cartesian product. Each roof surface is paired with every possible roof intervention. The result is a dataset (design_space_df) with all the possible roof types on each roof surface.

Constraint: Before optimization, a feasibility constraint is already applied. This is explained in the section on constraints and bounds, where the actual slope of the current roof surface must be less than or equal to the maximum allowable slope of the proposed roof type intervention.

In [44]:
```
from itertools import product

# Merge roof surfaces with buurt needs
roof_with_needs = pd.merge(roof_df, buurten_df, on='Buurt', how='left')

# Create cartesian product (each surface with every roof type)
surface_ids = roof_with_needs.index
roof_type_ids = dak_df.index
combinations = list(product(surface_ids, roof_type_ids))

# Build expanded dataframe
expanded_rows = []
for surface_idx, roof_type_idx in combinations:
    surface = roof_with_needs.loc[surface_idx]
    roof_type = dak_df.loc[roof_type_idx]

    # Always include 'Geen' regardless of slope
    if roof_type['Dak kleur'].strip().lower() != 'geen':
        # FEASIBILITY CONSTRAINT: the actual slope of current roofsurface <= max
        if surface['Slope'] > roof_type['Max hellingsgraad']:
            continue

    expanded_rows.append({
        'Surface_ID': surface['Surface_ID'],
        'Building': surface['Building'],
        'Buurt': surface['Buurt'],
        'Area': surface['Area'],
        'Slope': surface['Slope'],
        'Nsoc': surface['Nsoc'],
        'Nbio': surface['Nbio'],
        'Nwat': surface['Nwat'],
        'Dak type': roof_type['Dak kleur'].strip(),
        'Crepl': roof_type['Crepl'],
        'Cinst': roof_type['Cinst'],
        'Cmaint': roof_type['Cmaint'],
        'ROI': roof_type['ROI'],
        'Biodiv effect': roof_type['Biodiv effect'],
        'Sociaal effect': roof_type['Sociaal effect'],
        'Water effect': roof_type['Water effect']
    })

# Final design space is the result, all the possible rooftypes which could be on
design_space_df = pd.DataFrame(expanded_rows)

print(f"Feasible design options generated: {len(design_space_df)} rows")
display(design_space_df.head())
```

```
Feasible design options generated: 372 rows
```

	Surface_ID	Building	Buurt	Area	Slope	Nsoc	Nbio	Nwat	Dak type	Crepl	Cinst	Cn
0	1	A_1	A	30	0	0	100	100	Geen	0	0	
1	1	A_1	A	30	0	0	100	100	Groen 1	45	50	
2	1	A_1	A	30	0	0	100	100	Groen 2	45	75	
3	1	A_1	A	30	0	0	100	100	Rood	45	500	
4	1	A_1	A	30	0	0	100	100	Blauw	45	150	

Above we see the head of the dataframe, which forms the design solution space. All unique Surface_IDs have gotten rows with their possible 'DAK Type ,' which is the possible rooftop intervention.

Objective functions formula definitions

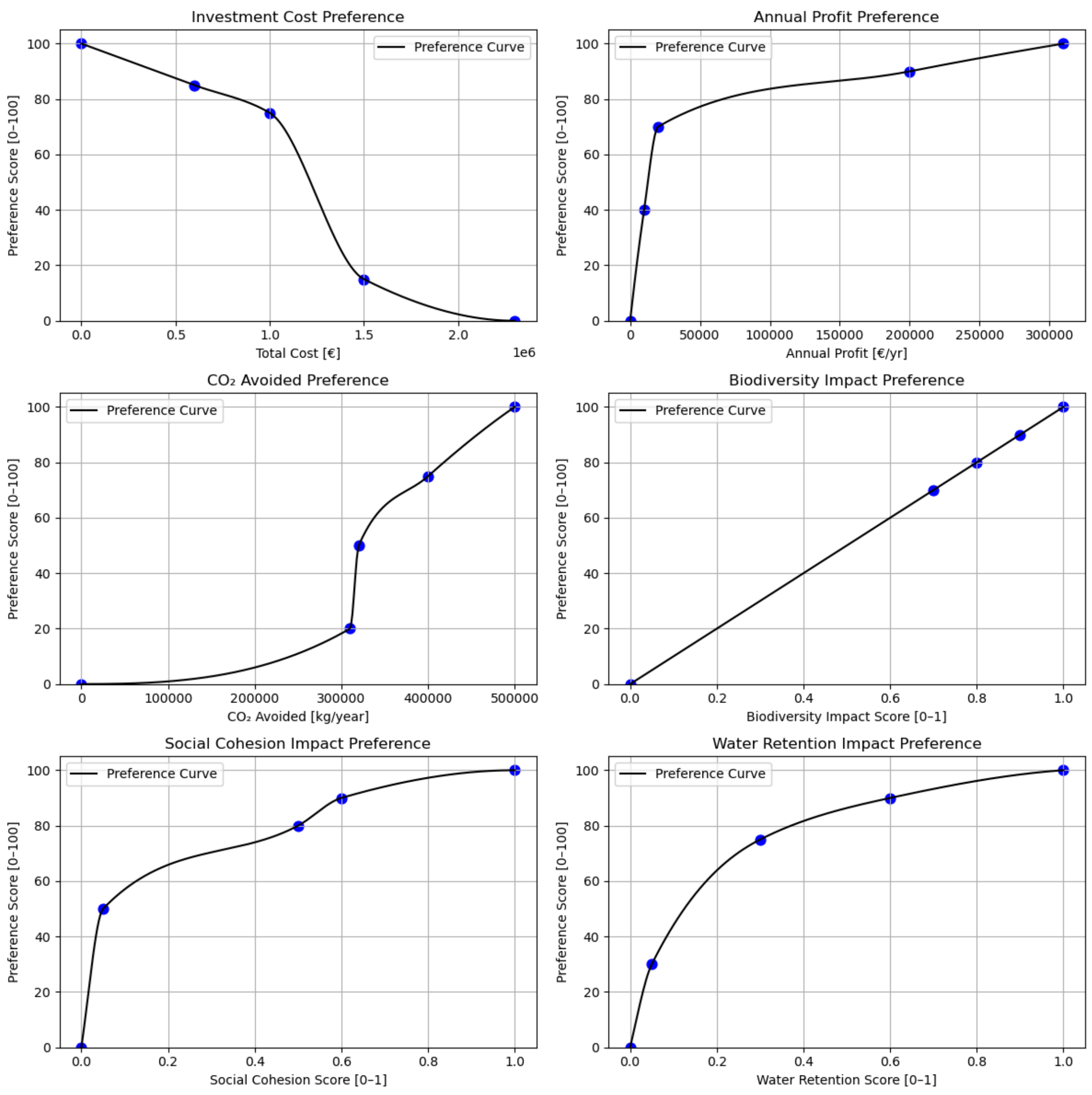
Formulas of the objective functions are defined below.

In [45]:
```
# Define formulas for 6 objective functions

def calculate_investment_cost(df):
    """[1] Calculate total investment cost across selected roof types."""
    return np.sum(df['Area'] * (df['Crepl'] + df['Cinst']))

def calculate_annual_profit(df):
    """[2] Calculate total net financial return per year."""
    return np.sum(df['Area'] * (df['ROI'] - df['Cmaint']))

def calculate_co2_impact(df, co2_per_m2=85.28):
    """[3] Calculate total CO₂ avoided (kg/year), only for yellow ('Geel') roofs
    geel_mask = df['Dak type'].str.lower() == 'geel'
    return np.sum(df.loc[geel_mask, 'Area']) * co2_per_m2

def calculate_biodiversity_impact(df, gamma=1.2, delta=2, max_value=0.2559):
    """[4] Calculate normalized Total Biodiversity Impact Score (scaled to max=1
    weighted_sum = np.sum(
        df['Area'] *
        (df['Biodiv effect'] / 100) ** gamma *
        (df['Nbio'] / 100) ** delta
    )
    total_area = np.sum(df['Area'])
    raw_score = weighted_sum / total_area if total_area > 0 else 0
    return min(raw_score / max_value, 1.0)  # Clamp at 1 to avoid overshoot

def calculate_social_impact(df, gamma=1.2, delta=2, max_value=0.1693):
    """[5] Calculate normalized Total Social Cohesion Impact Score (scaled to ma
    weighted_sum = np.sum(
        df['Area'] *
        (df['Sociaal effect'] / 100) ** gamma *
        (df['Nsoc'] / 100) ** delta
    )
    total_area = np.sum(df['Area'])
    raw_score = weighted_sum / total_area if total_area > 0 else 0
    return min(raw_score / max_value, 1.0)

def calculate_water_impact(df, gamma=1.2, delta=2, max_value=0.1288):
    """[6] Calculate normalized Total Water Retention Impact Score (scaled to ma
    weighted_sum = np.sum(
        df['Area'] *
        (df['Water effect'] / 100) ** gamma *
        (df['Nwat'] / 100) ** delta
    )
    total_area = np.sum(df['Area'])
    raw_score = weighted_sum / total_area if total_area > 0 else 0
    return min(raw_score / max_value, 1.0)
```

Defining and Plotting preference curves

Preferences and outcomes of objectives are defined and plotted.

In [46]:
```
# Load libraries
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import pchip_interpolate

# Decision-maker-defined preference points:
# x point: is the ouctome of the objective to which a preference score is define
# y pont: corresponding preference score

# 1. Investment Cost (€) – lower is better
x_cost = [0, 600000, 1_000_000, 1_500_000, 2_300_000]
y_cost = [100, 85, 75, 15, 0]

# 2. Annual Profit (€/yr) – higher is better
x_profit = [0, 10_000, 20_000, 200_000, 310_000]
y_profit = [0, 40, 70, 90, 100]

# 3. CO₂ Avoided (kg/year) – higher is better
x_co2 = [0, 310_000, 320_000, 400_000, 500_000]
y_co2 = [0, 20, 50, 75, 100]

# 4. Biodiversity Impact Score [0–1] – higher is better
x_bio = [0, 0.7, 0.8, 0.9, 1.0]
y_bio = [0, 70, 80, 90, 100]

# 5. Social Cohesion Impact Score [0–1] – higher is better
x_soc = [0.0, 0.05, 0.5, 0.6, 1.0]
y_soc = [0, 50, 80, 90, 100]

# 6. Water Retention Impact Score [0–1] – higher is better
x_water = [0.0, 0.05, 0.3, 0.6, 1.0]
y_water = [0, 30, 75, 90, 100]

# Plot the defined preference functions with points defined from above so decisi

fig, axs = plt.subplots(3, 2, figsize=(12, 12))
axs = axs.flatten()

x_curves = [x_cost, x_profit, x_co2, x_bio, x_soc, x_water]
y_curves = [y_cost, y_profit, y_co2, y_bio, y_soc, y_water]
titles = [
    'Investment Cost Preference',
    'Annual Profit Preference',
    'CO₂ Avoided Preference',
    'Biodiversity Impact Preference',
    'Social Cohesion Impact Preference',
    'Water Retention Impact Preference'
]
xlabels = [
    'Total Cost [€]',
    'Annual Profit [€/yr]',
    'CO₂ Avoided [kg/year]',
    'Biodiversity Impact Score [0–1]',
    'Social Cohesion Score [0–1]',
    'Water Retention Score [0–1]'
]

for i in range(6):
    x_vals = np.linspace(min(x_curves[i]), max(x_curves[i]), 300)
    y_vals = pchip_interpolate(x_curves[i], y_curves[i], x_vals)
    axs[i].plot(x_vals, y_vals, label='Preference Curve', color='black')
    axs[i].scatter(x_curves[i], y_curves[i], color='blue', s=60)
    axs[i].set_title(titles[i])
    axs[i].set_xlabel(xlabels[i])
    axs[i].set_ylabel('Preference Score [0–100]')
    axs[i].grid(True)
    axs[i].set_ylim(0, 105)
    axs[i].legend()

plt.tight_layout()
plt.show()
```



The plots above help stakeholders to understand how they value the outcome of objectives.

Bounds

Below the bounds are set. The bound ensures that eaxh surface roofsurface n gets at leas one rooftype intervention.

In [47]:
```python
# Get all unique surface IDs from the design space
unique_surface_ids = design_space_df['Surface_ID'].unique()

# Count how many surfaces there are
N = len(unique_surface_ids)

# Create a list of bounds for each surface
bounds = []
for surface_id in unique_surface_ids:
    # Count how many roof type options are available for this surface
    num_options = len(design_space_df[design_space_df['Surface_ID'] == surface_i

    # Select from the range of values in the designspace one rooftype
    bounds.append([0, num_options - 1])  # Choose one integer rooftype for each
```

Constraints

No constraints have been used for this optimization, instead constraints have been introduced prior in creating the final design_space_df.

Define weights

In [48]:
```
# Stakeholder weights
w_cost   = 0.30
w_profit = 0.30
w_co2    = 0.1
w_bio    = 0.1
w_soc    = 0.1
w_wat    = 0.1

weights = [w_cost, w_profit, w_co2, w_bio, w_soc, w_wat] # make a list of weight
weights = [w / sum(weights) for w in weights]  # Safety check: if weights do not
```

Define objective function for algorithm

In [49]:
```
# Clamp preference score between 0 and 100
def check_p_score(p):
    """Clamp preference scores between 0 and 100"""
    if np.isnan(p) or np.isinf(p):
        return 0  # Return lowest possible score if invalid
    return max(0, min(100, p))  # Clamp scores [0, 100]

# Decode selection indices into actual design choices (SELECTED ROOFTYPES)
def decode_solution(selection_indices, design_space_df):
    """Extract the selected row per roof surface using index selections."""
    # Get all unique surface IDs from the design space
    unique_surfaces = design_space_df['Surface_ID'].unique()
    selected_rows = []
    # For each surface, find the selected intervention based on the index

    for surface_id, option_index in zip(unique_surfaces, selection_indices):
        # Get all intervention options for the current surface
        options_for_surface = design_space_df[design_space_df['Surface_ID'] == s
        # Check if the index is valid; raise error if not

        if not (0 <= option_index < len(options_for_surface)):
            raise IndexError(f"Invalid index {option_index} for surface {surface
        # Append the selected option (row) to the result

        selected_rows.append(options_for_surface.iloc[option_index])
    # Return DataFrame with one selected intervention per surface

    return pd.DataFrame(selected_rows)
# Final objective function for the genetic algorithm

def objective(selection_matrix):
    """Evaluate preference scores for a population of roof intervention selectio
    prefs_list = []  # Will store the 6 objective scores for each individual
    for selection_indices in selection_matrix:

        selected_df = decode_solution(selection_indices, design_space_df)
        cost = calculate_investment_cost(selected_df)
        profit = calculate_annual_profit(selected_df)
        co2 = calculate_co2_impact(selected_df)
        bio = calculate_biodiversity_impact(selected_df)
        soc = calculate_social_impact(selected_df)
        wat = calculate_water_impact(selected_df)

        scores = [
            check_p_score(pchip_interpolate(x_cost,   y_cost,   cost)),
            check_p_score(pchip_interpolate(x_profit, y_profit, profit)),
            check_p_score(pchip_interpolate(x_co2,    y_co2,    co2)),
            check_p_score(pchip_interpolate(x_bio,    y_bio,    bio)),
            check_p_score(pchip_interpolate(x_soc,    y_soc,    soc)),
            check_p_score(pchip_interpolate(x_water,  y_water,  wat))
        ]

        prefs_list.append(scores)
    prefs_array = np.array(prefs_list).T # Transpose into proper format
    assert prefs_array.shape[1] > 0, "No valid individuals evaluated."


    return weights, prefs_array.tolist()
```

Run and adjust algorithm

In [50]:
```
# Run the Genetic Algorithm
# change how algorithm explores options
options = {
    'n_bits': 4,      # Number of bits per variable
    'n_iter': 100,     # Number of generations
    'n_pop': 150,     # Number of populations
    'r_cross': 0.8,    # Cross over rate
    'max_stall': 10,  # Stop early if no improvement
    'aggregation': 'tetra',
    'var_type': 'int'
}

print("Running Genetic Algorithm...")
ga = GeneticAlgorithm(objective=objective, constraints=[], bounds=bounds, option
score_opt, selection_opt, _ = ga.run()

# Decode and show result
best_design_df = decode_solution(selection_opt, design_space_df)
display(best_design_df[['Surface_ID', 'Building', 'Buurt', 'Area', 'Dak type']])
```

```
Running Genetic Algorithm...
The type of aggregation is set to tetra
Generation    Best score    Mean              Max stall    Diversity    Number of no
n-feasible results
No initial starting point for the optimization with tetra is given. A random popu
lation is generated.
0             -100.0        -43.0046          1            0.255        0
1             -100.0        -53.2015          1            0.298        0
2             -100.0        -68.2053          3            0.289        0
3             -100.0        -67.8445          4            0.298        0
4             -100.0        -78.2344          1            0.294        0
5             -100.0        -75.686           2            0.288        0
6             -100.0        -78.8351          3            0.297        0
7             -100.0        -79.3388          1            0.296        0
8             -100.0        -82.4039          2            0.309        0
9             -100.0        -82.7839          3            0.291        0
10            -100.0        -84.5332          4            0.307        0
11            -100.0        -74.3835          5            0.305        0
12            -100.0        -73.3821          1            0.303        0
13            -100.0        -78.7345          2            0.294        0
14            -100.0        -81.879           2            0.294        0
15            -100.0        -67.2336          3            0.299        0
16            -100.0        -76.588           4            0.306        0
17            -100.0        -83.0113          5            0.32         0
18            -100.0        -78.6949          6            0.324        0
19            -100.0        -73.5363          4            0.32         0
20            -100.0        -92.411           5            0.314        0
21            -100.0        -84.5804          1            0.318        0
22            -100.0        -78.145           2            0.318        0
23            -100.0        -77.4849          3            0.329        0
24            -100.0        -82.5133          1            0.311        0
25            -100.0        -82.6246          1            0.318        0
26            -100.0        -76.5495          2            0.328        0
27            -100.0        -78.6892          3            0.335        0
28            -100.0        -79.1584          3            0.338        0
29            -100.0        -71.2116          4            0.342        0
30            -100.0        -71.0405          5            0.34         0
31            -100.0        -78.3992          6            0.327        0
32            -100.0        -74.9935          7            0.323        0
33            -100.0        -59.6729          8            0.318        0
34            -100.0        -77.1594          9            0.319        0
35            -100.0        -62.1503          10           0.319        0
Stopped at gen 35
Execution time was 909.3716 seconds
```

	Surface_ID	Building	Buurt	Area	Dak type
2	1	A_1	A	30	Groen 2
2	2	A_1	A	110	Groen 2
2	3	A_1	A	50	Geel
1	4	A_1	A	40	Groen 1
2	5	A_1	A	30	Geel
...	...	...	...	...	...
0	80	E_4	E	80	Geen
5	81	E_5	E	20	Geel
5	82	E_5	E	110	Geel
3	83	E_5	E	80	Rood
3	84	E_5	E	40	Rood

84 rows × 5 columns

3. Visualizing and loading results

The code blocks below present the optimization results in the form of visualizations, summary statistics, and tables.

Optimized output statistcis

In [51]:
```
import pandas as pd
from IPython.display import display, Markdown

# Decode the optimal design and recalculate indicators
best_design_df = decode_solution(selection_opt, design_space_df)

cost = calculate_investment_cost(best_design_df)
profit = calculate_annual_profit(best_design_df)
co2 = calculate_co2_impact(best_design_df)
bio = calculate_biodiversity_impact(best_design_df)
soc = calculate_social_impact(best_design_df)
wat = calculate_water_impact(best_design_df)

# Preference scores
scores = [
    check_p_score(pchip_interpolate(x_cost,   y_cost,   cost)),
    check_p_score(pchip_interpolate(x_profit, y_profit, profit)),
    check_p_score(pchip_interpolate(x_co2,    y_co2,    co2)),
    check_p_score(pchip_interpolate(x_bio,    y_bio,    bio)),
    check_p_score(pchip_interpolate(x_soc,    y_soc,    soc)),
    check_p_score(pchip_interpolate(x_water,  y_water,  wat))
]

# Roof statistics
roof_area_per_type = best_design_df.groupby('Dak type')['Area'].sum()
```

```
roof_count_per_type = best_design_df['Dak type'].value_counts()

num_geen = best_design_df['Dak type'].str.strip().str.lower().eq('geen').sum()
percent_geen = 100 * num_geen / len(best_design_df) if len(best_design_df) > 0 e

# Main preference table
overview_df = pd.DataFrame({
    'Indicator': [
        'Cost (€)',
        'Profit (€)',
        'CO₂ Reduction (kg)',
        'Biodiversity',
        'Social Cohesion',
        'Water Retention'
    ],
    'Raw Value': [
        f"{cost:,.0f}",
        f"{profit:,.0f}",
        f"{co2:,.0f}",
        f"{bio:.4f}",
        f"{soc:.4f}",
        f"{wat:.4f}"
    ],
    'Preference Score': [f"{s:.1f}" for s in scores],
    'Weight': [f"{w:.2f}" for w in weights]
})

# Overall weighted preference score
overall_score = sum(w * s for w, s in zip(weights, scores))

final_row = pd.DataFrame({
    'Indicator': ['Overall Preference Score'],
    'Raw Value': [''],
    'Preference Score': [f"{overall_score:.2f}"],
    'Weight': ['']
})
overview_df_final = pd.concat([overview_df, final_row], ignore_index=True)

# Styling
def highlight_overall(row):
    is_last = row.name == len(overview_df_final) - 1
    return [
        'font-weight: bold' if is_last and col in ['Indicator', 'Preference Scor
        for col in overview_df_final.columns
    ]
styled_table = overview_df_final.style.apply(highlight_overall, axis=1)

# Rename 'geen' to 'No intervention'
roof_area_per_type_cleaned = roof_area_per_type.rename(index=lambda x: 'No inter
roof_count_per_type_cleaned = roof_count_per_type.rename(index=lambda x: 'No int

# Summary metrics
summary_data = [
    ['Summary', 'Total Roof Area Assigned', None, f"{roof_area_per_type.sum():,.
    ['Summary', 'Total Roof Surfaces', None, f"{len(best_design_df)}", 'count'],
    ['Summary', 'No Intervention (%)', None, f"{percent_geen:.1f}", '%']
]

# Area per type
area_data = [
    ['Area per Type', 'Area', rtype, f"{area:,.2f}", 'm²']
    for rtype, area in roof_area_per_type_cleaned.items()
]

# Count per type
count_data = [
    ['Count per Type', 'Count', rtype, f"{count}", 'surfaces']
    for rtype, count in roof_count_per_type_cleaned.items()
]

# Combine tables
combined_data = summary_data + area_data + count_data
columns = ['Category', 'Metric', 'Roof Type', 'Value', 'Unit']
combined_df = pd.DataFrame(combined_data, columns=columns)

# Display
display(Markdown("#### Summary optimized objectives and preferences"))
display(styled_table)

display(Markdown("#### Summary assigned optimized roofs"))
display(combined_df)
```

Summary optimized objectives and preferences

	Indicator	Raw Value	Preference Score	Weight
0	Cost (€)	1,002,200	74.9	0.30
1	Profit (€)	102,238	83.9	0.30
2	CO_2 Reduction (kg)	224,286	8.2	0.10
3	Biodiversity	0.2958	29.6	0.10
4	Social Cohesion	0.4301	75.4	0.10
5	Water Retention	0.3413	78.1	0.10
6	**Overall Preference Score**		**66.75**	

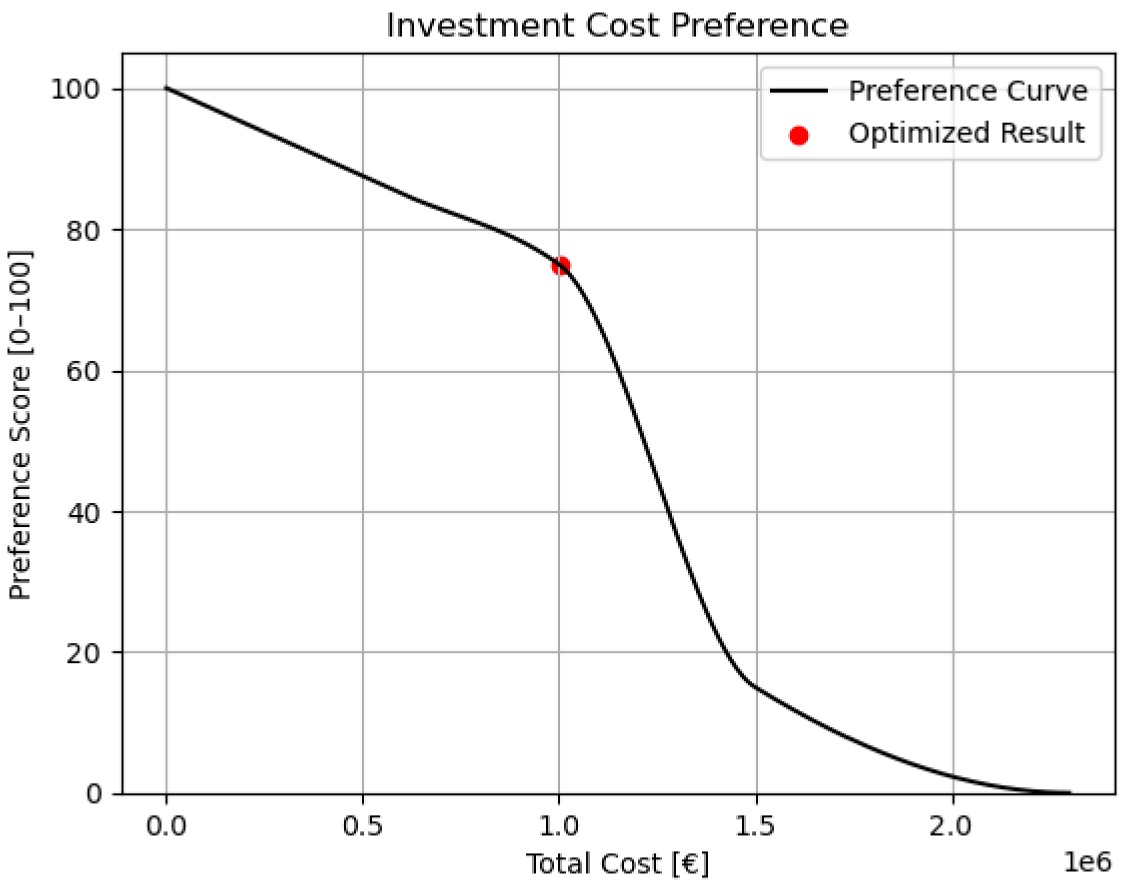
Summary assigned optimized roofs

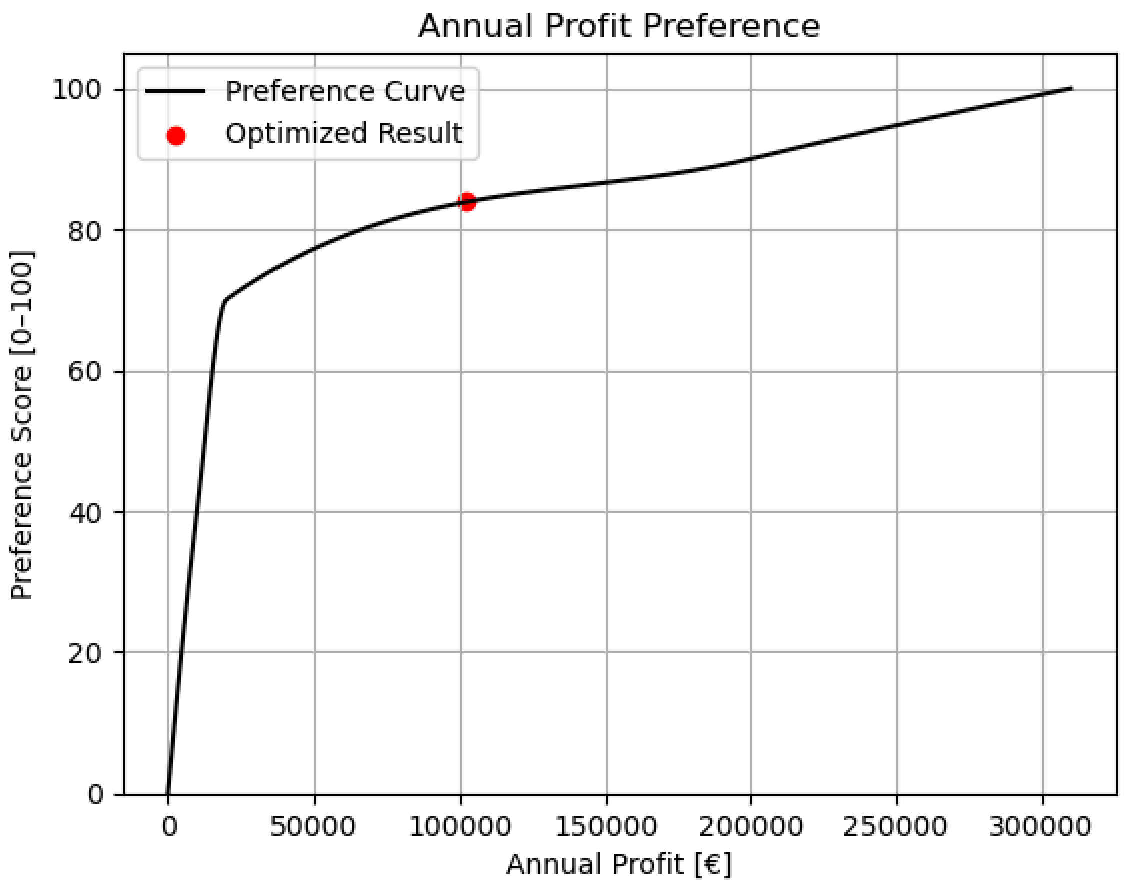
	Category	Metric	Roof Type	Value	Unit
0	Summary	Total Roof Area Assigned	None	5,810.00	m²
1	Summary	Total Roof Surfaces	None	84	count
2	Summary	No Intervention (%)	None	31.0	%
3	Area per Type	Area	Blauw	140.00	m²
4	Area per Type	Area	Geel	2,630.00	m²
5	Area per Type	Area	No intervention	1,860.00	m²
6	Area per Type	Area	Groen 1	230.00	m²
7	Area per Type	Area	Groen 2	430.00	m²
8	Area per Type	Area	Rood	520.00	m²
9	Count per Type	Count	Geel	38	surfaces
10	Count per Type	Count	No intervention	26	surfaces
11	Count per Type	Count	Rood	9	surfaces
12	Count per Type	Count	Groen 2	6	surfaces
13	Count per Type	Count	Groen 1	3	surfaces
14	Count per Type	Count	Blauw	2	surfaces

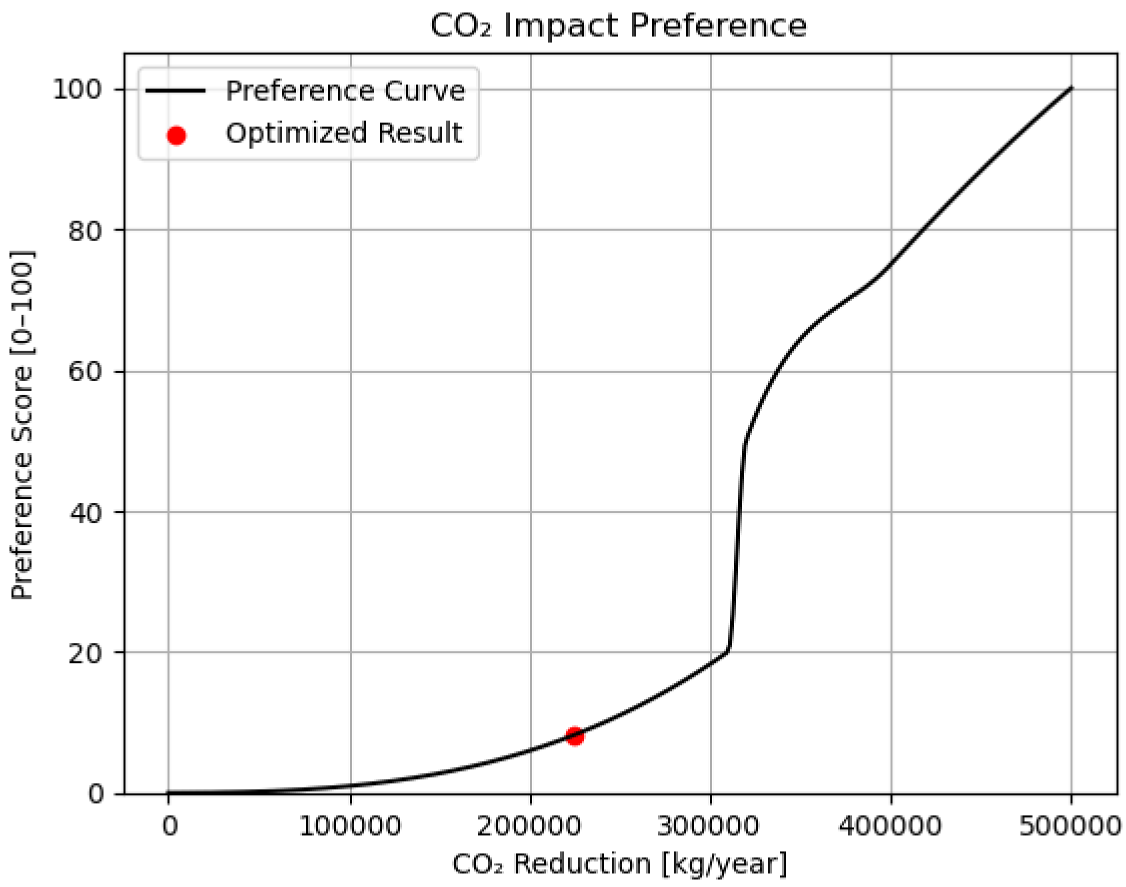
Preference functions with optimized points

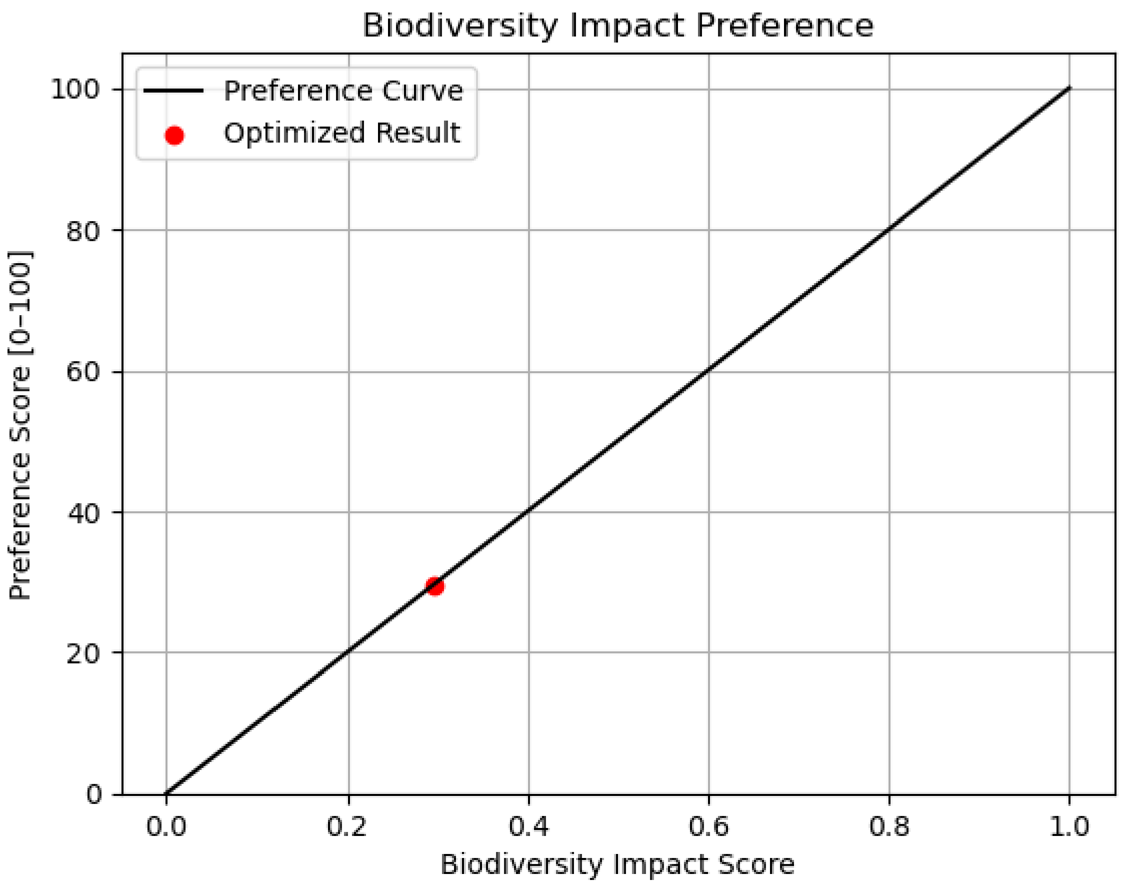
This code plots the earlier defined preference function with the optimized result.

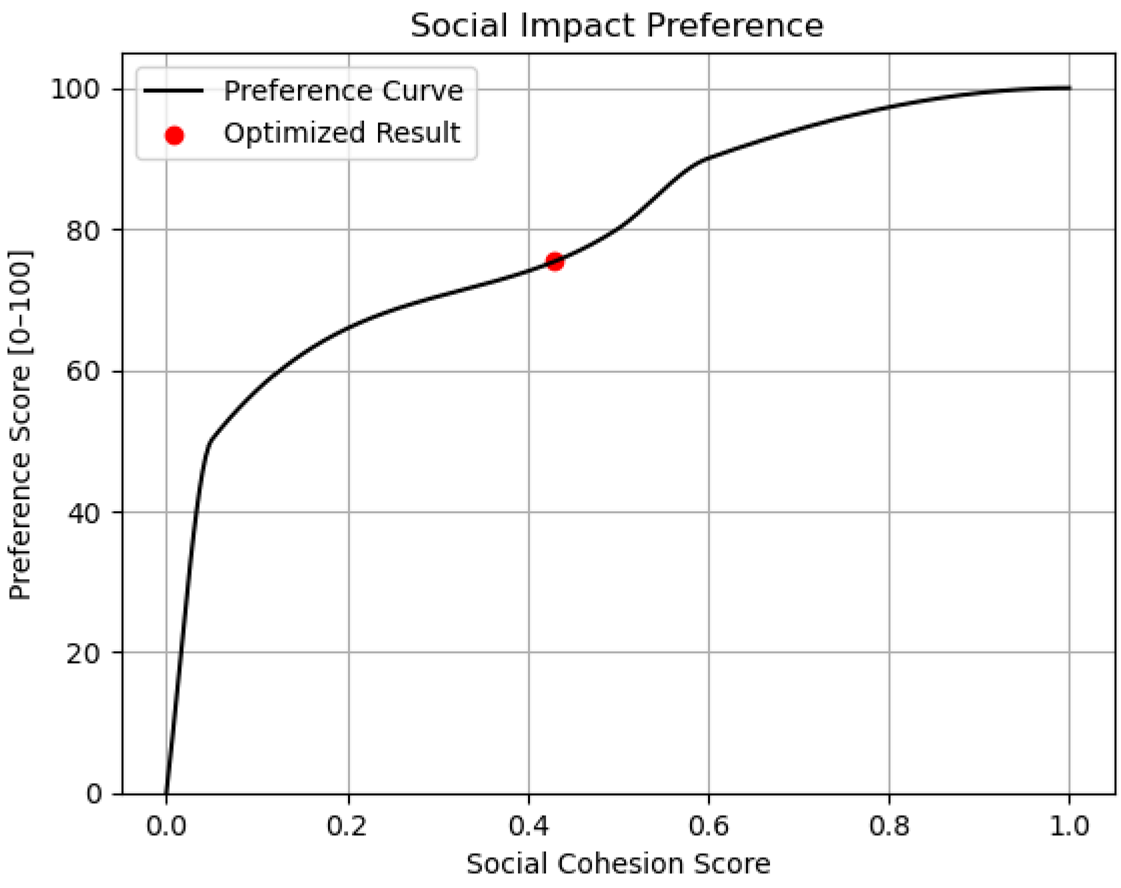
In [52]:
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import pchip_interpolate

# Use the actual optimized result dataframe
cost   = calculate_investment_cost(best_design_df)
profit = calculate_annual_profit(best_design_df)
co2    = calculate_co2_impact(best_design_df)
biodiv = calculate_biodiversity_impact(best_design_df)
soc    = calculate_social_impact(best_design_df)
water  = calculate_water_impact(best_design_df)

# Define all input data: x/y values and optimized result values
preference_data = [
    ("Investment Cost Preference", "Total Cost [€]", x_cost, y_cost, cost),
    ("Annual Profit Preference", "Annual Profit [€]", x_profit, y_profit, profit
    ("CO₂ Impact Preference", "CO₂ Reduction [kg/year]", x_co2, y_co2, co2),
    ("Biodiversity Impact Preference", "Biodiversity Impact Score", x_bio, y_bio
    ("Social Impact Preference", "Social Cohesion Score", x_soc, y_soc, soc),
    ("Water Retention Impact Preference", "Water Retention Score", x_water, y_wa
]

# Plot each preference function with optimized result
for title, xlabel, x_vals, y_vals, result_val in preference_data:
    c = np.linspace(min(x_vals), max(x_vals), 300)
    pref = pchip_interpolate(x_vals, y_vals, c)
    result_point = pchip_interpolate(x_vals, y_vals, result_val)

    plt.figure()
    plt.plot(c, pref, label="Preference Curve", color='black')
    plt.scatter([result_val], [result_point], color='red', label='Optimized Resu
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel("Preference Score [0-100]")
    plt.ylim(0, 105)
    plt.grid(True)
    plt.legend()
    plt.show()
```

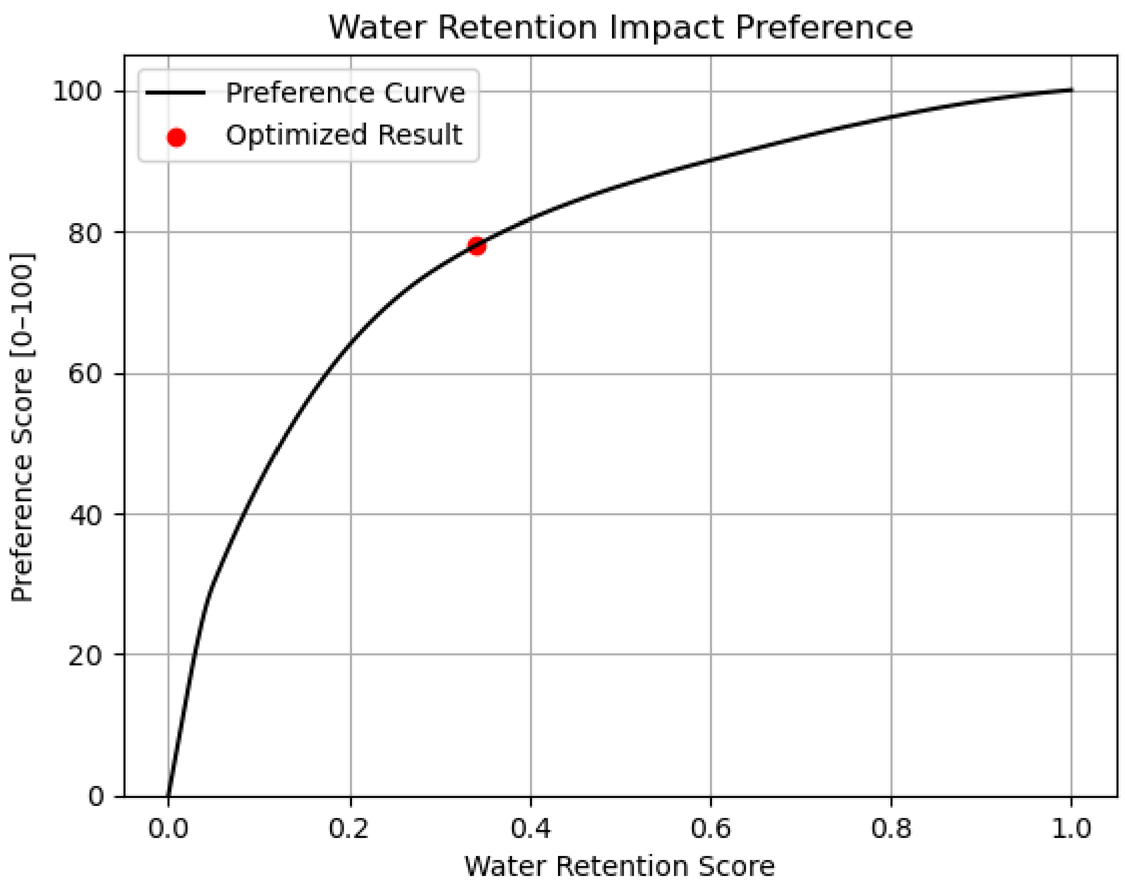



Annual Profit Preference
Preference Curve
Optimized Result
Preference Score [0–100]
Annual Profit [€]
0
20
40
60
80
100
0
50000
100000
150000
200000
250000
300000


CO₂ Impact Preference
Preference Curve
Optimized Result
Preference Score [0–100]
CO₂ Reduction [kg/year]
0
20
40
60
80
100
0
100000
200000
300000
400000
500000


Biodiversity Impact Preference
Preference Curve
Optimized Result
Preference Score [0–100]
Biodiversity Impact Score
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0


Social Impact Preference
Preference Curve
Optimized Result
Preference Score [0–100]
Social Cohesion Score
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

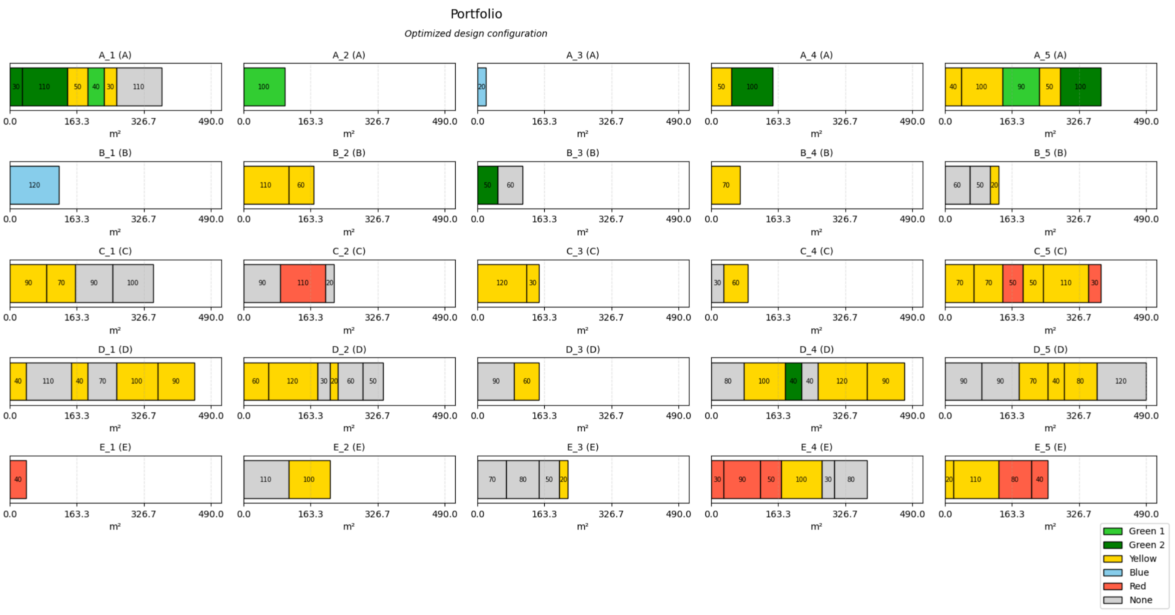


Plot portfolio and optimized roof surfaces

To directly interpretate result, it is handy to plot the assigned roof types after optimization.

In [53]:
```
import matplotlib.pyplot as plt
import pandas as pd
import matplotlib.patches as patches
import numpy as np

# Use the optimized assignment
df = best_design_df.copy()

# Color map (harmonized with other plots)
color_map = {
    'Green 1': 'limegreen',
    'Green 2': 'green',
    'Yellow': 'gold',
    'Blue': 'skyblue',
    'Red': 'tomato',
    'None': 'lightgray'
}

# Translate Dutch Dak type to English
dak_type_translation = {
    'Groen 1': 'Green 1',
    'Groen 2': 'Green 2',
    'Geel': 'Yellow',
    'Blauw': 'Blue',
    'Rood': 'Red',
    'Geen': 'None'
```

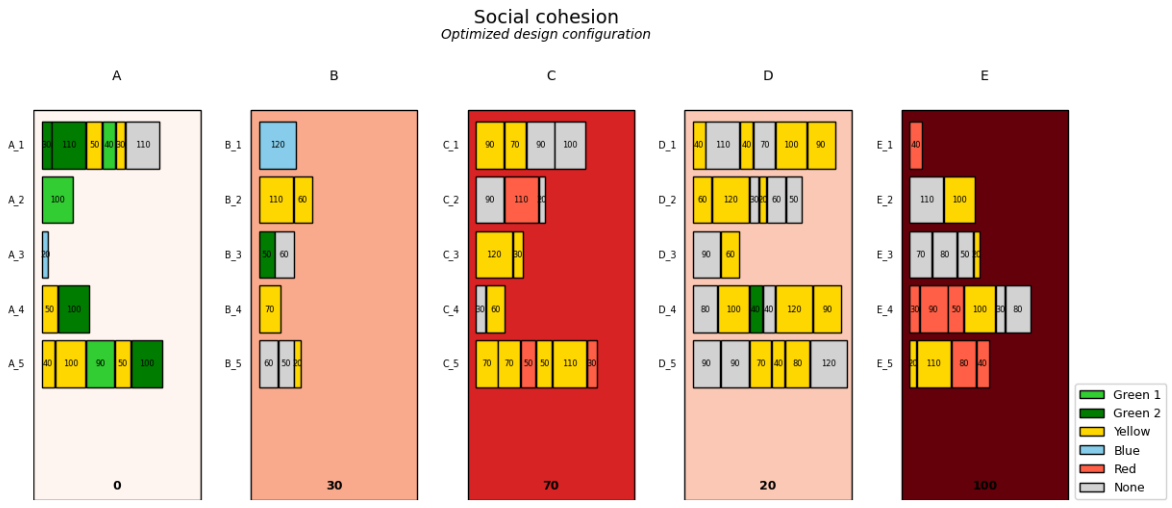
```
}

# Compute total area per building
building_totals = df.groupby('Building')['Area'].sum()
global_max_area = building_totals.max()

# Group by building
grouped = df.groupby('Building')

# Create plot
fig, axs = plt.subplots(6, 5, figsize=(18, 10))
axs = axs.flatten()

for i, (building_id, group) in enumerate(grouped):
    ax = axs[i]
    buurt = group['Buurt'].iloc[0]
    ax.set_title(f"{building_id} ({buurt})", fontsize=10)

    x_start = 0
    for _, row in group.iterrows():
        dak_type_nl = str(row['Dak type']).strip()
        dak_type_en = dak_type_translation.get(dak_type_nl, dak_type_nl)
        color = color_map.get(dak_type_en, 'gray')  # fallback to gray
        width = row['Area']
        rect = patches.Rectangle((x_start, 0.1), width, 0.8, edgecolor='black',
        ax.add_patch(rect)
        ax.text(x_start + width / 2, 0.5, f"{int(width)}", ha='center', va='cent
        x_start += width

    ax.set_xlim(0, global_max_area * 1.05)
    ax.set_ylim(0, 1)
    ax.set_xticks(np.linspace(0, global_max_area, 4))
    ax.set_xlabel("m²")
    ax.set_yticks([])
    ax.grid(True, axis='x', linestyle='--', alpha=0.3)

# Hide unused subplots
for j in range(i + 1, len(axs)):
    axs[j].axis('off')

# Add legend for harmonized roof types
legend_elements = [patches.Patch(facecolor=color, edgecolor='black', label=label
                   for label, color in color_map.items()]
fig.legend(handles=legend_elements, loc='lower right', fontsize=10)

# Add title and subtitle aligned slightly to the left
fig.text(0.41, 0.91, 'Portfolio', ha='center', va='center', fontsize=14)
fig.text(0.41, 0.88, 'Optimized design configuration', ha='center', va='center',

plt.tight_layout(rect=[0, 0, 1, 0.87])  # Leave space at the top for the title
plt.show()
```

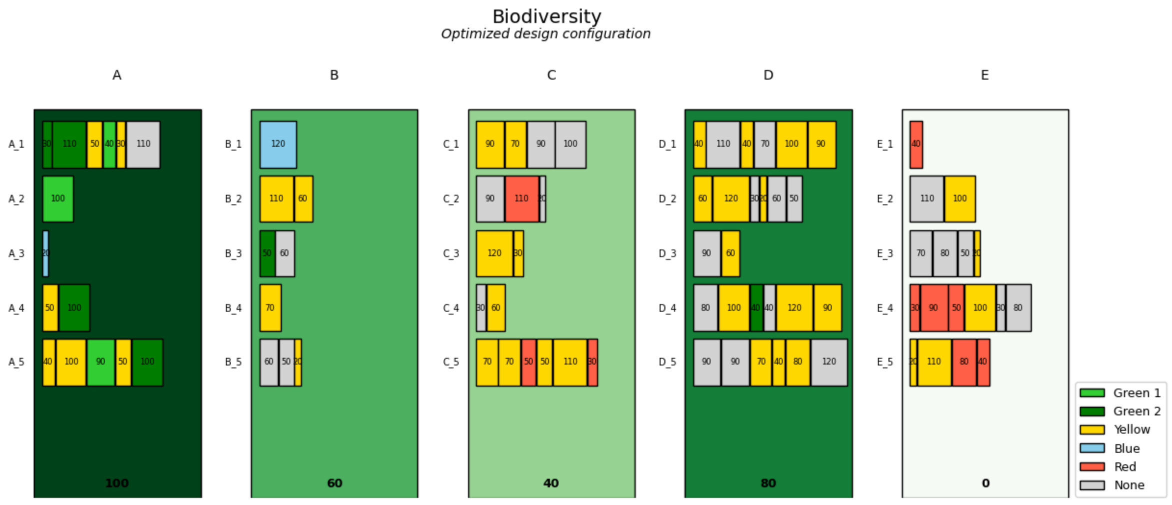


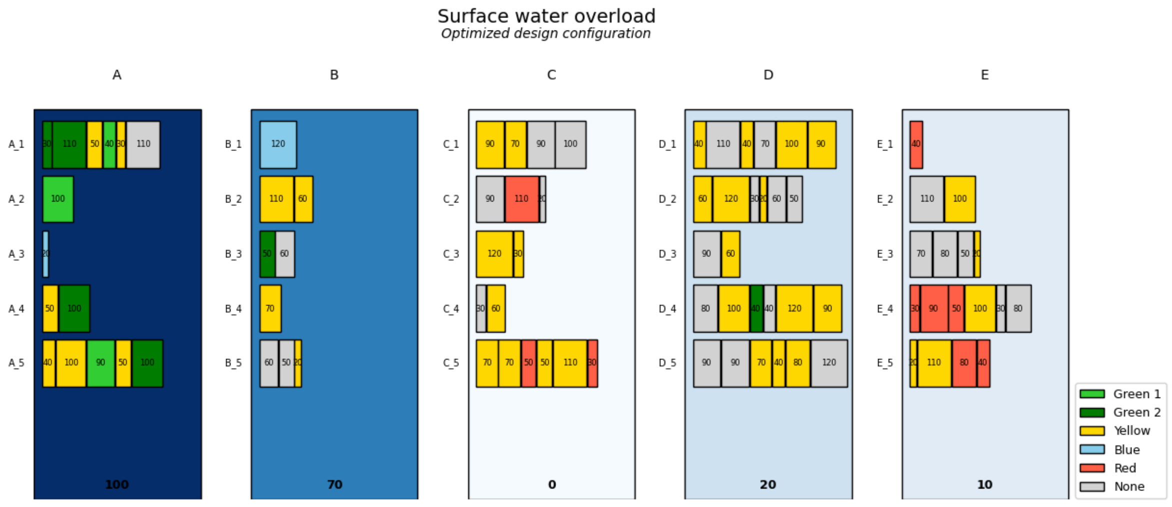
Optimized portfolio as overlayer over neighborhoods

Since objectives 3–6 are linked to specific locations, it's useful to see how roof types are allocated in relation to neighborhood needs. Therefore the code below creates a plot. However note, that if one of these spatial objectives is set to zero weight, its associated neighborhood needs no longer influence the outcome, making the corresponding choropleth irrelevant.

In [57]:
```python
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import matplotlib.colors as mcolors
import pandas as pd
import numpy as np

# Define custom color map for optimized roof types
color_map = {
    'Green 1': 'limegreen',
    'Green 2': 'green',
    'Yellow': 'gold',
    'Blue': 'skyblue',
    'Red': 'tomato',
    'None': 'lightgray'
}

# Translate Dutch to English
dak_type_translation = {
    'Groen 1': 'Green 1',
    'Groen 2': 'Green 2',
    'Geel': 'Yellow',
    'Blauw': 'Blue',
    'Rood': 'Red',
    'Geen': 'None'
}

def plot_optimized_roofs_per_neighborhood(best_design_df, theme_col, title, cmap
    best_design_df['Buurt'] = best_design_df['Buurt'].astype(str).str.strip()
    df.index = df.index.astype(str).str.strip()

    challenge_data = df[theme_col].to_dict()
    building_totals = best_design_df.groupby('Building')['Area'].sum()
    global_max_area = building_totals.max()

    neighborhoods = sorted(best_design_df['Buurt'].unique())
    buildings_per_neigh = {buurt: best_design_df[best_design_df['Buurt'] == buur

    fig, ax = plt.subplots(figsize=(14, 6))

    bar_width = 1
    bar_gap = 0.3
    building_height = 0.12
    roof_gap = 0.02

    cmap = plt.colormaps.get_cmap(cmap_name)
    norm = mcolors.Normalize(vmin=0, vmax=100)

    for i, buurt in enumerate(neighborhoods):
        challenge_val = challenge_data.get(buurt, 0)
        neigh_color = cmap(norm(challenge_val))

        x_pos = i * (bar_width + bar_gap)
        ax.add_patch(patches.Rectangle((x_pos, 0), bar_width, 1, facecolor=neigh

        # Neighborhood letter above the bar
        ax.text(x_pos + bar_width / 2, 1.07, f"{buurt}",
                ha='center', va='bottom', fontsize=10)

        # Bold challenge value inside the bar
        ax.text(x_pos + bar_width / 2, 0.02, f"{challenge_val:.0f}",
                ha='center', va='bottom', fontsize=9, color='black', fontweight=

        grouped = buildings_per_neigh[buurt]
        for j, (building_id, group) in enumerate(grouped):
            y_base = 0.85 - j * (building_height + roof_gap)
            x_building = x_pos + 0.05
            x_start = x_building
            for _, row in group.iterrows():
                dak_type_nl = str(row['Dak type']).strip()
                dak_type_en = dak_type_translation.get(dak_type_nl, dak_type_nl)
                color = color_map.get(dak_type_en, 'gray')
                width = row['Area'] / global_max_area * (bar_width - 0.1)
                rect = patches.Rectangle((x_start, y_base), width, building_heig
                ax.add_patch(rect)

                # Only show area (not dak type)
                ax.text(x_start + width / 2, y_base + building_height / 2, f"{in
                        ha='center', va='center', fontsize=6, color='black')
                x_start += width + 0.005

            # Building ID on left side
            ax.text(x_pos - 0.05, y_base + building_height / 2, building_id,
                    va='center', ha='right', fontsize=7)

    ax.set_xlim(0, len(neighborhoods) * (bar_width + bar_gap))
    ax.set_ylim(0, 1.2)
    ax.axis('off')

    # Legend in bottom-right corner
    legend_elements = [patches.Patch(facecolor=color, edgecolor='black', label=l
                       for label, color in color_map.items()]
    ax.legend(handles=legend_elements, loc='lower right', bbox_to_anchor=(1.05,

    # Title and subtitle aligned more to the left
    fig.text(0.41, 0.91, f"{title}", ha='center', va='center', fontsize=14)
    fig.text(0.41, 0.88, 'Optimized design configuration', ha='center', va='cent

    plt.tight_layout(rect=[0, 0.05, 0.88, 0.91])
    plt.show()

# Run for each theme
plot_optimized_roofs_per_neighborhood(best_design_df, 'Nsoc', 'Social cohesion',
plot_optimized_roofs_per_neighborhood(best_design_df, 'Nbio', 'Biodiversity', 'G
plot_optimized_roofs_per_neighborhood(best_design_df, 'Nwat', 'Surface water ove
```



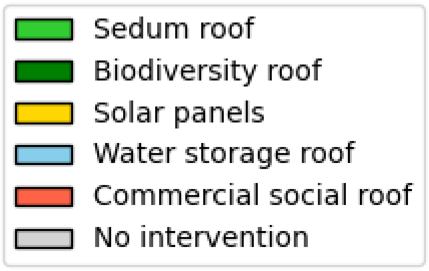




Extra info

A different legend makes output understandable.

In [55]:
```
import matplotlib.pyplot as plt
import matplotlib.patches as patches

# Define custom color map with descriptive rooftop labels
color_map = {
    'Sedum roof': 'limegreen',
    'Biodiversity roof': 'green',
    'Solar panels': 'gold',
    'Water storage roof': 'skyblue',
    'Commercial social roof': 'tomato',
    'No intervention': 'lightgray'
}

# Create figure and axis only for the legend
fig, ax = plt.subplots(figsize=(4, 2))
ax.axis('off')

# Create legend patches
legend_elements = [patches.Patch(facecolor=color, edgecolor='black', label=label
                   for label, color in color_map.items()]

# Add legend to plot
ax.legend(handles=legend_elements, loc='center', fontsize=10, frameon=True, ncol

plt.tight_layout()
plt.show()
```



C Python code - choropleths

See code next page.

Plotting Choropleths

The code below presents how the Choropleths were created.

Import libraries

In [35]:
```
import pandas as pd
import geopandas as gpd
from shapely.geometry import Point
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
```

Load and prepare ata

This code loads three data sets: the real estate portfolio of The Hague, neighborhood-level indicators (e.g., heat stress, social cohesion), and a shape file of The Hague's neighborhood boundaries. It merges the indicators with the shape file and converts the real estate data into a spatial format for mapping and analysis. Data and SHP files have been collected from: https://denhaag.incijfers.nl

In [36]:
```
# Load data
real_estate_df = pd.read_excel("Data/RealEstate_Cleaned_Final2.xlsx")
neighborhood_data = pd.read_excel("Data/Final data all themes clean - cleaned.xl
gdf_buurten = gpd.read_file("Data/buurten.shp")

# Preprocess neighborhood data and merge on common column
gdf_plot = gdf_buurten.copy()
neighborhood_data_plot = neighborhood_data.copy()
gdf_plot["BUURTNAAM"] = gdf_plot["BUURTNAAM"].str.strip().str.lower()
neighborhood_data_plot["BUURTNAAM"] = neighborhood_data_plot["BUURTNAAM"].str.st
gdf_merged = gdf_plot.merge(neighborhood_data_plot, on="BUURTNAAM", how="left")

# Create GeoDataFrame from real estate
geometry = [Point(xy) for xy in zip(real_estate_df["x_clean"], real_estate_df["y
gdf_real_estate_with_buurt = gpd.GeoDataFrame(real_estate_df, geometry=geometry,
```

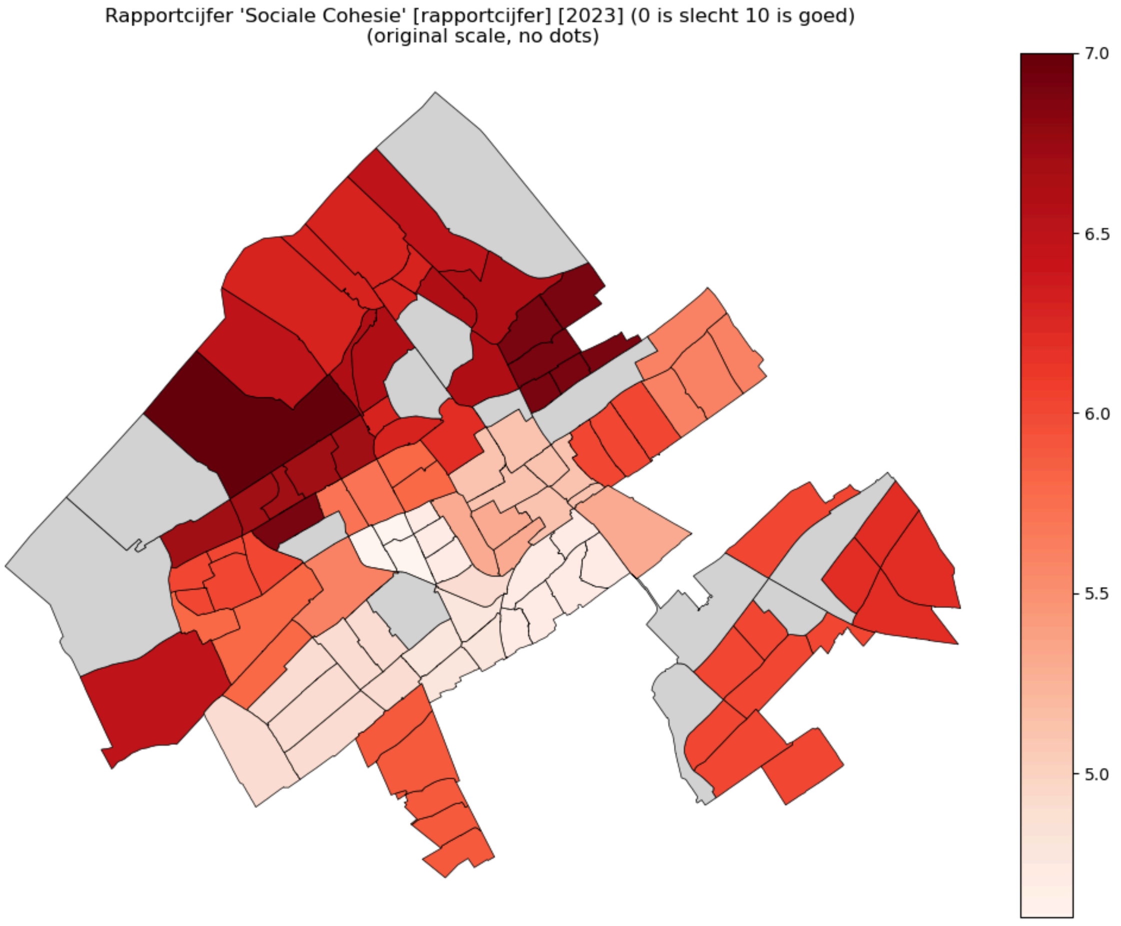
Select columns to plot

In [37]:
```
columns_to_plot = [
    ("Rapportcijfer 'Sociale Cohesie' [rapportcijfer] [2023] (0 is slecht 10 is
    ("Oppervlakte gebied 'meer dan 10 cm overstroming' [m²] [2021]", 'Blues', Fa
    ('waarnemingen in de 10-jaar periode 2007-2016\nvan de volgende soorten: bro
    ('Percentage van gebied dat zeer gevoelig is voor hittestress op zomerse dag
]
```

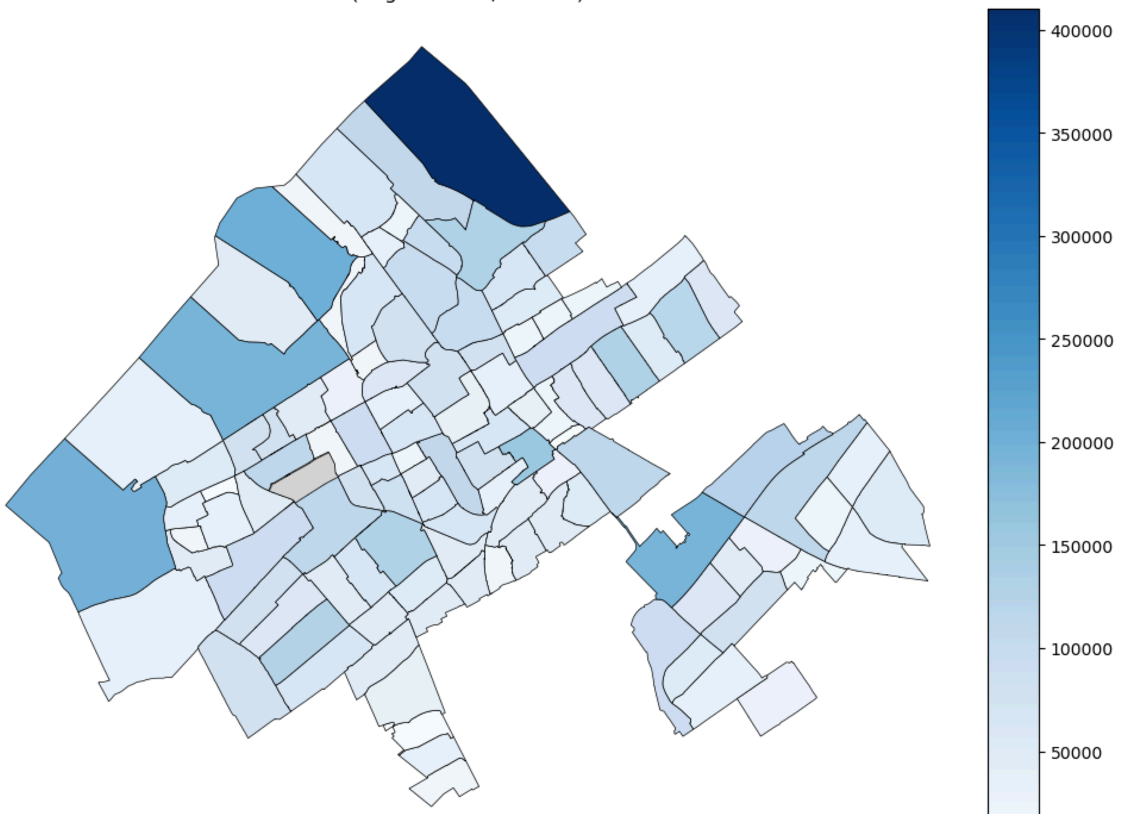
Plot neighborhood challenges as choropleths

Let's first plot the data as choropleths.

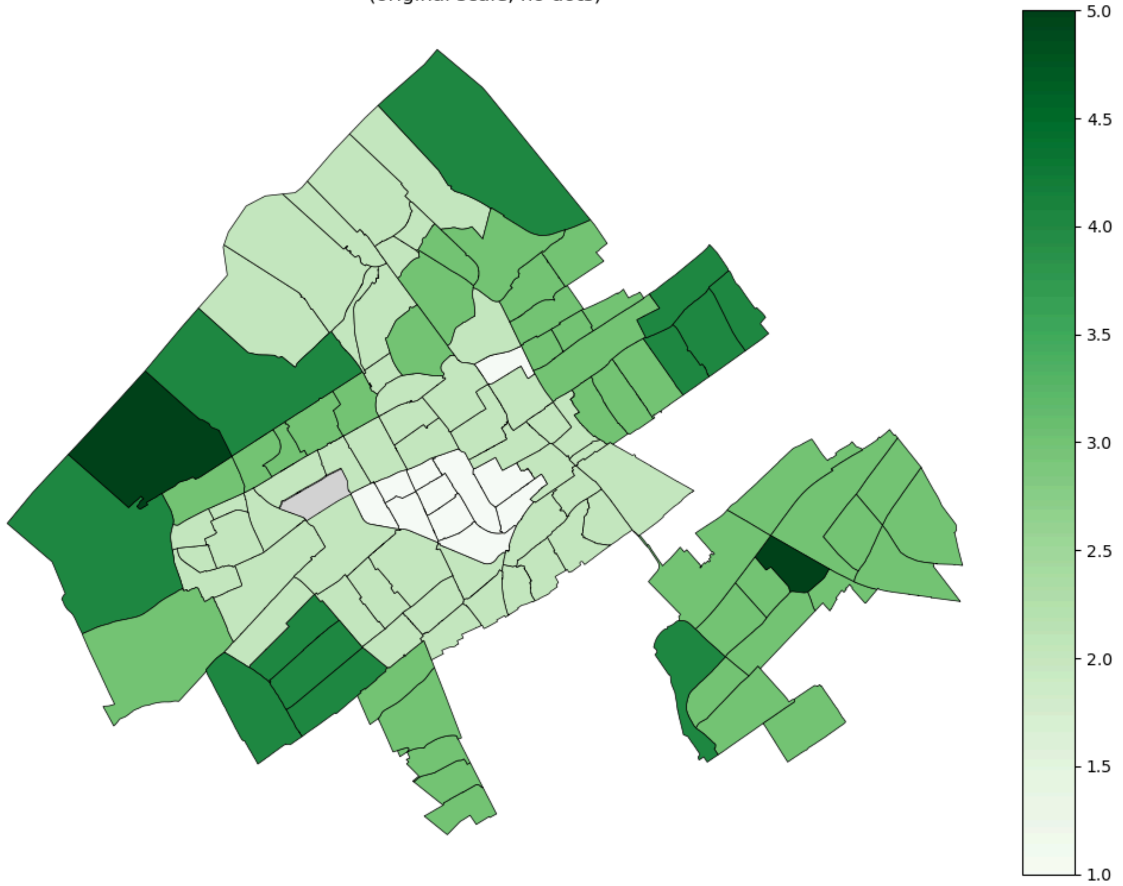
In [38]:
```
for col, cmap, reverse, title in columns_to_plot:
    # Plot original data (no normalization, no dots)
    fig, ax = plt.subplots(figsize=(10, 10))
    divider = make_axes_locatable(ax)
    cax = divider.append_axes("right", size="5%", pad=0.1)

    gdf_merged.plot(
        column=col,
        cmap=cmap,
        linewidth=0.5,
        ax=ax,
        edgecolor="black",
        legend=True,
        cax=cax,
        missing_kwds={"color": "lightgrey", "label": "No data"}
    )

    ax.set_title(f"{col}\n(original scale, no dots)", fontsize=12)
    ax.axis("off")
    plt.tight_layout()
    plt.show()
```

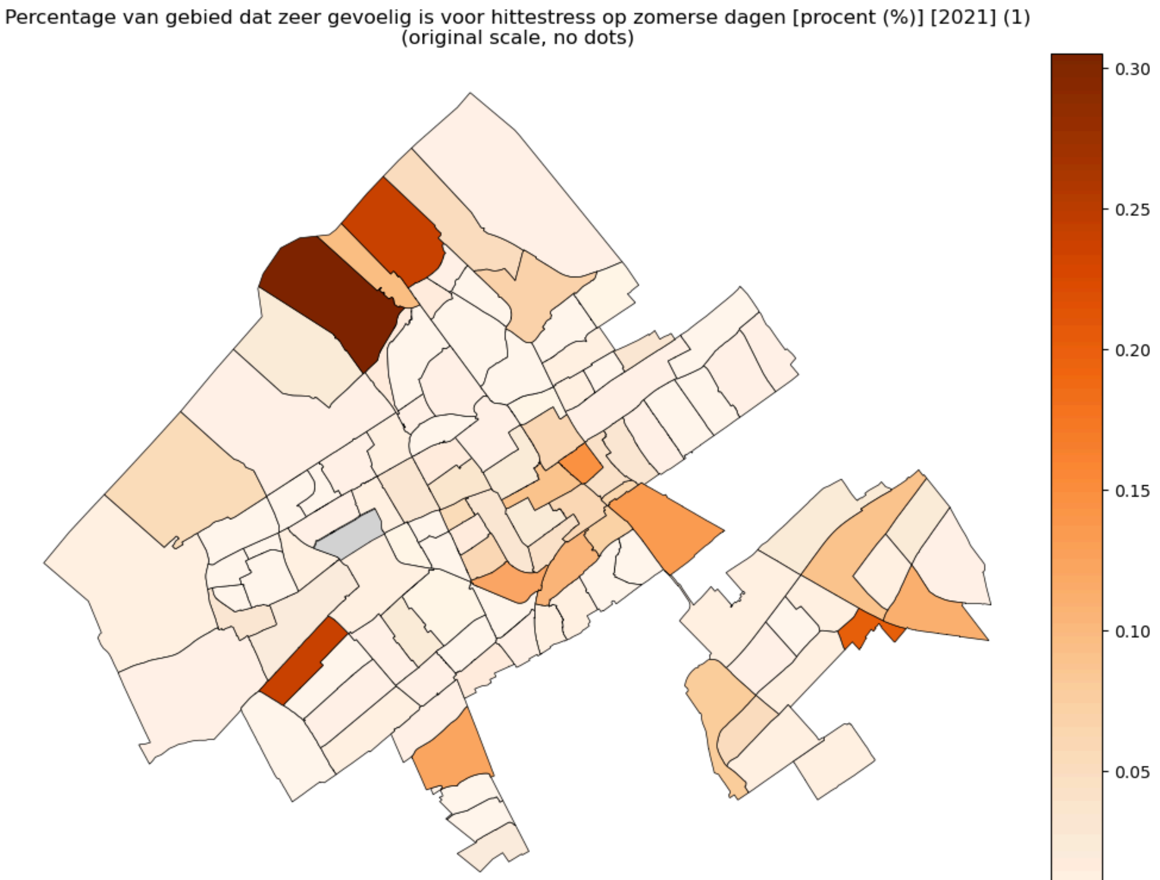

Rapportcijfer 'Sociale Cohesie' [rapportcijfer] [2023] (0 is slecht 10 is goed)
(original scale, no dots)
7.0
6.5
6.0
5.5
5.0

Oppervlakte gebied 'meer dan 10 cm overstroming' [m²] [2021]
(original scale, no dots)



waarnemingen in de 10-jaar periode 2007-2016
van de volgende soorten: broedvogels; vleermuizen;
landzoogdieren; reptielen; amfibieën &
zoetwatervissen; libellen; dagvlinders; sprinkhanen
& krekels; vaatplanten.
(original scale, no dots)

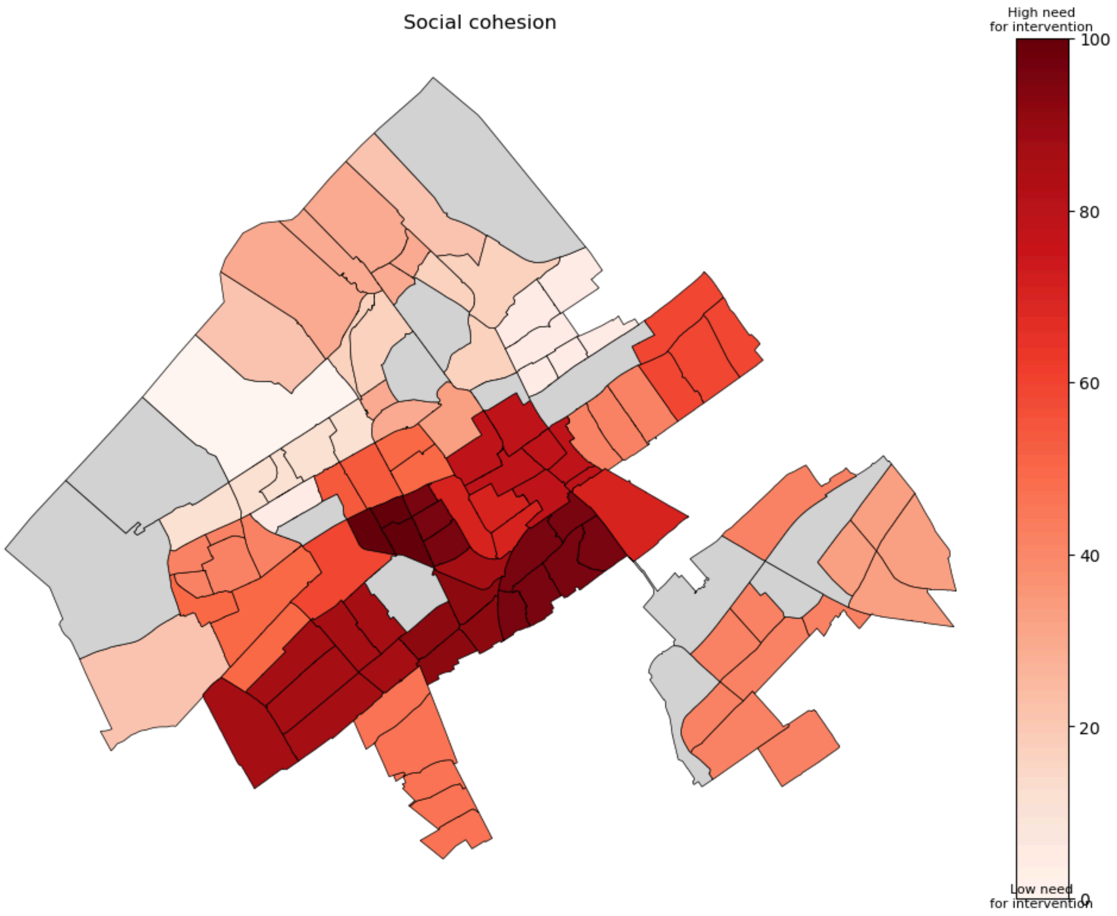


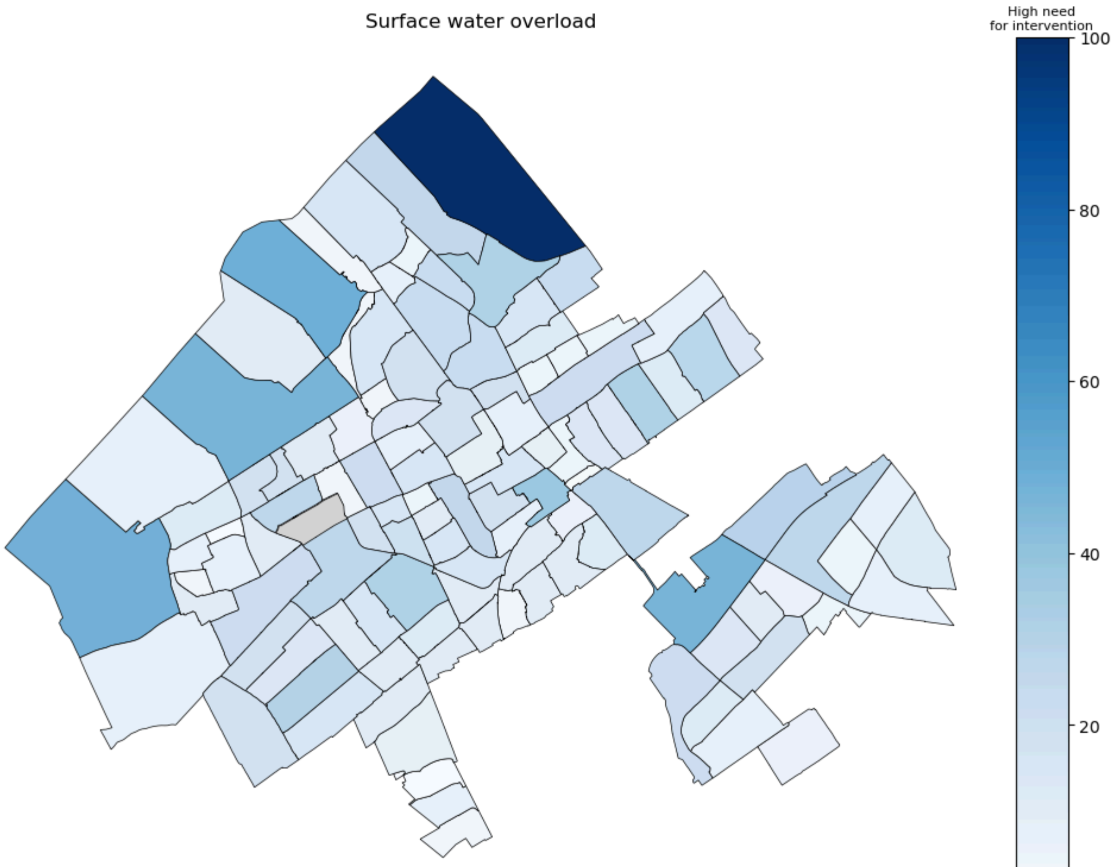

Percentage van gebied dat zeer gevoelig is voor hittestress op zomerse dagen [procent (%)] [2021] (1)
(original scale, no dots)
0.30
0.25
0.20
0.15
0.10
0.05
0.00

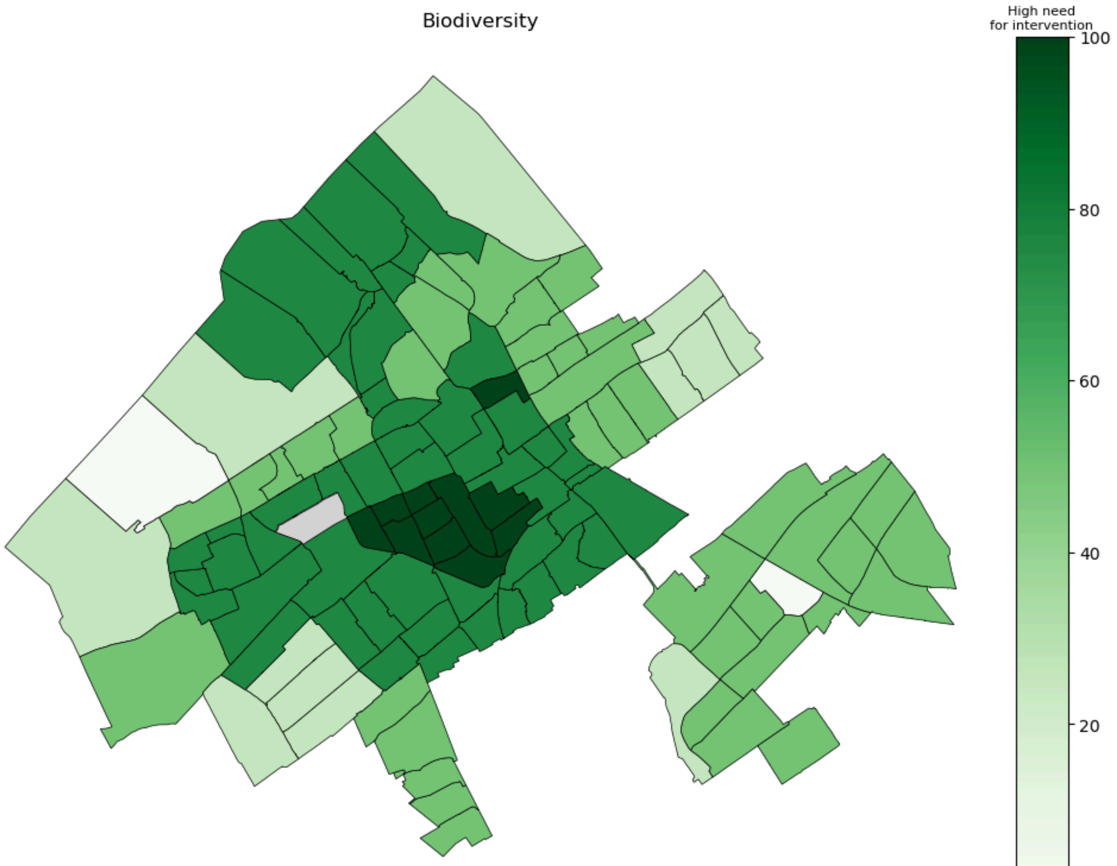
Normalize scale for optimization

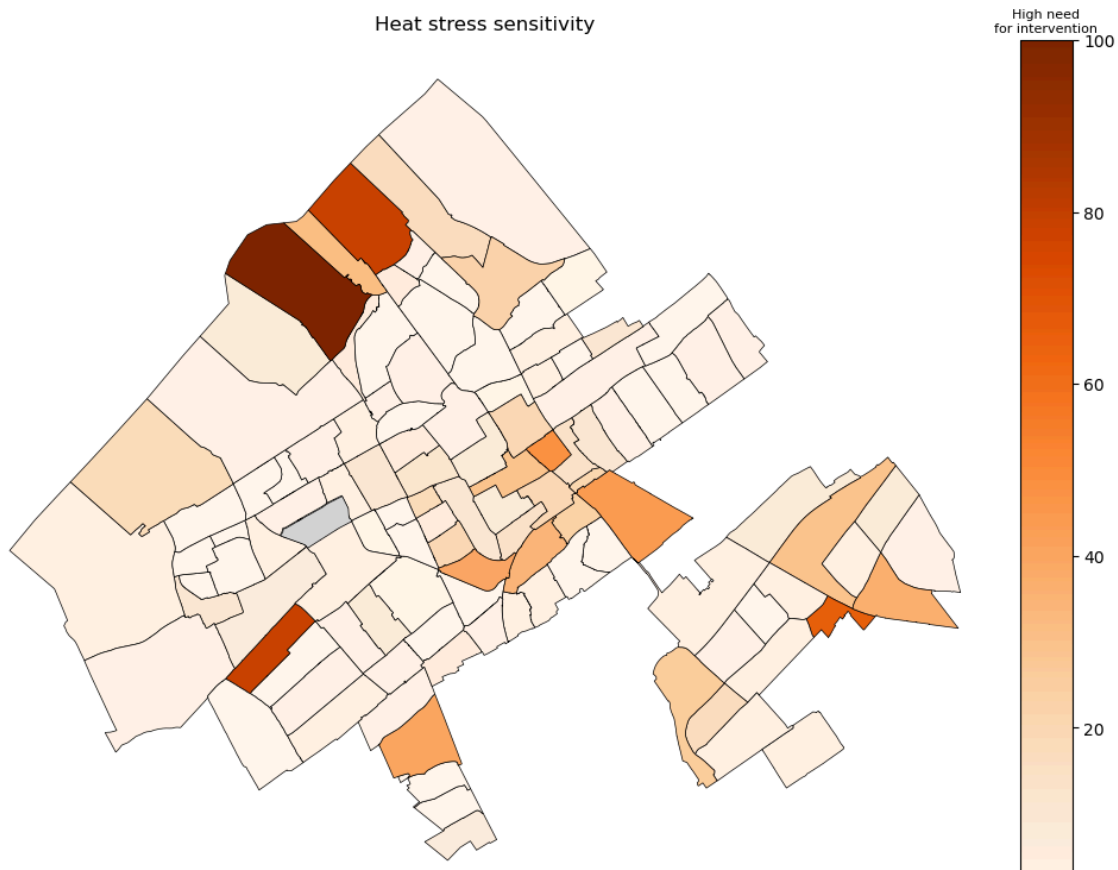
To enable optimization, the original data has been normalized to a 0–100 scale, where higher values (and more intense colors) indicate a greater need for intervention.

In [39]:
```python
for col, cmap, reverse, title in columns_to_plot:
    # Normalize to 0-100
    col_min, col_max = gdf_merged[col].min(), gdf_merged[col].max()
    norm_col = f"{col}_norm"
    gdf_merged[norm_col] = ((gdf_merged[col] - col_min) / (col_max - col_min)) *
    if reverse:
        gdf_merged[norm_col] = 100 - gdf_merged[norm_col]

    # Plot normalized data without dots
    fig, ax = plt.subplots(figsize=(10, 10))
    divider = make_axes_locatable(ax)
    cax = divider.append_axes("right", size="5%", pad=0.1)

    gdf_merged.plot(
        column=norm_col,
        cmap=cmap,
        linewidth=0.5,
        ax=ax,
        edgecolor="black",
        legend=True,
        cax=cax,
        missing_kwds={"color": "lightgrey", "label": "No data"}
    )

    cax.text(0.5, 1.01, 'High need\nfor intervention', ha='center', fontsize=8,
    cax.text(0.5, -0.01, 'Low need\nfor intervention', ha='center', fontsize=8,

    ax.set_title(title, fontsize=12)
    ax.axis("off")
    plt.tight_layout()
    plt.show()
```




Surface water overload
High need
for intervention
100
80
60
40
20
0
Low need
for intervention


Biodiversity
High need
for intervention
100
80
60
40
20
0
Low need
for intervention

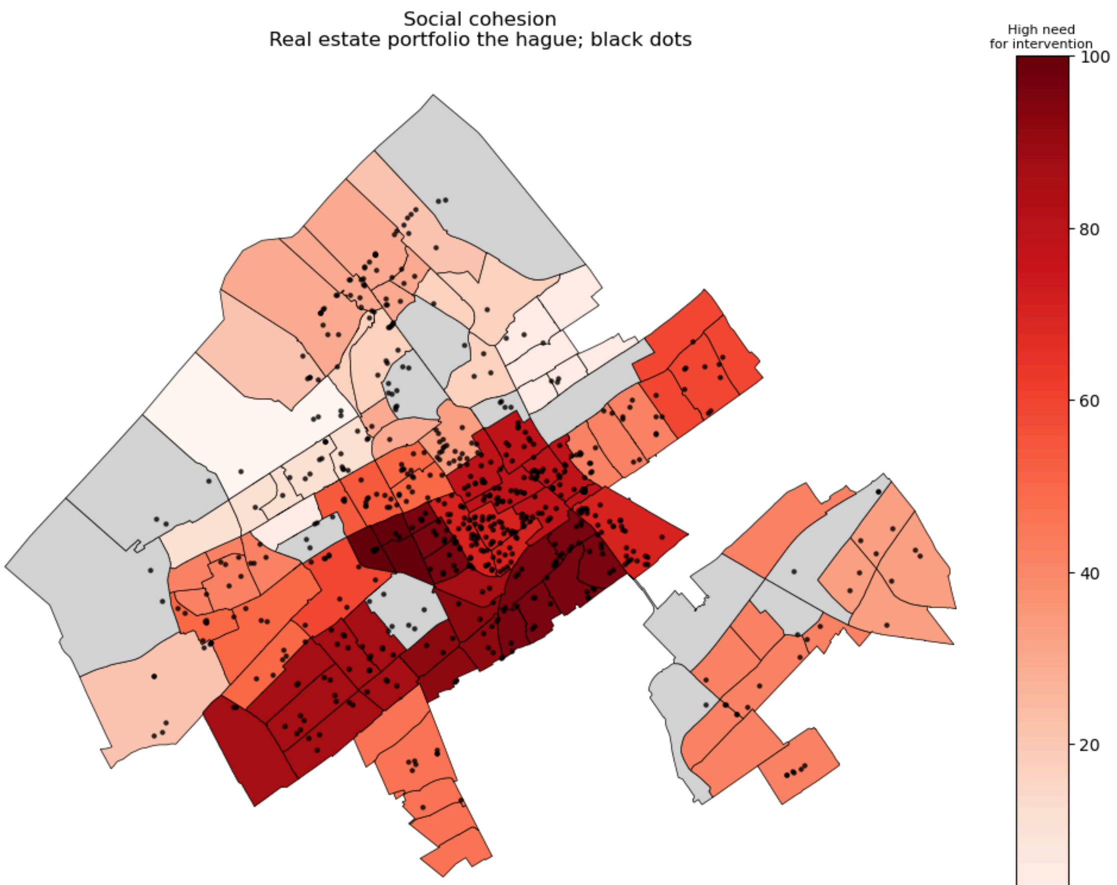


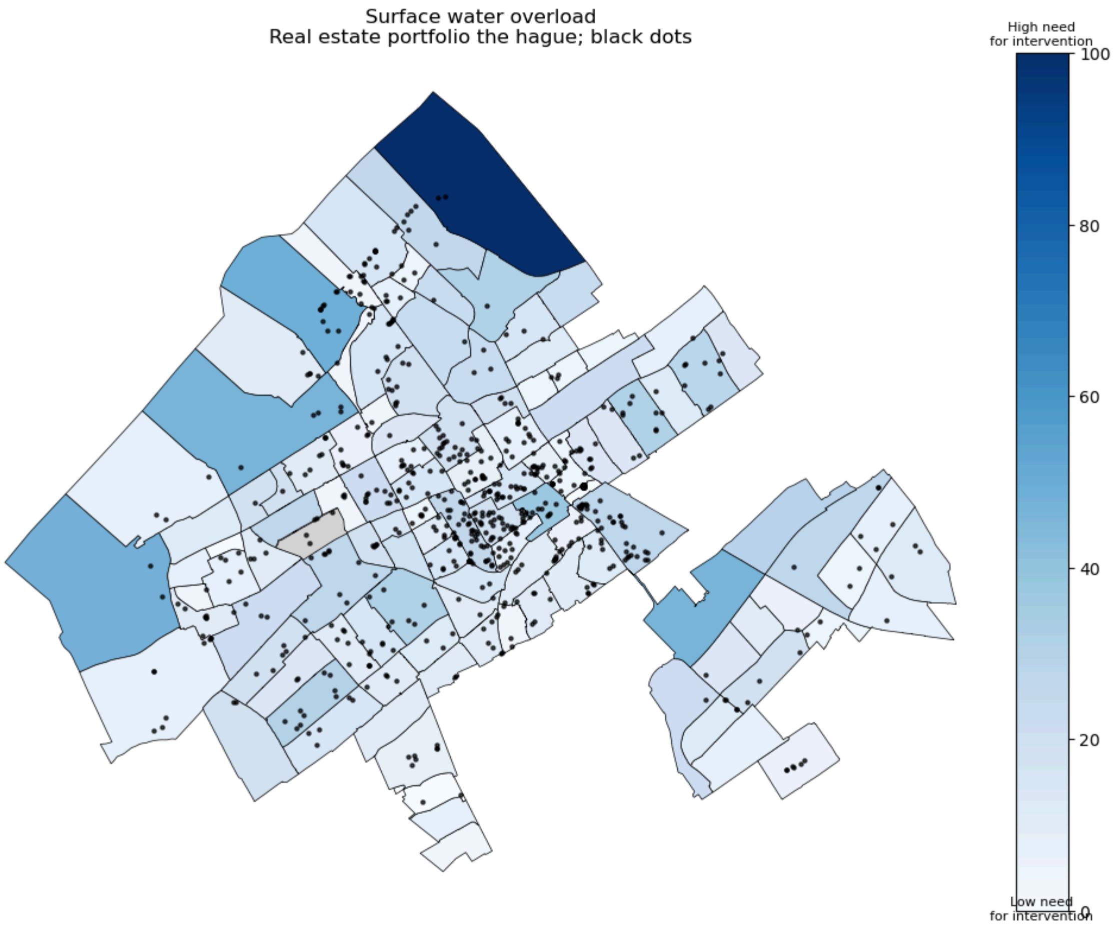
Neighborhood challenges and portfolio

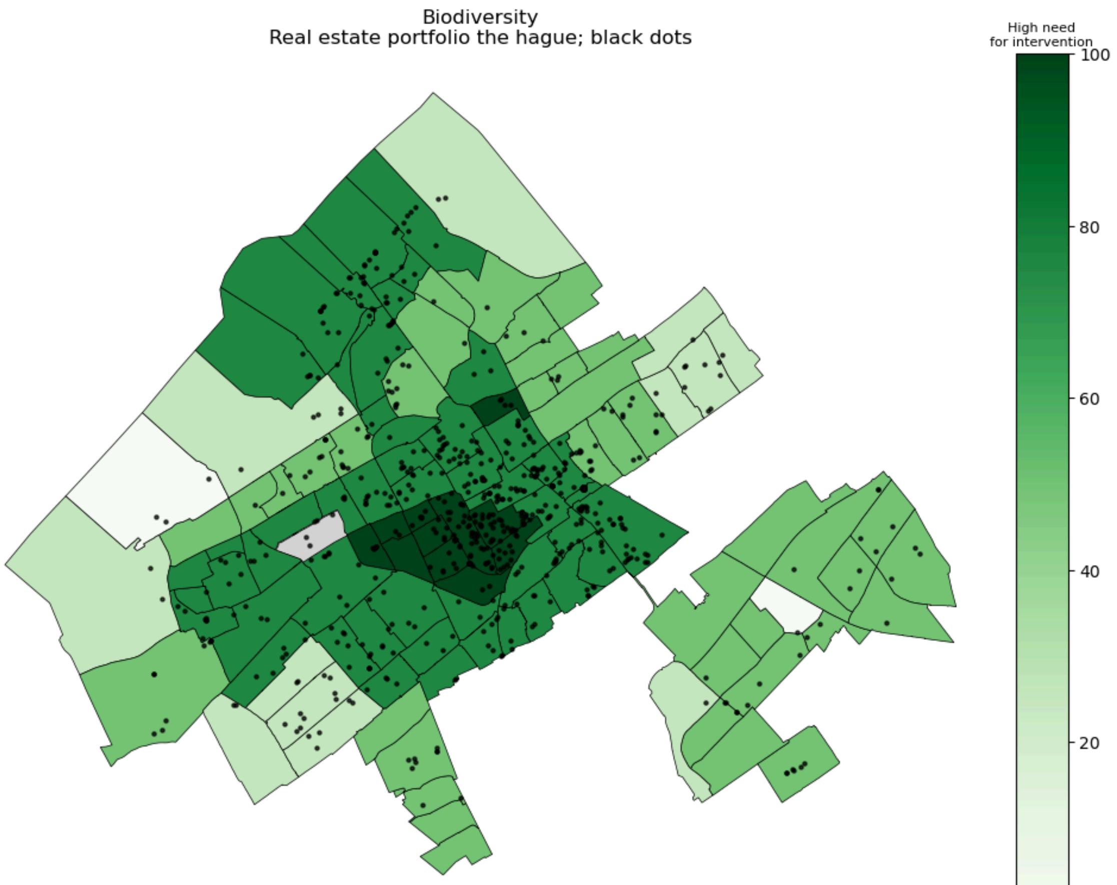
To see which buildings of the portfolio are located in areas with high needs, the portfolio is plotted as visual overlay.

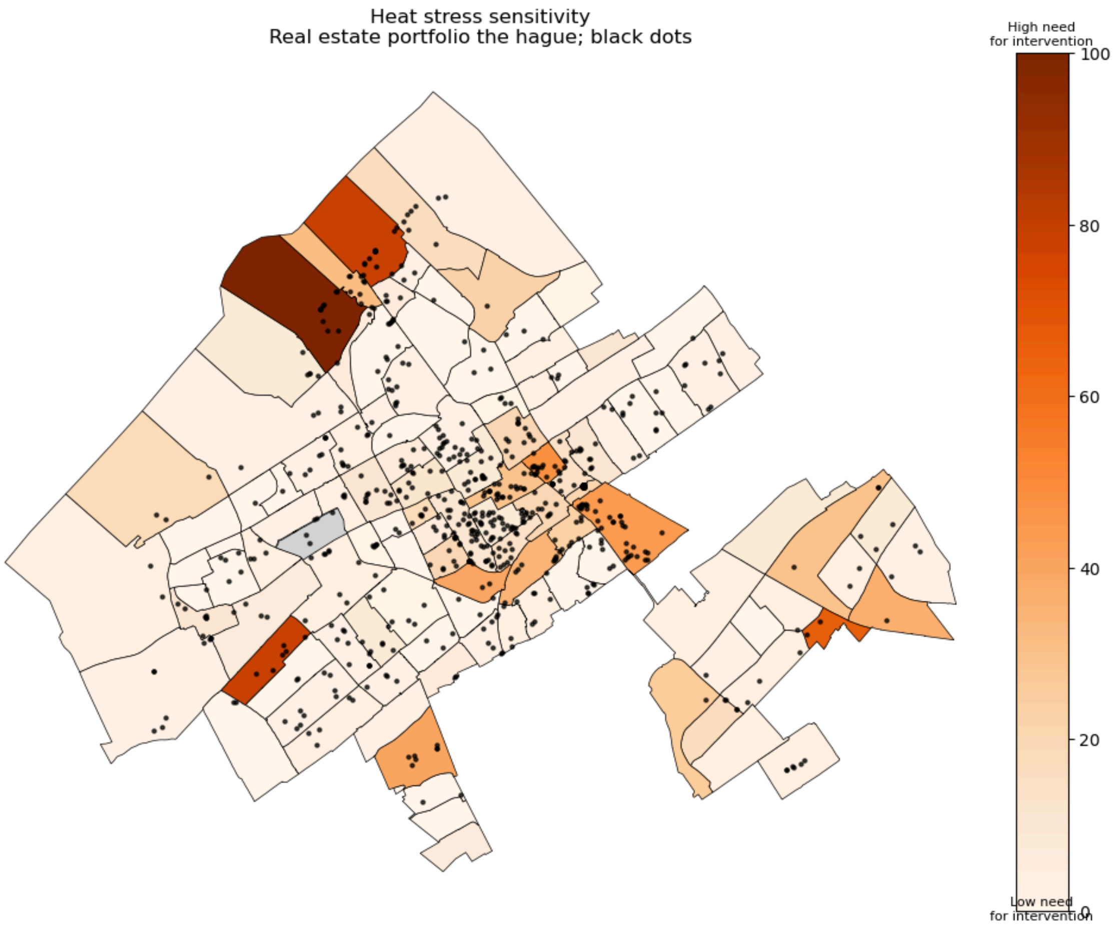
```
In [40]: for col, cmap, reverse, title in columns_to_plot:
             norm_col = f"{col}_norm"

             # Plot normalized data with real estate buildings as dots
             fig, ax = plt.subplots(figsize=(10, 10))
             divider = make_axes_locatable(ax)
             cax = divider.append_axes("right", size="5%", pad=0.1)

             gdf_merged.plot(
                 column=norm_col,
                 cmap=cmap,
                 linewidth=0.5,
                 ax=ax,
                 edgecolor="black",
                 legend=True,
                 cax=cax,
                 missing_kwds={"color": "lightgrey", "label": "No data"}
             )

             gdf_real_estate_with_buurt.plot(ax=ax, color="black", markersize=5, alpha=0.

             cax.text(0.5, 1.01, 'High need\nfor intervention', ha='center', fontsize=8,
             cax.text(0.5, -0.01, 'Low need\nfor intervention', ha='center', fontsize=8,

             ax.set_title(f"{title}\nReal estate portfolio the hague; black dots", fontsi
             ax.axis("off")
```

```
    plt.tight_layout()
    plt.show()
```






Biodiversity
Real estate portfolio the hague; black dots
High need
for intervention
100
80
60
40
20
0
Low need
for intervention


Heat stress sensitivity
Real estate portfolio the hague; black dots
High need
for intervention
100
80
60
40
20
0
Low need
for intervention

```
In [ ]:
```