Boeckmann, Jan; Thielen, Clemens

Article — Published Version

New Ways in Municipal Flood Mitigation: a Mixed-Integer Programming Approach and its Practical Application

Operations Research Forum

Suggested Citation: Boeckmann, Jan; Thielen, Clemens (2023) : New Ways in Municipal Flood Mitigation: a Mixed-Integer Programming Approach and its Practical Application, Operations Research Forum, ISSN 2662-2556, Springer International Publishing, Vol. 4, Iss. 4, https://doi.org/10.1007/s43069-023-00246-z

This Version is available at:
https://hdl.handle.net/10419/313672

Operations Research Forum (2023) 4:86
https://doi.org/10.1007/s43069-023-00246-z

RESEARCH

New Ways in Municipal Flood Mitigation: a Mixed-Integer Programming Approach and its Practical Application

Jan Boeckmann[1,2] · **Clemens Thielen**[1,2]

Received: 9 September 2022 / Accepted: 22 August 2023 / Published online: 28 October 2023

Abstract
Adapting to the consequences of climate change is one of the central challenges faced by humanity in the next decades. One of these consequences are intense heavy rain events, which can cause severe damage to buildings due to flooding. In this paper, we present the first use of optimization techniques that scales well enough to be applicable for supporting decision-making in planning precautionary measures for flash floods caused by heavy rain events in realistic scenarios. Our mixed-integer programming model has been implemented as an innovative decision support tool in the form of a web application, which has already been used by more than 30 engineering offices, municipalities, universities, and other institutions. The model aims to minimize the damage caused in the case of a heavy rain event by taking best-possible actions subject to a limited budget and constraints on the cooperation of residents. We further present an efficient, graph-based representation and preprocessing of the surface terrain, a combinatorial algorithm for computing an initial solution of the mixed-integer program, and computational results obtained on real-word data from different municipalities.

Keywords Mixed-integer programming · Flood mitigation · Graph algorithms

1 Introduction

Adapting to the consequences of climate change is without doubt one of the central challenges faced by humanity in the next decades. One of these consequences are intense heavy rain events, which scientists agree will increase both in their intensity and their frequency within the next years [1–3].

✉ Jan Boeckmann
jan.boeckmann@tum.de

Clemens Thielen
clemens.thielen@hswt.de

[1] TUM Campus Straubing for Biotechnology and Sustainability, Weihenstephan-Triesdorf University of Applied Sciences, Am Essigberg 3, 94315 Straubing, Germany

[2] Department of Mathematics, School of Computation, Information and Technology, Technical University of Munich, Boltzmannstr. 3, 85748 Garching bei München, Germany

The flash flood from 14 to 15 July 2021 in Germany, Luxembourg, and Belgium at the latest has caused a special public interest in adapting to such events. This event claimed more than 180 lives [4] and caused tremendous damage, which has been estimated at a total of 32 billion euros [5].

Typically, flood mitigation concepts are created based on simulations rather than using optimization methods [6]. Prioritization of actions is then often done by a simple point scheme [7]. This method, like any other purely simulation-based method, lacks the consideration of site-specific interplay of actions, which motivates using optimization methods in the context of flood mitigation.

Although there is clear evidence of the efficacy of precautionary measures [8], the literature on the use of optimization techniques in order to design flood mitigation concepts is surprisingly limited, which is also pointed out in [9, 10]. The closest related study to this work is [9], where the "Optimal Flood Mitigation Problem," which aims to optimize the positioning of a single type of precautionary measure (embankments) to protect critical assets in the case of a flood scenario, is introduced. Furthermore, two time-indexed mixed-integer programming formulations that over- and underestimate the water flows during a flooding scenario are presented. Here, the over- and underestimation is caused by the linearization of nonlinear constraints. Due to the time-indexed formulation, however, the MIPs do not tractably scale to realistic scenarios (as noted in the abstract of Tasseff [9]).

The problem of designing mitigation concepts for coastal floods in the Netherlands is an impressive example of the potential of using optimization techniques in flood mitigation. A mixed-integer programming formulation for a cost-efficient design of dike heights is presented in [11, 12]. Furthermore, a greedy search algorithm to compute a combination of reinforcement measures for dike segments, which is 42% cheaper than the combination obtained from the common approach, is implemented in [13].

Moreover, a genetic algorithm is used to compute efficient mitigation concepts for fluvial (river-caused) flash floods on the Thames Estuary (London, England) in a multi-objective setting [10]. Apart from the measures themselves, they also compute a threshold value for the timing to make an intervention given the uncertainty of the development of climate change and its impact on fluvial flash floods.

Another approach to the design of flood mitigation concepts can be found in [14], where a simulated annealing algorithm is used to determine an allocation of low-impact actions such as porous pavements and green roofs to districts in a megacity. Moreover, a particle swarm optimization algorithm is used in [15] to determine an optimal pumping schedule and optimal weir crest heights for detention reservoirs to minimize downstream flood damage.

An often neglected but crucial factor in creating successful flood mitigation concepts is taking the cooperation of residents into account since; in many cases, the most effective actions are located on private properties. Indeed, the potential of incentives in flood prevention has already been established as promising [16–18]. In practice, however, plans are often made before involving critical private actors. A holistic review on using market-based instruments for flood risk management is provided in [19].

Besides these approaches for flood mitigation, a wide variety of optimization techniques are used in post-disaster flood management. The design of evacuation plans including shelter location planning and helicopter assignment in a multi-objective robust setting is investigated in [20], and real-time operation procedures that specify reservoir releases during a flood are examined in [21, 22]. For a more extensive review of optimization and machine learning approaches in post-disaster flood management, we refer to [23].

Other applications of optimization techniques in water management involve the design of sewage water systems [24], a real-time release schedule for reservoirs during a flood [22], and the geometrical design of retention basins [25].

1.1 The Project AKUT

The work presented in this paper has been performed within the project AKUT—an acronym for the German translation of "Incentive Systems for Municipal Flood Prevention"—which has been funded by the German Federal Ministry for the Environment, Nature Conservation, and Nuclear Safety from January 2019 to March 2021. Within the project, a mixed-integer programming approach has been developed to find an optimal combination of actions to be taken such that the resulting damage on buildings is minimized while respecting a given budget and constraints on the cooperation of the residents. The resulting MIP has been implemented in a web application (also referred to as AKUT) using the Flask framework and Python 3.8. The application is available for municipalities free of charge (so far only in German language).

The project team included a municipality providing us with real-world data and the engineering office "igr GmbH" validating our results by comparing them to results of state-of-the-art simulations. Furthermore, the Professorship of Water Resource Management and Sanitary Environmental Engineering at Mainz University of Applied Sciences formulated and developed the engineering methodology for the model while the authors of this paper formulated the mathematical model and implemented the web application.

1.2 Our Contribution

In this paper, we present a novel mixed-integer programming approach for computing optimized flood mitigation concepts that minimize the damage to buildings due to flash floods caused by heavy rain events. To the best of our knowledge, this approach marks the first usage of optimization techniques in the context of planning precautionary measures for flood mitigation that scales well enough to be applied to real-world instances. Our model allows for different types of precautionary measures (basins, ditches, and embankments) that lead to elevations or depressions of the terrain surface. Moreover, the model takes constraints on the cooperation of the residents into account. One of the central challenges to make this approach work for realistic scenarios is modeling the surface terrain efficiently while still maintaining a realistic representation. We tackle this challenge via an efficient graph-based approach together with suitable preprocessing methods. Moreover, we present

Table 1 Comparison of our work to existing literature. A tick in the column "optimization" indicates that optimization algorithms are used and a tick in the column "pluvial" represents that the paper considers a pluvial flood scenario (as opposed to a fluvial or coastal flood scenario). A tick in the column "scalable" indicates that the developed method scales well enough to be applied to realistic scenarios. Finally, a tick in the column "incentivation" means that incentives or cooperation of residents is considered

Reference	Pre-/post-disaster	Optimization	Pluvial	Scalable	Incentivation
[9]	Pre	✓	✓		
[10]	Pre	✓		✓	
[11, 12]	Pre	✓		✓	
[21, 22]	Post	✓		✓	
[20]	Post	✓	✓	✓	
[18]	Pre		✓		✓
This paper	Pre	✓	✓	✓	✓

a combinatorial algorithm that is able to quickly compute an initial feasible solution of the presented MIP.

Our approach has been implemented as an innovative decision support tool in the form of a web application, which has already been used in practice by more than 30 engineering offices, municipalities, universities, and other institutions from all over Germany. We compare the results obtained from our model on real-world instances from different municipalities to results obtained from established simulation software, and investigate the main drivers for the running time and the quality of the obtained solutions. The novelty of our approach in comparison to a selection of the previously presented existing literature is summarized in Table 1.

2 Problem Description and Input Data

In this section, we define the underlying problem and describe the input data on which our approach is based.

In short, given a set of possible locations for retention basins (simply called basins in the following), ditches, and embankments, the goal in the problem is to determine a subset of these actions to take such that the resulting damage to buildings is minimized while respecting a given budget and constraints on the cooperation of residents.

The terrain surface is given as a digital terrain model (DTM), which is an established standard in engineering [6]. A DTM contains 2D coordinates in UTM format on a given grid together with their corresponding geodesic height, which is similar to the elevation above sea level. In our case, we use a grid size of 1 m.

Each of the data points in the DTM then determines the geodesic height of the one by one meter square centered at the 2D coordinate. This square is called the *shape* of a coordinate, and two 2D coordinates are called *adjacent* if their distance is 1 m, i.e., if one coordinate is 1 m to the north, south, west, or east of the other. These data are available for all German municipalities and, hence, suitable for applying our model in practice.

To estimate the damage that occurs due to flooding in the case of a rain event, information about the buildings' locations is required. To this end, the *shape* of a building is defined as the polygon derived from its outline. The outlines of the buildings are obtained from ALKIS,[1] which is a digital land information system. Just like the data for the DTM, these data are available to all German municipalities.

To link the positions of the buildings to the DTM coordinates, we say that a building is *on* a coordinate, if the shape of the building intersects with the shape of the coordinate. Conversely, we say that a coordinate *intersects* with a building in this case.

The definition of damage caused to buildings is based on the advisory leaflet DWA-M 119 [6] published by the German Association for Water, Wastewater and Waste (DWA) in 2016. The DWA is a politically and economically independent organization that supports safe and sustainable water management and prepares the DWA set of rules, which includes a large number of standards and advisory leaflets. Within the advisory leaflet DWA-M 119, they identify two main factors for the damage caused to a building, the first of which is the maximum water level at the building.

Here, the *maximum water level* at a building is the maximum over all water levels at the coordinates intersecting with the building. The *hazard class*, which represents the maximum water level at a building, is a categorical measure attaining the values zero to four. It is derived from the maximum water level at a building by the following rules[2]:

- **Zero:** The maximum water level at a building is 0 cm, i.e., none of the coordinates intersecting with the building has a strictly positive water level.
- **One:** The maximum water level at the building is strictly larger than 0 cm and less than or equal to 10 cm.
- **Two:** The maximum water level at the building is strictly larger than 10 cm and less than or equal to 30 cm.
- **Three:** The maximum water level at the building is strictly larger than 30 cm and less than or equal to 50 cm.
- **Four:** The maximum water level at the building is strictly larger than 50 cm.

The second main factor describes the (quite intuitive) fact that not every building suffers an equal amount of damage at a given water level. As an example, it is by far less severe if a garage is affected by the rain event compared to the case where a hospital is affected. To take this into account, the damage at a building does not only depend on the water level at the building (represented by its hazard class), but also on its *damage class*. The damage class is a categorical measure of the damage occurring at a building if water accumulates at it. It can attain the values one to four, where one corresponds to the lowest damage class (the garage in our example), i.e., the least amount of damage, and four corresponds to the highest damage class (the hospital in our example). The data from ALKIS, aside from just the shape of the building, provide additional information about the building like its usage, which allows to preset the damage class for some of the buildings automatically. For the remaining buildings, the damage class has to be specified manually.

[1] German acronym for "official real estate cadastre information system"

[2] See advisory leaflet DWA-M 119 [6].

The combination of the hazard class and the damage class yields the *need for protection* of a building, which is rated using a point system with a scale from zero to seven. For buildings with hazard class zero, i.e., none of the coordinates intersecting with the building have a strictly positive water level, the need for protection is also zero. Every other building has a strictly positive need for protection, which is increasing in both the building's damage class and its hazard class. The objective of the problem is to minimize the sum of all buildings' needs for protection.

In order to protect the buildings, a set of potential basins, ditches, and embankments is given, which together make up the possible *actions*. Each possible action is given by the polygon of its location, its construction costs, and its depth (in the case of a basin or ditch) or height (in the case of an embankment). The polygon of an action's location is called its *shape*.

Similar to the buildings, we say that an action is *on* a coordinate if its shape intersects with the shape of the coordinate. In this case, we also say that the coordinate *intersects* with the action. If an action is taken, i.e., a basin, ditch, or embankment is built, the geodesic height of all coordinates intersecting with the action is decreased by the action's depth or increased by its height. The change in geodesic height affects the flow of the water on the terrain surface and, hence, can protect buildings. The overall cost for taking actions is bounded from above by a given budget.

Taking an action requires the consent of the owners of the properties on which the action is located. The owners of the properties are called *actors* in the following. The outlines of the properties are also obtained from ALKIS. As before, the *shape* of a property is defined as the polygon derived from its outline. An action is *on* a property if their shapes intersect. Convincing actors to cooperate might be more or less hard. To guarantee that the recommended combination of actions can realistically be implemented, the number of hard-to-convince actors on whose properties actions are to be taken is bounded from above. To this end, an extended traffic light rating system with the following characterizations is used:

- **Green:** The actor is willing to cooperate.
- **Yellow:** The actor needs minor incentives to cooperate.
- **Red:** The actor needs major incentives to cooperate.
- **Black:** The actor does not cooperate at all.

For simplicity, we also refer to *green, yellow, red*, and *black properties* in the following. The willingness to cooperate has to be assigned manually by the user for each property.

3 Mathematical Modeling

We now present a graph-based model for the problem described in Section 2 as well as an approach for reducing the size of the underlying graph (Section 3.1). Afterwards, in Section 3.2, we derive our mixed-integer programming formulation that is used for solving the graph-based model, and we describe valid inequalities and presolve techniques that are used to improve performance.

3.1 Graph-Based Model

3.1.1 Construction of the Graph

In this section, we construct the directed graph $G_{\text{or}} = (V_{\text{or}}, R_{\text{or}})$, which we call the *original graph*, from the DTM.

Recall that, for a directed graph $G = (V, R)$, a node $v \in V$ is called a child of $u \in V$ if there is an arc from u to v. Conversely, u is then called a parent of v. Moreover, a node $v \in V$ is called a successor of $u \in V$ if there exists a (directed) path from u to v. Conversely, u is then called a predecessor of v. The set of outgoing arcs of $v \in V$ is denoted by $\delta^+(v)$ and the set of incoming arcs by $\delta^-(v)$. A node with no outgoing arcs is called a *leaf*, and a node without incoming arcs is called a *root*. For an arc $r \in R$, we denote its start node by $\alpha(r)$ and its end node by $\omega(r)$.

For the construction of $G_{\text{or}} = (V_{\text{or}}, R_{\text{or}})$, recall that the DTM contains data about the geodesic height for coordinates on a 1-m grid. The set V_{or} of nodes is constructed by associating one node with each of these coordinates, i.e., there is a one to one correspondence between the coordinates in the DTM and the nodes in the graph. To keep track of this correspondence, each node gets the coordinate as an additional attribute.

The *geodesic height* of a node is defined as the geodesic height of its corresponding coordinate. We store the geodesic height as an attribute for each node in the graph. We then index the nodes in $V_{\text{or}} = \{v_1, \ldots, v_n\}$ in non-decreasing order of geodesic height, where ties are broken arbitrarily. Furthermore, we define the *shape* of a node $v \in V_{\text{or}}$ as the shape of its corresponding coordinate, i.e., in this case, the one by one meter square with its center at the corresponding coordinate. The definitions of whether a building, action, or property intersects with (the shape of) a node are analogous to the ones for coordinates provided in the previous section. Finally, each node $v \in V_{\text{or}}$ is assigned an *area*, which we denote by area_v. In the case of the original graph, the area is 1 m² for each node. This changes though for the graphs we construct in Section 3.1.3.

For any two nodes whose corresponding coordinates are adjacent on the grid, there is an arc in R_{or} between the nodes, which is oriented from the node with the higher index to the node with the lower index. This means that arcs are directed from the node with larger geodesic height (the *higher* node) to the node with lower geodesic height (the *lower* node) whenever the two nodes do not have the same geodesic height. Note that, if all nodes in V_{or} have pairwise distinct geodesic heights, this makes the original graph $G_{\text{or}} = (V_{\text{or}}, R_{\text{or}})$ acyclic since the geodesic heights induce a topological sorting (both in the mathematical and literal sense) in this case. An example of the original graph is provided in Fig. 1.

To model the runoff behavior of the precipitation water, we compute flows in the graph, which are determined by the nodes' geodesic heights. To this end, for an arc $r \in R_{\text{or}}$, we define its *slope* as the absolute difference of its incident nodes' geodesic heights and denote it by slope_r. When distributing the outflow of a node $v \in V_{\text{or}}$ among its downhill arcs, we want to ensure that a higher slope causes more water flow on an arc. This is modeled by the *ratios* of the arcs, which are introduced next and are based on the concept of *processing networks* [26, 27], in which flow is distributed among the outgoing arcs of a node according to fixed ratios.

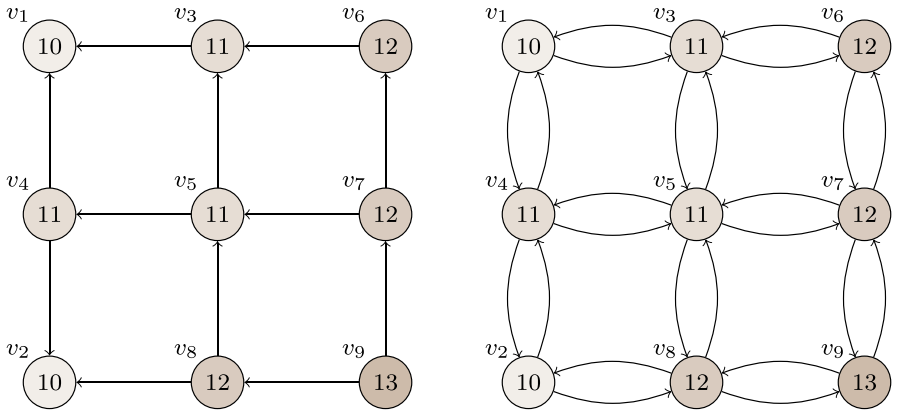


Fig. 1 An example of the original graph $G_{\text{or}} = (V_{\text{or}}, R_{\text{or}})$ on the left and an example of the extended original graph $G_{\text{or}}^{\text{ex}} = (V_{\text{or}}, R_{\text{or}}^{\text{ex}})$ on the right. The number in each node corresponds to its geodesic height, also indicated by the node's color. The nodes are indexed in non-decreasing order of geodesic height, where ties are broken arbitrarily as, e.g., for v_4 and v_5. The arcs in the original graph are directed such that they start at the node with the higher index

To compute the ratio of an arc $r \in R_{\text{or}}$, which we denote by ratio_r, we have to distinguish two cases. If the sum of the slopes of all outgoing arcs of the node $\alpha(r)$ is nonzero, we define the ratio of r by $\text{ratio}_r := {}^{\text{slope}_r}/_{\sum_{\hat{r} \in \delta^+(\alpha(r))} \text{slope}_{\hat{r}}}$. If the sum is zero, the ratio of r is defined as one divided by the number of successors, i.e., as $\text{ratio}_r := {}^{1}/_{|\delta^+(\alpha(r))|}$.

In some situations, however, actions that are taken lead to water flowing in the opposite direction of an arc $r \in R_{\text{or}}$, which means that the original graph does not suffice for our model. A simple example for such a situation is illustrated in Fig. 2. To this end, for an arc $r \in R_{\text{or}}$, we denote the inverse arc by $\overleftarrow{r}$. The *extended original graph* $G_{\text{or}}^{\text{ex}} = (V_{\text{or}}, R_{\text{or}}^{\text{ex}})$ is then constructed by adding the inverse arc $\overleftarrow{r}$ for each $r \in R_{\text{or}}$ to the original graph and setting the ratio of the arc $\overleftarrow{r}$ to the ratio of r. Note that this does not change the node set V_{or}.

3.1.2 Description of the Graph-Based Model

The goal in our problem is to provide best-possible protection for the buildings by taking a combination of actions respecting a given budget and the cooperation of the actors. In this section, we formulate the corresponding optimization problem formally by using the extended original graph introduced in the previous section.

The input of our graph-based model consists of the following:

- The **original graph** $G_{\text{or}} = (V_{\text{or}}, R_{\text{or}})$ and the **extended original graph** $G_{\text{or}}^{\text{ex}} = (V_{\text{or}}, R_{\text{or}}^{\text{ex}})$
- The set B of **buildings**, each of which is given by its shape and its damage class
- The set $\mathcal{A}$ of possible **actions**, each of which is given by its shape, its construction costs, and its depth/height

Fig. 2 The instance consists of two nodes u and v, where v is the higher of the two nodes. This means that water flows from v to u, which is illustrated on the left-hand side. If a basin with depth strictly larger than the absolute difference of the nodes' geodesic heights can be built on v, the resulting geodesic height of v after building the basin is less than the geodesic height of u. Therefore, the water flows in the opposite direction after building the basin, which is illustrated on the right-hand side

- The set P of **properties**, each of which is given by its shape and the willingness to cooperate of the corresponding actor
- A **budget** denoted by budget, which represents an upper bound on the total cost for taking actions
- The **maximum combined number of yellow and red properties** denoted by maxAllowedYellow + maxAllowedRed on which actions can be taken
- The **maximum number of red properties** maxAllowedRed on which actions can be taken
- The **rain per square meter** denoted by rain

A feasible solution is a set of actions whose total cost does not exceed the given budget and where neither the combined number of yellow and red properties on which actions are taken nor the number of red properties on which actions are taken exceeds the allowed maximum. The objective is to minimize the sum of all buildings' needs for protection, which is computed from a given feasible solution as described in the following.

The decision on which actions are to be taken changes the geodesic heights of the nodes intersecting with these actions, which may in turn change the flows in the graph. The change of the geodesic height is straightforward if there is at most one action taken on a node. However, if there are several actions with different depths/heights taken on one node, like, for example, a ditch leading into a deeper basin, we need a more sophisticated rule, which is given as follows:

(GH1) If at least one action decreasing the geodesic height (i.e., a basin or a ditch) is built on a node $v \in V_{\text{or}}$, then the geodesic height of v is set to the node's original geodesic height minus the maximum depth of any of the basins or ditches built on v.
(GH2) If no actions decreasing the geodesic height are built on a node $v \in V_{\text{or}}$ (i.e., neither basins nor ditches are built on v), then the geodesic height of v is

set to the node's original geodesic height plus the maximum height of any of the embankments built on v.

Once this decision has been made, the resulting geodesic heights (after taking the actions) determine the flows between the nodes, which then allows us to compute the water levels. Before we describe how the flows are computed, we describe the connection between the flows and the water levels. To this end, we define the *excess* of a node $v \in V_{\text{or}}$ as the amount of water accumulating at v, i.e., as the initial water from the rainfall plus the node's inflow minus its outflow. The *water level* at v is defined as the excess of v divided by the node's area.

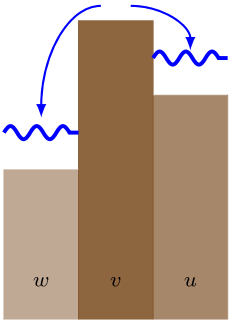
We next describe how the water levels are computed. An efficient implementation of a combinatorial algorithm for this computation is provided as Algorithm 6 in Appendix 2, which uses Algorithms 4 and 5 as subroutines for computing the flows in the graph and joining nodes, respectively.

We define $G^D = (V^D, R^D)$ for a subset $D \subseteq \mathcal{A}$ of actions as the graph that is obtained from G when adjusting the geodesic heights as described in (GH1) and (GH2) and changing arc directions where necessary. This graph then represents the input to Algorithm 6 if when computing the water levels in the scenario where exactly the actions in D are taken. Throughout the computation, we keep track of both the water levels and the excesses of all nodes in V. At the start of the algorithm, each node receives its initial water from the rainfall, which is computed by multiplying the given rain per square meter with the node's area. This water may then flow over the node's outgoing arcs.

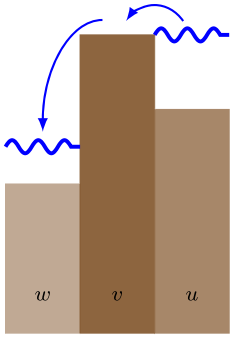
The outflow of a node $v \in V$ (i.e., the water flowing from the node to its adjacent nodes) is distributed proportionally to the ratios of the outgoing arcs if v is not a leaf. The leaves then start flooding until the water level at some leaf $u \in V$ equals the absolute difference of the geodesic height of u and its lowest parent node $v \in V$, in which case we say that the water level at u *matches* the geodesic height of v. The nodes are then joined and from there on represented by u. Joining nodes within Algorithm 6 is done via the subroutine presented in Algorithm 5. This process is then repeated until there is no node that is not a leaf in the graph and has strictly positive excess. The behavior of the flows during the algorithm is illustrated in Fig. 3.

After the flows are computed, the water levels at the nodes follow immediately by dividing the excess of each node by the nodes' area. This allows us to determine the maximum water level at each of the buildings and, hence, its resulting hazard class, which is obtained as described in Section 2. The combination of the buildings' hazard and damage classes yield the corresponding needs for protections whose sum is to be minimized.

Note that implementing the described procedure for computing the water levels efficiently is important for obtaining feasible running times of our overall approach on real-world instances. In fact, this procedure is used both when reducing the graph size as described in the following subsection and for obtaining a feasible initial solution of our MIP as discussed in Section 4.2. The implementation as an efficient combinatorial algorithm presented in Appendix 2 uses an intelligent update of the



(a) The water at node v is distributed to both its children u and w. Neither of the childrens' water levels matches the geodesic height of v, so the water at v flows from v to the nodes u and w.



(b) The water level at node u matches the geodesic height of v, which means that the nodes u and v are joined and afterwards represented by v. As the water level at w does not yet match the geodesic height of v, all water at v now flows to w.

Fig. 3 An illustration of the flows in Algorithm 6

flows, which decreases the running time by about 90% on average compared to computing the flows from scratch in each iteration. Moreover, using a Fibonacci heap to store leaf nodes and the corresponding times until the water level at the leaf matches the geodesic height of its lowest parent further reduces the running time by about 25% on average compared to a simple sorted-array implementation.

3.1.3 Reducing the Graph Size

The size of the original graph $G_{\text{or}} = (V_{\text{or}}, R_{\text{or}})$ or the extended graph $G_{\text{or}}^{\text{ex}} = (V_{\text{or}}, R_{\text{or}}^{\text{ex}})$ is the main determinant for the size of a problem instance. In this section, we describe how the graph size can be reduced while still maintaining a realistic model of the problem described in Section 2.

It is worth noting that the sizes of both the original graph and the extended graph are linear in the cardinality of the node set V_{or}, which we therefore use as a natural measure of the size of these graphs. The aim of this section is to derive the *reduced graph* $G_{\text{red}} = (V_{\text{red}}, R_{\text{red}})$ from the original graph. In fact, applying our MIP presented in the next section based on the original graph only works for unrealistically small instances. Hence, reducing the size of the graph is actually crucial in order to obtain a model that is applicable in practice.

As a quick outline of this section, we provide a short summary of the ideas of our graph size reduction techniques:

1. Instead of a fixed grid size of 1 m, we use a **dynamic grid size**, which means that certain parts of the terrain surface are modeled using coarser 25 m or 5 m grids.
2. We **remove nodes** that do not cause flow into critical locations.

3. We **contract all nodes in non-critical locations** that dispense water to critical locations into one source node.
4. We **contract adjacent nodes of similar geodesic heights**.

Before we can apply these ideas, we have to introduce some further definitions. To model the terrain surface using a grid size of 25 m, we construct the graph $G_{25} = (V_{25}, R_{25})$ from those coordinates in the DTM where both UTM coordinates are integer multiples of 25 m. This works completely analogously to the construction of the original graph. The only difference is that the shape of a node in V_{25} is no longer a square with an edge length of 1 m, but now a square with an edge length of 25 ms. Consequently, each node's area in G_{25} amounts to 625 m². Also note that, for example, a building is *on* a node $v \in V_{25}$ if its shape intersects with the shape of v, which in G_{25} is a square with an edge length of 25 ms.

In the same fashion, we construct the graph $G_5 = (V_5, R_5)$ from those coordinates in the DTM where both UTM coordinates are multiples of 5. It is important that, although the nodes in V_{25}, V_5, and V_{or} stem from the same coordinates, the sets are disjoint as the attributes of the nodes (e.g., their areas) differ.

To obtain more information about the graphs G_{25}, G_5, and G_{or}, we first assess for each node whether there are a buildings or possible actions on it. This means that, for each node $v \in \hat{V}$, where $\hat{V}$ is one of the sets V_{25}, V_5, or V_{or}, we store a set of buildings on the node, which we denote by B_v, and a set of possible actions, which we denote by $\mathcal{A}_v$. An efficient algorithm for obtaining these sets is provided in Appendix 1.

1. Using a dynamic grid size: We construct a graph with a dynamic grid size, which we denote by $G_{\text{dg}} = (V_{\text{dg}}, R_{\text{dg}})$. To this end, we first construct the node set V_{dg} and then the arc set R_{dg}. To construct the node set, we initialize V_{dg} as a copy of V_{25}, then resolve each node that intersects with a building or an action at a 5 m grid size, and finally resolve each node that has been resolved at a 5 m grid size and that intersects with a ditch or an embankment at a grid size of 1 m. This procedure is described in Algorithm 1. To keep track of the resolution at the single nodes, we further store the resolution res_v for each node $v \in V_{\text{dg}}$.

Algorithm 1 Construct-nodes

Procedure `constructNodes`$(V_{25}, V_5, V_{\text{or}})$
  Initialize $V_{\text{dg}} = V_{25}$, queue $= \emptyset$, and $\text{res}_v = 25$ for all $v \in V_{\text{dg}}$
  for $v \in V_{25}$ **do**
    if *v intersects with a building or an action* **then**
      Replace v in V_{dg} by the nodes in $V_5^v \subseteq V_5$, where V_5^v is the set of nodes in V_{25} whose shapes intersect with the shape of v (i.e., resolve this the node v at a 5 m grid size). For each $v' \in V_5^v$, set $\text{res}_{v'} := 5$.
      Insert all newly added nodes into the queue.
    end
  end
  for $v \in$ queue **do**
    if *There is a ditch or embankment on v* **then**
      Replace v in V_{dg} by the nodes in $V_{\text{or}}^v \subseteq V_{\text{or}}$, where V_{or}^v is the set of nodes in V_{or} whose shapes intersect with the shape of v (i.e., resolve the node v at a 1 m grid size). For each $v' \in V_{\text{or}}^v$, set $\text{res}_{v'} := 1$.
    end
  end
  return V_{dg}

The set of arcs R_{dg} is then constructed by adding an arc between two nodes in V_{dg} if and only if they are adjacent on the dynamic grid (i.e., their shapes have a common edge). The arc is again directed from the node with the higher index to the node with the lower index according to the ordering of the corresponding nodes with the same coordinates in V_{or} (i.e., from the higher node to the lower node whenever the two nodes do not have the same geodesic height).

To compute the ratios, we also have to take the resolutions of the nodes into account. This stems from the fact that, with the dynamic grid size, the length of the common edge of two adjacent nodes' shapes can be 1 m, 5 m, or 25 m. The ratio of an arc $r \in R_{\text{dg}}$, hence, depends on the slopes of the outgoing arcs of $\alpha(r)$ and on the proportion of the boundary of the shape of $\alpha(r)$ that the shapes of $\alpha(r)$ and $\omega(r)$ have in common. The ratio of r is computed as

$$\text{ratio}_r := \left(\text{slope}_r / \sum_{\hat{r} \in \delta^+(\alpha(r))} \text{slope}_{\hat{r}}\right) \cdot \text{correction}_r,$$

where

$$\text{correction}_r := \begin{cases} 1 & \text{if } \text{res}_{\alpha(r)} \leq \text{res}_{\omega(r)} \\ \text{res}_{\omega(r)} / \text{res}_{\alpha(r)} & \text{else} \end{cases}.$$

An example of shapes of nodes and the corresponding graph $G_{\text{dg}} = (V_{\text{dg}}, R_{\text{dg}})$ is provided in Fig. 4.

Modeling all buildings at a grid size of 5 m is still overly exact. Buildings at which no (or only negligible) water levels are to be expected can still be modeled at a grid size of 25 m. To assess a good grid size, we compute the water levels on the graphs $G_{\text{dg}} = (V_{\text{dg}}, R_{\text{dg}})$ and $G_{25} = (V_{25}, R_{25})$ using Algorithm 6, which is presented Appendix 2.

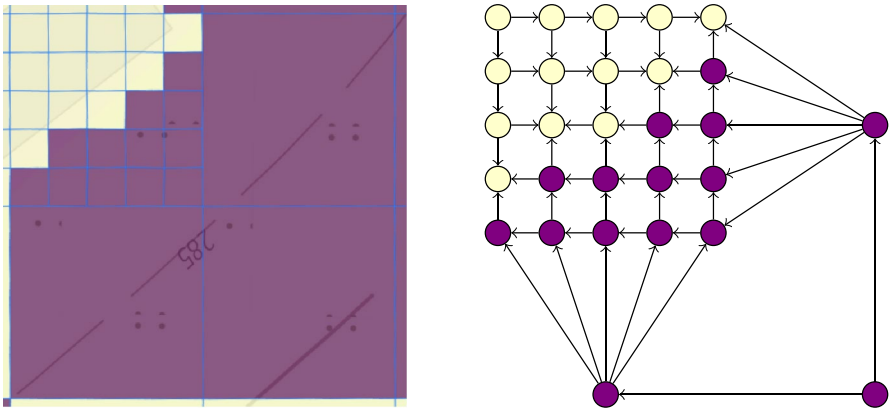


Fig. 4 A screenshot from our web application on the left-hand side, where the dynamic grid size is visualized and the nodes intersecting with a building are colored yellow. The corresponding part of the graph $G_{\text{dg}} = (V_{\text{dg}}, R_{\text{dg}})$ is visualized on the right-hand side

We call a node $v \in V_{25}$ *threatened*, if it has a strictly positive water level in the computation on $G_{25} = (V_{25}, R_{25})$ or if any node in V_{dg} whose shape intersects with the shape of v has a water level greater than or equal to 1 cm in the computation on $G_{\text{dg}} = (V_{\text{dg}}, R_{\text{dg}})$.

For each non-threatened node $v \in V_{25}$ that only intersects with buildings and not with actions, we rescale its resolution in $G_{\text{dg}} = (V_{\text{dg}}, R_{\text{dg}})$ back to 25 m, i.e., we contract all nodes in V_{dg} whose shapes intersect with the shape of v into v. Afterwards, we recompute the arc set R_{dg} and the ratios with the updated node set V_{dg} as we have done before, which yields the final version of $G_{\text{dg}} = (V_{\text{dg}}, R_{\text{dg}})$.

The reduction in the overall number of nodes achieved by this step highly depends on the number of nodes in V_{or} that do not intersect with any buildings or actions, as the number of these nodes is reduced by the highest factor of 625. In the instances presented in Section 4, the overall number of nodes is usually reduced by a factor of about 500.

2. Removing nodes not causing flow into critical locations: Our next goal is to remove nodes from the graph that do not cause any flow into critical locations. To this end, we define four new properties for nodes. A node $v \in V_{\text{dg}}$ is called ...

- *critical* if its shape intersects with a building or a potential action.
- *relevant* if it is critical, its resolution is not 25 m, or it is a successor of a critical node in $G_{\text{dg}} = (V_{\text{dg}}, R_{\text{dg}})$. Apart from critical nodes, relevant nodes are either nodes where water may accumulate and then cause critical nodes to be flooded due to back pressure, or nodes that are needed to complete the grid without gaps.
- *water-dispensing* if it is not relevant, but it is a predecessor of a relevant node in $G_{\text{dg}} = (V_{\text{dg}}, R_{\text{dg}})$. Water accumulating on such nodes does not cause flooding of relevant nodes due to back pressure. These nodes are, however, still interesting as they dispense water to relevant nodes.
- *irrelevant* if it is neither of the above. Irrelevant nodes do not contribute in any way to the flooding of relevant nodes.

As an example, think of a village at the foot of a mountain. Here, the nodes at coordinates within the village are the relevant nodes, the nodes at coordinates on the side of the mountain facing the village are the water-dispensing nodes, and the nodes at coordinates on sides of the mountain not facing the municipality are the irrelevant nodes.

The first step of the node removal consists of removing all irrelevant nodes from V_{dg}. It is worth noting that this may cause the graph to be no longer weakly connected. In practice though, this only happens if buildings are spread widely apart from each other, which is seldom the case. Apart from this, our model still works if the graph is not weakly connected. We denote the graph obtained by this method by $G_{\text{ri}} = (V_{\text{ri}}, R_{\text{ri}})$.[3]

The reduction in the overall number of nodes achieved by this step highly depends on the number of irrelevant nodes, which in turn depends on the choice of

[3] **ri**: **r**emove **i**rrelevant.

the input DTM. Barely any nodes are irrelevant in cases where the region covered by the DTM has been chosen relatively tight around the build-up region to be protected, whereas a lot of nodes are irrelevant if the region covered by the DTM has been chosen relatively large. However, since the region covered by the DTM is composed of 1 by 1 km rectangles and must always be chosen large enough so that no potentially relevant or water-dispensing nodes are omitted, a certain number of irrelevant nodes is usually unavoidable, so the removal of irrelevant nodes represents an important first step in reducing the overall number of nodes.

3. Contracting nodes in non-critical locations: In the next step, we deal with the water-dispensing nodes. By construction, flow through these nodes is not affected by the decision on which actions are taken. Next, we contract all water-dispensing nodes into a single node s, which we call the *source node*. Note that this contraction also changes the arc set. The arcs that are incident to s arise from arcs in G_{ri} that are directed from a water-dispensing node to a relevant node. In particular, this means that the in-degree of s is zero.

The area of the source node is set to the sum of the areas of all water-dispensing nodes. To compute the ratios of the arcs that are incident to s, we first compute the flows in the graph G_{ri} using Algorithm 4 and denote the resulting flow on $r \in R_{\text{ri}}$ by f_r. The ratio of an arc $r \in R_{\text{wd}}$ starting in s is then set to the sum of the inflow into $\omega(r)$ from water-dispensing nodes divided by the total inflow from water-dispensing into relevant nodes in G_{ri}:

$$\text{ratio}_r := \sum_{\substack{\hat{r} \in R\text{wd}: \\ \alpha(\hat{r}) \text{ is water-dispensing} \\ \text{and } \omega(\hat{r}) = \omega(r)}} f_{\hat{r}} \Big/ \sum_{\substack{\tilde{r} \in R\text{wd}: \\ \alpha(\tilde{r}) \text{ is water-dispensing} \\ \text{and } \omega(\tilde{r}) \text{ is relevant}}} f_{\tilde{r}}$$

For completeness, we set the geodesic height of s to the largest geodesic height in the graph before contraction plus 1 m. This ensures that the source node is never flooded unless an unrealistically large amount of rain per m^2 is used. We denote the obtained graph by $G_{\text{wd}} = (V_{\text{wd}}, R_{\text{wd}})$.[4]

4. Contracting adjacent nodes of similar geodesic heights: As a last step, we contract adjacent nodes into a new node if they have the same geodesic height up to a given threshold and the same combination of actions and buildings on them, which yields the desired *reduced graph* $G_{\text{red}} = (V_{\text{red}}, R_{\text{red}})$. The exact procedure for computing G_{red} is presented in Algorithm 2, which will be explained in the following paragraphs. The corresponding reduction step has two benefits. First, it further reduces the number of nodes. Second, and far more beneficially, it greatly improved the numerical stability of the MIP. Indeed, numerical issues caused the MIP to be infeasible before we introduced this procedure. The improved numerical stability stems from the fact that, after the procedure, all nodes in the resulting reduced graph G_{red} have pairwise distinct geodesic heights, and there are only very few adjacent nodes that have similar geodesic heights.

[4] **wd**: **w**ater **d**ispensing.

Algorithm 2 is divided into four parts. In the first part, we contract nodes that intersect with the same sets of buildings and actions and have a similar geodesic height into a new node representing the contracted nodes. The shape of such a new node is defined as the union of the shapes of the contracted nodes, and the boundary of such a node is the boundary of its shape. The geodesic height of the new node is then set to the area-weighted average over the geodesic heights of the nodes that have been contracted into the new node $v \in V_{\text{red}}$:

$$\text{gh}_v := \sum_{\substack{v' \in V_{\text{wd}}: \\ v' \text{ is contracted into } v}} \text{gh}_{v'} \cdot \text{area}_{v'} \Big/ \sum_{\substack{v' \in V_{\text{wd}}: \\ v' \text{ is contracted into } v}} \text{area}_{v'}$$

In practice, this procedure usually leads to all nodes in V_{red} having pairwise distinct geodesic heights. However, if this is not the case, we add a slight noise to the geodesic heights of each pair of nodes that have the same geodesic height. This is important in order to guarantee that the MIP produces a feasible solution of the problem.

In the second part, we remove uphill arcs $r \in R_{\text{red}}$ that might arise during this procedure.

In the third part, we recompute the ratios of the newly obtained arcs. This time, for a node $v \in V_{\text{red}}$, the ratio of an arc $r \in \delta^+_{G_{\text{red}}}(v)$ is set proportionally to the slopes of the arcs leaving v and to the length of the intersection of the boundaries of v and $\omega(r)$.

In the final part, we remove the node s and instead increase the area of nodes that are adjacent to s. This only decreases the size of the graph by a single node, but greatly improves the numerical stability of the MIP.

Finding a good value for the threshold is critical here. An overly high value leads to unrealistic results, whereas an overly low value decreases the performance gain obtained from the contraction. We found that, depending on the terrain surface, a value between 5 and 15 cm works best. On hilly surfaces, the value can preferably be set a bit higher, whereas on smooth surfaces, it is better to stick to low values.

The reduction in the overall number of nodes achieved in this last step mainly depends on the threshold parameter and the hilliness of the modeled region. The higher the threshold parameter and the flatter the region, the greater the reduction in the number of nodes.

For three representative regions, which are revisited later in Section 4.2, an overview of the reduction in the overall number of nodes from G_{or} to G_{red} is provided in Table 2.

Table 2 Reduction in the total number of nodes achieved for three representative regions, where the factor provided in the third column is obtained as $|V_{\text{or}}| / |V_{\text{red}}|$

Region	$\lvert V_{\text{or}} \rvert$	$\lvert V_{\text{red}} \rvert$	Factor
Hilly region	12,239,475	4719	2594
Flat region 1	2,523,799	6613	382
Flat region 2	1,789,498	3778	474

Algorithm 2 Contract-components

Procedure `contractComponents`$(G_{\text{wd}} = (V_{\text{wd}}, R_{\text{wd}}), \text{threshold}, \varepsilon)$
  Initialize $G_{\text{red}} = (V_{\text{red}}, R_{\text{red}})$ as a copy of $G_{\text{wd}} = (V_{\text{wd}}, R_{\text{wd}})$
  For each node $v \in V_{\text{red}}$, compute its geodesic height rounded to a multiple of the threshold and store it in rgh_v
  Initialize $\text{original\_nodes}_v = \{v\}$ for all $v \in V_{\text{red}}$
  # Contract nodes
  while *There are adjactent nodes* $u, v \in V_{\text{red}}$ *with* $\text{rgh}_v = \text{rgh}_u$ *and the buildings and actions on both nodes are the same* **do**
    Contract u, v into $\hat{v}$ and set $\text{original\_nodes}_{\hat{v}} = \text{original\_nodes}_u \cup \text{original\_nodes}_v$
  end
  # Enforce pairwise distinct geodesic heights
  while *There are two nodes* $u, v \in V_{\text{red}}$ *with* $\text{gh}_u = \text{gh}_v$ **do**
    if $\text{gh}_u = \min_{v' \in V_{\text{red}}} \text{gh}_{v'}$ **then**
      Set $\text{gh}_v = \text{gh}_v - \varepsilon$
    else
      Let $v' \in V_{\text{red}}$ with $\text{gh}_{v'}$ maximal such that $\text{gh}_{v'} < \text{gh}_v$
      Set $\text{gh}_v = \text{gh}_v - \min\{\varepsilon, \frac{1}{2} \cdot (\text{gh}_v - \text{gh}_{v'})\}$
    end
  end
  # Remove uphill arcs
  for $r \in R_{\text{red}}$ **do**
    if $\text{gh}_{\alpha(r)} < \text{gh}_{\omega(r)}$ **then**
      Remove r from R_{red} and add $\overleftarrow{r}$ to R_{red} if it does not already exist
    end
  end
  # Recompute ratios
  for $v \in V_{\text{red}}$ **do**
    $\text{sum\_of\_slopes}_v = \sum_{r \in \delta^+_{G_{\text{red}}}(v)} \text{slope}_r$
    Compute the length of the boundary of v in meters and store the value in $\text{length\_of\_boundary}_v$
    for $r \in \delta^+_{G_{\text{red}}}(v)$ **do**
      Compute the length of the intersection of the boundaries of v and $\omega(r)$ and store the value in common_boundary_r
      $\text{ratio}_r = (\text{slope}_r \cdot \text{common\_boundary}_r) / (\text{sum\_of\_slopes}_v \cdot \text{length\_of\_boundary}_v)$
    end
  end
  # Remove s
  for $r \in \delta^+_{G_{\text{red}}}(s)$ **do**
    $\text{area}_{\omega(r)} = \text{area}_{\omega(r)} + \text{area}_s \cdot \text{ratio}_r / \sum_{r' \in \delta^+_{G_{\text{red}}}(s)} \text{ratio}_{r'}$
  end
  Remove s and all its incident arcs from G_{red}
  return G_{red}

The *extended reduced graph* $G^{\text{ex}}_{\text{red}} = (V^{\text{ex}}_{\text{red}}, R^{\text{ex}}_{\text{red}})$ is constructed from the reduced graph $G_{\text{red}} = (V_{\text{red}}, R_{\text{red}})$ returned by Algorithm 2 in the same manner as we constructed it for the original graph, i.e., for each arc $r \in R_{\text{red}}$, we add a copy of r in reverse direction.

3.2 Mixed-Integer Programming Formulation and Presolve Techniques

In Section 3.2.1, we present our mixed-integer programming formulation of the problem defined in Section 3.1 as well as several intuitive valid inequalities that

improve solution times. The constraints are formulated verbally, while the mathematical formulation can be found in Appendix 3. We then describe methods to preset some of the variables in Section 3.2.2.

3.2.1 Mixed-Integer Programming Formulation

Before stating the mixed-integer programming formulation, we provide complete lists of the sets, parameters, and variables for better readability. The MIP takes, among other things, a graph and its extended graph as an input. Any of the graphs we constructed before could be used, but, as already mentioned, we highly recommend to use the reduced graph (and the corresponding extended reduced graph) here as all other graphs make the model too large or numerically unstable. The graph used in the MIP is denoted by $G = (V, R)$ and the corresponding extended graph by $G^{\text{ex}} = (V, R^{\text{ex}})$. Throughout this section, we assume that the nodes in V have pairwise distinct geodesic heights, which is the case if $G = G_{\text{red}}$.

Sets

V	node set of the graph
R	arc set of the graph
R^{ex}	arc set of the extended graph
B	set of buildings
$\mathcal{B}$	set of possible retention basins
$\mathcal{D}$	set of possible ditches
$\mathcal{E}$	set of possible embankments
$\mathcal{A}$	set of all possible actions, where $\mathcal{A} = \mathcal{B} \cup \mathcal{D} \cup \mathcal{E}$
P	set of properties
$P_{\text{yellow}} \subseteq P$	set of properties where the corresponding actor needs minor incentives to cooperate
$P_{\text{red}} \subseteq P$	set of properties where the corresponding actor needs major incentives to cooperate
$P_{\text{black}} \subseteq P$	set of properties where the corresponding actor does not cooperate at all

The sets corresponding to possible actions are denoted by calligraphic letters. We further introduce the set $\mathcal{B}_v \subseteq \mathcal{B}$ for each $v \in V$ as the set of basins on v. The sets $\mathcal{D}_v$ and $\mathcal{E}_v$ are defined analogously, and we let V_β denote the set of all nodes intersecting with building $\beta \in B$.

Parameters

rain	total rain per m^2 in m
budget	budget for the total cost of taken actions
GH_v	original geodesic height of node $v \in V$
area_v	area of node $v \in V$ in m^2
ratio_r	ratio of outflow of node $\alpha(r)$ allocated to arc $r \in R^{\text{ex}}$
depth_a	depth of basin or ditch $a \in \mathcal{B} \cup \mathcal{D}$ in m

height_e	height of embankment $e \in \mathcal{E}$ in m
cost_a	cost of action $a \in \mathcal{A}$
thresholdWL_k	threshold water level in m for hazard class $k \in \{0, 1, 2, 3\}$
$\text{damage}_{k,\beta}$	damage in the objective function if building $\beta \in B$ belongs to hazard class $k \in \{1, 2, 3, 4\}$
maxAllowedYellow	maximum number of properties needing minor incentives to cooperate that actions can be built on
maxAllowedRed	maximum number of properties needing major incentives to cooperate that actions can be built on

Variables

f_r	total flow on arc $r \in R^{\text{ex}}$ in m^3
excess_v	excess of node $v \in V$ in m^3
wl_v	water level at node $v \in V$ in m
flooded_v	1 if $\text{wl}_v > 0$, 0 otherwise
active_r	1 if there is flow along arc $r \in R^{\text{ex}}$, 0 otherwise
full_r	1 if $\text{wl}_{\alpha(r)} > 0$ for $r \in R$, 0 otherwise
decBasin_b	1 if basin $b \in \mathcal{B}$ is built, 0 otherwise
decDitch_d	1 if ditch $d \in \mathcal{D}$ is built, 0 otherwise
decEmb_e	1 if embankment $e \in \mathcal{E}$ is built, 0 otherwise
gh_v	geodesic height of node $v \in V$ after actions have been built in m
down_v	1 if a ditch or basin is built on $v \in V$, 0 otherwise
max_inc_v	maximum increase of height through building embankments on $v \in V$ in m
max_dec_v	maximum decrease of height through building ditches or basins on $v \in V$ in m
aux_fd_r	binary auxiliary variable for the flow distribution over arc $r \in R^{\text{ex}}$: 1 if arc is active and not full, 0 otherwise
od_r	1 if node $\alpha(r)$ is higher than node $\omega(r)$ after building the actions for $r \in R$, 0 otherwise
auxO1F1_r	binary auxiliary variable for $r \in R$: 1 if $\text{od}_r = 1$ and $\text{full}_r = 1$, 0 otherwise
auxO1F0_r	binary auxiliary variable for $r \in R$: 1 if $\text{od}_r = 1$ and $\text{full}_r = 0$, 0 otherwise
max_wl_β	maximum water level at any node intersecting with building $\beta \in B$ in m
$\text{hc}_{k,\beta}$	1 if building $\beta \in B$ belongs to hazard class $k \in \{0, \ldots, 4\}$, 0 otherwise
action_p	1 if an action is taken on property $p \in P$, 0 otherwise
hdb_b	absolute value of the height difference in m that is caused by building basin $b \in \mathcal{B}$ if basin b is built, 0 otherwise
hdd_d	absolute value of the height difference in m that is caused by building ditch $d \in \mathcal{D}$ if ditch d is built, 0 otherwise
hde_e	absolute value of the height difference in m that is caused by building embankment $e \in \mathcal{E}$ if it is built, 0 otherwise

Objective Function The only term in the objective function is the damage caused to the buildings, which depends on their hazard class and their damage class. Thus, the objective function to be minimized is given as

$$\sum_{\beta \in B} \sum_{k=1}^{4} \text{damage}_{k,\beta} \cdot \text{hc}_{k,\beta}.$$

Constraints To enhance readability, we use the max operator within our formulation. This operator takes a set of variables and/or parameters as an argument and returns the maximum among their values. Note that the operator can alternatively be implemented using big M constraints. This, however, may lead to numerical instability if finding a suitable value M is difficult. We therefore use the max operator, which is pre-implemented in most modern MIP solvers.

Furthermore, we make use of indicator constraints. An indicator constraint is of the form

$$bin = val \quad \Longrightarrow \quad a^T x \leq b$$

and states that the constraint $a^T x \leq b$ must be satisfied if the binary variable bin has value $val \in \{0, 1\}$. An indicator constraint can also be implemented using a big M constraint. It is, however, well known that indicator constraints have many advantages compared to big M formulations [28]. Indicator constraints are, like the max operator, pre-implemented in many modern MIP solvers.

The formulation of some constraints requires using strict inequalities, which is not possible theoretically in a MIP. In practice, however, values are encoded as floats with a bounded number of decimal places. Therefore, a strict inequality $x < y$ can be formulated as $x \leq y - \varepsilon$ for some small $\varepsilon > 0$.

Water Levels at Nodes To determine the water levels, we first compute the excess of each node $v \in V$:

1. The excess of node $v \in V$ is the inflow minus the outflow plus the rain volume on the node. The excess of a node $v \in V$ immediately yields the water level at the node:
2. The water level at node $v \in V$ is the excess of node v divided by its area.

Geodesic Heights of Nodes In contrast to most traditional flow problems, we do not aim to optimize the flow in the graph, but the terrain surface determining the flows. The following constraints therefore set the geodesic height variable gh_v for each node $v \in V$. First, to distinguish the two cases (GH1) and (GH2) from Section 3.1.2, the variable down_v is set to one in case (GH1), and to zero otherwise:

3. If a basin $b \in \mathcal{B}$ is built on node $v \in V$, the variable down_v is set to one.
4. If a ditch $d \in \mathcal{D}$ is built on node $v \in V$, the variable down_v is set to one.
5. If neither ditches nor basins are built on node $v \in V$, the variable down_v is set to zero.

Next, the variables hdb_b, hdd_d, and hde_e for $b \in \mathcal{B}$, $d \in \mathcal{D}$, and $e \in \mathcal{E}$ that determine the height differences that result from taking actions are set:

6. The variable hdb_b is set to depth_b if basin $b \in \mathcal{B}$ is built (i.e., if $\text{decBasin}_b = 1$), and to zero otherwise.
7. The variable hdd_d is set to depth_d if ditch $b \in \mathcal{B}$ is built (i.e., if $\text{decDitch}_d = 1$), and to zero otherwise.
8. The variable hde_e is set to height_e if embankment $e \in \mathcal{E}$ is built (i.e., if $\text{decEmb}_e = 1$), and to zero otherwise.

To enable setting the geodesic height variables as described in the case distinction, the maximum depth of any of the basins or ditches built on v in case (GH1) and the maximum height of any of the embankments built on v in case (GH2) is now computed:

9. The maximum decrease max_dec_v of the geodesic height at node $v \in V$ is set to the maximum of the height differences that result from building basins or ditches on v and 0.
10. The maximum increase of the geodesic height max_inc_v at node $v \in V$ is set to the maximum of the height differences that result from building embankments on v and 0.

Finally, the geodesic height variable gh_v is set for each node $v \in V$:

11. The geodesic height gh_v of node $v \in V$ is greater than or equal to the original geodesic height of v minus the maximum decrease caused by basins and ditches.
12. The geodesic height gh_v of node $v \in V$ is less than or equal to the original geodesic height of v plus the maximum increase caused by embankments.
13. If a basin or ditch is built on node $v \in V$ (i.e., $\text{down}_v = 1$), the geodesic height gh_v of v is less than or equal to the original geodesic height of v minus the maximum decrease caused by basins and ditches and, hence, in combination with Constraint (11), equal to the original geodesic height of v minus the maximum decrease caused by basins and ditches. This is modeled using a big M constraint where $M_v := \max(\{\text{depth}_b | b \in \mathcal{B}_v\} \cup \{\text{depth}_d | d \in \mathcal{D}_v\} \cup \{0\}) + \max(\{\text{height}_e | e \in \mathcal{E}_v\} \cup \{0\})$.
14. If no basin or ditch is built on node $v \in V$ (i.e., $\text{down}_v = 0$), the geodesic height gh_v of v is greater than or equal to the original geodesic height of v plus the maximum increase caused by embankments and, hence, in combination with Constraint (12), equal to the original geodesic of v height plus the maximum increase caused by embankments. This is again modeled using a big M constraint with the same M_v as in the previous constraint.

Arc Directions There might be arcs in the input graph where, after taking actions and thereby changing the geodesic heights of nodes, the start node has a lower geodesic height than the end node, so the direction of the arc has to be reversed. If this is *not* the case for an arc $r \in R$, the arc is said to have *original direction* and the variable od_r is set to one by using indicator constraints:

15. If arc $r \in R$ has original direction, the variable od_r is set to one.
16. Otherwise, the variable od_r is set to zero.

Full Arcs The following constraints deal with the behavior of the flows on the arcs in the extended graph $G^{\text{ex}} = (V, R^{\text{ex}})$. To this end, we introduce the following terminology: An

arc $r \in R$ is called *full* if the water level at the lower of the two nodes $\alpha(r)$ and $\omega(r)$ is greater than or equal to the absolute difference of their geodesic heights. For its inverse arc $\overleftarrow{r} \in R^{\text{ex}} \setminus R$, we say that this arc is full if and only if r is full.[5] Note that this definition refers to the geodesic heights after taking actions, where it is possible that $\alpha(r)$ has a smaller geodesic height than $\omega(r)$. To connect the variables full_r to the water levels, some binary auxiliary variables incorporating the original direction variables are first introduced:

17. The variable auxO1F1_r for arc $r \in R$ is set to one if and only if $\text{od}_r = 1$ and $\text{full}_r = 1$.
18. The variable auxO1F0_r for arc $r \in R$ is set to one if and only if $\text{od}_r = 1$ and $\text{full}_r = 0$.
19. The variable auxO0F1_r for arc $r \in R$ is set to one if and only if $\text{od}_r = 0$ and $\text{full}_r = 1$.
20. The variable auxO0F0_r for arc $r \in R$ is set to one if and only if $\text{od}_r = 0$ and $\text{full}_r = 0$.

The following constraints connect the variables full_r to the water levels using the auxiliary variables:

21. If arc $r \in R$ has original direction and is full, the water level at $\omega(r)$ must be greater than or equal to the absolute difference of the geodesic heights of $\alpha(r)$ and $\omega(r)$.
22. If arc $r \in R$ has original direction and is not full, the water level at $\omega(r)$ must be less than the absolute difference of the geodesic heights of $\alpha(r)$ and $\omega(r)$.
23. If arc $r \in R$ does not have original direction and is full, the water level at $\alpha(r)$ must be greater than or equal to the absolute difference of the geodesic heights of $\alpha(r)$ and $\omega(r)$.
24. If arc $r \in R$ does not have original direction and is not full, the water level at $\alpha(r)$ must be less than the absolute difference of the geodesic heights of $\alpha(r)$ and $\omega(r)$.

Flooded Nodes A node $v \in V$ is called *flooded* if its water level wl_v is strictly positive, and *non-flooded* otherwise. The following indicator constraints set the variables flooded_v for $v \in V$ that indicate flooded nodes:

25. If the water level wl_v at node $v \in V$ is strictly positive, the variable flooded_v is set to one.
26. If the water level wl_v at node v is zero, the variable flooded_v is set to zero.

Active Arcs The net flow between two adjacent nodes in the extended graph can be in either one or the other direction. An arc $r \in R^{\text{ex}}$ is called *active* if the flow on r is strictly positive. The following constraints set the variables active_r for$r \in R^{\text{ex}}$ that indicate active arcs:

[5] Nodes in G_{red} have pairwise distinct geodesic heights, so the lower node is always well-defined. In practice, the geodesic heights after taking actions are also pairwise distinct. If this is not the case, one can decrease the depth of height of the action that causes the issue by a small value similarly to how pairwise distinct geodesic heights of nodes are enforced in G_{red}.

27 For arc $r \in R$ and its inverse arc $\overleftarrow{r} \in R^{\text{ex}}$, at most one of the variables active_r and $\text{active}_{\overleftarrow{r}}$ can be equal to one.
28 If an arc $r \in R^{\text{ex}}$ is not active, the flow on the arc must be zero.

Flow on Arcs that Are Not Full The outflow of a node $v \in V$ is to be distributed according to the ratios of its outgoing arcs in the extended graph $G^{ex} = (V, R^{\text{ex}})$ that are active and not full. The following constraints set the auxiliary variables aux_fd_r and $\text{aux\_fd}_{\overleftarrow{r}}$ for $r \in R$ that indicate arcs that are both active and full:

29. For arc $r \in R$, the auxiliary variable aux_fd_r is to one if and only if the arc is active and not full.
30. For arc $r \in R$, the auxiliary variable $\text{aux\_fd}_{\overleftarrow{r}}$ for the inverse arc is set to one if and only if $\overleftarrow{r}$ is active and not full.[6]

The outflow of each node $v \in V$ is now distributed among its outgoing arcs in the extended graph that are active and not full:

31. For node $v \in V$ and each pair of arcs $r_1, r_2 \in \delta^+_{G^{\text{ex}}}(v)$, if both arcs are active and not full, the flow is distributed proportionally to the ratios ratio_{r_1} and ratio_{r_2}.

For each arc $r \in R$ that is not full, the water level at the higher of the two nodes $\alpha(r)$ and $\omega(r)$ must be zero.

32. For each arc $r \in R$ that is not full and has original direction, the water level at $\alpha(r)$ is set to zero.
33. For each arc $r \in R$ that is not full and does not have original direction, the water level at $\omega(r)$ is set to zero.

Water can only flow on downhill arcs $r \in R^{\text{ex}}$ that are not full:

34. For each arc $r \in R$ that is not full and has original direction, the arc $\overleftarrow{r}$ is not active.
35. For each arc $r \in R$ that is not full and does not have original direction, the arc r is not active.

Flow on Full Arcs As the flow is immediately connected to the water levels by Constraints (1) and (2), the flow on each full arc $r \in R$ can be set indirectly by connecting the water levels at its start node and its end node:

36. For each full arc $r \in R$, the sum of the geodesic height and the water level must be equal in $\alpha(r)$ and $\omega(r)$.

Maximum Water Levels at Buildings

37. For each building $\beta \in B$, the maximum water level variable max_wl_β is set to the maximum of the water levels at nodes intersecting with the building.

Note that, strictly speaking, the maximum is not taken here, but the maximum water level at the building is only bounded from below by each water level at an intersecting node. The objective function then aims to minimize the maximum water levels at the buildings to achieve equality.

[6] Recall that $\overleftarrow{r}$ is full if and only if r is full.

Hazard Classes of Buildings

38. Each building $\beta \in B$ belongs to exactly one hazard class.
39. If building $\beta \in B$ belongs to hazard class $k \in \{0, \dots, 4\}$, its maximum water level must be less than or equal to the upper threshold of this hazard class.

    Again, the maximum water levels are only bounded from above as a higher hazard class leads to a higher penalty in the objective function.

Budget Constraint

40. The total cost for building basins, ditches, and embankments must not exceed the given budget.

Incentives for Actors The following constraints enforce the given upper bounds on the incentives required for cooperation of actors and ensure that no actions are taken on properties of actors that do not cooperate at all. This is done by means of the variables action_p for $p \in P$ that indicate properties on which at least one action is taken:

41. Actions are taken on at most maxAllowedYellow + maxAllowedRed yellow and red properties in total.
42. Actions are taken on at most maxAllowedRed red properties.
43. No actions are taken on black properties.
44. The variable action_p for property $p \in P$ is set to one if at least one action is taken on property p.

    It is worth noting that it is not trivial to see that the MIP is indeed a correct formulation of the problem defined in Section 3.1. However, we show this in Appendix 2 by proving that any feasible solution of the MIP taking exactly the actions in $D \subseteq \mathcal{A}$ leads to the same water levels at the nodes as the result of Algorithm 6 applied on G^D, which is the graph that results from taking the actions in D and adjusting the geodesic heights and arc directions accordingly as described in Section 2.

Valid Inequalities We finish the description of the MIP by presenting three intuitive sets of valid inequalities that improve the solution times of the model:

45. For each pair of consecutive original-direction (i.e., downhill) arcs $r_1, r_2 \in R$ with $\omega(r_1) = \alpha(r_2)$, the first arc r_1 can only be full if the second arc r_2 is full as well.
46. If node $v \in V$ is flooded, then each arc $r \in \delta^+_{G^{\text{ex}}}(v)$ with $\text{gh}_v > \text{gh}_{\omega(r)}$ must be full (otherwise, water could still flow in downhill direction from v).
47. If node $v \in V$ is not flooded, then no arc $r \in \delta^-_{G^{\text{ex}}}(v)$ with $\text{gh}_v < \text{gh}_{\alpha(r)}$ can be full.

3.2.2 Presolve Techniques

We close this chapter by presenting two methods to preset some of the variables.

Through our analysis, we found that the variables flooded_v for $v \in V$ are the major bottleneck of the MIP. It is therefore natural to investigate which nodes must always be flooded and which nodes can never be flooded in a feasible solution in order to preset some of these variables to one or zero, respectively.

We start by presetting variables for nodes that must always be flooded. To this end, we consider the leaves of the graph $G = (V, R)$. If there is no possible embankment on a leaf $l \in V$ and no possible ditches or basins on any of the nodes in $\delta^-(l)$, the leaf will also be a leaf after taking actions—independent of which actions are selected. This means that l is flooded in any feasible solution since at least the initial water from the rain event will build up a water level strictly larger than zero at l. For all such leaves, we can therefore preset the variable flooded_l to one.

Identifying nodes $v \in V$ for which the variable flooded_v can be preset to zero (i.e., nodes that can never be flooded in any feasible solution) is more involved. The idea here is that, if no possible action is located on v, the water levels at all successors of v must match the geodesic height of v in order for v to be flooded. Thus, if the total amount of rain on the whole area does not suffice for raising the water level at each successor to the absolute difference of the geodesic height of the successor and the geodesic height of v, then v can never be flooded in any feasible solution.

In order to find such non-flooded nodes, we start by computing the maximum possible geodesic height of each node than can be obtained after taking actions,[7] and then construct a new graph $G_{\text{nf}} = (V_{\text{nf}}, R_{\text{nf}})$ where each node is assigned its corresponding maximum possible geodesic height and arcs are directed in downhill direction with respect to these geodesic heights.[8] For each node on which no actions are located, we compute its successors in G_{nf}.[9] If the amount of rain that is needed to raise the water level at each of these successors to the absolute difference of the geodesic height of the successor and the geodesic height of v exceeds the total rain volume on the whole area, node v can never be flooded in any feasible solution. If this is not the case, we can apply the same idea using a larger set of nodes instead of the successors of v. To this end, we consider the undirected version of G_{nf} and remove all nodes that have strictly larger geodesic height than v. We then compute all nodes different from v that are in the same connected component as v in the remaining undirected graph. It is clear that the set of these nodes is a superset of the set of successors of v in G_{nf}, so we can apply the reasoning as before to this larger set of nodes.

The pseudocode of the corresponding algorithm is presented as Algorithm 7 in Appendix 4. Note that one could of course use the larger set of nodes right away, but this would cause a non-negligible overhead in computation.

4 Computational Results

In this section, we present a comparison of the results obtained from our MIP to results obtained from established simulation software (Section 4.1). Afterwards, we use real-world instances from different municipalities to identify and analyze the

[7] Recall that building embankments can increase the geodesic heights of nodes.

[8] **nf**: **n**on-**f**looded.

[9] Note that nodes on which no actions are located have the same geodesic height in G and in G_{nf}.

main drivers for the running time of our method and the quality of the obtained solutions (Section 4.2).

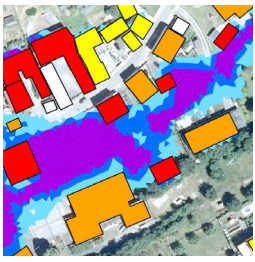
4.1 Comparison with Established Simulation Software

To validate our approach, we compared the results obtained on real-world instances to results obtained on these instances from the well-established simulation software "HYSTEM-EXTRAN" [29]. HYSTEM-EXTRAN is the German industry standard for hydro-dynamic simulations in urban water management and is used by most engineering offices and municipalities when evaluating precautionary measures for flash floods caused by heavy rain events. It does, however, not support any kind of optimization, but can only be used to simulate the water levels resulting from a given (usually manually chosen) combination of actions for a given amount of rain. Thus, we compared the water levels resulting from our approach for the status quo of each instance (which contains only the already implemented actions, if any) without allowing any additional actions to the water levels obtained from HYSTEM-EXTRAN's simulation for the same situation. The results obtained from HYSTEM-EXTRAN have been provided and validated by the engineering office igr AG, which was one of our partners in the project AKUT.

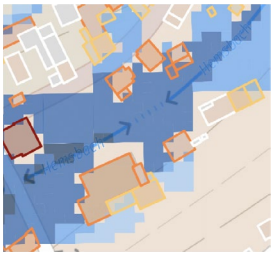
All in all, we found that the results predominantly coincide, with only slight differences that usually occur at the periphery of flooded areas. An illustrative example, in which a 30-year rain event (i.e., the heaviest rain to be expected in the chosen area over a time span of 30 years) has been simulated in a hilly region, is provided in Fig. 5. We further found that AKUT slightly underestimates the damage to buildings in hilly regions whereas it slightly overestimates the damage to buildings in flat regions. This is due to the fact that HYSTEM-EXTRAN also takes damage caused by high current velocity into account, which is neglected in AKUT.

4.2 Running Time and Performance

We now investigate the running times of our MIP and the quality of the obtained solutions. For both of them, we present the most important drivers that have been identified



(a) Water levels obtained from HYSTEM-EXTRAN.



(b) Water levels obtained from the software AKUT.

Fig. 5 Extract from a comparison of water levels obtained from HYSTEM-EXTRAN and AKUT for a 30-year rain event in a hilly region. The darker the blue color, the higher the water level, where the highest obtained levels are illustrated in purple in the case of HYSTEM-EXTRAN

Table 3 Computational results for nine representative instances. The column "HM" (**H**illiness **M**eassure) contains the median of the values obtained by dividing the slope of each arc by the Euclidean distance of the centers of its incident nodes, which is a measure of how hilly the terrain is. The column "FSF" contains the time until the final solution is found. The column "IOV" contains the objective value of the initial solution of the MIP provided by applying Algorithm 6. The column "BOV" contains the objective value of the best solution returned by the MIP. Note that the different values of $|V_{\text{red}}|$ among instances with the same region result from different merging of nodes during preprocessing due to different rain events

Region	Total area (m^2)	# Buildings	HM		
Hilly region (HR)	2,294,375	579	7.2%		
Flat region 1 (FR1)	1,728,799	573	1.9%		
Flat region 2 (FR2)	585,123	957	2.0%		
Instance	$\lvert V_{\text{red}}\rvert$	Running time	FSF	IOV	BOV
HR, 30-year, 4 actions	4719	48 min	31 min	821	807
HR, 50-year, 4 actions	4750	2 h, 20 min	44 min	863	855
HR, 50-year, 6 actions	4750	56 min	42 min	863	854
FR1, 30-year, 4 actions	6613	1 h, 8 min	33 min	405	392
FR1, 50-year, 4 actions	6646	24 h	4 h, 26 min	418	399
FR1, 50-year, 6 actions	6646	24 h	2 h, 35 min	418	398
FR2, 30-year, 4 actions	3778	6 h, 36 min	6 h, 36 min	571	475
FR2, 50-year, 4 actions	3790	3 h, 2 min	2 h, 33 min	575	483
FR2, 50-year, 6 actions	3790	2 h, 27 min	2 h, 27 min	575	480

by applying our approach to a wide range of different real-world problem instances. As an illustration, results for nine representative instances obtained from three different regions (two municipalities and a part of a city) that are considered in three relevant scenarios are presented. The three scenarios are a 30-year rain event with a budget that allows to take four actions, a 50-year rain event with with a budget that allows to take four actions, and a 50-year rain event with with a larger budget that allows to take six actions.[10] The regions are a municipality on a hilly terrain, called "Hilly Region" (HR) in the following, a municipality on a flat terrain, called "Flat Region 1" (FR1) in the following, and a part of a city on a flat terrain, called "Flat Region 2" (FR2) in the following. For each region, the set $\mathcal{A}$ of possible actions is the same for all three scenarios and consists of about 20 actions that have been selected according to the local circumstances such that each of them could be implemented in reality.

For each instance, an initial solution taking no actions, which is computed using Algorithm 6, is given to the MIP. As termination criterion, a 3% MIP gap is used for the hilly region, and a 5% MIP gap for the flat regions. Furthermore, a time limit of 24 h is set. All computations in this section were executed using Gurobi 9.5.0 on a server with 32 AMD EPYC 7542 processors (2.9GHz). The most important characteristics of the instances together with the results and the running times are provided in Table 3.

[10] Recall that a 30-year (50-year) rain event corresponds to the heaviest rain to be expected in the chosen area over a time span of 30 years (50 years).

In general, it is found that the maximum possible number of actions is taken in each of the nine instances. It is worth noting that there are instances, which are not presented here, where this is not the case. Possible reasons for not taking an action although it would be possible are (1) an overabundance of possible actions and a large budget such that the action that is not taken does not contribute to the quality of the solution anymore and (2) a poor-quality location of the action such that the action does not protect any buildings.

Among the 42 actions that are selected by the MIP in the presented instances, 40 are basins, while only two are ditches or embankments. This confirms observations made on numerous real-world instances indicating that retention basins, if they can be built, are usually the most efficient actions. However, building retention basins requires free space, which is not always available, especially in densely populated urban areas. Embankments and ditches are usually built together with a retention basin such that the actions overlap geographically. An intuitive explanation for this behavior of the MIP is that retention basins act as a storage for the water, while ditches or embankments connect the inflow from a larger area to the basins.

Another interesting finding is that, in hilly regions, the most efficient retention basins, i.e., the ones that are typically selected by the MIP, are often low-lying and located centrally. As an illustration, in the three scenarios considered for HR, 10 of the 14 selected basins are low-lying and located centrally.

Although the rain volume is significantly higher in the instances modeling the 50-year rain events, it is found that neither the budget nor the rain volume dramatically change the set of taken actions. Among the 12 actions taken in the three considered instances with a 50-year rain event and four possible actions per instance, eight are also taken in the corresponding instances with a 30-year rain event. Furthermore, among the 12 actions taken in the three instances with a 50-year rain event and six possible actions per instance, nine are also taken in the corresponding instances with four possible actions.

In general, the running times show high fluctuations as can be seen in FR1, where the instance with the 30-year rain event takes significantly less time to solve than the instances with the 50-year rain event. Still, several factors influencing the running time and the quality of the obtained solutions can be identified.

Concerning the running time, the most important factor is the number of nodes in the graph (i.e., $|V_{\text{red}}|$). The instances of FR1, which are the instances with the largest number of nodes, with a 50-year rain event are the only instances in our experiments on which the MIP gap could not be closed before reaching the time limit, whereas all other instances could be solved within less than 7 h.

The second most important factor is the hilliness of the terrain surface. Comparing HR and FR2, the instances for HR are solved significantly faster than the instances for FR2 despite the graph for FR2 being slightly smaller than the graph for HR. Further, in hillier regions, the MIP tends to be numerically more stable.

Although the parameter "MIPFocus" is set to 2 and the parameter "Heuristics" is set to 0.01, which both enforce a stronger attention on improving the lower bound, the final solution is usually found relatively quickly, and the solver spends a significant part of the overall running time on improving the lower bound afterwards, as can be seen when comparing the values in the columns "Running Time" and "FSF"

(final solution found) in Table 3. Without tuning the parameters accordingly, the running time increases drastically since the solver struggles to close the MIP gap.

Concerning the quality of the solutions, we observe that hillier regions usually have more damage potential overall. To illustrate this, we compare HR to FR1, which have almost the same number of buildings. However, the objective values of both the initial solution and the solution returned by the MIP are more than twice as large in HR as in FR1. This is due to two reasons. Firstly, hilly regions have heavier rainfalls than flat regions due to orographic precipitation [30]. In our example, a 30-year rain event in HR has a precipitation level of 44.9 mm whereas a 30-year rain event in FR1 has a precipitation level of only 35.9 mm. Secondly, hilly regions tend to have a larger drainage area, which can also be seen comparing the total areas of HR and FR1. Indeed, the difference in the total areas result almost entirely from a higher number of water-dispensing nodes in instances of HR.

Lastly, the density of the buildings (i.e., the number of buildings per area) affects the potential of how much better the solution returned by the MIP can be compared to the initial solution (i.e., the solution where no actions are taken). In our case, there is a significantly higher density of buildings in FR2 than there is in HR and FR1. We see that the difference between the objective values of the initial solution and the obtained solution from the MIP is considerably higher in FR2 than it is in HR and FR1. This stems from the fact that, if a high water level at a critical location is prevented by an action, the action protects more buildings in FR2 than it does in the other two regions. It is worth noting, however, that there are other instances with a high density of buildings where planning impactful actions becomes hard due to lack of space. In such cases, a high density of buildings can decrease the potential for damage reduction by taking actions significantly.

5 Conclusion

To the best of our knowledge, the web application AKUT is the first software that uses optimization techniques to support decision-making in planning precautionary measures for flash floods caused by heavy rain events and scales well enough to be applied to realistic scenarios. It is currently used by more than 25 organizations from all over Germany. The usage of optimization techniques in this context has evidently provided valuable support in handling a challenging and highly topical task for various organizations like municipalities, engineering offices, and research institutes.

A mixed-integer program is used to minimize the damage in the case of a heavy rain event by taking best-possible actions subject to a limited budget and constraints on the cooperation of residents. To model the terrain surface, a grid graph obtained from a digital terrain model is transformed by several preprocessing methods such that the cardinality of its node set becomes small enough to apply the previously mentioned mixed-integer program while still maintaining a realistic representation of the terrain surface. Comparisons with results from established software provide strong evidence that solutions obtained from our approach yield realistic results.

When applying the software to large cities, these must currently be subdivided into several parts due to performance reasons. Hence, an interesting question would

be how the performance of our approach could further be improved such that it can handle larger instances. As the problem decomposes into several smaller subproblems, using decomposition methods could be a promising attempt. Additionally, even though an efficiently implemented combinatorial algorithm (Algorithm 6) is already used to generate an initial feasible solution of our MIP, another possible approach for improving the running times of the model could be to implement callbacks that use this algorithm to compute new solutions at later stages of the branch and bound process.

Appendix 1. Obtaining the Buildings and Actions on the Nodes

A straightforward algorithm to obtain the sets B_v of buildings and $\mathcal{A}_v$ of actions for all nodes $v \in V$ is to loop over the nodes and, within this loop, iterate over all buildings and actions and check if the shape of the node and the shape of the building or action intersect. However, this has a horrendous running time and can be done way more efficiently using the connectivity of the shapes of buildings and actions.

For simplicity, we only present the algorithm to compute the set B_v for all nodes $v \in V$. The computation of the sets $\mathcal{A}_v$ works similarly. For each building, we initialize a queue q containing a single node $v' \in V$ whose shape contains the coordinate of some vertex of the building's shape. Note that the coordinate of such a node can easily be computed by rounding both components of the vertex's coordinate to the next multiple of 1, 5, or 25 depending on which of the node sets the algorithm is called for.

While the queue is not empty, we take a node v from the queue and check whether it intersects with the building. If this is the case, we add the building to $B(v)$ and add the nodes north, south, west, and east of v that have not yet been processed for this building to the queue. The pseudocode is provided in Algorithm 3, which takes the set B of buildings and a node set $\hat{V} \in \{V_{25}, V_5, V_{\text{or}}\}$ as its two arguments.

Algorithm 3 COMPUTE-BUILDINGS-ON-NODES

```
1  Procedure computeBuildingsOnNodes(B, V̂)
2      Initialize B_v = ∅ for all v ∈ V̂
3      for β ∈ B do
4          Compute v' ∈ V̂ containing an arbitrary vertex of the building's shape
5          Initialize q = [v'] and visited = ∅
6          while q is nonempty do
7              v = q.pop()
8              if β is on v and v ∉ visited then
9                  Add the nodes north, south, west, and east of v that are not
                     in visited to q
10                 Add β to B_v
11             end
12             Add v to visited
13         end
14     end
15     return B_v for all v ∈ V̂
```

Appendix 2. Computing the Water Levels

In this part of the appendix, the algorithm for computing water levels, which has already been introduced in Section 3.1.2, is presented. The algorithm uses two subroutines that are presented first.

Computing the Flows in the Graph We start by describing the subroutine for computing the flows in the graph. The idea is to start at the highest root node in the graph and dispense all its water to its children according to the ratios of the corresponding arcs. Then, this root node is removed from the graph and we continue with the next highest root node in the graph. This process is repeated until all nodes have been visited. The pseudocode is provided in Algorithm 4.

Algorithm 4 COMPUTE-FLOWS

```
Procedure computeFlows(G̃ = (Ṽ, R̃))
    Initialize f_r = 0 for all r ∈ R̃
    Initialize excess_v = rain · area_v for all v ∈ V
    Save a copy G' = (V', R') of the input graph G̃ = (Ṽ, R̃)
    while V' ≠ ∅ do
        Choose v ∈ V' as a root node with largest geodesic height among all root
          nodes in G'
        if v is not a leaf in G' then
            Distribute the whole excess of v among its outgoing arcs proportionally
              to their ratios and save the flow on r in f_r for each r ∈ δ⁺_{G'}(v)
            for r ∈ δ⁺_{G'}(v) do
                excess_{ω(r)} = excess_{ω(r)} + f_r
            end
        end
    end
    return f = (f_r)_{r∈R̃}
```

Note that the flows have to be recomputed in each iteration of Algorithm 6. Using Algorithm 4 to compute the flows from scratch for the whole graph in each iteration is highly inefficient and causes significant overhead. Instead, if one is provided the flows from the previous iteration and the nodes $u, v \in \tilde{V}$ that have just been joined using Algorithm 5, the flows can be updated. It is easy to see that the excess of nodes that are neither v nor one of its successors remain unchanged. Further, it is also easy to see that excess_v increases exactly by $\text{area}_u \cdot \text{rain} + \sum_{r \in \delta^-(u)} f_r$. This additional excess is then distributed along the subgraph that is induced by v and its successors, and the new flows are added to the flows that have been computed in the previous iteration. In fact, updating the flows in this way decreases the running time of Algorithm 6 by more than 90% compared to recomputing the flows from scratch in each iteration.

Joining Nodes The second subroutine that is called in Algorithm 6 joins two nodes u and v in the graph $\tilde{G}_t$ of the current iteration t, which then becomes the current graph $\tilde{G}_{t+1}$ in the next iteration $t+1$. Whenever the subroutine is called, it holds that v is the unique parent node of u with lowest geodesic height. Within the subroutine, all arcs in $\delta^-_{\tilde{G}_t}(u)$ are redirected such that v is their new end node, the new set $\text{repres}_{v,t+1}$ is

set to $\text{repres}_{v,t} \cup \text{repres}_{u,t}$, and the new area of v in the next iteration $t+1$ is set to $\text{area}_{v,t} + \text{area}_{u,t}$. The pseudocode of this subroutine is provided in Algorithm 5. It is worth noting that, when u is removed from $\tilde{V}_{t+1}$ in line 7 of the algorithm, u has no incident arcs. It has no outgoing arcs because the algorithm is only called for u being a leaf in G, and the incoming arcs are redirected in the for loop starting in line 2. It is worth noting that subsequent calls of this routine as it is done in Algorithm 6 can lead to parallel arcs in the graph.

Algorithm 5 JOIN-NODES

```
1  Procedure joinNodes(u, v)
2      Save a copy G̃_{t+1} = (Ṽ_{t+1}, R̃_{t+1}) of G̃_t = (Ṽ_t, R̃_t)
3      for r ∈ δ⁻_{G̃_{t+1}}(u) with α(r) ≠ v do
4          Set ω(r) = v in G̃_{t+1}
5      end
6      repres_{v,t+1} = repres_{v,t} ∪ repres_{u,t}
7      area_{v,t+1} = area_{v,t} + area_{u,t}
8      Remove u from Ṽ_{t+1}
9      for v' ∈ Ṽ_{t+1} \ {v} do
10         repres_{v',t+1} = repres_{v',t}
11         area_{v',t+1} = area_{v',t}
12     end
```

Computing the Water Levels We present the algorithm for computing the water levels for a given graph, whose pseudocode is provided in Algorithm 6. Initially, a copy $\tilde{G}_1 = (\tilde{V}_1, \tilde{R}_1)$ of the input graph is saved. This graph is modified in each iteration of the algorithm and we denote the graph at the beginning of the t-th iteration of the algorithm by $\tilde{G}_t = (\tilde{V}_t, \tilde{R}_t)$. With the flows that are computed using Algorithm 4 in the first iteration or its above-mentioned modified version using the update method in the following iterations, we compute for each leaf the proportion of the rain event that is needed to fill up the water level at the leaf to the geodesic height of its parent with lowest index (and, hence, lowest geodesic height), which we call the *lowest parent*. The leaf for which the least such proportion is needed is called the *first flooded leaf*.[11] For a node $v \in V$, we denote its lowest parent in $\tilde{G}_t$ by $\text{lp}_{\tilde{G}_t}(v)$. If the graph is clear from the context, we omit the graph in the index. The extraction of the first flooded leaf is implemented using a Fibonacci heap, which results in a speedup of about 25% on average as opposed to extracting it using a simple sorted-array implementation.

The first flooded leaf and its lowest parent are then joined into a single node representing both those nodes by using Algorithm 5. The excesses until then are saved, and, from there on, the water is increased until either (1) the water level at the next leaf reaches the absolute difference of its own geodesic height and the geodesic height of its lowest parent in the graph $\tilde{G}_t$ of the current iteration of the algorithm, or (2) the sum of those proportions, which we denote by sp,[12] equals or exceeds one, i.e., we have simulated the whole rain event.

[11] In case that several leaves have the least proportion, any of them can be designated as the first flooded leaf.

[12] **sp: s**um of **p**roportions.

Throughout this process, for each node v that is created joining nodes as described above, we store the nodes in the input graph that are represented by v in the current iteration t in a set $\text{repres}_{v,t}$, which is updated within Algorithm 5. Afterwards, we recompute the water levels of all nodes in the input graph using these sets, which is done by setting the water level at each node $v \in V$ that has been removed from the graph during the algorithm to $\text{wl}_{\tilde{v},t} + \text{gh}_{\tilde{v}} - \text{gh}_v$, where $\tilde{v}$ is the unique node such that $v \in \text{repres}_{\tilde{v},T}$ with T denoting the number of iterations of the algorithm.[13]

We would like to point out that the iteration indices in Algorithm 6 are chosen such that all variables concerning the graph, i.e., the graph $\tilde{G}_t$ itself, the area area_t and the sets repres_t, are initialized with one, and all variables concerning the flows on the graph, i.e., the flows f_t themselves, the excesses excess_t, the proportions $\hat{p}_t$, and the sum of proportions sp, are initialized with zero. This enforces that, in each iteration t of the algorithm, the flows $f_{r,t}$, $r \in \tilde{R}_t$, are the flows on the graph $\tilde{G}_t$.

Algorithm 6 COMPUTE-WATER-LEVELS

Procedure `computeWaterLevels`$(G = (V, R))$
  Initialize $\text{wl}_v = 0$ for all $v \in V$, $t = 1$, and $\text{sp}_0 = 0$
  Save a copy $\tilde{G}_1 = (\tilde{V}_1, \tilde{R}_1)$ of the input graph
  for $v \in V$ **do**
    $\text{repres}_{v,1} = \{v\}$
    $\text{area}_{v,1} = \text{area}_v$
    $\text{excess}_{v,0} = 0$
  end
  while $\text{sp}_t < 1$ **do**
    For each leaf $u \in \tilde{V}_t$, compute its incoming water over the whole rain event given that the graph remains unchanged as

$$\text{iwr}_{u,t} = \text{area}_{u,t} \cdot \text{rain} + \sum_{r \in \delta^-_{\tilde{G}_t}(u)} f_{r,t},$$

    where the flows $f_{r,t}$ are the flows computed by Algorithm 4 in the first iteration $t = 1$ or its modified version using the update method in all later iterations $t > 1$
    For each leaf $u \in \tilde{V}_t$, compute the proportion $p_{u,t}$ of the rain event that is needed to fill up the water level at the leaf to the absolute difference of its own geodesic height and the geodesic height of its lowest parent $\text{lp}(u)$, i.e., such that $\text{excess}_{u,t-1} + \text{iwr}_{u,t} \cdot p_{u,t} = \left(\text{gh}_{\text{lp}_{\tilde{G}_t}(u)} - \text{gh}_u\right) \cdot \text{area}_{u,t}$
    Denote by $\hat{u}$ a leaf whose corresponding proportion $\hat{p}_t := p_{\hat{u},t}$ is the lowest among the computed proportions where ties are broken arbitrarily, and by $\hat{v}$ its lowest parent
    if $\text{sp}_{t-1} + \hat{p}_t \geq 1$ **then**
      $\hat{p}_t = 1 - \text{sp}_{t-1}$ and $\text{sp}_t = 1$
    else
      $\text{sp}_t = \text{sp}_{t-1} + \hat{p}_t$
    end
    For each leaf node $u \in \tilde{V}$, set $\text{excess}_{u,t} = \text{excess}_{u,t-1} + \hat{p}_t \cdot \text{iwr}_{u,t}$
    if $\text{sp}_t \neq 1$ **then**
      Call `joinNodes`$(\hat{u}, \hat{v})$
      $t = t + 1$
    end
  end
  for $\tilde{v} \in \tilde{V}_t$ **do**
    for $v \in \text{repres}_{\tilde{v},t}$ **do**
      $\text{wl}_v = \text{wl}_{\tilde{v},t} + \text{gh}_{\tilde{v}} - \text{gh}_v$
    end
  end
  return $\text{wl} = (\text{wl}_v)_{v \in V}$

[13] The existence and uniqueness of $\tilde{v}$ is shown in Lemma and Definition 1 in Appendix 2.

Appendix 3. Constraints of the Mixed-Integer Programming Formulation

We provide the mathematical formulation of the constraints of the MIP presented in Section 3.

Water Levels at Nodes Computing excess_v for each node $v \in V$:

$$\text{excess}_v = \sum_{r \in \delta^+_{G^{ex}}(v)} f_r - \sum_{r \in \delta^-_{G^{ex}}(v)} f_r + \text{rain} \cdot \text{area}_v \quad \forall v \in V \tag{1}$$

Computing the water level wl_v at each node $v \in V$:

$$\text{wl}_v = \frac{\text{excess}_v}{\text{area}_v} \quad \forall v \in V \tag{2}$$

Geodesic Heights of Nodes Setting the variable down_v for each node $v \in V$:

$$\text{down}_v \geq \text{decBasin}_b \quad \forall v \in V, b \in \mathcal{B}_v \tag{3}$$

$$\text{down}_v \geq \text{decDitch}_d \quad \forall v \in V, d \in \mathcal{D}_v \tag{4}$$

$$\text{down}_v \leq \sum_{b \in \mathcal{B}_v} \text{decBasin}_b + \sum_{d \in \mathcal{D}_v} \text{decDitch}_d \quad \forall v \in V \tag{5}$$

Setting the variables hdb_b, hdd_d, and hde_e for $b \in \mathcal{B}$, $d \in \mathcal{D}$, $e \in \mathcal{E}$:

$$\text{hdb}_b = \text{depth}_b \cdot \text{decBasin}_b \quad \forall b \in \mathcal{B} \tag{6}$$

$$\text{hdd}_d = \text{depth}_d \cdot \text{decDitch}_d \quad \forall d \in \mathcal{D} \tag{7}$$

$$\text{hde}_e = \text{height}_e \cdot \text{decEmb}_e \quad \forall e \in \mathcal{E} \tag{8}$$

Computing the maximum height of an embankment for each node $v \in V$:

$$\text{max\_dec}_v = max(\{\text{hdb}_b | b \in \mathcal{B}_v\} \cup \{\text{hdd}_d | d \in \mathcal{D}_v\} \cup \{0\}) \quad \forall v \in V \tag{9}$$

$$\text{max\_inc}_v = \max(\{\text{hde}_e | e \in \mathcal{E}(v)\} \cup \{0\}) \quad \forall v \in V \tag{10}$$

Setting the geodesic height variable gh_v for each node $v \in V$:

$$\text{gh}_v \geq \text{GH}_v - \text{max\_dec}_v \quad \forall v \in V \tag{11}$$

$$\text{gh}_v \leq \text{GH}_v + \text{max\_inc}_v \quad \forall v \in V \tag{12}$$

$$\text{gh}_v \leq \text{GH}_v - \text{max\_dec}_v + (1 - \text{down}_v) \cdot M_v \quad \forall v \in V \tag{13}$$

$$\mathrm{gh}_v \geq \mathrm{GH}_v + \mathrm{max\_inc}_v - \mathrm{down}_v \cdot M_v \quad \forall v \in V \tag{14}$$

Arc Directions Setting the variable od_r for each arc $r \in R$:

$$\mathrm{od}_r = 1 \Rightarrow \mathrm{gh}_{\alpha(r)} \geq \mathrm{gh}_{\omega(r)} \quad \forall r \in R \tag{15}$$

$$\mathrm{od}_r = 0 \Rightarrow \mathrm{gh}_{\alpha(r)} < \mathrm{gh}_{\omega(r)} \quad \forall r \in R \tag{16}$$

Full Arcs Setting the auxiliary variables $\mathrm{auxO1F1}_r$, $\mathrm{auxO1F0}_r$, $\mathrm{auxO0F1}_r$, and $\mathrm{auxO0F0}_r$ for each arc $r \in R$:

$$\mathrm{auxO1F1}_r \geq -1 + \mathrm{od}_r + \mathrm{full}_r \quad \forall r \in R \tag{17.1}$$

$$\mathrm{auxO1F1}_r \leq \mathrm{full}_r \quad \forall r \in R \tag{17.2}$$

$$\mathrm{auxO1F1}_r \leq \mathrm{od}_r \quad \forall r \in R \tag{17.3}$$

$$\mathrm{auxO1F0}_r \geq \mathrm{od}_r - \mathrm{full}_r \quad \forall r \in R \tag{18.1}$$

$$\mathrm{auxO1F0}_r \leq 1 - \mathrm{full}_r \quad \forall r \in R \tag{18.2}$$

$$\mathrm{auxO1F0}_r \leq \mathrm{od}_r \quad \forall r \in R \tag{18.3}$$

$$\mathrm{auxO0F1}_r \geq -\mathrm{od}_r + \mathrm{full}_r \quad \forall r \in R \tag{19.1}$$

$$\mathrm{auxO0F1}_r \leq \mathrm{full}_r \quad \forall r \in R \tag{19.2}$$

$$\mathrm{auxO0F1}_r \leq 1 - \mathrm{od}_r \quad \forall r \in R \tag{19.3}$$

$$\mathrm{auxO0F0}_r \geq 1 - \mathrm{od}_r - \mathrm{full}_r \quad \forall r \in R \tag{20.1}$$

$$\mathrm{auxO0F0}_r \leq 1 - \mathrm{full}_r \quad \forall r \in R \tag{20.2}$$

$$\mathrm{auxO0F0}_r \leq 1 - \mathrm{od}_r \quad \forall r \in R \tag{20.3}$$

Connecting the variables full_r to the water levels using the auxiliary variables:

$$\mathrm{auxO1F1}_r = 1 \Rightarrow \mathrm{wl}_{\omega(r)} \geq \mathrm{gh}_{\alpha(r)} - \mathrm{gh}_{\omega(r)} \quad \forall r \in R \tag{21}$$

$$\mathrm{auxO1F0}_r = 1 \Rightarrow \mathrm{wl}_{\omega(r)} < \mathrm{gh}_{\alpha(r)} - \mathrm{gh}_{\omega(r)} \quad \forall r \in R \tag{22}$$

$$\mathrm{auxO0F1}_r = 1 \Rightarrow \mathrm{wl}_{\alpha(r)} \geq \mathrm{gh}_{\omega(r)} - \mathrm{gh}_{\alpha(r)} \quad \forall r \in R \tag{23}$$

$$\text{auxO0F0}_r = 1 \Rightarrow \text{wl}_{\alpha(r)} < \text{gh}_{\omega(r)} - \text{gh}_{\alpha(r)} \quad \forall r \in R \tag{24}$$

Flooded Nodes Setting the variable flooded_v for each node $v \in V$:

$$\text{flooded}_v = 0 \Rightarrow \text{wl}_v = 0 \quad \forall v \in V \tag{25}$$

$$\text{flooded}_v = 1 \Rightarrow \text{wl}_v > 0 \quad \forall v \in V \tag{26}$$

Active Arcs Setting the variable active_r for each arc $r \in R^{\text{ex}}$:

$$\text{active}_r + \text{active}_{\overleftarrow{r}} = 1 \quad \forall r \in R \tag{27}$$

$$\text{active}_r = 0 \Rightarrow f_r = 0 \quad \forall r \in R^{\text{ex}} \tag{28}$$

Flow on Arcs that are not Full Setting the auxiliary variables aux_fd_r and $\text{aux\_fd}_{\overleftarrow{r}}$ for each arc $r \in R$:

$$\text{aux\_fd}_r \geq \text{active}_r - \text{full}_r \quad \forall r \in R \tag{29.1}$$

$$\text{aux\_fd}_r \leq \text{active}_r \quad \forall r \in R \tag{29.2}$$

$$\text{aux\_fd}_r \leq 1 - \text{full}_r \quad \forall r \in R \tag{29.3}$$

$$\text{aux\_fd}_{\overleftarrow{r}} \geq \text{active}_{\overleftarrow{r}} - \text{full}_r \quad \forall r \in R \tag{30.1}$$

$$\text{aux\_fd}_{\overleftarrow{r}} \leq \text{active}_{\overleftarrow{r}} \quad \forall r \in R \tag{30.2}$$

$$\text{aux\_fd}_{\overleftarrow{r}} \leq 1 - \text{full}_r \quad \forall r \in R \tag{30.3}$$

Distributing the outflow of each node $v \in V$ among its outgoing arcs in the extended graph that are active and not full: Setting the auxiliary variables aux_fd_r and $\text{aux\_fd}_{\overleftarrow{r}}$ for each arc $r \in R$:

$$\text{aux\_fd}_{r_2} = 1 \Rightarrow f_{r_1} \leq \frac{\text{ratio}_{r_1}}{\text{ratio}_{r_2}} \cdot f_{r_2} \quad \forall v \in V, r_1, r_2 \in \delta^+_{G^{\text{ex}}}(v) \tag{31.1}$$

$$\text{aux\_fd}_{r_1} = 1 \Rightarrow f_{r_2} \leq \frac{\text{ratio}_{r_2}}{\text{ratio}_{r_1}} \cdot f_{r_1} \quad \forall v \in V, r_1, r_2 \in \delta^+_{G^{\text{ex}}}(v) \tag{31.2}$$

For each arc $r \in R$ that is not full, the water level at the higher of the nodes $\alpha(r)$ and $\omega(r)$ must be zero:

$$\text{auxO1F0}_r = 1 \Rightarrow \text{wl}_{\alpha(r)} = 0 \quad \forall r \in R \tag{32}$$

$$\text{auxO0F0}_r = 1 \Rightarrow \text{wl}_{\omega(r)} = 0 \quad \forall r \in R \tag{33}$$

Water can only flow on downhill arcs $r \in R^{\text{ex}}$ that are not full:

$$\text{auxO1F0}_r = 1 \Rightarrow \text{active}_{\overleftarrow{r}} = 0 \quad \forall r \in R \tag{34}$$

$$\text{auxO0F0}_r = 1 \Rightarrow \text{active}_r = 0 \quad \forall r \in R \tag{35}$$

Flow on Full Arcs Setting the flow on each full arc $r \in R$ indirectly by connecting the water levels at its start node and its end node:

$$\text{full}_r = 1 \Rightarrow \text{gh}_{\alpha(r)} + \text{wl}_{\alpha(r)} = \text{gh}_{\omega(r)} + \text{wl}_{\omega(r)} \quad \forall r \in R \tag{36}$$

Maximum Water Levels at Buildings Bounding the maximum water level variable max_wl_β from below for each building $\beta \in B$:

$$\text{max\_wl}_\beta \geq \text{wl}_v \quad \forall \beta \in B, v \in V_\beta \tag{37}$$

Hazard Classes of Buildings Setting a hazard class for each building $\beta \in B$ via its maximum water level:

$$\sum_{k=0}^{4} \text{hc}_{k,\beta} = 1 \quad \forall \beta \in B \tag{38}$$

$$\text{hc}_{k,\beta} = 1 \Rightarrow \text{max\_wl}_\beta \leq \text{thresholdWL}_k \quad \forall \beta \in B, k \in \{0,1,2,3\} \tag{39}$$

$$\sum_{b \in \mathcal{B}} \text{cost}_b \cdot \text{decBasin}_b + \sum_{d \in \mathcal{D}} \text{cost}_d \cdot \text{decDitch}_d + \sum_{e \in \mathcal{E}} \text{cost}_e \cdot \text{decEmb}_e \leq \text{budget} \tag{40}$$

Budget Constraint

Incentives for Actors Enforcing the given upper bounds on the incentives required for cooperation of actors and ensuring that no actions are taken on properties of actors that do not cooperate at all:

$$\sum_{p \in P_{\text{yellow}}} \text{action}_p + \sum_{p \in P_{\text{red}}} \text{action}_p \leq \text{maxAllowedYellow} + \text{maxAllowedRed} \tag{41}$$

$$\sum_{p \in P_{\text{red}}} \text{action}_p \leq \text{maxAllowedRed} \tag{42}$$

$$\text{action}_p = 0 \quad \forall p \in P_{\text{black}} \tag{43}$$

$$\text{decBasin}_b \leq \text{action}_p \quad \forall b \in \mathcal{B}, p \in P : b \text{ is located on } p \tag{44.1}$$

$$\text{decDitch}_d \leq \text{action}_p \quad \forall d \in \mathcal{D}, p \in P : d \text{ is located on } p \tag{44.2}$$

$$\text{decEmb}_e \leq \text{action}_p \quad \forall e \in \mathcal{E}, p \in P : e \text{ is located on } p \tag{44.3}$$

Valid Inequalities The first arc in a pair of consecutive original-direction (i.e., downhill) arcs can only be full if the second arc is full as well:

$$\text{full}_{r_2} \geq \text{full}_{r_1} - (2 - \text{od}_{r_1} - \text{od}_{r_2}) \quad \forall r_1, r_2 \in R : \omega(r_1) = \alpha(r_2) \tag{45}$$

If node $v \in V$ is flooded, then each arc $r \in \delta^+_{G^{\text{ex}}}(v)$ with $\text{gh}_v > \text{gh}_{\omega(r)}$ must be full:

$$\text{flooded}_v = 1 \Rightarrow \text{full}_r \geq \text{od}_r \qquad \forall v \in V, r \in \delta^+_G(v) \tag{46.1}$$

$$\text{flooded}_v = 1 \Rightarrow \text{full}_r \geq 1 - \text{od}_r \qquad \forall v \in V, r \in \delta^-_G(v) \tag{46.2}$$

If node $v \in V$ is not flooded, then no arc $r \in \delta^-_{G^{\text{ex}}}(v)$ with $\text{gh}_v < \text{gh}_{\alpha(r)}$ can be full:

$$\text{flooded}_v = 0 \Rightarrow \text{full}_r \leq 1 - \text{od}_r \qquad \forall v \in V, r \in \delta^-_G(v) \tag{47.1}$$

$$\text{flooded}_v = 0 \Rightarrow \text{full}_r \leq \text{od}_r \qquad \forall v \in V, r \in \delta^+_G(v) \tag{47.2}$$

Appendix 4. Proof of Validity of the Mixed-Integer Programming Formulation

In this section, we prove that the MIP is a valid formulation of the problem described in Section 3.1. To formally define the statement we want to prove, let $D \subseteq \mathcal{A}$ be a set of actions such that building exactly the actions in D fulfills the budget Constraint (40) and does not violate any bounds on the incentives in Constraints (41)–(44.3). Throughout this section, we let x denote a feasible solution of the MIP taking exactly the actions in D, and let y denote the result of Algorithm 6 applied on $G^D = (V^D, R^D)$, which is the graph that results from taking the actions in D and adjusting the geodesic heights and arc directions accordingly as described in Section 2.

As some variables are denoted the same in the MIP and in Algorithm 6, we write, e.g., x.wl and y.wl, respectively, in case of the water levels, to distinguish between them whenever the distinction is not clear from the context.

The aim of this section is to prove that, for each node $v \in V(= V^D)$, the water level at v in x is the same as the water level at v in y, i.e., $x.\text{wl}_v = y.\text{wl}_v$ for all $v \in V$. As the value of the objective value is determined by the water levels, this in particular means that the objective value of a solution of the MIP only depends on the taken actions.

It is worth noting that, although we prove that the water levels in any feasible solution x of the MIP taking a given set D of actions coincide with those of the corresponding result y of Algorithm 6, the flows in different MIP solutions taking the same actions can still differ. The reason is that, if a cycle of flooded nodes exists

in G^{ex}, an arbitrary amount of flow might be sent over this cycle, which conserves feasibility but can cause the flows to be different in the different solutions even when the same actions are taken.

For the proof, we make the following assumptions:

1. The graph G and, hence, also the graph G^D is weakly connected. This is the case in all realistic instances and, moreover, can be assumed without loss of generality because the arguments in the proof can be applied to each weakly connected component individually in case that there are multiple weakly connected components.
2. The highest node in the graph G is non-flooded in both x and y. This assumption is satisfied in all real-world problem instances since rain events that flood each single node are unrealistic, and damage on buildings could not be mitigated by any realistic actions anyway in such cases.
3. The geodesic heights of nodes in G^D are pairwise distinct, which is true if $G = G_{\text{red}}$.

Firstly note that Constraints (3)–(14) imply that, for each $v \in V$, it holds that $x.\text{gh}_v$ is the geodesic height of v in G^D. We therefore omit the "x." for the geodesic height in the following. Also note that taking actions only directly affects the geodesic heights, but the flows and, hence, the water levels are only affected indirectly via their dependence on the geodesic heights in G^D. As, furthermore, the variables od only act as a case distinction in the MIP, it suffices to show that $x.\text{wl}_v = y.\text{wl}_v$ for all $v \in V$ in the case where $D = \emptyset$ and, thus, $G^D = G$, i.e., for the case that no actions are taken.

Appendix 4.1. Characterization of Flooded Subgraphs

In this section, for both the feasible solution x of the MIP and the result y of Algorithm 6, we present a characterization of inclusionwise-maximal weakly connected subgraphs consisting of nodes that are flooded, i.e, have a strictly positive water level. These subgraphs will be referred to as *sinks* and are formally defined as follows:

Definition 1 Given x or y, each weakly connected component of the subgraph G_{flooded} of G induced by the set of flooded nodes is called a *sink*. The set of all sinks is denoted by $\mathcal{S}(x)$ and $\mathcal{S}(y)$ for x and y, respectively.

The goal of this section is to show that, for both x and y, every sink is a *pre-sink*, i.e., it consists of a node v and all nodes that can be reached from v via undirected paths that contain no nodes of geodesic height larger than v. Formally, pre-sinks are defined as follows:

Definition 2 For $v \in V$, the weakly connected component containing v in the induced subgraph $G_{\leq v} := G|_{\{v' \in V : \text{gh}_{v'} \leq \text{gh}_v\}}$ is called the *pre-sink induced by* v. The set of nodes contained in the pre-sink induced by v is denoted by $\text{PS}(v)$.

Note that, in the following, we slightly abuse notation by identifying a sink or pre-sink with the set of nodes it contains as long as this does not lead to any confusion.

Appendix 4.1.1. Characterization of Flooded Subgraphs for x

We start by proving the desired connection between sinks and pre-sinks for the feasible solution x of the MIP. Throughout this subsection, we omit the "x." when referring to variables, so, e.g., the water level at a node v is denoted by wl_v instead of $x.\mathrm{wl}_v$.

Observation 1 Let $r \in R^{\mathrm{ex}}$ with $\alpha(r) = u$, $\omega(r) = v$, and $\mathrm{gh}_v < \mathrm{gh}_u$. If $\mathrm{wl}_u > 0$, then $\mathrm{wl}_v + \mathrm{gh}_v = \mathrm{wl}_u + \mathrm{gh}_u$.

Proof Constraints (46.1) and (46.2) imply that r is full. Constraint (36) then yields $\mathrm{wl}_v + \mathrm{gh}_v = \mathrm{wl}_u + \mathrm{gh}_u$. □

Observation 2 Let $r \in R^{\mathrm{ex}}$ with $\alpha(r) = u$, $\omega(r) = v$, and $\mathrm{gh}_v < \mathrm{gh}_u$. If $\mathrm{wl}_v > \mathrm{gh}_u - \mathrm{gh}_v$, then $\mathrm{wl}_v + \mathrm{gh}_v = \mathrm{wl}_u + \mathrm{gh}_u$.

Proof Constraint (22) (if $r \in R$) or (24) (if $r \notin R$) forces $\mathrm{full}_r = 1$ as $\mathrm{wl}_v > \mathrm{gh}_u - \mathrm{gh}_v$. As before, Constraint (36) then yields $\mathrm{wl}_v + \mathrm{gh}_v = \mathrm{wl}_u + \mathrm{gh}_u$. □

The two observations are used to prove the following important proposition:

Proposition 1 *Let $v \in V$. If $\mathrm{gh}_{v'} + \mathrm{wl}_{v'} > \mathrm{gh}_v$ for some $v' \in \mathrm{PS}(v)$, then every node $\hat{v} \in \mathrm{PS}(v)$ is flooded with $\mathrm{gh}_{\hat{v}} + \mathrm{wl}_{\hat{v}} = \mathrm{gh}_{v'} + \mathrm{wl}_{v'}$.*

Proof Let $v' \in \mathrm{PS}(v)$ with $\mathrm{gh}_{v'} + \mathrm{wl}_{v'} > \mathrm{gh}_v$ and let $\hat{v}$ be an arbitrary node in $\mathrm{PS}(v)$. Then, since the pre-sink is weakly connected, there exists an undirected path P with $\mathrm{trace}(P) = (v', \tilde{v}_1, \dots, \tilde{v}_k, \hat{v})$ in $G_{\leq v}$. In particular, this means that $\mathrm{gh}_{\tilde{v}_i} \leq \mathrm{gh}_v$ for all $i \in \{1, \dots, k\}$. Applying Observations 1 and 2 inductively on the path yields

$$\begin{aligned} \mathrm{wl}_{\tilde{v}_i} &= \mathrm{wl}_{v'} + \mathrm{gh}_{v'} - \mathrm{gh}_{\tilde{v}_i} > 0 \text{ for all } i \in \{1, \dots, k\}, \text{ and} \\ \mathrm{wl}_{\hat{v}} &= \mathrm{wl}_{v'} + \mathrm{gh}_{v'} - \mathrm{gh}_{\hat{v}} > 0, \end{aligned}$$

which proves the claim. □

Using Proposition 1, we can now prove the desired connection between sinks and pre-sinks for x:

Proposition 2 *Let $S \in \mathcal{S}(x)$ and let $v \in S$ be the node with highest geodesic height among all nodes in S. Then, $S = \mathrm{PS}(v)$.*

Proof By definition of $\mathrm{PS}(v)$, it holds that $v \in \mathrm{PS}(v)$. Moreover, it is easy to see that $\mathrm{PS}(v) \subseteq S$ by applying Proposition 1 for $v' = v$. Hence, it only remains to show that $S \subseteq \mathrm{PS}(v)$. To this end, let $v' \in S$ be an arbitrary node. By definition of a sink, S is weakly connected, which means that there exists an undirected path P in G only containing nodes in S with $\mathrm{trace}(P) = (v, \tilde{v}_1, \dots, \tilde{v}_k, v')$ and $\mathrm{gh}_{\tilde{v}_i} \leq \mathrm{gh}_v$ for all $i \in \{1, \dots, k\}$. This means that P is also a path in $G_{\leq_v}$ and, hence, that v' is in the same connected component of $G_{\leq_v}$ as v. Therefore, we obtain that $v' \in \mathrm{PS}(v)$. □

The proofs of Propositions 1 and 2 also yield the following helpful property about the water levels at nodes within the same sink:

Corollary 1 *Let $S \in \mathcal{S}(x)$ and $u, v \in S$. Then,* $\mathrm{wl}_v + \mathrm{gh}_v = \mathrm{wl}_u + \mathrm{gh}_u$.

When investigating the water levels, it therefore suffices to know the water level at one node within each sink.

Appendix 4.1.2. Characterization of Flooded Subgraphs for *y*

We now prove the desired connection between sinks and pre-sinks for the result y of Algorithm 6. Throughout this subsection, we again omit the "y." when referring to variables, so, e.g., the water level at a node v is denoted by wl_v instead of $y.\mathrm{wl}_v$. Although the proof is a bit more involved, the basic idea is similar to the proof of Proposition 2. Before we show two observations similar to Observations 1 and 2, we introduce some notation and obtain some further structural results.

Notation 1 The total number of iterations of the while-loop in Algorithm 6 is denoted by T. Furthermore, we write $\overline{G} = (\overline{V}, \overline{R}) := (\tilde{V}_T, \tilde{R}_T)$ and $\mathrm{repres}_v := \mathrm{repres}_{v,T}$ for $v \in V$.

Further, let $t \in \{1, \dots, T-1\}$ and let $v \in V$ such that v is the first flooded leaf in iteration t. We then say that v *leaves the graph* in iteration t. Given a node $v \notin \overline{V}$, we denote the unique iteration in which v leaves the graph by t_v. In the following, we present some structural results about the sets repres_v for $v \in V$.

Observation 3 Let $v \in \overline{V}$. For two nodes $u_1, u_2 \in \mathrm{repres}_v$, it holds that $\mathrm{gh}_{u_1} + \mathrm{wl}_{u_1} = \mathrm{gh}_{u_2} + \mathrm{wl}_{u_2}$

Proof Due to lines 23 to 27 of the algorithm, it holds that

$$\mathrm{gh}_{u_1} + \mathrm{wl}_{u_1} = \mathrm{gh}_v + \mathrm{wl}_v \text{ and } \mathrm{gh}_{u_2} + \mathrm{wl}_{u_2} = \mathrm{gh}_v + \mathrm{wl}_v.$$

□

As shown in the previous observation, the water level at a node $v \notin \overline{V}$ is determined by a node $\bar{v} \in \overline{V}$ such that $v \in \mathrm{repres}_{\bar{v}}$. We now introduce a suitable notion for this node and show that it is uniquely defined.

Lemma and Definition 1 *For each $t \in \{1, \dots, T\}$ and each node $v \in V$, there exists exactly one node $\tilde{v} \in \tilde{V}_t$ such that $v \in \text{repres}_{\tilde{v},t}$. For a node $v \in V$, we call the (unique) node $\bar{v} \in \overline{V}$ such that $v \in \text{repres}_{\bar{v}}$ the highest representative of v and write $\bar{v} = \text{hr}_v$.*

Proof For $t = 1$, the claim is clear as $\text{repres}_{v,1} = \{v\}$ for all $v \in V$. In each iteration $t \in \{1, \dots, T-1\}$ of the algorithm, one node u leaves the graph and is joined with its lowest parent v. All nodes in $\text{repres}_{u,t}$ are then in $\text{repres}_{v,t+1}$ in the next iteration and all other sets $\text{repres}_{v',t}$ for $v' \in \tilde{V}_t \setminus \{u\}$ remain unchanged. Hence, the property is conserved in each iteration of the algorithm, which proves the claim. □

Note that the set repres_{v,t_v} is not deleted when a node $v \in V$ leaves the graph, which means a node $v \in V$ can be in several sets $\text{repres}_{v'}$ for $v' \in V$. Also note that, as soon as a node $v \in V$ joins a set $\text{repres}_{u,t}$ for some other node $u \in V$ in some iteration $t \in \{1, \dots, T-1\}$, the node never leaves this set again, which implies that $v \in \text{repres}_u$ in this case. Moreover, the previous proof yields the following observation:

Observation 4 Let $v \in V$. Then, $v \in \overline{V}$ if and only if $v = \text{hr}_v$.

We proceed by proving some further structural results in order to obtain the analogous statements to Observations 1 and 2 in the context of y.

Lemma 1 *Let $r \in R$ with $\alpha(r) = u$ and $\omega(r) = v$. If $u \notin \overline{V}$, then it holds that $v \in \text{repres}_u$.*

Proof Let $u \notin \overline{V}$. We start by showing that also $v \notin \overline{V}$. Suppose for the sake of a contradiction that $v \in \overline{V}$. Then, the child v of u never leaves the graph, which implies that u is never a leaf. Consequently, u can never be removed and, thus, $u \in \overline{V}$, which contradicts the assumption that $u \in \overline{V}$.

Hence, we obtain that $v \notin \overline{V}$, so there exists an iteration t_v where v leaves the graph, i.e., `joinNodes` $v, \tilde{v}_1$ is called for $\tilde{v}_1 = \text{lp}_{\tilde{G}_{t_v}}(v)$. Thus, we have $v \in \text{repres}_{\tilde{v}_1,t_v+1}$ in the following iteration, so also $v \in \text{repres}_{\tilde{v}_1}$.

If $\tilde{v}_1 = u$, we are done. Otherwise, $\text{gh}_u > \text{gh}_{\tilde{v}_1}$ and there exists an arc $r_1 \in \tilde{R}_{t_v+1}$ from u to $\tilde{v}_1$. In the same way as for v, it then follows that $\tilde{v}_1 \notin \overline{V}$ and that it must, hence, be joined into another node $\tilde{v}_2 \in V$ in some iteration. Furthermore, it holds that $v \in \text{repres}_{\tilde{v}_2}$. Applying this argument iteratively induces a sequence of nodes with strictly increasing geodesic heights until eventually $\tilde{v}_k = u$ for some $k \in \mathbb{N}$. Thus, it holds that $v \in \text{repres}_u$. □

Lemma 2 *Let $r \in R$ with $\alpha(r) = u$ and $\omega(r) = v$. If $\text{wl}_u > 0$, then $v \in \text{repres}_u$.*

Proof Case 1: $u \notin \overline{V}$ Then, the claim follows directly from Lemma 1.

Case 2: $u \in \overline{V}$ As $\text{wl}_u > 0$, u must be a leaf in $\overline{G}$, which implies that $v \notin \overline{V}$. Hence, there exists an iteration t_v where v leaves the graph, i.e., `joinNodes` $v, \tilde{v}_1$ is called for $\tilde{v}_1 = \text{lp}_{G_{t_v}}(v)$. As in the proof of Lemma 1, this implies that $v \in \text{repres}_{\tilde{v}_1}$.

If $\tilde{v}_1 = u$, we are done. Otherwise, $\mathrm{gh}_u > \mathrm{gh}_{\tilde{v}_1}$ and there exists an arc $r_1 \in \tilde{R}_{t_v+1}$ from u to $\tilde{v}_1$. In this case, we show that $\tilde{v}_1 \notin \overline{V}$. For the sake of a contradiction, suppose that $\tilde{v}_1 \in \overline{V}$. Then, r_1 remains in the graph from iteration $t_v + 1$ until the termination of the algorithm, meaning that u is not a leaf in $\overline{G}$ and, hence, is not flooded. This contradicts the assumption that $\mathrm{wl}_u > 0$, so we obtain that $\tilde{v}_1 \notin \overline{V}$. Thus, $\tilde{v}_1$ must be joined into another node $\tilde{v}_2 \in V$ in some iteration. Furthermore, it holds that $v \in \mathrm{repres}_{\tilde{v}_2}$. Applying this argument iteratively induces a sequence of nodes with strictly increasing geodesic heights until eventually $\tilde{v}_k = u$ for some $k \in \mathbb{N}$. Thus, it holds that $v \in \mathrm{repres}_u$. □

One further structural result is needed, which can be interpreted as the transitivity of the representatives.

Observation 5 Let $u, v, w \in V$ such that $u \in \mathrm{repres}_v$ and $v \in \mathrm{repres}_w$. Then, $u \in \mathrm{repres}_w$.

Proof Let u join repres_v in iteration t_u, and let v join repres_w in iteration t_v. Since the set repres_v remains unchanged after v leaves the graph, it must then hold that $t_u < t_v$. Moreover, in all iterations $t > t_u$, the nodes u and v are always in the same set $\mathrm{repres}_{\tilde{v},t}$ for $\tilde{v} \in \tilde{V}_t$. Thus, when v joins repres_w in iteration $t_v > t_u$, so does u, which proves the claim. □

The technical results above allow proving the following observation, which is the analogue of Observation 1 in the context of y:

Observation 6 Let $r \in R$ with $\alpha(r) = u$ and $\omega(r) = v$. If $\mathrm{wl}_u > 0$ then $\mathrm{hr}_u = \mathrm{hr}_v$ and $\mathrm{wl}_u + \mathrm{gh}_u = \mathrm{wl}_v + \mathrm{gh}_v$.

Proof Lemma 2 implies that $v \in \mathrm{repres}_u$. From Observation 5, we obtain that $v \in \mathrm{repres}_{\mathrm{hr}_u}$ and, hence, that $\mathrm{hr}_u = \mathrm{hr}_v$. Observation 3 then implies that $\mathrm{wl}_u + \mathrm{gh}_u = \mathrm{wl}_v + \mathrm{gh}_v$. □

To prove the analogue of Observation 2, we need one more structural result:

Lemma 3 *Let $r \in R$ with $\alpha(r) = u$ and $\omega(r) = v$ with $\mathrm{wl}_v > \mathrm{gh}_u - \mathrm{gh}_v$, then $v \in \mathrm{repres}_u$.*

Proof We start by proving that $v \notin \overline{V}$. For the sake of a contradiction, suppose that $v \in \overline{V}$. Due to Lemma 1, it must then hold that $u \in \overline{V}$ as, otherwise, this would imply that $v \notin \overline{V}$. Therefore, the arc r is never removed or changed during the algorithm, so $r \in \overline{R}_t$ for all $t \in \{1, \ldots, T\}$. As $\mathrm{wl}_v > \mathrm{gh}_u - \mathrm{gh}_v$, and $v \in \overline{V}$, the node v must be a leaf in $\overline{G}$. Let $\tilde{v} = \mathrm{lp}_{\overline{G}}(v)$. It must hold that $\mathrm{wl}_v \leq \mathrm{gh}_{\tilde{v}} - \mathrm{gh}_v$ as, otherwise, v would have been joined into its lowest parent during the algorithm. As u is a parent of v in each iteration, it must hold that $\mathrm{gh}_{\tilde{v}} \leq \mathrm{gh}_u$. This implies that $\mathrm{wl}_v \leq \mathrm{gh}_u - \mathrm{gh}_v$, which is a contradiction to $\mathrm{wl}_v > \mathrm{gh}_u - \mathrm{gh}_v$. Thus, it holds that $v \notin \overline{V}$.

As $v \notin \overline{V}$, there exists an iteration t_v where v leaves the graph, i.e., `joinNodes` $v, \tilde{v}_1$ is called for $\tilde{v}_1 = \mathrm{lp}_{\tilde{G}_{t_v}}(v)$. Hence, it holds that $v \in \mathrm{repres}_{\tilde{v}_1}$. If $\tilde{v}_1 = u$, we are done. Otherwise, $\mathrm{gh}_u > \mathrm{gh}_{\tilde{v}_1}$ and there exists an arc $r_1 \in \tilde{R}_{t_v+1}$ from u to $\tilde{v}_1$.

We now prove that $\tilde{v}_1 \notin \overline{V}$. For the sake of a contradiction, suppose $\tilde{v}_1 \in \overline{V}$. Then, $\tilde{v}_1 = \mathrm{hr}_v$ and Observation 3 implies that $\mathrm{wl}_{\tilde{v}_1} = \mathrm{wl}_v + \mathrm{gh}_v - \mathrm{gh}_{\tilde{v}_1} > \mathrm{gh}_u - \mathrm{gh}_{\tilde{v}_1}$. The desired contradiction is now obtained analogously to the argumentation above showing that $v \notin \overline{V}$.

Thus, $\tilde{v}_1$ must be joined into another node $\tilde{v}_2 \in V$ in some iteration. Furthermore, it holds that $v \in \mathrm{repres}_{\tilde{v}_2}$. Applying this argument iteratively induces a sequence of nodes with strictly increasing geodesic heights until eventually $\tilde{v}_k = u$ for some $k \in \mathbb{N}$. Thus, it holds that $v \in \mathrm{repres}_u$. □

Observation 7 Let $r \in R$ with $\alpha(r) = u$ and $\omega(r) = v$ with $\mathrm{wl}_v > \mathrm{gh}_u - \mathrm{gh}_v$. Then, it holds that $\mathrm{hr}_u = \mathrm{hr}_v$ and $\mathrm{wl}_u + \mathrm{gh}_u = \mathrm{wl}_v + \mathrm{gh}_v$.

Proof Lemma 3 implies that $v \in \mathrm{repres}_u$. From Observation 5, we obtain that $v \in \mathrm{repres}_{\mathrm{hr}_u}$ and, hence, that $\mathrm{hr}_u = \mathrm{hr}_v$. Observation 3 then implies that $\mathrm{wl}_u + \mathrm{gh}_u = \mathrm{wl}_v + \mathrm{gh}_v$. □

We now use the two previous observations to prove the analogue of Proposition 1 in the context of y:

Proposition 3 *Let* $v \in V$. *If* $\mathrm{gh}_{v'} + \mathrm{wl}_{v'} > \mathrm{gh}_v$ *for some* $v' \in \mathrm{PS}(v)$, *then every node* $\hat{v} \in \mathrm{PS}(v)$ *is flooded with* $\mathrm{gh}_{\hat{v}} + \mathrm{wl}_{\hat{v}} = \mathrm{gh}_{v'} + \mathrm{wl}_{v'}$.

Proof The proof is completely analogous to the proof of Proposition 1 except for using Observations 6 and 7 instead of Observations 1 and 2. □

Proposition 3 finally allows us to prove the desired connection between sinks and pre-sinks for y:

Proposition 4 *Let* $S \in \mathcal{S}(y)$ *and let* $v \in S$ *be the node with highest geodesic height among all nodes in* S. *Then*, $S = \mathrm{PS}(v)$.

Proof The proof is completely analogous to the proof of Proposition 2, but uses Proposition 3 instead of Proposition 1. □

Similar to the case of Proposition 2, the proof of Proposition 4 also yields the following corollary:

Corollary 2 *Let* $S \in \mathcal{S}(y)$ *and* $u, v \in S$. *Then*, $\mathrm{wl}_v + \mathrm{gh}_v = \mathrm{wl}_u + \mathrm{gh}_u$.

Appendix 4.2. Characterization of flooded pre-sinks

Using Propositions 2 and 4 together with Corollaries 1 and 2, it remains to show that $\mathcal{S}(x) = \mathcal{S}(y)$ and that, for each node $v \in V$ with $\mathrm{PS}(v) \in \mathcal{S}(x)$, it holds that $x.\mathrm{wl}_v = y.\mathrm{wl}_v$. To this end, we now characterize, for each of x and y, when all nodes

in a pre-sink are flooded, in which case the corresponding pre-sink will also be called *flooded*. The following definition is useful in this context:

Definition 3 Let $v \in V$. If v is not the highest node in G, the *lowest parent* $\mathrm{lp}(\mathrm{PS}(v))$ of the pre-sink $\mathrm{PS}(v)$ is defined as the node with minimal geodesic height in $\delta^-_G(\mathrm{PS}(v))$.[14] If v is the highest node in G, we set $\mathrm{lp}(\mathrm{PS}(v)) := v$ to avoid notation issues. Further, the *threshold* of the pre-sink $\mathrm{PS}(v)$ is defined as

$$\mathrm{thr}(\mathrm{PS}(v)) := \sum_{v' \in \mathrm{PS}(v)} (\mathrm{gh}_v - \mathrm{gh}_{v'}) \cdot \mathrm{area}_{v'}$$

and the *capacity* of the pre-sink $\mathrm{PS}(v)$ is defined as

$$\mathrm{cap}(\mathrm{PS}(v)) := \sum_{v' \in \mathrm{PS}(v)} (\mathrm{gh}_{\mathrm{lp}(\mathrm{PS}(v))} - \mathrm{gh}_{v'}) \cdot \mathrm{area}_{v'}.$$

Intuitively, the threshold of a pre-sink is the maximum amount of water it can hold before it becomes flooded, and the capacity is the maximum amount of water it can hold before its water level matches the geodesic height of its lowest parent. We next show that a pre-sink is flooded if and only if the amount of rain on the pre-sink plus the inflow from uphill nodes, which is exactly the positive contribution to the excess of the node, exceeds its threshold.

Before we prove this characterization separately for x and y, we show a simpler characterization of when a pre-sink is flooded:

Observation 8 Let $v \in V$. Then, all nodes in $\mathrm{PS}(v)$ are flooded if and only if v is flooded.

Proof The forward direction is clear. The backward direction follows immediately from Proposition 1 for x and from Proposition 3 for y. □

Appendix 4.2.1. Characterization of Flooded Pre-sinks in *x*

We again start by presenting the characterization for x and omit the "x." when referring to variables whenever this does not lead to any confusion.

Definition 4 Let $v \in V$. The *positive contribution to the excess* of the pre-sink $\mathrm{PS}(v)$ in x is defined as

$$x.\mathrm{pce}_{\mathrm{PS}(v)} := \sum_{r \in \delta^-_{G^{\mathrm{ex}}}(\mathrm{PS}(v))} x.f_r + \sum_{v' \in \mathrm{PS}(v)} \mathrm{rain} \cdot \mathrm{area}_{v'}.$$

Next, we prove that any water that enters a non-flooded pre-sink remains in this pre-sink.

[14] Note that the lowest parent exists in this case as G is assumed to be weakly connected. In general, it does not hold that $\mathrm{lp}(\mathrm{PS}(v)) = \mathrm{lp}(v)$.

Lemma 4 *Let $v \in V$ be non-flooded. Then, it holds that*

$$\mathrm{pce}_{\mathrm{PS}(v)} = \sum_{v' \in \mathrm{PS}(v)} \mathrm{excess}_{v'} = \mathrm{excess}_{\mathrm{PS}(v)}.$$

Proof The latter equality is clear by definition. To prove the first equality, we reformulate the excess of the pre-sink:

$$\sum_{v' \in \mathrm{PS}(v)} \mathrm{excess}_{v'} = \sum_{v' \in \mathrm{PS}(v)} \left[\sum_{r \in \delta^-_{G^{\mathrm{ex}}}(v')} f_r - \sum_{r \in \delta^+_{G^{\mathrm{ex}}}(v')} f_r \right] + \sum_{v' \in \mathrm{PS}(v)} \mathrm{rain} \cdot \mathrm{area}_{v'}$$

We investigate the first sum and observe:

1. The flow on any arc $r \in R^{\mathrm{ex}}$ with $\alpha(r), \omega(r) \in \mathrm{PS}(v)$ appears exactly once in each of the two inner sums. Therefore, the flow on this arc does not contribute to the overall value of the sum.
2. The flow on ayn arc $r \in \delta^-_{G^{\mathrm{ex}}}(\mathrm{PS}(v))$ appears exactly once in the first inner sum.
3. We claim that any arc $r \in \delta^+_{G^{\mathrm{ex}}}(\mathrm{PS}(v))$ has $x.f_r = 0$. This holds since $\mathrm{gh}_{\omega(r)} > \mathrm{gh}_v$ (otherwise, $\omega(r) \in \mathrm{PS}(v)$) and $\mathrm{gh}_{\alpha(r)} + \mathrm{wl}_{\alpha(r)} \leq \mathrm{gh}_v$ (since v would be flooded due to Proposition 1 otherwise). Constraints (21) and (23) then force full_r to be zero and, hence, $f_r = 0$, which yields the desired result.
4. Any other arc does not appear in the sum at all.

Therefore, we obtain that

$$\sum_{v' \in \mathrm{PS}(v)} x.\mathrm{excess}_{v'} = \sum_{r \in \delta^-_{G^{\mathrm{ex}}}(\mathrm{PS}(v))} x.f_r + \sum_{v' \in \mathrm{PS}(v)} \mathrm{rain} \cdot \mathrm{area}_{v'} = x.\mathrm{pce}_{\mathrm{PS}(v)}.$$

□

Using this lemma, we can prove the first direction of the desired characterization of flooded pre-sinks:

Lemma 5 *Let $v \in V$. If $\mathrm{pce}_{\mathrm{PS}(v)} > \mathrm{thr}(\mathrm{PS}(v))$, then $\mathrm{PS}(v)$ is flooded.*

Proof Due to Observation 8, it suffices to show that v is flooded. For the sake of a contradiction, suppose v is not flooded. Then, it holds that

$$\begin{aligned}\sum_{v' \in \mathrm{PS}(v)} (\mathrm{gh}_v - \mathrm{gh}_{v'}) \cdot \mathrm{area}_{v'} &= \mathrm{thr}(\mathrm{PS}(v)) \\ &< \mathrm{pce}_{\mathrm{PS}(v)} \\ &\overset{\text{Lemma 4}}{=} \sum_{v' \in \mathrm{PS}(v)} \mathrm{excess}_{v'} \\ &= \sum_{v' \in \mathrm{PS}(v)} \mathrm{wl}_{v'} \cdot \mathrm{area}_{v'}.\end{aligned}$$

This means there exists a node $\hat{v} \in \mathrm{PS}(v)$ with $\mathrm{wl}_{\hat{v}} > \mathrm{gh}_v - \mathrm{gh}_{\hat{v}}$. Proposition 1 then implies that $\mathrm{wl}_v > 0$, which yields the desired contradiction. □

Before proving the other direction, we firstly observe that, for every node $v \in V$, it holds that $\mathrm{pce}_{\mathrm{PS}(v)} \geq \mathrm{excess}_{\mathrm{PS}(v)}$ since, by definition, $\mathrm{pce}_{\mathrm{PS}(v)}$ contains all positive summands from the definition of $\mathrm{excess}_{\mathrm{PS}(v)}$.

Lemma 6 *Let $v \in V$. If $\mathrm{pce}_{\mathrm{PS}(v)} \leq \mathrm{thr}(\mathrm{PS}(v))$, then $\mathrm{PS}(v)$ is not flooded.*

Proof Again, due to Observation 8, it suffices to show that v is not flooded. Similar to the proof of Lemma 5, it holds that

$$\begin{aligned}\sum_{v' \in \mathrm{PS}(v)} (\mathrm{gh}_v - \mathrm{gh}_{v'}) \cdot \mathrm{area}_{v'} &= \mathrm{thr}(\mathrm{PS}(v)) \\ &\geq \mathrm{pce}_{\mathrm{PS}(v)} \\ &\geq \sum_{v' \in \mathrm{PS}(v)} \mathrm{excess}_{v'} \\ &= \sum_{v' \in \mathrm{PS}(v)} \mathrm{wl}_{v'} \cdot \mathrm{area}_{v'}.\end{aligned}$$

For the sake of a contradiction, suppose that $\mathrm{wl}_v > 0$. Applying Proposition 1 for every $v' \in \mathrm{PS}(v)$ yields that $\mathrm{wl}_{v'} > \mathrm{gh}_v - \mathrm{gh}_{v'}$, which is a contradiction to the above inequality. □

Lemmas 5 and 6 finally enable us to prove the desired characterization of flooded pre-sinks for x:

Proposition 5 *Let $v \in V$. Then, $\mathrm{PS}(v)$ is flooded in x if and only if it holds that $x.\mathrm{pce}_{\mathrm{PS}(v)} > \mathrm{thr}(\mathrm{PS}(v))$.*

Appendix 4.2.2. Characterization of Flooded Pre-sinks in *y*

We now present the characterization for y and omit the "y." when referring to variables whenever this does not lead to any confusion. The proof, however, is remarkably more technical than for x. A major part of the proof involves showing that

$y.\text{excess}_{\text{PS}(v)} = \sum_{v' \in \text{PS}(v)} y.\text{wl}_{v'} \cdot \text{area}_{v'}$, which is the first milestone of this subsection. To this end, we start by showing that any two pre-sinks are either disjoint or one is contained in the other.

Lemma 7 *Let* $u, v \in V$ *with* $\text{gh}_u > \text{gh}_v$. *Then, either* $\text{PS}(v) \cap \text{PS}(u) = \emptyset$ *or* $\text{PS}(v) \subseteq \text{PS}(u)$.

Proof Let K be the set of nodes of the weakly connected component in $G_{\leq u}$ that contains v. Then, it immediately follows that $\text{PS}(v) \subseteq K$. If it holds that $\text{PS}(v) \cap \text{PS}(u) \neq \emptyset$, it also holds that $K \cap \text{PS}(u) \neq \emptyset$. It follows that $\text{PS}(u) = K \subseteq \text{PS}(v)$, which proves the claim. □

Next, we prove that, when the end node of an arc $r \in R$ is changed during the `joinNodes`-routine, its original end node is in the pre-sink of the new end node. To this end, we first introduce a notation that keeps track of changes of arcs during the algorithm.

Definition 5 For $t \in \{1, \dots, T\}$ and $r \in \tilde{R}_t$, we call the unique arc $r' \in R$ that r stems from the *original arc* of r and denote it by $\text{oa}(r)$. Conversely, we call r the *changed arc of r' at iteration t* and write $\text{ca}_t(r') = r$.

Note that $\text{ca}_t(r)$ is not necessarily defined for every arc $r \in R$ in every iteration $t \in \{1, \dots, T\}$. We next prove that, when the end node of an arc is changed, it stays within the same pre-sink.

Observation 9 Let $r \in R$ and $t \in \{1, \dots, T\}$ such that $\text{ca}_t(r)$ is defined. Then, $\omega(r) \in \text{PS}(\omega(\text{ca}_t(r)))$.

Proof For the sake of a contradiction, suppose that $\omega(r) \notin \text{PS}(\omega(\text{ca}_t(r)))$. Without loss of generality, we may assume that r is chosen such that $\alpha(r)$ has minimal geodesic height among all arcs with this property and that $\omega(r) \in \text{PS}(\omega(\text{ca}_{t-1}(r)))$.[15]

To enhance readability, we name the nodes as follows. The start node of the arcs $r, \text{ca}_t(r)$, and $\text{ca}_{t-1}(r)$ is called u.[16] The end nodes of $r, \text{ca}_t(r)$, and $\text{ca}_{t-1}(r)$ are called v, w, and w' respectively.

The second assumption implies that `joinNodes` w', w has been called in iteration $t-1$. Hence, there exists an arc $\hat{r} \in \tilde{R}_{t-1}$ from w to w'. This means that $\alpha(\text{oa}(\hat{r})) = w$. Due to the first assumption, the claim of the observation holds true for $\text{oa}(\hat{r})$, which implies that $\omega(\text{oa}(\hat{r})) \in \text{PS}(\omega(\text{ca}_t(\text{oa}(r)))) = \text{PS}(\omega(\hat{r})) = \text{PS}(w')$ for all $t \in \{1, \dots, T\}$. By definition of a pre-sink, it also holds that $\omega(\text{oa}(\hat{r})) \in \text{PS}(\alpha(\text{oa}(\hat{r}))) = \text{PS}(w)$. It, therefore, must hold that $\omega(\text{oa}(\hat{r})) \in \text{PS}(w') \cap \text{PS}(w)$. Lemma 7 implies that $\text{PS}(w') \subseteq \text{PS}(w)$. As $u \in \text{PS}(w')$, it then also holds that $\omega(r) = u \in \text{PS}(w) = \text{PS}(\omega(\text{ca}_t(r)))$, which is a contradiction. □

[15] Note that $t \geq 2$ as $\tilde{R}_1 = R$.

[16] Note that the start node of an arc is never changed in Algorithm 5, so it holds that $\alpha(r) = \alpha(\text{ca}_t(r)) = \alpha(\text{ca}_{t-1}(r))$.

This observation can be utilized to obtain a useful result about the nodes in the sets repres_v for $v \in V$ at the end of the algorithm.

Lemma 8 *Let $v \in V$ and $t \in \{1, \dots, T\}$. Then,* $\text{repres}_{v,T} \subseteq \text{PS}(v)$.

Proof We prove this by induction over the iterations. In the first iteration, the claim clearly holds true as $\text{repres}_{v,1} = \{v\} \subseteq \text{PS}(v)$ for all $v \in V$. Now let the claim hold in some iteration $t \in \{1, \dots, T-1\}$, i.e., $\text{repres}_{v,T} \subseteq \text{PS}(v)$ for all $v \in V$. Let $u, w \in V$ such that `joinNodes` u, w is called in iteration t. For any node $v \neq w$, the set $\text{repres}_{v,T}$ remains unchanged in iteration t, so the claim still holds in the next iteration $t+1$. Due to the update rule in the `joinNodes`-routine, it remains to show that $\text{repres}_{u,t} \subseteq \text{PS}(w)$. From the induction hypothesis, we already know that $\text{repres}_{u,t} \subseteq \text{PS}(u)$. Further, we know that $\text{gh}_u < \text{gh}_w$. Using Lemma 7, it remains to show that $\text{PS}(u) \cap \text{PS}(w) \neq \emptyset$. As `joinNodes` u, w is called in iteration t, there must be an arc $r \in \tilde{R}_t$ with $\alpha(r) = w$ and $\omega(r) = u$. Using Observation 9, we get that $\omega(\text{oa}(r)) \in \text{PS}(u)$. By definition of a pre-sink, it also holds that $\omega(\text{oa}(r)) \in \text{PS}(\alpha(r)) = \text{PS}(w)$. Hence, it holds that $\text{PS}(u) \cap \text{PS}(w) \neq \emptyset$, which completes the proof. □

If $v \notin \overline{V}$, we can even show a stronger statement.

Lemma 9 *Let $t \in \{1, \dots, T\}$ and $v \in V \setminus \tilde{V}_t$. Then,* $\text{repres}_{v,T} = \text{PS}(v)$.

Proof We already showed $\text{repres}_{v,T} \subseteq \text{PS}(v)$ in Lemma 8. For the other direction, let $w \in \text{PS}(v)$. This means there exists an undirected path with trace $(v, \tilde{w}_1, \dots, \tilde{w}_k, w)$ of nodes in $\text{PS}(v)$. As $v \notin \tilde{V}_t$, it holds that $\text{wl}_{v,t} > 0$. Applying Lemmas 2 and 3 inductively on this path yields $w \in \text{repres}_{v,T}$, which proves the claim. □

The proof immediately shows another result.

Corollary 3 *If $v \in V$ is a leaf in $\tilde{G}_t$ for some $t \in \{1, \dots, T\}$, then* $\text{repres}_{v,T} = \text{PS}(v)$.

We use this statement to show that, whenever two nodes are joined, the higher one is the lowest parent of the pre-sink induced by the lower one.

Observation 10 Let `joinNodes` u, v be called during the algorithm. Then, it holds that $v = \text{lp}(\text{PS}(u))$.

Proof We first prove that v is a parent of $\text{PS}(u)$. Let `joinNodes` u, v be called in iteration t. This means that there exists an arc $r \in \tilde{R}_t$ with $\alpha(r) = v$ and $\omega(r) = u$. Due to Observation 9, it holds that $\omega(\text{oa}(r)) \in \text{PS}(u)$. As the source node of an arc is never changed, it further holds that $\alpha(\text{oa}(r)) = v$, which means that v is a parent of $\text{PS}(u)$.

We conclude the proof by showing that v is the *lowest* parent of $\text{PS}(u)$. As `joinNodes` u, v is called in iteration t, it holds that $u \notin \overline{V}$. Lemma 9 then implies that $\text{repres}_u = \text{PS}(u)$. This means that, for every arc $r \in R$ with $\omega(r) \in \text{PS}(u)$, if $\text{ca}_t(r)$

exists, it holds that $\omega(\mathrm{ca}_t(r)) = u$. As v is by choice of the algorithm the lowest parent of u in $\tilde{G}_t$, it also is the lowest parent of $\mathrm{PS}(u)$ in G, which proves the claim. □

A further consequence of Lemma 9 is stated in the following corollary.

Corollary 4 *Let $v \in V \setminus \overline{V}$. Then,* $\mathrm{excess}_v = \sum_{v' \in \mathrm{PS}(v)} (\mathrm{gh}_{\mathrm{lp}(\mathrm{PS}(v))} - \mathrm{gh}_v) \cdot \mathrm{area}_{v'}$.

Proof Due to Lemma 9, it holds that $\mathrm{repres}_v = \mathrm{PS}(v)$. As $v \notin \overline{V}$, the node v must leave the graph in some iteration t. In order for the node to leave the graph, it must become the first flooded leaf, which only happens if

$$\mathrm{excess}_v / \mathrm{area}_{v,t} = \mathrm{gh}_{\mathrm{lp}(\mathrm{PS}(v))} - \mathrm{gh}_v.$$

Rearranging and plugging in $\mathrm{area}_{v,t} = \sum_{v' \in \mathrm{PS}(v)} \mathrm{area}_{v'}$ yields the desired result. □

We next investigate the structure of the sets repres_v for $v \in V$ in more detail. This requires a further definition, which will be particularly important in a later stage of the proof as well.

Lemma and Definition 2 *Let $v \in V$. Then, all weakly connected components of $G|_{\mathrm{PS}(v)\setminus\{v\}}$ are pre-sinks. We call a node u inducing such a pre-sink a follow-up node of v and denote the set of all follow-up nodes of v by* $\mathrm{FUN}(v)$.

Proof Let C be a weakly connected component of $G|_{\mathrm{PS}(v)\setminus\{v\}}$ and let u be the highest node in C. We show that $C = \mathrm{PS}(u)$.

Let $w \in C$. As C is weakly connected and u is the highest node in C, there exists an undirected path with trace $(u, \tilde{u}_1, \ldots, \tilde{u}_k, w)$ where all intermediate nodes are in C as well. This means that $w \in \mathrm{PS}(u)$.

Let $w \in \mathrm{PS}(u)$. This means there exists an undirected path P with trace $(u, \tilde{u}_1, \ldots, \tilde{u}_k, w)$ of nodes in $\mathrm{PS}(u)$. As $u \in \mathrm{PS}(v)$, it holds that $\mathrm{PS}(u) \subseteq \mathrm{PS}(v)$ due to Lemma 7, which implies that $\tilde{u}_i \in \mathrm{PS}(v)$ for all $i \in \{1, \ldots, k\}$. Further, as $\tilde{u}_i \in \mathrm{PS}(u)$, it holds that $\tilde{u}_i \neq v$ for all $i \in \{1, \ldots, k\}$. This implies P is an undirected path in $G|_{\mathrm{PS}(v)\setminus\{v\}}$, which means that $w \in C$. □

Using this definition, an advanced structural result about the sets repres_v for $v \in V$ can be shown.

Corollary 5 *Let $v \in V$ and $t \in \{1, \ldots, T\}$. Then, there exists a set $U \subseteq \mathrm{FUN}(v)$ such that* $\mathrm{repres}_{v,T} = \{v\} \cup \bigcup_{u \in U} \mathrm{PS}(u)$.

Proof Let $u \in \mathrm{FUN}(v)$. If $u \in \mathrm{repres}_{v,T}$, then this implies that $u \notin \tilde{V}_t$. Lemma 9 implies that $\mathrm{repres}_{u,t} = \mathrm{PS}(u)$. As $u \in \mathrm{repres}_{v,T}$, so must be all nodes in $\mathrm{repres}_{u,t}$, which implies $\mathrm{PS}(u) \subseteq \mathrm{repres}_{v,T}$.

If $u \notin \mathrm{repres}_{v,T}$, then Observation 10 implies that $u \in \tilde{V}_t$. This means that v cannot become the lowest parent of any of the nodes in $\mathrm{PS}(u)$, which implies that no node in $\mathrm{PS}(u)$ is in $\mathrm{repres}_{v,T}$. □

We use Corollaries 4 and 5 to prove the next result.

Lemma 10 *Let $v \in \overline{V}$ and $u \in \mathrm{FUN}(v)$ such that $u \in \mathrm{repres}_v$. Then*

$$\mathrm{excess}_{\mathrm{PS}(u)} = \sum_{v' \in \mathrm{PS}(u)} (\mathrm{gh}_v - \mathrm{gh}_{v'}) \cdot \mathrm{area}_{v'}.$$

Proof As $u \in \mathrm{repres}_v$, it also holds that $\mathrm{PS}(u) \subseteq \mathrm{repres}_v$ due to Corollary 5. This then implies that any node $w \in \mathrm{PS}(u)$ must leave the graph in some iteration, i.e., $w \notin \overline{V}$, which allows us to apply Corollary 4 for any node in $\mathrm{PS}(u)$:

$$\begin{aligned}
\mathrm{excess}_{\mathrm{PS}(u)} &= \sum_{v' \in \mathrm{PS}(u)} \mathrm{excess}_{v'} \\
&= \sum_{v' \in \mathrm{PS}(u)} \sum_{\tilde{v} \in \mathrm{PS}(v')} (\mathrm{gh}_{\mathrm{lp}(\mathrm{PS}(v'))} - \mathrm{gh}_{v'}) \cdot \mathrm{area}_{\tilde{v}} \\
&= \sum_{v' \in \mathrm{PS}(u)} \sum_{\tilde{v} \in \mathrm{repres}_{v'}} (\mathrm{gh}_{\mathrm{lp}(\mathrm{PS}(v'))} - \mathrm{gh}_{v'}) \cdot \mathrm{area}_{\tilde{v}}
\end{aligned}$$

Let $v_1, \ldots, v_k \in \mathrm{PS}(u)$ be the nodes in order of ascending geodesic heights such that $v' \in \mathrm{repres}_{v_i}$ for all $i \in \{1, \ldots, k\}$. This implies that $v_1 = v'$ and $v_k = u$. Further, Observation 10 implies that $v_{i+1} = \mathrm{lp}(\mathrm{PS}(v_i))$ for all $i \in \{1, \ldots, k-1\}$. The terms in the above sum involving $\mathrm{area}_{v'}$ can then be written as

$$\begin{aligned}
&[(\mathrm{gh}_{\mathrm{lp}(\mathrm{PS}(u))} - \mathrm{gh}_u) + (\mathrm{gh}_u - \mathrm{gh}(v_{k-1})) + \cdots + (\mathrm{gh}(v_2) - \mathrm{gh}_{v'})] \cdot \mathrm{area}_{v'} \\
=&(\mathrm{gh}_v - \mathrm{gh}_{v'}) \cdot \mathrm{area}_{v'}
\end{aligned}$$

Inserting this into the sum above yields the desired result. □

A similar idea can be used to prove the following lemma.

Lemma 11 *Let $v \in V \setminus \overline{V}$. Then, it holds that*

$$\mathrm{excess}_{\mathrm{PS}(v)} = \sum_{v' \in \mathrm{PS}(v)} (\mathrm{gh}_{\mathrm{lp}(\mathrm{PS}(v))} - \mathrm{gh}_{v'}) \cdot \mathrm{area}_{v'}.$$

Proof As $v \notin \overline{V}$, it must hold that $\mathrm{repres}_v = \mathrm{PS}(v)$ due to Lemma 9. This means that, for every $w \in \mathrm{PS}(v)$, it must holds that $w \notin \overline{V}$. The rest of the proof is then along the same lines as the proof of Lemma 10. □

We can now prove that, for $v \in \overline{V}$, the excess is obtained by summing up the product of the water level and the area over all nodes in $\mathrm{PS}(v)$.

Proposition 6 *Let $v \in \overline{V}$. Then, it holds that*

$$\mathrm{excess}_{\mathrm{PS}(v)} = \sum_{v' \in \mathrm{PS}(v)} \mathrm{wl}_{v'} \cdot \mathrm{area}_{v'}.$$

Proof For a non-flooded node $v' \in V$, it is clear that $\text{excess}_{v'} = 0$. Due to Proposition 4 and Lemma 7, any sink $S \in \mathcal{S}(y)$ is either contained in $\text{PS}(v)$ or disjoint from $\text{PS}(v)$. Let $S_1, \dots, S_k \in \mathcal{S}(y)$ be the sinks contained in $\text{PS}(v)$ and let $u_1, \dots, u_k \in V$ be the nodes inducing their corresponding pre-sinks respectively. Then, we can write the excess as

$$\text{excess}_{\text{PS}(v)} = \sum_{i=1}^{k} \text{excess}_{\text{PS}(u_i)} \tag{3}$$

We fix some $i \in \{1, \dots, k\}$ and distinguish two cases.

Case 1: $u_i \in \overline{V}$ As u_i is in a sink, it must hold that $\text{wl}_{u_i} > 0$, which means that u_i is a leaf in $\overline{G}$. Corollary 3 then yields that $\text{repres}_{u_i} = \text{PS}(u_i)$. For any follow-up node $u' \in \text{FUN}(u_i)$, Lemma 10 yields

$$\text{excess}_{\text{PS}(u')} = \sum_{v' \in \text{PS}(u)} (\text{gh}_{u_i} - \text{gh}_{v'}) \cdot \text{area}_{v'}.$$

Further, as $\text{repres}_{u_i} = \text{PS}(u_i)$, we get that

$$\text{excess}_{u_i} = \text{area}_{\text{PS}(u_i)} \cdot \text{wl}_{u_i} = \sum_{v' \in \text{PS}(u_i)} \text{area}_{v'} \cdot \text{wl}_{u_i}.$$

This yields

$$\begin{aligned}
\text{excess}_{\text{PS}(u_i)} &= \sum_{u' \in \text{FUN}(u_i)} \sum_{v' \in \text{PS}(u')} (\text{gh}_{u_i} - \text{gh}_{v'}) \cdot \text{area}_{v'} \\
&\quad + \sum_{v' \in \text{PS}(u_i)} \text{area}_{v'} \cdot \text{wl}_{u_i} \\
&= \sum_{v' \in \text{PS}(u_i)} (\text{wl}_{u_i} + \text{gh}_{u_i} - \text{gh}_{v'}) \cdot \text{area}_{v'} \\
&\overset{\text{Prop. 3}}{=} \sum_{v' \in \text{PS}(u_i)} \text{wl}_{v'} \cdot \text{area}_{v'}.
\end{aligned}$$

Case 2: $u_i \notin \overline{V}$ Lemma 11 yields that

$$\text{excess}_{\text{PS}(u_i)} = \sum_{v' \in \text{PS}(u_i)} (\text{gh}_{\text{lp}(\text{PS}(u_i))} - \text{gh}_{v'}) \cdot \text{area}_{v'}.$$

Let $\tilde{u} = \text{lp}(\text{PS}(u_i))$. As $\tilde{u} \notin \text{PS}(u_i) = S_i$, it must hold that $\text{wl}(\tilde{u}) = 0$ and, hence, that $\tilde{u} \in \overline{V}$. As $u_i \notin \overline{V}$, it must hold that $u_i \in \text{repres}(\tilde{u})$. Observation 3 then yields that

$$\text{gh}_{\text{lp}(\text{PS}(u_i))} - \text{gh}_{v'} = \text{wl}_{v'},$$

which then implies

$$\text{excess}_{\text{PS}(u_i)} = \sum_{v' \in \text{PS}(u_i)} \text{wl}_{v'} \cdot \text{area}_{v'},$$

which is the same result as in the previous case. Resubstituting this into (3) yields

$$\text{excess}_{\text{PS}(v)} = \sum_{v' \in \text{PS}(v)} \text{wl}_{v'} \cdot \text{area}_{v'}.$$

□

Now that we have reached this milestone, we next prove the analogue of Proposition 5 in the context of Algorithm 6. To this end, we firstly define the positive contribution to the excess.

Definition 6 Let $v \in V$ and $t \in \{1, \dots, T\}$. We define the *positive contribution to the excess* of pre-sink $\text{PS}(v)$ in iteration t as

$$y.\text{pce}_{\text{PS}(v),t} := \sum_{\substack{r \in \delta_G^-(\text{PS}(v)): \\ \text{ca}_t(r) \text{ exists}}} f_{\text{ca}_t(r),t} + \sum_{v' \in \text{PS}(v)} \text{area}_{v'} \cdot \text{rain} \cdot \hat{p}_t$$

and the *positive contribution to the excess of pre-sink* $\text{PS}(v)$ *until iteration* t as

$$y.\text{pce}_{\text{PS}(v),\leq t} := \sum_{t'=1}^{t} y.\text{pce}_{\text{PS}(v),t'}.$$

As a short-hand notation, we define $y.\text{pce}_{\text{PS}(v)} := y.\text{pce}_{\text{PS}(v),\leq T}$. Further, for $t \in \{1, \dots, T\}$, the *change of excess* in iteration t is defined as $\Delta\text{excess}_{v,t} := \text{excess}_{v,t} - \text{excess}_{v,t-1}$, where $\text{excess}_{v,0} = 0$ for all $v \in V$.

It is worth noting that we will introduce a similar definition for the MIP solution x later. To avoid confusion, we keep the "y." in the central results. For smaller technical results, however, we omit the "y." whenever it is clear from the context.

We start by proving a structural result, whose statement is similar to the one of Observation 9.

Lemma 12 *Let $t \in \{1, \dots, T\}$ and $v \in \tilde{V}_t$. Further let $r \in R$ such that $\alpha(r) \notin \text{PS}(v)$ and $\omega(r) \in \text{PS}(v)$. Then, $\text{ca}_{t'}(r)$ exists for every $t' \leq t$ and $\omega(\text{ca}_{t'}(r)) \in \text{PS}(v)$.*

Proof We start by proving that $\text{ca}_{t'}(r)$ exists for every $t' \leq t$. Suppose for the sake of a contradiction that this is not the case. Then, there exists a last iteration $\hat{t} \leq t$, in which $\text{ca}_{\hat{t}}(r)$ exists. This means that `joinNodes` $\omega(\text{ca}_{\hat{t}}(r)), \alpha(r)$ is called in iteration $\hat{t}$. As $\alpha(r) \notin \text{PS}(v)$, it must hold that $\text{gh}_{\alpha(r)} > \text{gh}_v$. Further, as $\alpha(r)$ is a parent of $\text{PS}(v)$, it holds that $v \in \text{PS}(\alpha(r))$. As $\omega(\text{ca}_{\hat{t}}(r)) \in \text{repres}_{\alpha(r),t}$, it also holds that $v \in \text{repres}_{\alpha(r),t}$ due to Corollary 5. This, however, implies that $v \notin \tilde{V}_t$, which is a contradiction.

We proceed by proving the second claim by contradiction. So suppose there exists an iteration $t' \leq t$ such that $\omega(\text{ca}_{t'-1}(r)) \in \text{PS}(v)$ and $\omega(\text{ca}_{t'}(r)) \notin \text{PS}(v)$. This means that `joinNodes` $\omega(\text{ca}_{t'-1}(r)), \omega(\text{ca}_{t'}(r))$ has been called in iteration $t'-1$. Hence, there exists an arc $r' \in \tilde{R}_{t'-1}$ from $\omega(\text{ca}_{t'}(r))$ to $\omega(\text{ca}_{t'-1}(r))$. Let $\hat{r} = \text{oa}(r')$. Clearly, it holds that $\alpha(\hat{r}) = \omega(\text{ca}_{t'}(r))$. Due to Observation 9, it holds that $\omega(\text{ca}_{t'-1}(r)) \in \text{PS}(\omega(\text{ca}_{t'}(r)))$. Hence, $\text{PS}(\omega(\text{ca}_{t'}(r))) \cap \text{PS}(v) \neq \emptyset$, which implies that $\text{PS}(\omega(\text{ca}_{t'}(r))) \subseteq \text{PS}(v)$ due to Lemma 7. In particular, this shows that $\omega(\text{ca}_{t'}(r)) \in \text{PS}(v)$. □

The lemma above allows proving that, as long as a node $v \in V$ has not left the graph, the change of the excess of its induced pre-sink in an interation t is exactly the positive contribution to the excess of the pre-sink in iteration t.

Lemma 13 *Let $t \in \{1, \dots, T\}$ and $v \in \tilde{V}_t$. Then, it holds that* $\text{pce}_{\text{PS}(v),t} = \Delta\text{excess}_{\text{PS}(v),t}$.

Proof Firstly, note that any water that enters a node $v' \in \text{PS}(v)$ or that arises from the rain on v' is sent over the outgoing arcs of v' in $\tilde{G}_t$ if there are any. Observation 9 guarantees that the child of such an arc is itself in $\text{PS}(v')$ and, hence, in $\text{PS}(v)$. Therefore, the water that arrives at the leaves of $\tilde{G}_t$ is exactly the inflow into the pre-sink plus the rain on the pre-sink.

The rain on the pre-sink is exactly $\sum_{v' \in \text{PS}(v)} \text{area}_{v'} \cdot \text{rain} \cdot \hat{p}_t$ and the inflow into the pre sink $\delta^-_{\tilde{G}_t}(\text{PS}(v)) = \{r \in \delta^-_G(\text{PS}(v)) | \text{ca}_t(r) \text{ exists}\}$ due to Lemma 12 and Observation 9, which proves the claim. □

There are two useful corollaries that directly follow from this.

Corollary 6 *Let $v \in V$ and $t \in \{1, \dots, T\}$ such that $v \in \tilde{V}_t$. Then, it holds that* $\text{pce}_{\text{PS}(v),\leq t} = \text{excess}_{\text{PS}(v),t}$.

Proof Sum over the iterations from one to t and use Lemma 13. □

Corollary 7 *Let $v \in V$. Then, it holds that* $\text{pce}_{\text{PS}(v)} \geq \text{excess}_{\text{PS}(v)}$.

We can now prove the characterization of a pre-sink being flooded.

Lemma 14 *Let $v \in V$. If* $\text{pce}_v > \text{thr}(\text{PS}(v))$, *then* $\text{PS}(v)$ *is flooded.*

Proof As in the context of the MIP, it suffices to show that v is flooded. For the sake of a contradiction, suppose that v is not flooded. Particularly, this means that $v \in \overline{V}$. Corollary 6 then implies that

$$\sum_{v' \in \mathrm{PS}(v)} (\mathrm{gh}_v - \mathrm{gh}_{v'}) \cdot \mathrm{area}_{v'} = \mathrm{thr}(\mathrm{PS}(v)) < \mathrm{pce}_{\mathrm{PS}(v)} = \mathrm{excess}_{\mathrm{PS}(v)} \overset{\text{Prop. 6}}{=} \sum_{v' \in \mathrm{PS}(v)} \mathrm{wl}_{v'} \cdot \mathrm{area}_{v'} \tag{4}$$

This means that there exists a node $v' \in \mathrm{PS}(v)$ with $\mathrm{wl}_{v'} > (\mathrm{gh}_v - \mathrm{gh}_{v'})$. Then, Proposition 3 implies that $\mathrm{wl}_v > 0$. This however directly implies that for all $\tilde{v} \in \mathrm{PS}(v)$, it holds that $\mathrm{wl}_{\tilde{v}} > (\mathrm{gh}_v - \mathrm{gh}_{\tilde{v}})$, which is a contradiction to (4). □

Next, the other direction is shown.

Lemma 15 *Let $v \in V$. If $\mathrm{pce}_v \leq \mathrm{thr}(\mathrm{PS}(v))$, then $\mathrm{PS}(v)$ is not flooded.*

Proof For the sake of a contradiction, suppose that v is flooded. We distinguish two cases:

Case 1: $v \notin \overline{V}$
Due to Lemma 11, it holds that

$$\mathrm{excess}_{\mathrm{PS}(v)} = \sum_{v' \in \mathrm{PS}(v)} (\mathrm{gh}_{\mathrm{lp}(\mathrm{PS}(v))} - \mathrm{gh}_{v'}) \cdot \mathrm{area}_{v'}.$$

Corollary 7 then implies that

$$\sum_{v' \in \mathrm{PS}(v)} (\mathrm{gh}_v - \mathrm{gh}_{v'}) \cdot \mathrm{area}_{v'} = \mathrm{thr}(\mathrm{PS}(v)) \geq \mathrm{pce}_{\mathrm{PS}(v)} \geq \mathrm{excess}_{\mathrm{PS}(v)} = \sum_{v' \in \mathrm{PS}(v)} (\mathrm{gh}_{\mathrm{lp}(\mathrm{PS}(v))} - \mathrm{gh}_{v'}) \cdot \mathrm{area}_{v'}.$$

This, however, is a contradiction to $\mathrm{gh}_v < \mathrm{gh}_{\mathrm{lp}(\mathrm{PS}(v))}$.
Case 2: $v \in \overline{V}$ Proposition 6 yields

$$\mathrm{excess}_{\mathrm{PS}(v)} = \sum_{v' \in \mathrm{PS}(v)} \mathrm{wl}_{v'} \cdot \mathrm{area}_{v'}.$$

Using Corollary 7, we get that

$$\sum_{v' \in \mathrm{PS}(v)} (\mathrm{gh}_v - \mathrm{gh}_{v'}) \cdot \mathrm{area}_{v'} = \mathrm{thr}(\mathrm{PS}(v)) \geq \mathrm{pce}_{\mathrm{PS}(v)} \geq \mathrm{excess}_{\mathrm{PS}(v)} = \sum_{v' \in \mathrm{PS}(v)} \mathrm{wl}_{v'} \cdot \mathrm{area}_{v'}.$$

Proposition 3, however, states that $\mathrm{gh}_{v'} + \mathrm{wl}_{v'} = \mathrm{gh}_v + \mathrm{wl}_v > \mathrm{gh}_v$ for each $v' \in \mathrm{PS}(v)$, which is a contradiction. □

As a result, we finally obtain the desired characterization of flooded pre-sinks for y:

Proposition 7 *Let $v \in V$. Then, $\mathrm{PS}(v)$ is flooded in y if and only if $y.\mathrm{pce}_v > \mathrm{thr}(\mathrm{PS}(v))$.*

Appendix 4.3. Characterization of Inflows and Outflows of Pre-sinks

Propositions 5 and 7 imply that, if $x.\mathrm{pce}_v = y.\mathrm{pce}_v$ for a node $v \in V$, then $\mathrm{PS}(v)$ is flooded in x if and only if it is flooded in y. The fact that the positive contribution to the excess highly depends on the flows in the graph motivates to investigate the flows in more detail. In this section, we present a characterization of the inflows and outflows of pre-sinks.

Appendix 4.3.1. Characterization of Inflows and Outflows of Pre-sinks for *x*

We start by presenting the characterization for x and omit the "x." when referring to variables whenever this does not lead to any confusion.

Definition 7 Let $S \in \mathcal{S}(x)$. The sink S is called *filled to capacity (ftc)* if $\mathrm{excess}_S = \mathrm{cap}(S)$, and it is called *backfloating* if

$$\sum_{\substack{r \in R^{\mathrm{ex}}: \\ \alpha(r) \in S, \\ \omega(r) = \mathrm{lp}(S)}} x.f_r > \sum_{\substack{r \in R^{\mathrm{ex}}: \\ \alpha(r) = \mathrm{lp}(S), \\ \omega(r) \in S}} x.f_r.$$

In this case, the sink's *backfloat* is defined as

$$x.\mathrm{bf}_S := \sum_{\substack{r \in R^{\mathrm{ex}}: \\ \alpha(r) \in S, \\ \omega(r) = \mathrm{lp}(S)}} x.f_r - \sum_{\substack{r \in R^{\mathrm{ex}}: \\ \alpha(r) = \mathrm{lp}(S), \\ \omega(r) \in S}} x.f_r.$$

The following lemma provides an alternative characterization of a pre-sink being ftc.

Lemma 16 *Let $S \in \mathcal{S}(x)$. Then S is ftc if and only if $\mathrm{wl}_v + \mathrm{gh}_v = \mathrm{gh}_{\mathrm{lp}(S)}$ for all $v \in S$.*

Proof Let $S \in \mathcal{S}(x)$ be ftc. Using Constraint (2) together with the definition of $\mathrm{cap}(S)$, this is equivalent to

$$\sum_{v \in S} \mathrm{wl}_v \cdot \mathrm{area}_v = \sum_{v \in S} (\mathrm{gh}_{\mathrm{lp}(S)} - \mathrm{gh}_v) \cdot \mathrm{area}_v.$$

Proposition 1 further implies that $\mathrm{wl}_v \leq \mathrm{gh}_{\mathrm{lp}(S)} - \mathrm{gh}_v$ for all $v \in S$ since otherwise $\mathrm{wl}_{\mathrm{lp}(S)} > 0$, which is a contradiction to $\mathrm{lp}(S) \notin S$. This means that the above equation holds if and only if $\mathrm{wl}_v + \mathrm{gh}_v = \mathrm{gh}_{\mathrm{lp}(S)}$ for all $v \in S$. □

Corollary 8 *Let $S \in \mathcal{S}(x)$ be non-ftc. Then, $\mathrm{wl}_v + \mathrm{gh}_v < \mathrm{gh}_{\mathrm{lp}(S)}$ for all $v \in S$.*

Proof Use Lemma 16 and Corollary 1. □

These two statements allow investigating the flows on arcs leading into a sink in more detail.

Corollary 9 *Let $S \in \mathcal{S}(x)$ be ftc. Then, every arc $r \in \delta_G^-(S) \cap \delta_G^+(\mathrm{lp}(S))$ is full and every arc $r \in \delta_G^-(S) \setminus \delta_G^+(\mathrm{lp}(S))$ is not full.*

Proof Due to Lemma 16, it holds that $\mathrm{wl}_v + \mathrm{gh}_v = \mathrm{gh}_{\mathrm{lp}(S)}$ for all $v \in S$. Let now $r \in \delta_G^-(S) \cap \delta_G^+(\mathrm{lp}(S))$, then r is full due to Constraints (22) and (24). For an arc $r \in \delta_G^-(S) \setminus \delta_G^+(\mathrm{lp}(S))$, it must hold that $\mathrm{gh}_{\alpha(r)} > \mathrm{gh}_{\mathrm{lp}(S)} = \mathrm{wl}_v + \mathrm{gh}_v$. Hence, Constraint (36) implies that r is not full. □

Corollary 10 *Let $S \in \mathcal{S}(x)$ be non-ftc. Then, no arc $r \in \delta_G^-(S)$ is full.*

Proof Use Corollary 8 and Constraint (36). □

Next, we provide a characterization of a sink being backfloating. The idea is that, if the rain on the sink plus the flow on incoming arcs from nodes that are not the lowest parent of the sink is larger than the sink's capacity, then the sink is backfloating. To this end, we split the positive contribution to the excess into two parts.

Definition 8 Let $v \in V$. We define the *positive contribution to the excess from non-lowest parents* of pre-sink $\mathrm{PS}(v)$ as

$$x.\mathrm{pcenlp}_{\mathrm{PS}(v)} := \sum_{\substack{r \in \delta^-_{G^{\mathrm{ex}}}(\mathrm{PS}(v)):\\ \alpha(r) \neq \mathrm{lp}(\mathrm{PS}(v))}} x.f_r + \sum_{v' \in \mathrm{PS}(v)} \mathrm{rain} \cdot \mathrm{area}_{v'}$$

and the *positive contribution to the excess from the lowest parent* of pre-sink $\mathrm{PS}(v)$ as

$$x.\mathrm{pcelp}_{\mathrm{PS}(v)} := \sum_{\substack{r \in \delta^-_{G^{\mathrm{ex}}}(\mathrm{PS}(v)):\\ \alpha(r) = \mathrm{lp}(\mathrm{PS}(v))}} x.f_r.$$

We firstly observe the following:

Observation 11 Let $S \in \mathcal{S}(x)$ be backfloating, then is is also ftc.

Proof Suppose S is not ftc. Then, due to Corollary 10, all incoming arcs into S are not full, which means that

$$\sum_{\substack{r\in R^{\text{ex}}:\\ \alpha(r)\in S,\\ \omega(r)=\text{lp}(S)}} x.f_r = 0.$$

Hence, S cannot be backfloating. □

We start proving the first direction of our characterization described above.

Lemma 17 *Let $S \in \mathcal{S}(x)$ be backfloating. Then, it holds that $x.\text{pcenlp}_S > \text{cap}(S)$ and $x.\text{bf}_S = x.\text{pcenlp}_S - \text{cap}(S)$.*

Proof Due to Observation 11, it holds that S is ftc. Furthermore, Corollary 9 implies that every arc $r \in \delta_G^-(S) \cap \delta_G^+(\text{lp}(S))$ is full and every arc $r \in \delta_G^-(S) \setminus \delta_G^+(\text{lp}(S))$ is not full, which implies that:

$$\begin{aligned}
\text{cap}(C) &= \text{Excess}_S \\
&= \sum_{v'\in\text{PS}(v)} \left[\sum_{r\in\delta^-_{G\text{ex}}(v')} x.f_r - \sum_{r\in\delta^+_{G\text{ex}}(v')} x.f_r \right] + \sum_{v'\in\text{PS}(v)} \text{rain}\cdot\text{area}_{v'} \\
&= \sum_{r\in\delta^-_{G\text{ex}}(S)} x.f_r + \sum_{v'\in\text{PS}(v)} \text{rain}\cdot\text{area}_{v'} - \sum_{r\in\delta^+_{G\text{ex}}(S)} x.f_r \\
&\overset{\text{Cor. 9}}{=} \sum_{\substack{r\in\delta^-_{G\text{ex}}(S):\\ \alpha(r)\neq\text{lp}(S)}} x.f_r + \sum_{v'\in\text{PS}(v)} \text{rain}\cdot\text{area}_{v'} \\
&\quad + \sum_{\substack{r\in\delta^-_{G\text{ex}}(S):\\ \alpha(r)=\text{lp}(S)}} x.f_r - \sum_{\substack{r\in\delta^+_{G\text{ex}}(S):\\ \omega(r)=\text{lp}(S)}} x.f_r \\
&= x.\text{pcenlp}_S + \sum_{\substack{r\in\delta^-_{G\text{ex}}(S):\\ \alpha(r)=\text{lp}(S)}} x.f_r - \sum_{\substack{r\in\delta^+_{G\text{ex}}(S):\\ \omega(r)=\text{lp}(S)}} x.f_r
\end{aligned}$$

Using that

$$\sum_{\substack{r\in\delta^-_{G\text{ex}}(S):\\ \alpha(r)=\text{lp}(S)}} x.f_r - \sum_{\substack{r\in\delta^+_{G\text{ex}}(S):\\ \omega(r)=\text{lp}(S)}} x.f_r < 0,$$

it follows that $\text{cap}(S) < \text{pcenlp}_S$ and that $\text{cap}(S) = \text{pcenlp}_S + \text{bf}_S$, which concludes the proof. □

The following corollary is a direct consequence of this proof:

Corollary 11 *Let $S \in \mathcal{S}(x)$ be ftc. Then, it holds that*

$$\sum_{\substack{r\in\delta^-_{G\text{ex}}(S):\\ \alpha(r)=\text{lp}(S)}} x.f_r - \sum_{\substack{r\in\delta^+_{G\text{ex}}(S):\\ \omega(r)=\text{lp}(S)}} x.f_r = \text{cap}(S) - x.\text{pcenlp}_S.$$

Next, the opposite direction is shown.

Lemma 18 *Let $S \in \mathcal{S}(x)$ with $x.\text{pcenlp}_S > \text{cap}(S)$. Then S is backfloating with $x.\text{bf}_S = x.\text{pcenlp}_S - \text{cap}(S)$.*

Proof We first prove that S is ftc. Suppose this was not the case, then, by Corollary 10, no arc $r \in \delta_G^-(S)$ is full. Hence, it holds that

$$\begin{aligned}\text{excess}_S &= \sum_{v' \in \text{PS}(v)} \left[\sum_{r \in \delta^-_{G^{\text{ex}}}(v')} x.f_r - \sum_{r \in \delta^+_{G^{\text{ex}}}(v')} x.f_r \right] \\ &\quad + \sum_{v' \in \text{PS}(v)} \text{rain} \cdot \text{area}_{v'} \\ &= \sum_{r \in \delta^-_{G^{\text{ex}}}(S)} x.f_r + \sum_{v' \in \text{PS}(v)} \text{rain} \cdot \text{area}_{v'} - \underbrace{\sum_{r \in \delta^+_{G^{\text{ex}}}(S)} x.f_r}_{=0} \\ &= x.\text{pcenlp}_S + x.\text{pcelp}_S > \text{cap}(S),\end{aligned}$$

which is a contradiction.

By the same arguments as in the proof of Lemma 17, the desired result is obtained. □

The two lemmas are summarized in the following proposition:

Proposition 8 *Let $S \in \mathcal{S}(x)$. Then, S is backfloating if and only if $x.\text{pcenlp}_S > \text{cap}(S)$. In this case, it holds that $x.\text{bf}_S = x.\text{pcenlp}_S - \text{cap}(S)$.*

Appendix 4.3.2. Characterization of Inflows and Outflows of Pre-sinks for *y*

We proceed by presenting an analogue characterization for y and omit the "y." when referring to variables whenever this does not lead to any confusion.

Definition 9 Let $S \in \mathcal{S}(y)$. We call *S filled to capacity (ftc)* if $\text{excess}_S = \text{cap}(S)$. Furthermore, let $v \in V$ such that $S = \text{PS}(v)$ and $v \notin \overline{V}$. We then call *S backfloating* if

$$\begin{aligned}&\sum_{t > t_v} \sum_{\substack{r \in \delta_G^-(\text{PS}(v)): \\ \text{ca}_t(r) \text{ exists}, \\ \alpha(r) \neq \text{lp}(\text{PS}(v))}} f_{\text{ca}_t(r),t} + \sum_{v' \in \text{PS}(v)} \text{area}_{v'} \cdot \text{rain} \cdot (1 - \text{sp}_{t_v}) \\ &> \sum_{t=1}^{T} \sum_{\substack{r \in \delta_G^-(\text{PS}(v)): \\ \text{ca}_t(r) \text{ exists}, \\ \alpha(r) = \text{lp}(\text{PS}(v))}} f_{\text{ca}_t(r),t}.\end{aligned}$$

In this case, the sink's *backfloat* is defined by

$$\begin{aligned}\mathrm{bf}_S := &\sum_{t>t_v} \sum_{\substack{r\in\delta_G^-(\mathrm{PS}(v)):\\ \mathrm{ca}_t(r)\text{ exists},\\ \alpha(r)\neq \mathrm{lp}(\mathrm{PS}(v))}} f_{\mathrm{ca}_t(r),t} + \sum_{v'\in\mathrm{PS}(v)} \mathrm{area}_{v'}\cdot \mathrm{rain}\cdot(1-\mathrm{sp}_{t_v})\\ &- \sum_{t=1}^{T} \sum_{\substack{r\in\delta_G^-(\mathrm{PS}(v)):\\ \mathrm{ca}_t(r)\text{ exists},\\ \alpha(r)=\mathrm{lp}(\mathrm{PS}(v))}} f_{\mathrm{ca}_t(r),t}.\end{aligned}$$

In contrast to the MIP, only one direction of the characterization of a sink being ftc is needed for the proof. It is worth noting that the other direction holds as well.

Lemma 19 *Let $S \in \mathcal{S}(y)$ be non-ftc. Then, for all $v \in S$, it holds that* $\mathrm{wl}_v + \mathrm{gh}_v < \mathrm{gh}_{\mathrm{lp}(S)}$.

Proof Clearly, it cannot hold that $\mathrm{wl}_v + \mathrm{gh}_v > \mathrm{gh}_{\mathrm{lp}(S)}$ as, in this case, due to Proposition 3, it would hold that $\mathrm{wl}_{\mathrm{lp}(S)} > 0$ and, hence, that $\mathrm{lp}(S) \in S$, which is clearly a contradiction. So suppose that $\mathrm{wl}_{\tilde{v}} + \mathrm{gh}_{\tilde{v}} = \mathrm{gh}_{\mathrm{lp}(S)}$ holds for some $\tilde{v} \in S$. Again, Proposition 3 implies that $\mathrm{wl}_{v'} + \mathrm{gh}_{v'} = \mathrm{gh}_{\mathrm{lp}(S)}$ holds for all $v' \in S$. Let $v \in V$ such that $S = \mathrm{PS}(v)$. We then distinguish two cases.

Case 1: $v \in \overline{V}$
We apply Proposition 6 and obtain

$$\begin{aligned}\mathrm{excess}_S &= \sum_{v'\in\mathrm{PS}(v)} \mathrm{wl}_{v'}\cdot\mathrm{area}_{v'}\\ &= \sum_{v'\in\mathrm{PS}(v)} (\mathrm{gh}_{\mathrm{lp}(S)} - \mathrm{gh}_{v'})\cdot\mathrm{area}_{v'}\\ &= \mathrm{cap}(S),\end{aligned}$$

which is a contradiction to S being non-ftc.
Case 2: $v \notin \overline{V}$
We apply Lemma 11 and obtain

$$\mathrm{excess}_S = \sum_{v'\in\mathrm{PS}(v)} (\mathrm{gh}_{\mathrm{lp}(\mathrm{PS}(v))} - \mathrm{gh}_{v'})\cdot\mathrm{area}_{v'} = \mathrm{cap}(S),$$

which, again, is a contradiction to S being non-ftc. □

We divide the positive contribution to the excess as before.

Definition 10 Let $v \in V$ and $t \in \{1, \ldots, T\}$. We define the *positive contribution to the excess from non-lowest parents* of pre-sink $\mathrm{PS}(v)$ until iteration t as

$$y.\mathrm{pcenlp}_{\mathrm{PS}(v),T} := \sum_{t'=1}^{t} \sum_{\substack{r\in\delta_G^-(\mathrm{PS}(v)):\\ \mathrm{ca}_{t'}(r)\text{ exists},\\ \alpha(r)\neq\mathrm{lp}(\mathrm{PS}(v))}} y.f_{\mathrm{ca}_t(r),t'} + \sum_{v'\in\mathrm{PS}(v)} \mathrm{area}_{v'}\cdot\mathrm{rain}\cdot\mathrm{sp}_t$$

and the *positive contribution to the excess from the lowest parent* of pre-sink $\mathrm{PS}(v)$ as

$$y.\mathrm{pcelp}_{\mathrm{PS}(v),T} := \sum_{t'=1}^{t} \sum_{\substack{r\in\delta_G^-(\mathrm{PS}(v)):\\ \mathrm{ca}_{t'}(r)\text{ exists},\\ \alpha(r)=\mathrm{lp}(\mathrm{PS}(v))}} y.f_{\mathrm{ca}_t(r),t'}.$$

Furthermore, we introduce the short-hand notations $\mathrm{pcenlp}_{\mathrm{PS}(v)} := \mathrm{pcenlp}_{\mathrm{PS}(v),T}$ and $\mathrm{pcelp}_{\mathrm{PS}(v)} := \mathrm{pcelp}_{\mathrm{PS}(v),T}$

This allows proving the following observation.

Observation 12 Let $S \in \mathcal{S}(y)$ be backfloating, then it is also ftc.

Proof Suppose this is not the case, then Lemma 19 implies that, for all $v' \in S$, it holds that $\mathrm{wl}_{v'} + \mathrm{gh}_{v'} < \mathrm{gh}_{\mathrm{lp}(S)}$. In particular, for $v \in V$ such that $S = \mathrm{PS}(v)$, this means that $v \in \overline{V}$, which means that S is not backfloating.

Using this observation, we prove the first direction of our characterization of a sink being backfloating.

Lemma 20 *Let $S \in \mathcal{S}(y)$ be backfloating. Then, it holds that* $\mathrm{pcenlp}_S > \mathrm{cap}(S)$ *and that* $\mathrm{bf}_S = \mathrm{pcenlp}_S - \mathrm{cap}(S)$.

Proof Let $v \in V$ such that $S = \mathrm{PS}(v)$. As S is backfloating, it must hold that $v \notin \overline{V}$ and, due to the above observation, it must also hold that $\mathrm{cap}(S) = \mathrm{excess}_S$. We then argue that

$$\begin{aligned}
\mathrm{cap}(S) &= \mathrm{excess}_S \\
&\overset{\text{Cor. 6}}{=} \mathrm{pce}_{S,\le t_v} = \mathrm{pcenlp}_{S,t_v} + \mathrm{pcelp}_{S,t_v} \\
&= \sum_{t'\le t_v} \sum_{\substack{r\in\delta_G^-(\mathrm{PS}(v)):\\ \mathrm{ca}_{t'}(r)\text{ exists},\\ \alpha(r)\neq\mathrm{lp}(\mathrm{PS}(v))}} f_{\mathrm{ca}_t(r),t'} + \sum_{v'\in\mathrm{PS}(v)} \mathrm{area}_{v'}\cdot\mathrm{rain}\cdot\mathrm{sp}_{t_v} \\
&\quad + \sum_{t'\le t_v} \sum_{\substack{r\in\delta_G^-(\mathrm{PS}(v)):\\ \mathrm{ca}_{t'}(r)\text{ exists},\\ \alpha(r)=\mathrm{lp}(\mathrm{PS}(v))}} f_{\mathrm{ca}_t(r),t'} + \underbrace{\sum_{t'>t_v} \sum_{\substack{r\in\delta_G^-(\mathrm{PS}(v)):\\ \mathrm{ca}_{t'}(r)\text{ exists},\\ \alpha(r)=\mathrm{lp}(\mathrm{PS}(v))}} f_{\mathrm{ca}_t(r),t'}}_{=0}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{t' \le t_v} \sum_{\substack{r \in \delta_G^-(\mathrm{PS}(v)) : \\ \mathrm{ca}_{t'}(r) \text{ exists}, \\ \alpha(r) \neq \mathrm{lp}(\mathrm{PS}(v))}} f_{\mathrm{ca}_t(r),t'} + \sum_{v' \in \mathrm{PS}(v)} \mathrm{area}_{v'} \cdot \mathrm{rain} \cdot \mathrm{sp}_{t_v} + \mathrm{pcelp}_S \\
&= \mathrm{pcenlp}_S - \sum_{t > t_v} \sum_{\substack{r \in \delta_G^-(\mathrm{PS}(v)) : \\ \mathrm{ca}_t(r) \text{ exists}, \\ \alpha(r) \neq \mathrm{lp}(\mathrm{PS}(v))}} f_{\mathrm{ca}_t(r),t} \\
&- \sum_{v' \in \mathrm{PS}(v)} \mathrm{area}_{v'} \cdot \mathrm{rain} \cdot (1 - \mathrm{sp}_{t_v}) + \mathrm{pcelp}_S.
\end{aligned}
$$

A further direct consequence of the proof is the following statement.Using that S is backfloating, by definition, it follows that

$$
\sum_{t > t_v} \sum_{\substack{r \in \delta_G^-(\mathrm{PS}(v)) : \\ \mathrm{ca}_t(r) \text{ exists}, \\ \alpha(r) \neq \mathrm{lp}(\mathrm{PS}(v))}} f_{\mathrm{ca}_t(r),t} + \sum_{v' \in \mathrm{PS}(v)} \mathrm{area}_{v'} \cdot \mathrm{rain} \cdot (1 - \mathrm{sp}_{t_v}) > \mathrm{pcelp}_S,
$$

which means that $\mathrm{cap}(S) < \mathrm{pcelp}_S$. Further, plugging in the formula for bf_S into the above equation yields $\mathrm{bf}_S = \mathrm{pcenlp}_S - \mathrm{cap}(S)$. □

Lemma 21 *Let $S \in \mathcal{S}(y)$ with $\mathrm{pcenlp}_S > \mathrm{cap}(S)$. Then, it holds that S is backfloating and that $\mathrm{bf}_S = \mathrm{pcenlp}_S - \mathrm{cap}(S)$.*

Proof Let $v \in V$ such that $S = \mathrm{PS}(v)$. We first show that S is ftc. Suppose the contrary, then it holds that $v \in \overline{V}$. Hence, applying Corollary 6 yields

$$
\mathrm{cap}(S) > \mathrm{excess}_S = \mathrm{pce}_S = \mathrm{pcenlp}_S + \mathrm{pcelp}_S \geq \mathrm{pcenlp}_S,
$$

which is a contradiction. Executing the same proof as for Lemma 20 yields the desired result. □

The two previous lemmas are summarized in the following proposition.

Proposition 9 *Let $S \in \mathcal{S}(y)$. Then S is backfloating if and only if $y.\mathrm{pcenlp}_S > \mathrm{cap}(S)$. In this case, it holds that $y.\mathrm{bf}_S = y.\mathrm{pcenlp}_S - \mathrm{cap}(S)$.*

A further direct consequence of the proof is the following statement.

Corollary 12 *Let $S \in \mathcal{S}(y)$ be ftc. Further, let $v \in V$ such that $S = \mathrm{PS}(v)$. Then, it holds that*

$$\operatorname{cap}(S) = y.\operatorname{pcenlp}_S - \sum_{t>t_v} \sum_{\substack{r\in\delta_G^-(\operatorname{PS}(v)):\\ \operatorname{ca}_t(r)\text{ exists},\\ \alpha(r)\neq \operatorname{lp}(\operatorname{PS}(v))}} y.f_{\operatorname{ca}_t(r),t}$$
$$- \sum_{v'\in\operatorname{PS}(v)} \operatorname{area}_{v'} \cdot \operatorname{rain} \cdot (1 - \operatorname{sp}_{t_v}) + y.\operatorname{pcelp}_S.$$

Appendix 4.4. Proof of Equal Water Levels in *x* and *y*

We now use the results obtained about the water levels and flows in pre-sinks to show that $x.\mathrm{wl}_v = y.\mathrm{wl}_v$ for all $v \in V$. The idea of the proof is to show from the highest node to the lowest node that either the water levels are the same or that the flows on all outgoing arcs that are not full in x are the same in x and y. The overall flows over an arc $r \in R$ in y are defined by $y.f_r := \sum_{\substack{t \in \{1,\dots,T\}:\\ \operatorname{ca}_t(r)\text{ exists}}} y.f_{\operatorname{ca}(r),t}$. We formalize the above-mentioned property of a node in the following definition:

Definition 11 A node $v \in V$ is called *explored above* if v is non-flooded in x and y and all $v' \in V$ with $\mathrm{gh}_{v'} > \mathrm{gh}_v$ fulfill the following properties:

1. $x.\mathrm{wl}_{v'} = y.\mathrm{wl}_{v'}$
2. For each arc $r \in \delta_G^+(v')$ that not full in x, it holds that $x.f_r = y.f_r$.

It is now shown that, if a node $v \in V$ is explored above, each pre-sink induced by a follow-up node of $\operatorname{PS}(v)$ is backfloating in x if and only if it is in y.

Lemma 22 *Let $v \in V$ be explored above. Then, for each $u_i \in \operatorname{FUN}(v)$, it holds that $\operatorname{PS}(u_i)$ is backfloating in x if and only if it is in y. If it is backfloating, it further holds that $x.\mathrm{bf}_{\operatorname{PS}(u_i)} = y.\mathrm{bf}_{\operatorname{PS}(u_i)}$.*

Proof As v is not flooded, it holds that every arc $r \in \delta^-(\operatorname{PS}(u_i))$ with $\alpha(r) \neq v$ is not full in x. Then, due to Property 2 of v being explored above, it holds that $x.\operatorname{pcenlp}_{\operatorname{PS}(u_i)} = y.\operatorname{pcenlp}_{\operatorname{PS}(u_i)}$. The claim then follows using Propositions 8 and 9. □

The lemma in particular shows that, if a pre-sink induced by a follow-up node u_i is backfloating, the water levels of all nodes in $\operatorname{PS}(u_i)$ are the same in x and y. We next investigate the follow-up nodes whose induced pre-sinks are not ftc. To this end, we introduce a further definition.

Definition 12 Let $v \in V$ and let $u_i \in \mathrm{FUN}(v)$. The *total ratio of the lowest parent* is defined by

$$\mathrm{ratiolp}_{\mathrm{PS}(u_i)} := \sum_{\substack{r \in \delta_G^-(\mathrm{PS}(u_i)):\\ \alpha(r)=v}} \mathrm{ratio}_r.$$

We observe two important properties of the total ratio of the lowest parent.

Observation 13 Let $v \in V$ and $z \in \{x, y\}$. Further, let $u_i, u_j \in \mathrm{FUN}(v)$ such that $\mathrm{PS}(u_i)$ and $\mathrm{PS}(u_j)$ are not ftc in z. Then, it holds that

$$\frac{z.\mathrm{pcelp}_{\mathrm{PS}(u_i)}}{z.\mathrm{pcelp}_{\mathrm{PS}(u_j)}} = \frac{\mathrm{ratiolp}_{\mathrm{PS}(u_i)}}{\mathrm{ratiolp}_{\mathrm{PS}(u_j)}}.$$

Proof For $z = x$, this is clear since all arcs in $\delta_G^-(\mathrm{PS}(u_i))$ and $\delta_G^-(\mathrm{PS}(u_j))$ are not full. The claim is then clear from Constraint (31.1 and 31.2). For $z = y$, the claim is also clear as all arcs in $\delta_G^-(\mathrm{PS}(u_i))$ and $\delta_G^-(\mathrm{PS}(u_j))$ are never removed from the graph (otherwise, $\mathrm{PS}(u_i)$ or $\mathrm{PS}(u_j)$ would be ftc). □

Observation 14 Let $z \in \{x, y\}$ and $v \in V$ be non-flooded with $z.\mathrm{pce}_{\mathrm{PS}(v)} < \mathrm{thr}(\mathrm{PS}(v))$. Then, there exists a pre-sink $\mathrm{PS}(u_i)$ for some $u_i \in \mathrm{FUN}(v)$ that is non-ftc in z.

The two previous observations show that the following is well-defined:

Definition 13 Let $z \in \{x, y\}$ and $v \in V$ be non-flooded with $z.\mathrm{pce}_{\mathrm{PS}(v)} < \mathrm{thr}(\mathrm{PS}(v))$. Furthermore, let $u_i \in \mathrm{FUN}(v)$ such that $\mathrm{PS}(u_i)$ is non-ftc in z. Then, we define the *non-full arc distribution* of v as

$$z.\mathrm{nfad}_v := \frac{z.\mathrm{pcelp}_{\mathrm{PS}(u_i)}}{\mathrm{ratiolp}_{\mathrm{PS}(u_i)}}.$$

We next show that $x.\mathrm{nfad}_v = y.\mathrm{nfad}_v$ for each non-flooded and explored-above node $v \in V$ with $z.\mathrm{pce}_{\mathrm{PS}(v)} < \mathrm{thr}(\mathrm{PS}(v))$. To prove this, an additional lemma is required.

Lemma 23 *Let $z \in \{x, y\}$ and $v \in V$ be non-flooded with $z.\mathrm{pce}_{\mathrm{PS}(v)} < \mathrm{thr}(\mathrm{PS}(v))$. Further let $u_i \in \mathrm{FUN}(v)$. If it further holds that $z.\mathrm{nfad}_v \cdot \mathrm{ratiolp}_{\mathrm{PS}(u_i)} \geq \mathrm{cap}(\mathrm{PS}(u_i)) - z.\mathrm{pcenlp}_{\mathrm{PS}(u_i)}$, then $\mathrm{PS}(u_i)$ is ftc in z.*

Proof Suppose $\mathrm{PS}(u_i)$ is not ftc. Then, by definition, it holds that

$$\begin{aligned} z.\mathrm{pcelp}_{\mathrm{PS}(u_i)} &= z.\mathrm{nfad}_v \cdot \mathrm{ratiolp}_{\mathrm{PS}(u_i)} \\ &\geq \mathrm{cap}(\mathrm{PS}(u_i)) - z.\mathrm{pcenlp}_{\mathrm{PS}(u_i)}, \end{aligned}$$

which means that $z.\mathrm{excess}_{\mathrm{PS}(u_i)} \geq \mathrm{cap}(\mathrm{PS}(u_i))$ (due to Lemma 4 in the case of $z = x$ and Corollary 6 in the case of $z = y$). As v is non-flooded, it must also hold that $z.\mathrm{excess}_{\mathrm{PS}(u_i)} \leq \mathrm{cap}(\mathrm{PS}(u_i))$, which means that equality holds and $\mathrm{PS}(u_i)$ is ftc. □

We proceed by showing that $x.\text{nfad}_v = y.\text{nfad}_v$ for each explored-above node $v \in V$ for which it holds that $z.\text{pce}_{\text{PS}(v)} < \text{thr}(\text{PS}(v))$.

Lemma 24 *Let $z \in \{x, y\}$ and $v \in V$ be explored above with $z.\text{pce}_{\text{PS}(v)} < \text{thr}(\text{PS}(v))$. Then, it holds that $x.\text{nfad}_v = y.\text{nfad}_v$.*

Proof As v is explored above, the node is non-flooded, and hence, it holds that $x.\text{excess}_{\text{PS}(v)} = y.\text{excess}_{\text{PS}(v)}$ due to Lemma 4 and Corollary 6.

For the sake of a contradiction, suppose that $x.\text{nfad}_v \neq y.\text{nfad}_v$. It is firstly assumed that $x.\text{nfad}_v > y.\text{nfad}_v$. The proof of the other case is along the same lines.

For all $u_i \in \text{FUN}(v)$ for which $\text{PS}(u_i)$ is ftc in y, Lemma 23 shows that $\text{PS}(u_i)$ is also ftc in x. In particular, this yields $x.\text{excess}_{\text{PS}(u_i)} = y.\text{excess}_{\text{PS}(u_i)}$. For all $u_i \in \text{FUN}(v)$, for which $\text{PS}(u_i)$ is not ftc in y, we distinguish two cases.

- Case 1: $\text{PS}(u_i)$ is ftc in x Then, $y.\text{excess}_{\text{PS}(u_i)} \leq \text{cap}(\text{PS}(u_i)) = x.\text{excess}_{\text{PS}(u_i)}$ by definition of ftc.
- Case 2: $\text{PS}(u_i)$ is not ftc in x. Then, by definition, it holds that $x.\text{pcelp}_{\text{PS}(u_i)} > y.\text{pcelp}_{\text{PS}(u_i)}$. As v is explored above, it further holds that $x.\text{pcenlp}_{\text{PS}(u_i)} = y.\text{pcenlp}_{\text{PS}(u_i)}$, which yields $x.\text{pce}_{\text{PS}(u_i)} > y.\text{pce}_{\text{PS}(u_i)}$. As $\text{PS}(u_i)$ is not ftc in both x and y, it holds that

  $$x.\text{excess}_{\text{PS}(u_i)} = x.\text{pce}_{\text{PS}(u_i)} > y.\text{pce}_{\text{PS}(u_i)} = y.\text{excess}_{\text{PS}(u_i)}.$$

  All in all, since there exists a $u_i \in \text{FUN}(v)$ such that $\text{PS}(u_i)$ is not ftc in y due to Observation 14, it holds that

  $$\begin{aligned} x.\text{excess}_{\text{PS}(v)} &= \sum_{u_i \in \text{FUN}(v)} x.\text{excess}_{\text{PS}(u_i)} \\ &> \sum_{u_i \in \text{FUN}(v)} y.\text{excess}_{\text{PS}(u_i)} = y.\text{excess}_{\text{PS}(v)}, \end{aligned}$$

  which is a contradiction. □

We now prove the final proposition, which shows that, if a node v is explored above, then its non-flooded follow-up nodes are as well.

Proposition 10 *Let $v \in V$ be explored above with $z.\text{pce}_{\text{PS}(v)} < \text{thr}(\text{PS}(v))$ for all $z \in \{x, y\}$. Then, it holds that*

(a) $x.\text{wl}_v = y.\text{wl}_v$
(b) *For all $u_i \in \text{FUN}(v)$, it holds that $x.\text{wl}_{u_i} = y.\text{wl}_{u_i}$*
(c) *For all $r \in \delta_G^+(v)$ where r is not full in x, it holds that $x.f_r = y.f_r$.*

Proof We prove the claims individually.

(a) As $z.\text{pce}_{\text{PS}(v)} < \text{thr}(\text{PS}(v))$ for all $z \in \{x, y\}$, it holds that v is non-flooded in both x and y.
(b) Let $u_i \in \text{FUN}(v)$. If $\text{PS}(u_i)$ is ftc in x, then it is also in y due to Lemmas 23 and 24, which means that the claim holds in this case. If $\text{PS}(u_i)$ is not ftc in x, it is, again because of Lemmas 23 and 24, not ftc in y. In this case, it holds that $x.\text{excess}_{\text{PS}(u_i)} = y.\text{excess}_{\text{PS}(u_i)}$, which shows the claim.
(c) Let $r \in \delta_G^+(v)$ where r is not full in x. Then, due to Lemma 24, it holds that $f_r = \text{nfad}_v \cdot \text{ratio}_r$ in both solutions, which proves the claim. □

Using this proposition, the final theorem can be shown.

Theorem 1 *For all $v \in V$, it holds that $x.\text{wl}_v = y.\text{wl}_v$.*

Proof Let $v' \in V$ be the highest node in the graph. As we assumed that v' is not flooded, it clearly holds that v' is explored above. If $\text{pce}(v') = \text{thr}(v')$, we know that all pre-sinks induced by the follow-up nodes of v' are ftc, which means that the claim holds. If this is not the case, Proposition 10 shows that, for each $u_i \in \text{FUN}(v')$, it holds that $x.\text{wl}_{u_i} = y.\text{wl}_{u_i}$. If $\text{wl}_{u_i} > 0$, it must hold that $\text{PS}(u_i)$ is a sink, and, hence, that all nodes in $\text{PS}(u_i)$ have the same water level in the two solutions. Otherwise, Proposition 10 shows that u_i is explored above. We then continue applying the same arguments to all such u_i and work our way down the graph until we have shown the claim for all nodes. □

Appendix 5. Algorithm for presolving non-flooded nodes

An algorithm for computing the nodes that can be preset to be non-flooded is provided as Algorithm 7.

Algorithm 7 PRESOLVE-NON-FLOODED

```
Procedure presolveNonFloodedNodes(G)
    Compute the maximal geodesic height for each node v ∈ V and obtain V_nf.
    Construct the graph G_nf = (V_nf, R_nf) by adding downhill arcs; Initialize
      presolve_non_flooded = ∅
    for v ∈ V_nf do
        volume_needed = 0
        if v ≠ s and A(v) = ∅ then
            for v' ∈ successors(v) do
                volume_needed = volume_needed + area_v' · (GH_v − GH_v')
            end
            if volume_needed > total rain volume then
                Add v to presolve_non_flooded
            else
                Take a copy of the undriected graph and remove all nodes with
                  geodesic height larger than GH_v
                volume_needed = 0
                for each v' in the connected component of v after removal with
                  v' ≠ v do
                    volume_needed = volume_needed + area_v' · (GH_v − GH_v')
                end
                if volume_needed > total rain volume then
                    Add v to presolve_non_flooded
                end
            end
        end
    end
    return presolve_non_flooded
```

Author Contribution JB and CT jointly developed the presented mathematical models. JB implemented the models and performed the computational experiments. Both authors contributed equally to the writing of the manuscript and have approved the final manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was partially supported by the German Federal Ministry for the Environment, Nature Conservation and Nuclear Safety (BMU) within the project "AKUT – Incentive Systems for Municipal Flood Prevention" (grant number 67DAS156C).

Data Availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Code Availability The code is available from the corresponding author on reasonable request.

Declarations

Ethics Approval Not applicable

Consent to Participate Not applicable

Consent for Publication Not applicable

Conflict of Interest The authors declare no competing interests.

References

1. Blenkinsop S, Fowler HJ, Barbero R, Chan SC, Guerreiro SB, Kendon E, Lenderink G, Lewis E, Li XF, Westra S et al (2018) The INTENSE project: using observations and models to understand the past, present and future of sub-daily rainfall extremes. Adv Sci Res 15:117–126
2. IPCC: Climate Change 2021 (2021) The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge University Press. Masson-Delmotte V, Zhai P, Pirani A, Connors SL, Péan C, Berger S, Caud N, Chen Y, Goldfarb L, Gomis MI, Huang M, Leitzell K, Lonnoy E, Matthews JBR, Maycock TK, Waterfield T, Yelekçi O, Yu R, Zhou B (eds.)
3. Rajczak J, Schär C (2017) Projections of future precipitation extremes over Europe: a multimodel assessment of climate simulations. Journal of Geophysical Research: Atmospheres 122(20):10773–10800
4. Fekete A, Sandholz S (2021) Here comes the flood, but not failure? Lessons to learn after the heavy rain and pluvial floods in Germany 2021. Water 13(21):3016
5. Mohr S, Ehret U, Kunz M, Ludwig P, Caldas-Alvarez A, Daniell JE, Ehmele F, Feldmann H, Franca MJ, Gattke C, Hundhausen M, Knippertz P, Küpfer K, Mühr B, Pinto JG, Quinting J, Schäfer AM, Scheibel M, Seidel F, Wisotzky C (2023) A multi-disciplinary analysis of the exceptional flood event of July 2021 in central Europe - Part 1: Event description and analysis. Nat Hazard 23(2):525–551

6. German Association for Water, Wastewater and Waste (2016) Merkblatt DWA-M 119, Risikomanagement in der kommunalen Überflutungsvorsorge für Entwässerungssysteme bei Starkregen (Risk management in municipal flood protection for drainage systems in the event of heavy rain)
7. Siekmann T (2018) Methodik zur Priorisierung von Maßnahmen der Sturzflutvorsorge. https://www.siekmann-ingenieure.de/media/priorisierung-massnahmen_methodik.pdf. Accessed 01 April 2023
8. Kreibich H, Thieken AH, Petrow T, Müller M, Merz B (2005) Flood loss reduction of private households due to building precautionary measures-lessons learned from the Elbe flood in August 2002. Nat Hazard 5(1):117–126
9. Tasseff B (2021) Optimization of critical infrastructure with fluids. Ph.D. thesis, University of Michigan
10. Woodward M, Kapelan Z, Gouldby B (2014) Adaptive flood risk management under climate change uncertainty using real options and optimization. Risk Anal 34(1):75–92
11. Brekelmans R, den Hertog D, Roos K, Eijgenraam C (2012) Safe dike heights at minimal costs: the nonhomogeneous case. Oper Res 60(6):1342–1355
12. Zwaneveld P, Verweij G, van Hoesel S (2018) Safe dike heights at minimal costs: an integer programming approach. Eur J Oper Res 270(1):294–301
13. Klerk W, Kanning W, Kok M, Wolfert R (2021) Optimal planning of flood defence system reinforcements using a greedy search algorithm. Reliability Engineering & System Safety 207:107344
14. Huang C, Hsu N, Liu H, Huang Y (2018) Optimization of low impact development layout designs for megacity flood mitigation. J Hydrol 564:542–558
15. Ngo TT, Yoo DG, Lee YS, Kim JH (2016) Optimization of upstream detention reservoir facilities for downstream flood mitigation in urban areas. Water 8(7):290
16. Jack B, Kousky C, Sims K (2008) Designing payments for ecosystem services: lessons from previous experience with incentive-based mechanisms. Proceedings of the National Academy of Sciences (PNAS) 105(28):9465–9470
17. Machac J, Hartmann T, Jilkova J (2018) Negotiating land for flood risk management: upstream-downstream in the light of economic game theory. Journal of Flood Risk Management 11(1):66–75
18. Poussin J, Botzen W, Aerts J (2014) Factors of influence on flood damage mitigation behaviour by households. Environmental Science & Policy 40:69–77
19. Filatova T (2014) Market-based instruments for flood risk management: a review of theory, practice and perspectives for climate adaptation policy. Environmental Science & Policy 37:227–242
20. Khalilpourazari S, Pasandideh SHR (2021) Designing emergency flood evacuation plans using robust optimization and artificial intelligence. J Comb Optim 41:640–677
21. Che D, Mays LW (2015) Development of an optimization/simulation model for real-time flood-control operation of river-reservoirs systems. Water Resour Manage 29(11):3987–4005
22. Wei C, Hsu N (2008) Multireservoir real-time operations for flood control using balanced water level index method. J Environ Manage 88(4):1624–1639
23. Munawar HS, Hammad AWA, Waller ST, Thaheem MJ, Shrestha A (2021) An integrated approach for post-disaster flood management via the use of cutting-edge technologies and UAVs: a review. Sustainability 13(14):7925
24. Schmitt TG, Worreschk S, Kaufmann Alves I, Herold F, Thielen C (2014) An optimization and decision support tool for long-term strategies in the transformation of urban water infrastructure. In: Proceedings of the 11th International Conference on Hydroinformatics (HIC), pp. 1–8
25. Nematollahi B, Parnian BH, Talebbeydokhti N, Rakhshandehroo GR, Nikoo MR, Gandomi AH (2022) A stochastic conflict resolution optimization model for flood management in detention basins: application of fuzzy graph model. Water 14(5):774
26. Holzhauser M, Krumke S, Thielen C (2017) Maximum flows in generalized processing networks. J Comb Optim 33:1226–1256
27. Koene J (1983) Minimal cost flow in processing networks: a primal approach. Ph.D. thesis, Centrum voor Wiskunde & Informatica, Amsterdam
28. Bonami P, Lodi A, Tramontani A, Wiese S (2015) On mathematical programming with indicator constraints. Math Program 151:191–223
29. Institut für technisch-wissenschaftliche Hydrologie GmbH: HYSTEM-EXTRAN. https://itwh.de/en/software-products/desktop/hystem-extran/. Accessed 1 April 2023
30. Roe GH (2005) Orographic precipitation. Annu Rev Earth Planet Sci 33(1):645–671