AGU100 ADVANCING EARTH AND SPACE SCIENCE

Water Resources Research

RESEARCH ARTICLE

10.1029/2018WR024362

**Key Points:**
- The structural optimal flood mitigation problem is introduced
- A problem discretization amenable to derivative-free optimization is developed
- Benefits of constraining the problem with additional physics-based restrictions are shown

**Correspondence to:**
B. Tasseff,
btasseff@lanl.gov

**Citation:**
Tasseff, B., Bent, R., & Van Hentenryck, P. (2019). Optimization of structural flood mitigation strategies. *Water Resources Research*, *55*, 1490–1509. https://doi.org/10.1029/2018WR024362

Received 5 NOV 2018
Accepted 21 JAN 2019
Accepted article online 25 JAN 2019
Published online 21 FEB 2019

# Optimization of Structural Flood Mitigation Strategies

**Byron Tasseff[1,2], Russell Bent[3], and Pascal Van Hentenryck[4]**

[1] Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, USA, [2] Information Systems and Modeling Group, Los Alamos National Laboratory, Los Alamos, NM, USA, [3] Applied Mathematics and Plasma Physics Group, Los Alamos National Laboratory, Los Alamos, NM, USA, [4] H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA

**Abstract** The dynamics of flooding are primarily influenced by the shape, height, and roughness (friction) of the underlying topography. For this reason, mechanisms to mitigate floods frequently employ structural measures that either modify topographic elevation, for example, through the placement of levees and sandbags, or increase roughness, for example, through revegetation projects. However, the configuration of these measures is typically decided in an ad hoc manner, limiting their overall effectiveness. The advent of high-performance surface-water modeling software and improvements in black-box optimization suggest that a more principled design methodology may be possible. This paper proposes a new computational approach to the problem of designing structural mitigation strategies under physical and budgetary constraints. It presents the development of a problem discretization amenable to simulation-based, derivative-free optimization. However, meta-heuristics alone are found to be insufficient for obtaining quality solutions in a reasonable amount of time. As a result, this paper proposes novel numerical and physics-based procedures to improve convergence to a high-quality mitigation. The efficiency of the approach is demonstrated on hypothetical dam break scenarios of varying complexity under various mitigation budget constraints. In particular, experimental results show that, on average, the final proposed algorithm results in a 65% improvement in solution quality compared to a direct implementation.

## 1. Introduction

Modern flood risk management (FRM) is a continuous process of identifying issues, defining objectives, assessing risks, appraising options, implementation, monitoring, and review. Within this framework, risk assessment is regarded as a cyclic process that includes the design and evaluation of alternative management strategies. Such strategies commonly include both "hard" and "soft" structural mitigation measures, for example, the construction of dams (hard) and wetland storage (soft; Sayers et al., 2013). Measures can also be temporary (e.g., sandbags) or permanent (e.g., levees). However, for complex scenarios, the number of feasible strategies is extremely large and computationally difficult to explore. As such, the manual design and assessment of these strategies, whether conducted in a real-world or simulation-based setting, can be time-consuming and expensive. This limitation may result in vastly suboptimal FRM strategies. To aid in the FRM process, an optimization-based decision support approach for proposing structural mitigation designs can serve as a useful tool within the overall risk assessment phase.

This paper develops such an optimization-based decision support approach for proposing flood protection strategies, whereby effective mitigation designs are realized through the exploration of various configurations in a computational setting. Specifically, the paper defines the Optimal Flood Mitigation Problem (OFMP), whose goal is to make topographic modifications that protect critical regions under a given flood scenario. This is a difficult optimization problem, as these modifications can have highly nonlinear effects on the flooding behavior. Moreover, physical models used to examine these effects are computationally expensive. Finally, as the OFMP aims at deciding several simultaneous modifications, an efficient exploration of the full search space is computationally intractable for realistic scenarios.

The literature associated with the OFMP is limited. The closest related studies are by Judi, Tasseff, et al. (2014) and Tasseff et al. (2016). In the former, an interdiction model for flood mitigation is proposed, and model surrogates constructed from simulation data are used as proxies for estimating flood sensitivity to hard structural mitigation measures. In the latter, an OFMP similar to that discussed herein is introduced, and mixed-integer linear programs constrained by approximate flooding dynamics are solved to obtain hard structural mitigation designs. However, the approach is shown to suffer from substantial scalability issues in space and time. Neither study uses an approach which relies upon repeated deterministic modeling of the partial differential equations (PDEs) underlying the flooding dynamics.

A number of studies discuss simulation-optimization approaches for reservoir operation, where the PDEs associated with the flood dynamics are treated as a black box. An extensive literature review of these studies can be found in Che and Mays (2015). The work described by Colombo et al. (2009) considers the full PDEs, but their focus is on optimizing normal operations of an open-channel system. Finally, the problem of optimizing dike heights with uncertainty in flooding estimates is considered by Brekelmans et al. (2012). However, in this study, the PDEs for flood propagation are not considered, and probability models for maximum flood depths are used in place of deterministic physical models.

This paper presents a new approach to the problem of designing structural FRM strategies over PDE constraints. It develops a problem discretization amenable to simulation-based derivative-free optimization. Moreover, the paper shows that meta-heuristics alone are insufficient for obtaining quality solutions in reasonable time. As a result, it presents several innovative computational and physics-based techniques to increase convergence to high-quality solutions. The efficiency of the proposed approach is compared using hypothetical dam break scenarios of varying complexity under multiple mitigation budgets. Experimental results show that the proposed algorithm results in a 65% improvement in solution quality compared to a direct implementation.

The rest of this paper is organized as follows: Section 2 discusses the background of flood modeling and formalization of the OFMP; section 3 describes solution methods for a specific OFMP; section 4 compares these methods using fictional dam break scenarios, with both simplistic (section 4.3) and realistic (section 4.4) topographies, and multiple mitigation budgets; and section 5 concludes the paper.

## 2. Model

In this paper, it is assumed that flood scenarios are modeled using the two-dimensional (2-D) shallow water equations. These PDEs are derived from the Navier-Stokes equations under the assumption that horizontal length scales are much larger than the vertical scale. This is reasonable for large-scale floods, where water depths are much smaller than typical flood wavelengths. Two-dimensional models, in particular, alleviate the fundamental disadvantages of their 1-D counterparts by allowing for higher-order representations of the topographic surface. Moreover, 2-D models readily make use of widely available topographic elevation data. Finally, with recent advances in high-performance computing, solutions to these PDEs have become numerically tractable for large-scale problems, making them of particular computational interest. With volumetric, bed slope, and bed shear stress source terms, these equations are expressed as

$$\frac{\partial h}{\partial t} + \frac{\partial (hu)}{\partial x} + \frac{\partial (hv)}{\partial y} = R(x, y, t), \tag{1a}$$

$$\frac{\partial (hu)}{\partial t} + \frac{\partial}{\partial x}\left(hu^2 + \frac{1}{2}gh^2\right) + \frac{\partial (huv)}{\partial y} = -gh\frac{\partial B}{\partial x} - g\frac{n^2}{h^{1/3}}|u|u, \tag{1b}$$

$$\frac{\partial (hv)}{\partial t} + \frac{\partial (huv)}{\partial x} + \frac{\partial}{\partial y}\left(hv^2 + \frac{1}{2}gh^2\right) = -gh\frac{\partial B}{\partial y} - g\frac{n^2}{h^{1/3}}|v|v, \tag{1c}$$

where $h$ is the water depth, $u$ and $v$ are horizontal velocities, $B$ is the bottom topography (or bathymetry), $g$ is the acceleration due to gravity, $n$ is the Manning's roughness coefficient, and $R$ is a volumetric source term (Chertock et al., 2015). Equation (1a) represents mass continuity, while equations (1b) and (1c) represent conservation of momentum over the two horizontal dimensions.

These equations can be rewritten in vector form by introducing the definitions

$$\begin{gathered}\mathbf{U} := (h, hu, hv), \mathbf{F}(\mathbf{U}) := \left(hu, hu^2 + \frac{1}{2}gh^2, huv\right), \\ \mathbf{G}(\mathbf{U}) := \left(hv, huv, hv^2 + \frac{1}{2}gh^2\right), \mathbf{S}_R(R) := (R(x, y, t), 0, 0), \\ \mathbf{S}_B(\mathbf{U}, B) := \left(0, -gh\frac{\partial B}{\partial x}, -gh\frac{\partial B}{\partial y}\right), \mathbf{S}_n := \left(0, -g\frac{n^2}{h^{1/3}}|u|u, -g\frac{n^2}{h^{1/3}}|v|v\right),\end{gathered} \tag{2}$$

where **U** is the vector of conserved variables; **F** and **G** are fluxes in the $x$ and $y$ directions, respectively; and $\mathbf{S}_R$, $\mathbf{S}_B$, and $\mathbf{S}_n$ are the volumetric, bed slope, and bed shear stress source terms, respectively. This allows equations (1a)–(1c) to be rewritten more concisely as

$$\mathbf{U}_t + \mathbf{F}_x + \mathbf{G}_y = \mathbf{S}_R + \mathbf{S}_B + \mathbf{S}_n, \tag{3}$$

where $t$, $x$, and $y$ indicate partial differentiation with respect to those variables.

The OFMP considers a flood scenario (e.g., a dam failure) and a set of 2-D regions ("assets") to protect. To minimize flooding at asset locations, the model must produce optimal topographic elevation and roughness fields using a set of $m$ mitigation measures. For each measure $i \in \{1, 2, \ldots, m\}$, the functions $\delta_B(\omega_i)$ and $\delta_n(\omega_i)$ define 2-D fields of height and roughness for a given tuple of field parameters $\omega_i$. Measures can first additively modify the elevation field $B$ to return a new field $\tilde{B}$, defined as

$$\tilde{B}\left(B, \left(\omega_1, \omega_2, \ldots, \omega_m\right)\right) := B + \sum_{i=1}^{m} \delta_B(\omega_i). \tag{4}$$

Similarly, measures can modify the roughness field $n$ to return a new field defined as

$$\tilde{n}\left(n, \left(\omega_1, \omega_2, \ldots, \omega_m\right)\right) := n + \{\max_i\{\delta_n(\omega(i))(x, y)\} : (x, y) \in \mathbb{R}^2\}, \tag{5}$$

that is, a field of maximum roughness. For notational ease, hereafter, $\tilde{B}$ refers to equation (4), ñ refers to equation (5), and the tuple $(\omega_1, \omega_2, \ldots, \omega_m)$ is referred to as the "parametric configuration" or simply the "configuration." With these definitions and shorthand notations, the *modified* bed slope source term is defined as

$$\tilde{\mathbf{S}}_B\left(\mathbf{U}, B, \left(\omega_1, \omega_2, \ldots, \omega_m\right)\right) := \left(0, -gh\frac{\partial \tilde{B}}{\partial x}, -gh\frac{\partial \tilde{B}}{\partial y}\right). \tag{6}$$

We note that the change in elevation may be a result of permanent structures such as levees or temporary measures such as sandbags. Similarly, the modified bed shear stress source term is defined as

$$\tilde{\mathbf{S}}_n\left(\mathbf{U}, n, \left(\omega_1, \omega_2, \ldots, \omega_m\right)\right) := \left(0, -g\frac{\tilde{n}^2}{h^{1/3}}|u|u, -g\frac{\tilde{n}^2}{h^{1/3}}|v|v\right). \tag{7}$$

Hereafter, $\tilde{\mathbf{S}}_B := \tilde{\mathbf{S}}_B\left(\mathbf{U}, B, (\omega_1, \omega_2, \ldots, \omega_m)\right)$ and $\tilde{\mathbf{S}}_n := \tilde{\mathbf{S}}_n\left(\mathbf{U}, n, (\omega_1, \omega_2, \ldots, \omega_m)\right)$ are used to concisely denote these two source terms that vary with the configuration.

The OFMP is then written in a form that embeds the 2-D shallow water equations as constraints and optimizes the tuple $(\omega_1, \omega_2, \ldots, \omega_m)$ (i.e., the configuration) via

$$\underset{\omega_1, \omega_2, \ldots, \omega_m}{\text{minimize}} \quad z\left(\omega_1, \omega_2, \ldots, \omega_m\right) = \sum_{a \in \mathcal{A}} \iint_a \max_t h(x, y, t)\,\mathrm{d}x\,\mathrm{d}y \tag{8a}$$

$$\text{subject to} \quad \mathbf{U}_t + \mathbf{F}_x + \mathbf{G}_y = \mathbf{S}_R + \tilde{\mathbf{S}}_B + \tilde{\mathbf{S}}_n \tag{8b}$$

$$\delta_B(\omega_i)(x, y) = 0, \forall i \in \{1, 2, \ldots, m\}, \quad \text{for} \quad (x, y) \in \bigcup \mathcal{A} \tag{8c}$$

$$\delta_n(\omega_i)(x, y) = 0, \forall i \in \{1, 2, \ldots, m\}, \quad \text{for} \quad (x, y) \in \bigcup \mathcal{A} \tag{8d}$$

$$\left(\omega_1, \omega_2, \ldots, \omega_m\right) \in \mathcal{F}. \tag{8e}$$

Here $\mathcal{A}$ denotes the set of asset regions to be protected and $z$ denotes the objective function. This function is defined in equation (8a) and captures the maximum water volume over all asset locations and times. Constraint (8b) denotes the solution to the shallow water equations in the presence of the $m$ mitigation measures. Constraints (8c) prohibit measures from being constructed "underneath" an asset. Similarly, constraints (8d) prohibit the roughness at an asset location from being modified. Finally, constraint (8e) ensures $(\omega_1, \omega_2, \ldots, \omega_m)$ resides within the set of all feasible parametric configurations $\mathcal{F}$, that is, $\mathcal{F}$ distinguishes valid and invalid mitigation designs.

For simplicity of presentation, this paper considers only two types of structural measures, although the approach can easily be generalized to include other soft and hard measures, both temporary and permanent. The first type is an immovable wall of fixed length ($\ell$), width ($w$), and height ($\bar{b}_i$). Each wall is defined using three continuously defined, bounded parameters: latitudinal position of the wall centroid ($\lambda_i$), longitudinal position of the wall centroid ($\phi_i$), and angle of the wall formed with respect to the longitudinal axis ($\theta_i$). In this paper, the centroid position is bounded by the scenario domain's spatial extent, and $\theta_i \in [0, \pi]$. The second structural type is a revegetation project defined by a 2-D circular region with center $(\lambda_i, \phi_i)$ and fixed radius $r$ that increases the area's Manning's roughness coefficient based on a fixed field $\bar{n}_i$. Under these assumptions, the OFMP aims at deciding $\omega_i = \left(\lambda_i, \phi_i, \theta_i, \bar{b}_i, \bar{n}_i\right)$ for each measure $i \in \{1, 2, \ldots, m\} = \mathcal{M}$, where $\bar{b}_i$ and $\bar{n}_i$ are decided a priori for each measure. More specifically, this produces an OFMP of the specialized form

$$\underset{\omega_1, \omega_2, \ldots, \omega_m}{\text{minimize}} \quad z\left(\omega_1, \omega_2, \ldots, \omega_m\right) = \sum_{a \in \mathcal{A}} \iint_a \max_t h(x, y, t)\, \mathrm{d}x\, \mathrm{d}y \tag{9a}$$

$$\text{subject to} \quad \mathbf{U}_t + \mathbf{F}_x + \mathbf{G}_y = \mathbf{S}_R + \tilde{\mathbf{S}}_B + \tilde{\mathbf{S}}_n \tag{9b}$$

$$\delta_B(\omega_i)(x, y) = 0, \quad \text{for} \quad (x, y) \in \bigcup \mathcal{A}, \forall i \in \mathcal{M} \tag{9c}$$

$$\delta_n(\omega_i)(x, y) = 0, \quad \text{for} \quad (x, y) \in \bigcup \mathcal{A}, \forall i \in \mathcal{M} \tag{9d}$$

$$\delta_B(\omega_i)(x, y) = \begin{cases} \bar{b}_i \ \text{for} \begin{cases} |(x - \phi_i)\cos\theta_i - (y - \lambda_i)\sin\theta_i| \le \frac{\ell}{2} \\ |(x - \phi_i)\sin\theta_i + (y - \lambda_i)\cos\theta_i| \le \frac{w}{2} \end{cases} & \forall i \in \mathcal{M} \\ 0 \ \text{otherwise} \end{cases} \tag{9e}$$

$$\delta_n(\omega_i)(x, y) = \begin{cases} \bar{n}_i(x, y) \ \text{for} \quad (x - \phi_i)^2 + (y - \lambda_i)^2 \le r^2 & \\ & \forall i \in \mathcal{M} \\ 0 \qquad \text{otherwise} \end{cases} \tag{9f}$$

$$\lambda_{lb} \le \lambda_i \le \lambda_{ub}, \phi_{lb} \le \phi_i \le \phi_{ub}, 0 \le \theta_i \le \pi, \forall i \in \mathcal{M}. \tag{9g}$$

Using this formulation, $i$ is a wall when $\bar{b}_i > 0$ and $\bar{n}_i = 0$, and $i$ is a revegetation project when $\bar{b}_i = 0$ and $\bar{n}_i > 0$. Constraints (9c) and (9d) emphasize that modifications cannot be made within asset regions; constraint (9e) imposes the wall height $\bar{b}_i$ within each rotated rectangle defined using the parameters $\lambda_i$, $\phi_i$, and $\theta_i$ and a standard 2-D rotation matrix; and constraint (9f) imposes additions to roughness within each revegetation circle defined by the center $(\lambda_i, \phi_i)$. Finally, constraint (9g) replaces constraint (8e) of the more general OFMP. Here $\lambda_{lb}$ and $\lambda_{ub}$ ($\phi_{lb}$ and $\phi_{ub}$) are the lower and upper latitudinal (longitudinal) boundaries of the scenario domain.

Constraint (9g) implies a large feasible region, as the spatial extent is typically much larger than the flood's extent. To reduce the solution space, the notion of a *restricted region* $\mathcal{P}$ is thus introduced, where centroids must reside in $\mathcal{P}$. That is,

$$(\lambda_i, \phi_i) \in \mathcal{P}, \forall i \in \{1, 2, \ldots, m\} \tag{9h}$$

is appended to the problem above, completing the primary model used in this paper.

## 3. Algorithm

The OFMP at the end of section 2 remains difficult to solve directly. However, with recent improvements in both numerical discretizations of the shallow water equations (e.g., Chertock et al., 2015) and high-performance implementations thereof (e.g., Brodtkorb et al., 2012; Tasseff & Judi, 2016), numerically efficient solutions of the PDEs described in constraint (9b) are possible. With this intuition, in algorithm 1, a time-limited search-based method is introduced to find a near-optimal solution $\left(\omega_1^*, \omega_2^*, \ldots, \omega_m^*\right)$ to the problem defined by equations (9a) through (9h).

**Algorithm 1** SolveOFMP: Solves the OFMP defined by equations (9a) through (9h)

```
1:  function SolveOFMP($B, n, \mathcal{A}, m, T_{\max}, \alpha$)
2:      $\tilde{\mathcal{P}} \leftarrow$ InitializeRestriction($B, n, \mathcal{A}, \alpha$)
3:      $\left(\omega_1^*, \omega_2^*, \ldots, \omega_m^*\right) \leftarrow$ InitializeSolution($m, \tilde{\mathcal{P}}$), $\Omega \leftarrow \emptyset$
4:      while Clock $< T_{\max}$ do
5:          $\left(\omega_1, \omega_2, \ldots, \omega_m\right) \leftarrow$ GenerateSolution($m, \tilde{\mathcal{P}}, \Omega$)
6:          Solve $\mathbf{U}_t + \mathbf{F}_x + \mathbf{G}_y = \mathbf{S}_R + \tilde{\mathbf{S}}_B + \tilde{\mathbf{S}}_n$
7:          $\Omega \leftarrow \Omega \cup \left\{\left(\omega_1, \omega_2, \ldots, \omega_m\right)\right\}$
8:          if $z(\omega_1, \omega_2, \ldots, \omega_m) < z(\omega_1^*, \omega_2^*, \ldots, \omega_m^*)$ then
9:              $\left(\omega_1^*, \omega_2^*, \ldots, \omega_m^*\right) \leftarrow \left(\omega_1, \omega_2, \ldots, \omega_m\right)$
10:             $\tilde{\mathcal{P}} \leftarrow$ UpdateRestriction($\mathbf{U}, \mathcal{A}, \tilde{\mathcal{P}}, \alpha$)
11:         end if
12:     end while
13:     return $(B + \sum_{i=1}^{m} \delta_B(\omega_i^*), n + \{\max_i\{\delta_n(\omega_i^*)(x, y)\} : (x, y) \in \mathbb{R}^2\})$
14: end function
```

Here $B$ and $n$ denote the initial topographic elevation and Manning's roughness coefficient fields; $\mathcal{A}$ denotes the set of assets; $m$ denotes the number of mitigation measures being configured; $T_{\max}$ denotes the maximum clock time; and $\alpha$ is a parameter used for computing restrictions. The function Clock returns the current clock time. Since a useful definition of $\mathcal{P}$ is difficult to compute a priori, $\tilde{\mathcal{P}}$ serves as an iterative approximation of some desired $\mathcal{P}$. In line 2, $\tilde{\mathcal{P}}$ is initialized; it is later modified in line 10 using UpdateRestriction. Both functions are described in section 3.1. In line 3, the best solution and the historical solution set $\Omega$ are initialized. In line 5, a configuration is generated via some history-dependent function GenerateSolution, described in section 4.1. In line 6, the shallow water equations are solved. In line 7, the historical solution set is updated. In lines 8 through 11, the best solution and $\tilde{\mathcal{P}}$ are updated. Finally, in line 13, the best elevation and roughness fields are returned.

### 3.1. Computation of the Restricted Region

#### 3.1.1. The Direct Methodology

The most obvious globally acceptable method for selecting $\mathcal{P}$ is to assume

$$\mathcal{P} = \mathbb{R}^2, \tag{10}$$

where, of course,

$$\left\{(x, y) \in \mathbb{R}^2 : \lambda_{lb} \leq x \leq \lambda_{ub}, \phi_{lb} \leq y \leq \phi_{ub}\right\} \subset \mathcal{P}, \tag{11}$$

indicating the bounds within constraint (9g) involving $\lambda_i$ and $\phi_i$ dominate those imposed by $\mathcal{P}$. This method for selecting $\mathcal{P}$ is hereafter referred to as the direct method. In practice, this method is used to define the direct implementations of the functions InitializeRestriction and UpdateRestriction, both of which return the set $\mathbb{R}^2$.

#### 3.1.2. The Pathline Methodology

A *pathline* is the trajectory an individual fluid element follows over time, beginning at position $\left(x_0, y_0\right)$ and time $t_0$. In 2-D, a pathline is defined by the two equations

$$x(t) = x_0 + \int_{t_0}^{t} u(x(t'), y(t'), t')\mathrm{d}t', \tag{12a}$$

$$y(t) = y_0 + \int_{t_0}^{t} v(x(t'), y(t'), t')\mathrm{d}t', \tag{12b}$$

where $u$ and $v$ are velocities in the $x$ and $y$ directions. To compute the pathline from a flood wave to an initially dry point $(x_0, y_0)$, the definition of $t_{\text{wet}}\,(x_0, y_0)$ is introduced as the time at which the depth at $(x_0, y_0)$ exceeds some threshold. More concisely,

$$t_{\text{wet}}(x_0, y_0) := \min\left\{t \in [t_0, t_f] \, : \, h(x_0, y_0, t) \geq \epsilon_h\right\}, \tag{13}$$

where $\epsilon_h$ is an arbitrarily small depth, taken in this study to be 1 mm. Using this definition, the pathline equations may be integrated *in reverse*, giving

$$x_{\text{wet}}(x_0, y_0, t) = x_0 + \int_{t_{\text{wet}}}^{t} u(x_{\text{wet}}(t'), y_{\text{wet}}(t'), t')\mathrm{d}t', \tag{14a}$$

$$y_{\text{wet}}(x_0, y_0, t) = y_0 + \int_{t_{\text{wet}}}^{t} v(x_{\text{wet}}(t'), y_{\text{wet}}(t'), t')\mathrm{d}t', \tag{14b}$$

where it is assumed that $t \leq t_{\text{wet}}$. The above equations approximate a path to flooding.

In this paper, a *pathtube* is defined as a set of pathlines satisfying equations (14a) and (14b). For a region $\mathcal{R}$, the pathtube $\mathcal{S}$ encompassing $\mathcal{R}$ with a start time of $t_0$ is

$$\mathcal{S}(\mathbf{U}, \mathcal{R}) = \left\{(x_{\text{wet}}(x_0, y_0, t), y_{\text{wet}}(x_0, y_0, t)) \in \mathbb{R}^2 \, : \, (x_0, y_0) \in \mathcal{R}, t \in [t_0, t_{\text{wet}}(x_0, y_0)]\right\}. \tag{15}$$

This region encompasses approximate paths of least resistance from a flood to $\mathcal{R}$. It is clear that good locations for structural mitigation measures are likely to reside in $\mathcal{S}$.

A robust selection of $\mathcal{P}$ would account for the change in $\mathbf{U}$ with respect to a large set of feasible configurations. In an ideal setting, a good selection for $\mathcal{P}$ would thus be

$$\mathcal{P} = \bigcup_{\boldsymbol{\omega} \in \mathcal{F}} \bigcup_{a \in \mathcal{A}} \left\{(x, y) \in \mathcal{S}(\mathbf{U}, a) \, : \, \mathbf{U}_t + \mathbf{F}_x + \mathbf{G}_y = \mathbf{S}_R + \tilde{\mathbf{S}}_B + \tilde{\mathbf{S}}_n\right\}. \tag{16}$$

In practice, defining $\mathcal{P}$ as per equation (16) is nontrivial. First, each $a \in \mathcal{A}$ may be a set of infinitely many points. There are also infinitely many moments $t$ in a solution $\mathbf{U}$ to the shallow water equations. Most importantly, the union over all feasible configurations $(\omega_1, \omega_2, \ldots, \omega_m) = \boldsymbol{\omega} \in \mathcal{F}$ assumes knowledge of $\mathbf{U}$ for any such feasible configuration $(\omega_1, \omega_2, \ldots, \omega_m)$. For these reasons, an iteratively constructed definition of the pathtube-like region $\tilde{\mathcal{P}}$ is instead proposed, which approximately captures the features of some unknown larger $\mathcal{P}$ relevant to the OFMP (e.g., equation (16)).

From a numerical perspective, each $a \in \mathcal{A}$ is actually a polygon whose exterior connects a set of points $P_a$. Solutions to the OFMP are likely to intersect the pathlines from a flood to each of these points. Also, in practice, numerical solutions to the shallow water equations are discrete in space and time. Assuming that solutions are obtained for a set of time stamps $\mathcal{T}$ on a rectangular grid $G$, $t_{\text{wet}}$ is first redefined as

$$t_{\text{wet}}(x_0, y_0) := \min\{t \in \mathcal{T} \, : \, h_{i_0, j_0, t} \geq \epsilon_h\}, \tag{17}$$

where $(i_0, j_0)$ is the unique index of the cell in grid $G$ that contains the point $(x_0, y_0)$.

For each point along an asset exterior, a numerical representation of the pathline leading *to* that point is desired. To accomplish this, it is assumed that a pathline can be approximated as a set $\mathcal{L}$ of discrete points. These points can be generated by solving equations (14a) and (14b) using any suitable ordinary differential equation integration technique. In this study, suggestions from Telea, (2014; initially described for *streamlines*, which trace a static field) are used to compute pathlines according to the function $\textsc{ComputePathline}(\mathbf{U}, x_0, y_0)$, whose arguments denote a solution $\mathbf{U}$ to the shallow water equations and the $x$ and $y$ positions of a seed point, respectively. A complete description of this function is given in algorithm 3.

The definition of $\textsc{ComputePathline}$ enables the computation of a set of points $Q$ approximating the pathtube leading to a set of exterior asset points $P_a \in a \in \mathcal{A}$ via

$$Q(\mathbf{U}, P_a) = \bigcup_{(x_0, y_0) \in P_a} \textsc{ComputePathline}(\mathbf{U}, x_0, y_0). \tag{18}$$

Since pathtubes are curvilinear, typical geometries that envelope $Q$ (e.g., the convex hull) do not effectively summarize this set. For this reason, the notion of an *alpha shape* is introduced, which minimally encompasses points of $Q$ using straight lines. A discussion on alpha shapes can be found in Fischer (2000). In this study, Edelsbrunner's algorithm (Edelsbrunner et al., 1983), presented in algorithm 4, is used to compute alpha shapes. The function that computes this shape for a set $Q$ and alpha value $\alpha$ is denoted as $\textsc{AlphaShape}(Q, \alpha)$.

**Algorithm 2** InitializeRestriction: Returns the initial restricted positional set

```
1: function InitializeRestriction($B, n, \mathcal{A}, \alpha$)
2:     Solve $\mathbf{U}_t + \mathbf{F}_x + \mathbf{G}_y = \mathbf{S}_R + \mathbf{S}_B + \mathbf{S}_n$
3:     return $\bigcup_{a \in \mathcal{A}} \textsc{AlphaShape}\left(Q(\mathbf{U}, P_a), \alpha\right) \setminus \bigcup \mathcal{A}$
4: end function
```

The definition of the function AlphaShape finally allows for definition of the functions InitializeRestriction and UpdateRestriction. Both assume restrictions are the unions of alpha shapes approximating the pathtubes leading to each asset. The functions are described in algorithms 2 and equation (19), respectively. In algorithm 2, line 2, the shallow water equations are solved *without* the presence of structural mitigation measures. In line 3, the union of alpha shapes for all pathtubes leading to the assets $a \in \mathcal{A}$ is computed. Asset regions are then subtracted from this set to ensure structural measures do not overlap with asset locations.

The function UpdateRestriction using the pathline approach is defined as

$$\textsc{UpdateRestriction}(\mathbf{U}, \mathcal{A}, \mathcal{P}, \alpha) = \mathcal{P} \cup \left( \bigcup_{a \in \mathcal{A}} \textsc{AlphaShape}(Q(\mathbf{U}, P_a), \alpha) \right) \setminus \bigcup \mathcal{A}. \tag{19}$$

The majority of this function resembles algorithm 2, although the union of the current set and previous $\mathcal{P}$ is computed to encourage exploration of a more representative (i.e., expanded) search space. Moreover, as per algorithm 1, this function is only called as better solutions to the OFMP are obtained. This decreases the burden of computing pathtubes and alpha shapes on each iteration of the algorithm.

### 3.2. Sequential Optimization Algorithm

Due to the nonlinear sensitivity of flooding behavior with respect to mitigation efforts, predictable and incremental changes to solutions of the OFMP while increasing the number of mitigation measures, $m$, are not ensured. This may be undesirable from a planning perspective. A separate algorithm is thus proposed to induce a sequential solution to the OFMP, whereby solutions with $m = 2$ include those of $m = 1$, solutions with $m = 3$ include those of $m = 2$, and so on. This ensures increasing utility for configurations of increasing sizes. It also allows policymakers to more clearly understand the effects of budgetary constraints with respect to the overall structural flood mitigation efforts. The recursion to compute sequential solutions may be defined as

$$(B_i, n_i) = \textsc{SolveOFMP}\left(B_{i-1}, n_{i-1}, \mathcal{A}, 1, \frac{T_{\max}}{m}, \alpha\right), \tag{20}$$

where $B_0 = B$, $n_0 = n$, and the time for each subproblem is an equal portion of $T_{\max}$. In this paper, line 10 is eliminated from algorithm 1 when using the sequential approach, as the best placement for a *single* structural mitigation measure is likely to reside within the *initial* $\tilde{\mathcal{P}}$ computed on line 2. As a consequence, for each structural measure placed using the sequential approach, pathtubes are constructed only once.

## 4. Results

### 4.1. Model Relaxation

The proposed approach uses the open source `scipy.optimize.differential_evolution` (DE) and RBFOpt libraries to produce two separate implementations of GenerateSolution in algorithm 1 (Costa & Nannicini, 2018; Storn & Price, 1997). Both only include support for simple bounds like those indicated in constraint (9g). Thus, these implementations of GenerateSolution may generate configurations that are infeasible with respect to constraints (9c) through (9f). To overcome this, the OFMP defined by

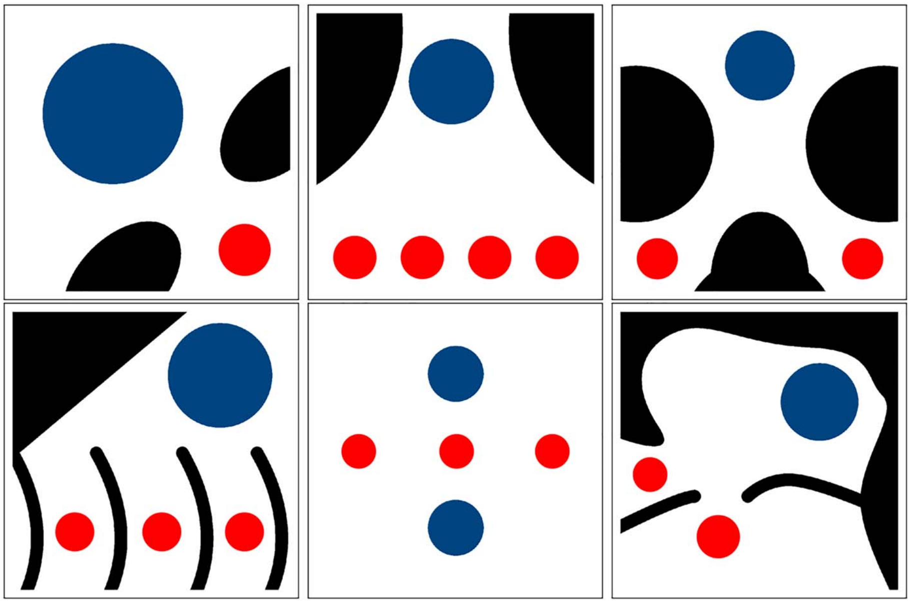

**Figure 1.** Pictorial descriptions of six simple Optimal Flood Mitigation Problem scenarios, ordered numerically (e.g., one in the upper left). Black represents nonzero topographic elevation (of height 1 m); blue represents nonzero initial water depth (of height 1 m); and red represents assets.

equations (9a) through (9h) is replaced with the relaxed formulation

$$\underset{\omega_1,\omega_2,\ldots,\omega_m}{\text{minimize}} \quad z\left(\omega_1,\omega_2,\ldots,\omega_m\right) = p_1 + p_2 + \sum_{a\in\mathcal{A}} \iint_a \max_t h(x,y,t)\,\mathrm{d}x\,\mathrm{d}y \tag{21a}$$

$$\text{subject to} \quad p_1 = c_1 \sum_{i=1}^{m} \min\left\{ \left\| (x,y) - \left(\lambda_i, \phi_i\right) \right\| \, : \, (x,y) \in \mathcal{P} \right\} \tag{21b}$$

$$p_2 = c_2 \sum_{i=1}^{m} \sum_{a\in\mathcal{A}} \iint_a \delta_B(\omega_i)\,\mathrm{d}x\,\mathrm{d}y + c_3 \sum_{i=1}^{m} \sum_{a\in\mathcal{A}} \iint_a \delta_n(\omega_i)\,\mathrm{d}x\,\mathrm{d}y \tag{21c}$$

$$\mathbf{U}_t + \mathbf{F}_x + \mathbf{G}_y = \mathbf{S}_R + \tilde{\mathbf{S}}_B + \tilde{\mathbf{S}}_n \tag{21d}$$

$$\delta_B(\omega_i)(x,y) = \begin{cases} \bar{b}_i \ \text{ for } \begin{cases} |(x-\phi_i)\cos\theta_i - (y-\lambda_i)\sin\theta_i| \le \frac{\ell}{2} \\ |(x-\phi_i)\sin\theta_i + (y-\lambda_i)\cos\theta_i| \le \frac{w}{2} \end{cases} \forall i \in \mathcal{M} \\ 0 \ \ \text{otherwise} \end{cases} \tag{21e}$$

$$\delta_n(\omega_i)(x,y) = \begin{cases} \bar{n}_i \ \text{ for } (x-\phi_i)^2 + (y-\lambda_i)^2 \le r^2 \\ \qquad\qquad\qquad\qquad\qquad\qquad \forall i \in \mathcal{M} \\ 0 \ \ \text{otherwise} \end{cases} \tag{21f}$$

$$\lambda_{lb} \le \lambda_i \le \lambda_{ub}, \phi_{lb} \le \phi_i \le \phi_{ub}, 0 \le \theta_i \le \pi, \forall i \in \mathcal{M}. \tag{21g}$$

In equation (21a), two penalty terms are included in the objective to capture infeasibilities in constraints (9c) through (9f). The first penalty, $p_1$, is defined in constraint (21b) and denotes the sum of all minimum distances between each measure's centroid and the nearest point of the restricted positional set $\mathcal{P}$. This term is scaled by the constant $c_1$, taken in this study to be equal to one. The second penalty, $p_2$, is defined in constraint (21c). Here the first term denotes the net modified elevation volume over all asset regions, and the second term denotes the net change in roughness over all asset regions. These terms are scaled by the constants $c_2$ and $c_3$, respectively. Herein, both are taken to be $(\Delta r)^{-2}$, where $\Delta r$ is the spatial resolution of the discretization.

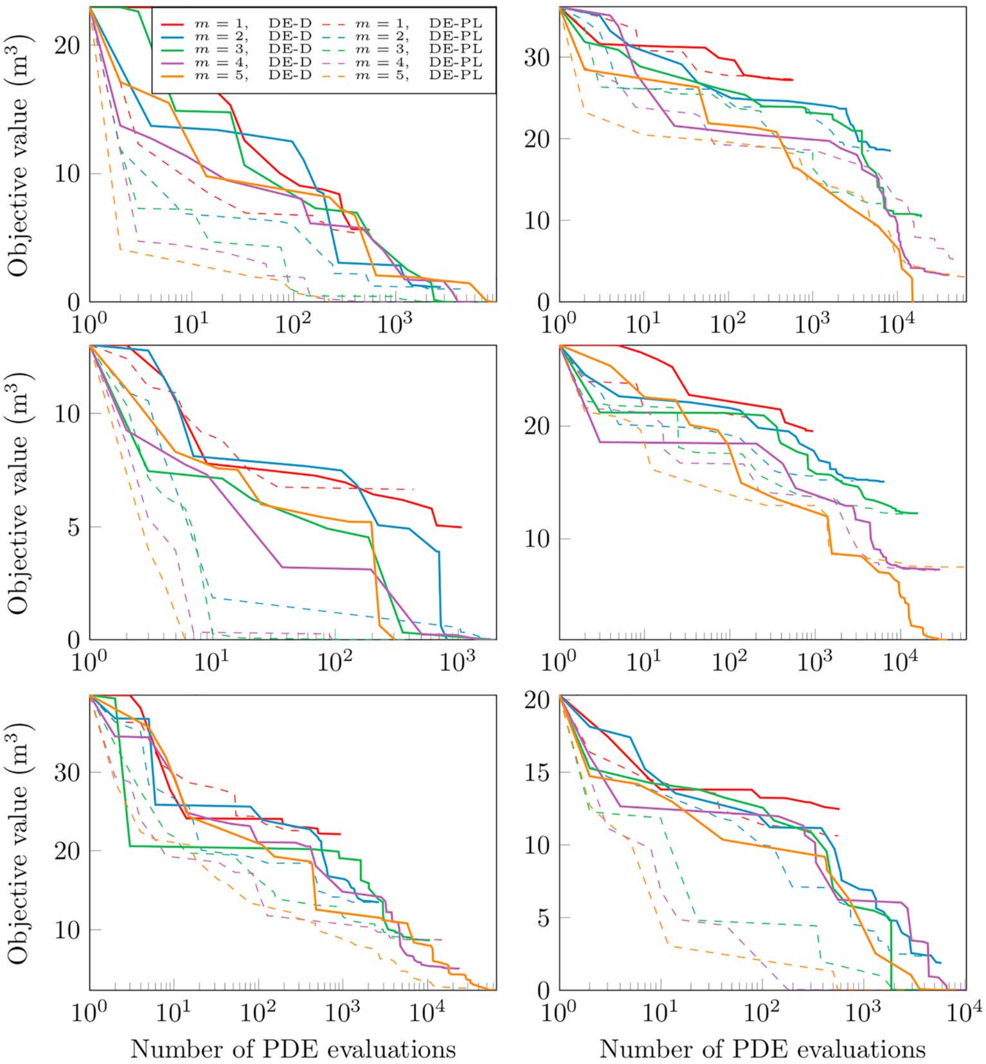

**Figure 2.** Comparison of objective value versus number of PDE evaluations for scenarios 1 through 6, using DE-D and DE-PL for configurations of one through five walls. PDE = partial differential equation.

### 4.2. Experimental Setting

For simplicity, sections 4.3 through 4.4.3 focus on OFMPs where only wall-type measures are considered (i.e., $\bar{b}_i > 0$), while section 4.4.4 presents an algorithmic proof of concept where only revegetation-type measures are considered (i.e., $\bar{n}_i > 0$). For each experiment, algorithm 1 was limited to 1 day of wall-clock time. When using DE, population sizes of 45 $m$ ($\bar{b}_i > 0$) and 30 $m$ ($\bar{n}_i > 0$) were employed; trial solutions were computed as the best solution plus scaled contributions of two random candidates; the mutation constant varied randomly within $[0.5, 1.0)$; and the recombination constant was set to 0.9. When using the direct InitializeRestriction and UpdateRestriction methods, Latin hypercube sampling was used to initialize the population. When using the pathline-based methods, the population was initialized via random sampling over the initial restricted set (i.e., $\tilde{\mathcal{P}}$) and $\theta_i \in [0, \pi]$. When using RBFOpt, the `sampling` method was used; most other parameters were left unchanged.

For computational considerations, if the configuration proposed by GenerateSolution was feasible, the shallow water equations (i.e., constraint (21d)) were solved using the proposed configuration. Otherwise, a solution containing *no structural mitigation measures* was referenced. That is, $\tilde{S}_B$ was replaced with $S_B$, and $\tilde{S}_n$ was replaced with $S_n$. To solve these PDEs, the open-source surface-water modeling software Nuflood (Tasseff & Judi, 2016) was used, where the shallow water equations are spatially discretized according to the scheme described by Kurganov and Petrova (2007).

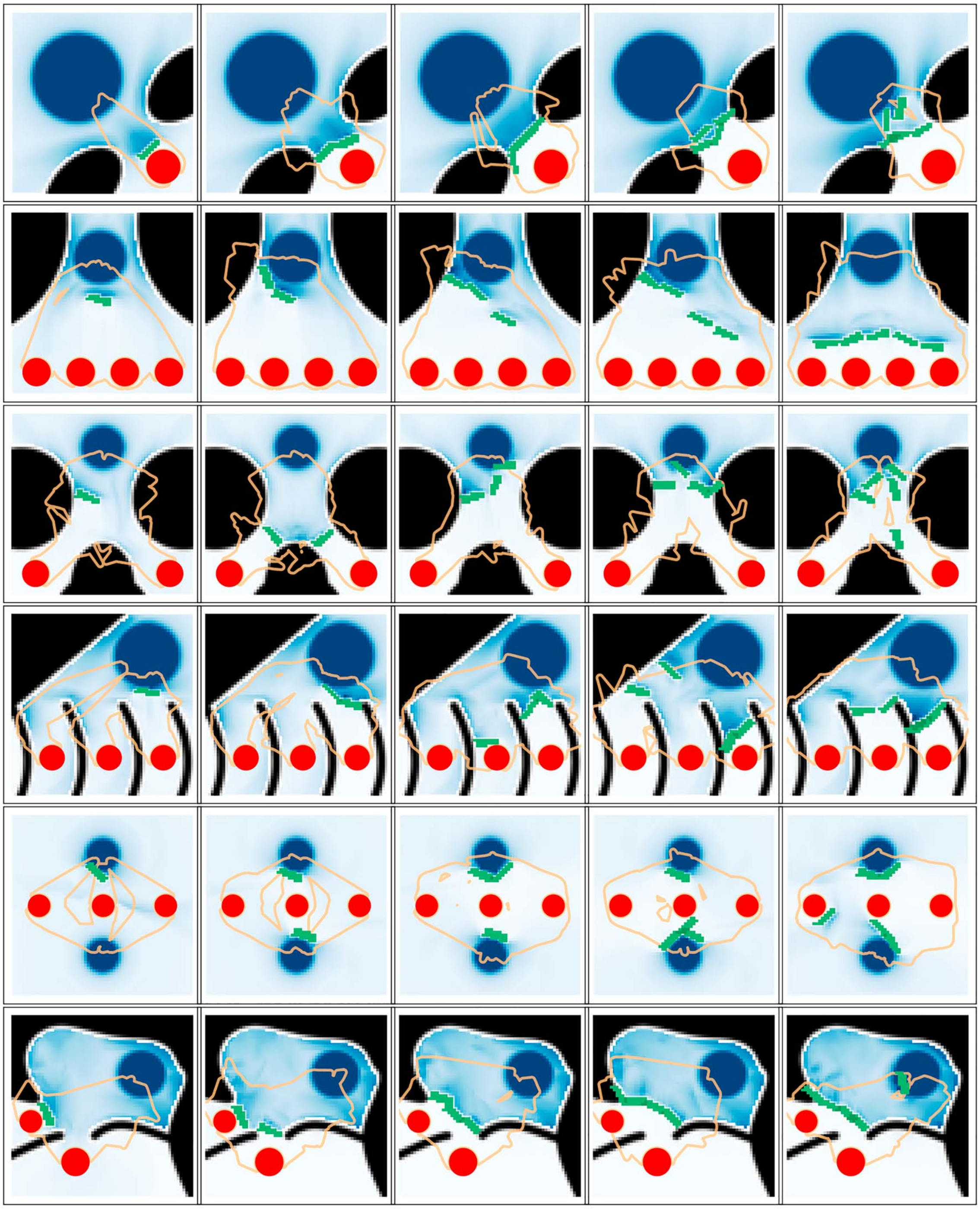

**Figure 3.** Best obtained elevations and maximum depths for configurations of one through five walls for the highly simplified flood scenarios, referenced as scenarios 1 through 6 (vertically). Darker blue corresponds to larger maximum depths; black corresponds to nonzero portions of the initial topographic elevation field; green corresponds to elevation additions via the placement of walls; and red corresponds to asset locations. The orange lines represent the exteriors of the final computed restricted positional sets $\tilde{\mathcal{P}}$ in algorithm 1.

Each experiment was conducted on one Intel Xeon E5-2695 V4 CPU containing 18 cores at 2.1 GHz and 125 GB of RAM. Nuflood was compiled in single-precision mode using the Intel C++ Compiler, version 17.0.1. The remainder of algorithm 1 was implemented in Python 3.6. Compared to the PDE evaluations, these other portions of the algorithm were found to be computationally negligible.

**Table 1**
*Table Comparing Objectives Using the (Direct) RBFOpt-D and (Pathline-Based) RBFOpt-PL Solvers Over 10 Random Seeds, With the Number of Walls (m) Ranging From 1 to 10, as Discussed in Section 4.4*

| | RBFOpt-D | | | | RBFOpt-PL | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $m$ | Mean | Min | Max | SD | Mean | Min | Max | SD | Mean improvement |
| 1 | 165.93 | 159.63 | 170.25 | 3.36 | 162.81 | 159.63 | 167.59 | 2.61 | 1.88% |
| 2 | 147.96 | 105.34 | 166.60 | 20.14 | 111.02 | 95.98 | 130.88 | 10.74 | 24.96% |
| 3 | 144.93 | 111.87 | 164.10 | 15.44 | 89.22 | 77.56 | 93.62 | 5.40 | 38.44% |
| 4 | 128.52 | 105.81 | 159.12 | 14.21 | 81.33 | 51.63 | 97.83 | 13.47 | 36.72% |
| 5 | 135.94 | 128.25 | 144.42 | 5.03 | 62.38 | 26.40 | 80.40 | 15.67 | 54.11% |
| 6 | 122.24 | 98.38 | 140.19 | 14.22 | 59.13 | 41.40 | 71.25 | 9.07 | 51.63% |
| 7 | 119.14 | 81.14 | 145.80 | 18.65 | 51.11 | 29.13 | 66.20 | 13.08 | 57.10% |
| 8 | 102.08 | 78.65 | 122.69 | 16.22 | 43.28 | 27.82 | 57.35 | 9.40 | 57.60% |
| 9 | 107.40 | 81.93 | 127.53 | 15.52 | 42.73 | 19.00 | 53.26 | 11.19 | 60.21% |
| 10 | 104.90 | 77.17 | 124.13 | 16.75 | 34.72 | 21.75 | 42.74 | 7.60 | 66.90% |

*Note*. Values are scaled by a factor of $10^{-4}$. SD = standard deviation.

### 4.3. Simplified Circular Dam Break Scenarios

To compare the two positional restriction methodologies described in section 3.1, six simple OFMP scenarios were constructed. All were intended to have human intuitive solutions, that is, optimal placement of structural mitigation measures could be inferred from a basic understanding of flood propagation. These scenarios are displayed pictorially in Figure 1. In each scenario, under the influence of gravity, the initial volume of water (colored with blue) is propagated outward; without mitigation measures, this water comes into contact with assets (colored with red), flooding them.

Each of the six scenarios was modeled using a spatial resolution of 1 m and 64 × 64 grid cells. The ground surface was assumed to be frictionless; critical depth boundary conditions were employed; and a simulation duration of 100 s was used. When necessary to compute pathlines, intermediate PDE solution data were reported for every 1 s of simulation time. In the experiments performed, each of the corresponding OFMPs was solved with the number of walls, $m$, ranging from one to five. Wall widths, lengths, and heights were fixed to 2.5, 8.0, and 1.0 m, respectively. Finally, all experiments were performed using a single fixed random seed.

In Figure 2, for each experiment, the objective behavior is plotted against the number of PDE evaluations required to reach that objective. These behaviors are compared for the direct differential evolution solver (DE-D) and its pathline-based counterpart (DE-PL). The DE-PL solver was generally able to find good solutions faster and improve upon them more rapidly, especially for configurations involving larger numbers of walls. However, there were some instances where the DE-D solver produced higher quality solutions than the DE-PL solver, for example, when optimizing the configuration of five walls in scenario 4. These anomalies could be a consequence of the random nature of the DE algorithm; they could also be due to the DE implementation's tendency to terminate once a population has sufficiently stabilized.

In Figure 3, the best obtained wall configurations using DE-PL are displayed pictorially for all pairs of scenarios and numbers of mitigation measures. The configurations resemble what might be intuited by a human. When applicable, configurations are nonoverlapping and well-connected. As the number of walls varies, configurations also show interesting nonincremental behavior. For example, in the first scenario, walls are initially placed close to the asset; as the number of walls increases, they are placed farther away to form connections with existing topographic features. However, such nonsequential behavior may be undesirable from a planning perspective.

### 4.4. Hypothetical Dam Break Scenario From Theme C of the Twelfth International Benchmark Workshop on Numerical Analysis of Dams

This section focuses on demonstrating the merits of the sequential optimization algorithm using the hypothetical dam break defined in Theme C of the Twelfth International Benchmark Workshop on Numerical Analysis of Dams (ICOLD 2013; Judi, Pasqualini, et al., 2014). To simulate this scenario, the dam break was

**Table 2**

*Table Comparing Objective Values Obtained Using the (Direct) DE-D and (Pathline-Based) DE-PL Solvers Over 10 Random Seeds, With the Number of Walls (m) Ranging From 1 to 10*

| | DE-D | | | | DE-PL | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *m* | Mean | Min | Max | SD | Mean | Min | Max | SD | Mean improvement |
| 1 | <u>162.18</u> | **158.59** | 167.59 | 3.85 | 163.86 | 159.63 | 170.07 | 4.52 | −1.04% |
| 2 | 99.49 | **84.87** | 104.24 | 6.29 | 102.61 | 94.24 | 105.39 | 3.19 | −3.13% |
| 3 | 66.78 | 39.56 | 119.20 | 27.76 | <u>50.75</u> | **33.36** | 65.93 | 14.80 | 24.01% |
| 4 | 101.06 | 79.57 | 134.61 | 20.57 | 31.84 | 17.51 | 56.87 | 13.20 | 68.49% |
| 5 | 115.43 | 100.56 | 145.39 | 16.04 | 22.74 | 12.82 | 36.39 | 8.04 | 80.30% |
| 6 | 124.15 | 101.22 | 145.75 | 15.85 | 24.79 | 7.24 | 60.66 | 14.87 | 80.03% |
| 7 | 129.17 | 110.94 | 153.14 | 12.52 | 18.14 | 5.59 | 26.77 | 6.64 | 85.96% |
| 8 | 125.51 | 96.68 | 144.92 | 16.47 | 14.02 | 4.03 | 19.74 | 5.59 | 88.83% |
| 9 | 129.23 | 99.39 | 146.58 | 14.02 | 17.83 | 8.56 | 22.46 | 4.85 | 86.20% |
| 10 | 119.09 | 86.16 | 140.87 | 17.91 | 19.68 | 10.43 | 30.38 | 5.61 | 83.47% |

*Note.* Values are scaled by a factor of $10^{-4}$. Best objectives over all seeds and solvers in Tables 1 through 4 are denoted in bold, while best *mean* objectives are underlined. SD = standard deviation.

modeled as a point source with time-dependent discharge. The initial topographic elevation field (with the dam excluded) was provided by the workshop and resampled from a resolution of 10 to 90 m to ease computational burden. The Manning's roughness coefficient was set to 0.035; critical depth boundary conditions were employed; a duration of 12 hr was used; and when necessary to compute pathlines, PDE solution data were reported every 10 min of simulation time.

Asset locations and sizes were selected to increase the difficulty of the OFMP, with two assets placed near the primary channel of the scenario and three placed farther away. The experimental setup remained similar to that described in section 4.3. However, in this case, the number of walls ranged from 1 to 10, while wall widths, lengths, and heights were fixed to 250, 1,000, and 10 m, respectively. To compare differences in OFMP solver performance, each solver was executed using 10 different random seeds for each possible number of walls. In total, the experiments described in this subsection thus required nearly 600 days of compute time.

### 4.4.1. Pathline-Based Algorithm Results

To confirm the effectiveness of the pathline-based solvers, two implementations of algorithm 1 using RBFOpt were benchmarked. In Table 1, the objective behavior of the pathline-based solver (RBFOpt-PL) is compared against its direct counterpart (RBFOpt-D). The pathline-based solver clearly outperformed

**Table 3**

*Table Comparing Objective Values Obtained Using the (Pathline-Based) DE-PL and (Direct Sequential) DE-D-S Solvers Over 10 Random Seeds, With the Number of Walls (m) Ranging From 1 to 10*

| | DE-PL | | | | DE-D-S | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *m* | Mean | Min | Max | SD | Mean | Min | Max | SD | Mean improvement |
| 1 | 163.86 | 159.63 | 170.07 | 4.52 | <u>162.18</u> | **158.59** | 167.59 | 3.85 | 1.02% |
| 2 | 102.61 | 94.24 | 105.39 | 3.19 | 102.38 | 100.97 | 103.68 | 0.87 | 0.22$ |
| 3 | <u>50.75</u> | **33.36** | 65.93 | 14.80 | 64.98 | 48.97 | 81.93 | 14.71 | −28.06% |
| 4 | 31.84 | 17.51 | 56.87 | 13.20 | 41.04 | 26.07 | 58.60 | 14.88 | −28.89% |
| 5 | 22.74 | 12.82 | 36.39 | 8.04 | 23.22 | 17.45 | 34.87 | 6.18 | −2.11% |
| 6 | 24.79 | 7.24 | 60.66 | 14.87 | 14.94 | 11.56 | 18.39 | 2.26 | 39.76% |
| 7 | 18.14 | 5.59 | 26.77 | 6.64 | 11.14 | 4.10 | 14.53 | 2.87 | 38.60% |
| 8 | 14.02 | 4.03 | 19.74 | 5.59 | 9.42 | 6.02 | 16.64 | 3.30 | 32.81% |
| 9 | 17.83 | 8.56 | 22.46 | 4.85 | 4.72 | **0.00** | 9.91 | 3.74 | 73.52% |
| 10 | 19.68 | 10.43 | 30.38 | 5.61 | 3.07 | **0.00** | 9.29 | 3.10 | 84.38% |

*Note.* Values are scaled by a factor of $10^{-4}$. Best objectives over all seeds and solvers in Tables 1 through 4 are denoted in bold, while best *mean* objectives are underlined. SD = standard deviation.

**Table 4**
*Table Comparing Objectives Obtained Using the (Direct Sequential) DE-D-S and (Pathline-Based Sequential) DE-PL-S Solvers Over 10 Random Seeds, With the Number of Walls (m) Ranging From 1 to 10*

| | DE-D-S | | | | DE-PL-S | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *m* | Mean | Min | Max | SD | Mean | Min | Max | SD | Mean improvement |
| 1 | 162.18 | **158.59** | 167.59 | 3.85 | 163.61 | 159.63 | 167.59 | 4.19 | −0.88% |
| 2 | 102.38 | 100.97 | 103.68 | 0.87 | 98.25 | 86.78 | 105.84 | 7.45 | 4.04% |
| 3 | 64.98 | 48.97 | 81.93 | 14.71 | 56.74 | 35.73 | 86.12 | 13.92 | 12.69% |
| 4 | 41.04 | 26.07 | 58.60 | 14.88 | 27.18 | **14.55** | 56.37 | 11.81 | 33.77% |
| 5 | 23.22 | 17.45 | 34.87 | 6.18 | 16.32 | **8.38** | 25.35 | 5.25 | 29.73% |
| 6 | 14.94 | 11.56 | 18.39 | 2.26 | 10.63 | **3.92** | 16.96 | 5.29 | 28.82% |
| 7 | 11.14 | 4.10 | 14.53 | 2.87 | 7.02 | **0.13** | 15.00 | 5.08 | 36.97% |
| 8 | 9.42 | 6.02 | 16.64 | 3.30 | 4.53 | **0.00** | 9.68 | 3.81 | 51.90% |
| 9 | 4.72 | **0.00** | 9.91 | 3.74 | 3.36 | **0.00** | 7.86 | 3.43 | 28.83% |
| 10 | 3.07 | **0.00** | 9.29 | 3.10 | 2.59 | 0.00 | 6.30 | 2.60 | 15.76% |

*Note*. Values are scaled by a factor of $10^{-4}$. Best objectives over all seeds and solvers in Tables 1 through 4 are denoted in bold, while best *mean* objectives are underlined. SD = standard deviation.

RBFOpt-D in nearly all instances; for example, it resulted in smaller minima, means, and standard deviations. The single exception appears to be for $m = 1$, where the direct solver produced an equivalent minimum to the pathline-based solver. Nonetheless, on average, the pathline-based solver provided a 45% improvement over the direct solver, with generally larger improvements for greater numbers of walls. This improvement was computed as

$$\text{percentage improvement} = 100\left(\frac{a-b}{a}\right), \tag{22}$$

where, here, $a$ and $b$ represent the mean objective values obtained from the RBFOpt-D and RBFOpt-PL solvers. The same metric is also used throughout Tables 2–4.

A similar comparison is made between DE-D and DE-PL in Table 2. Again, the pathline-based solver (DE-PL) outperformed its direct counterpart (DE-D) in nearly all metrics, providing an overall mean improvement of 59%. The pathline-based solver also displayed mostly monotonic decreases in the objective as the number of walls increased, while the objectives associated with the direct solver generally *increased* as the number of walls increased. However, note that for small numbers of walls (i.e., one and two), the direct DE solver outperformed its pathline-based counterpart. This could be a consequence of the more complicated objective penalty in constraint (21b) when $\mathcal{P}$ is restricted. For example, Deb (2000) describes various means by which penalty-based genetic algorithms can result in nonoptimal solutions. Nonetheless, overall, the direct penalization method considered herein works well.

It is important to note the differences between the RBFOpt-based and DE-based solvers benchmarked in Tables 1 and 2, respectively. In general, DE-PL greatly outperformed both RBFOpt-based solvers; for example, DE-PL provided a 47% mean improvement over RBFOpt-PL. These differences could be for multiple reasons. For example, there are many more hyperparameters associated with RBFOpt than DE; more careful tuning may have increased RBFOpt's convergence. Furthermore, RBFOpt's `sampling` search strategy was used to show the efficacy of the pathline-based approach when applied to other (nonevolutionary) search techniques; the solver software may have performed more favorably using some other strategy.

Figure 4 displays the best obtained wall configuration for each possible number of walls using the DE-PL solver. Structure placement appears highly nonincremental as the number of walls increases, especially for smaller numbers of walls. Also, when optimizing for a number of walls greater than eight, solutions generally deteriorated, indicating the search space becomes prohibitively large. Interestingly, the size of the restricted set $\tilde{\mathcal{P}}$ did not increase substantially as the configuration size grew. Finally, in Figure 5, the best obtained solution for 10 walls using DE-D is displayed; this underscores the difficulty of such a problem when applying a conventional algorithm.

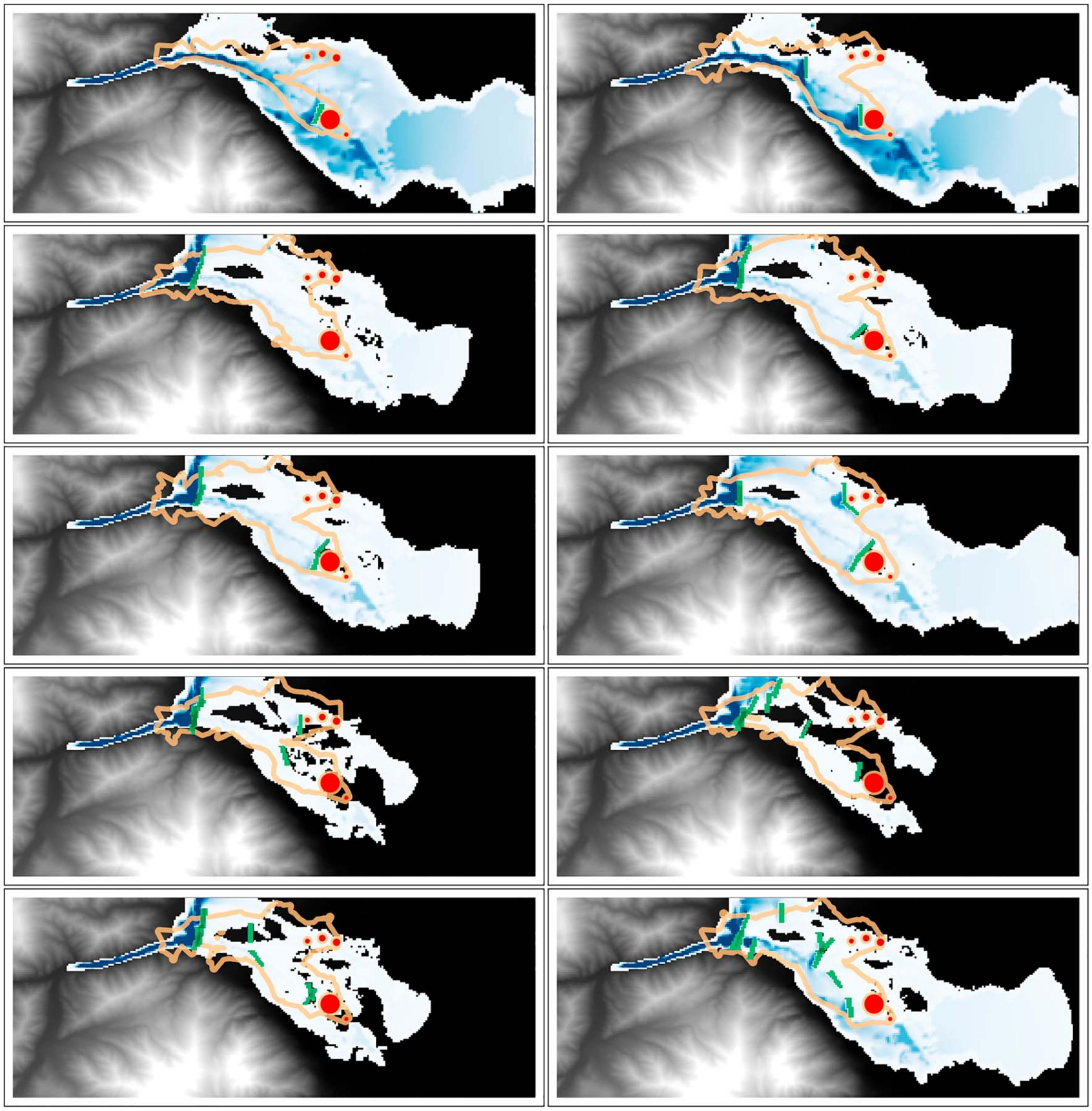

**Figure 4.** Best obtained elevations and maximum depths for configurations of 1 through 10 walls for the ICOLD 2013 dam failure scenario using DE-PL. Darker blue corresponds to larger maximum depths; gray corresponds to the initial topographic elevation field; green corresponds to elevation additions via the placement of walls; and red corresponds to asset locations. Orange lines represent the exteriors of the final restricted positional sets $\tilde{P}$ in algorithm 1.

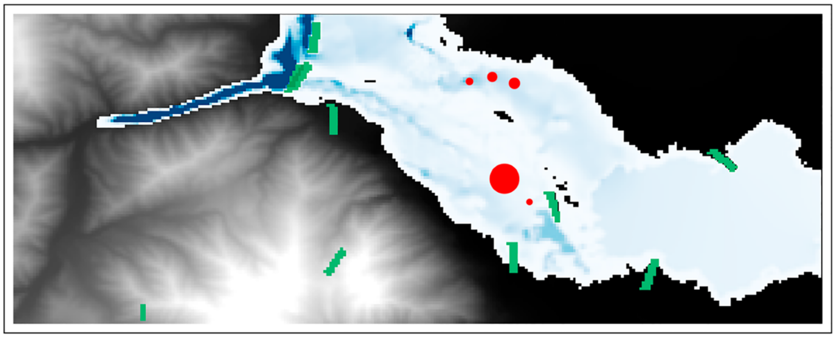

**Figure 5.** Best solution in a setting equivalent to Figure 4 for 10 walls using DE-D.

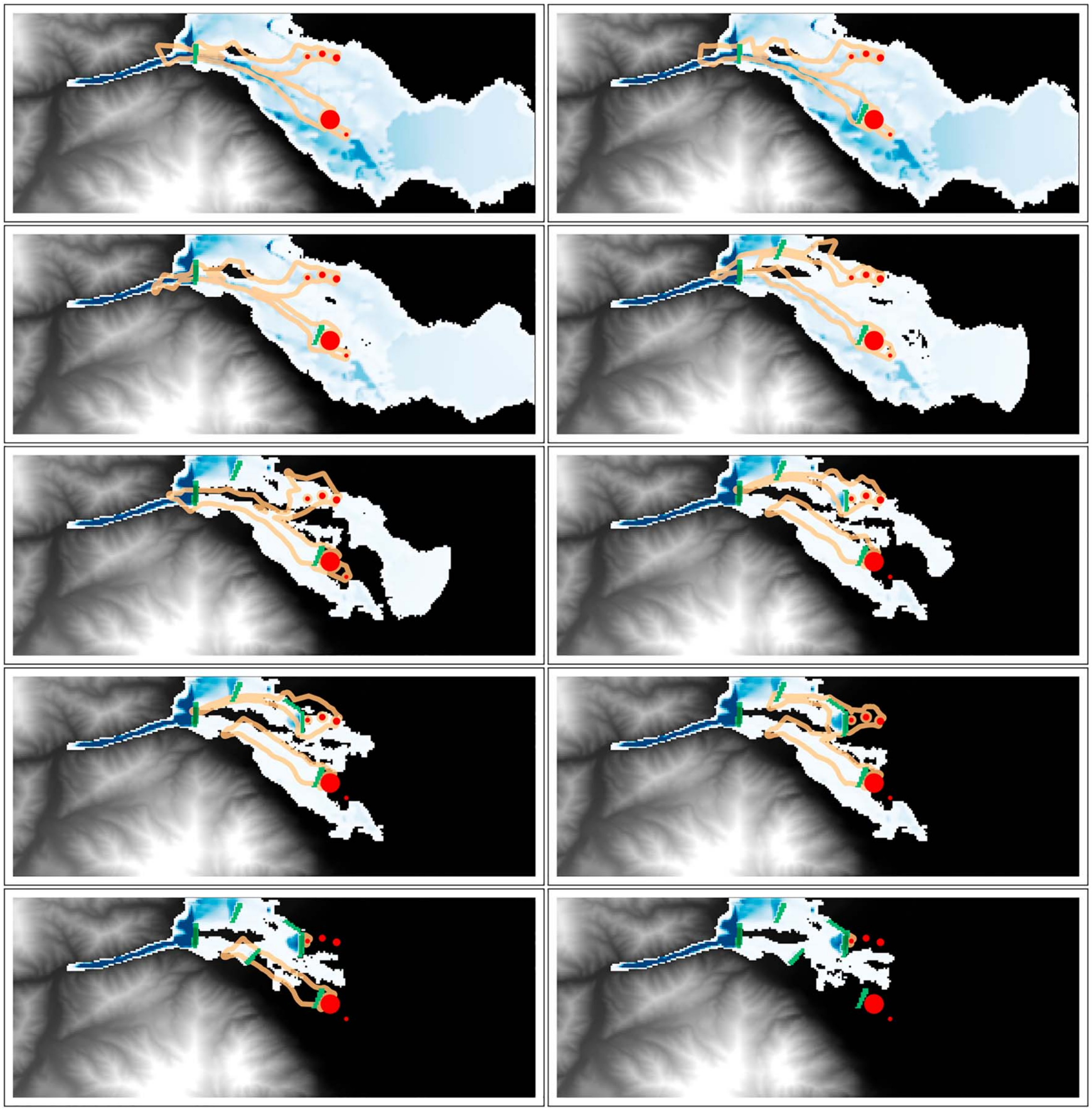

**Figure 6.** Best obtained elevations and maximum depths for configurations of 1 through 10 walls for the ICOLD 2013 dam failure scenario using DE-PL-S. Darker blue corresponds to larger maximum depths; gray corresponds to the initial topographic elevation field; green corresponds to elevation additions via the placement of walls; and red corresponds to asset locations. Orange lines represent the exteriors of the restricted positional sets $\tilde{\mathcal{P}}$ initialized in algorithm 1.

### 4.4.2. Sequential Algorithm Results

To counteract the degradation of solutions for larger configurations, the sequential approach presented in section 3.2 was benchmarked in a similar setting. In Table 3, performance of the direct sequential DE solver (DE-D-S) is compared against DE-PL. Interestingly, DE-D-S performed much better than DE-PL for configurations containing many walls, providing improvements as large as 84%. This result indicates the difficulty in optimizing configurations of multiple structural mitigation measures simultaneously, which may lead to a worse objective when running the previous algorithms with more measures. Note, however, that the sequential approach generally did not provide improvements over DE-PL for configurations consisting of three, four, and five walls. These results indicate that sequential optimization is most beneficial when the number of structural measures becomes larger (e.g., greater than five).

Finally, a comparison between DE-D-S and the sequential DE-PL solver (DE-PL-S) is made in Table 4. On average, DE-PL-S provided a 24% improvement over its direct counterpart. The sequential DE-PL solver was also capable of finding a solution which completely mitigated the flood using a smaller structural budget. That is, the direct sequential solver found a totally mitigating solution at $m = 9$, but DE-PL-S accomplished

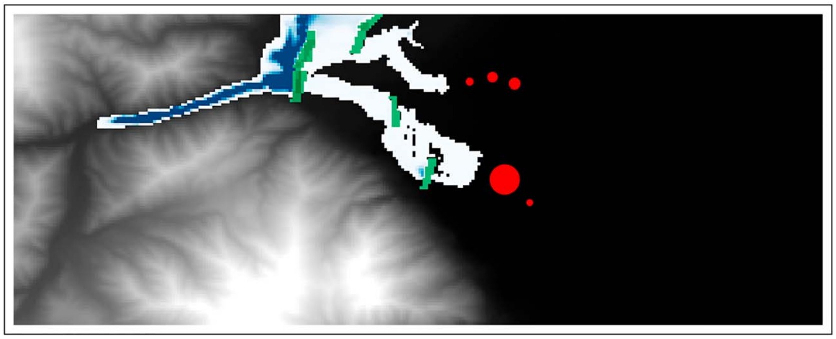

**Figure 7.** Best solution in a setting equivalent to Figure 6 for 10 walls using DE-D-S.

this for $m = 8$. Interestingly, however, for $m = 10$, DE-D-S found a totally mitigating solution, whereas DE-PL-S only found a *nearly* mitigating solution. This again may be a consequence of the relatively small number of experiments performed. Overall, except for small $m$ (i.e., $m = 1$), the pathline-based sequential approach appears highly superior to the direct sequential approach. This result indicates that DE-PL-S serves as a good general purpose OFMP solver.

Figure 6 displays the 10 incremental configurations obtained via DE-PL-S for $m = 10$ and the random seed that gave the minimum corresponding objective in Table 4. The ultimate solution for $m = 10$ shows remarkable similarity to the solution obtained via DE-PL for $m = 8$, as shown in Figure 4. That is, both solutions appear to exploit the critical depth boundary condition to divert water outside of the domain's uppermost boundary. However, the sequential solution appears to place a larger number of walls in more intuitive locations. Similarly, as displayed by the solution for $m = 10$ shown in Figure 7, DE-D-S also produced a configuration which diverted flow out of the domain's uppermost boundary, although one wall was placed extraordinarily near this boundary. Such solutions may not be possible when using the pathline approach, as pathlines typically do not reside near domain boundaries.

### 4.4.3. Summary of Algorithm Comparisons

Tables 1 through 4 compare the performance of solvers against one another. Within these tables, the best objectives over *all* seeds and solvers are denoted in bold, while the best *mean* objectives are underlined. It is first apparent that for $m \in \{1, 2\}$, minima were obtained through use of DE-D. Good mean objectives were also obtained using this solver. This result indicates that direct local search algorithms are capable of performing well on OFMPs that contain a small number of structural measures. It also implies that more careful tuning of these algorithms may hold great promise.

For $m = 3$, DE-PL performed most favorably, providing the best overall and best mean objectives. This implies for a moderate number of structural measures, DE-PL effectively uses pathlines to restrict the search space. Moreover, if the optimal solution is nonincremental, it is capable of finding solutions that sequential

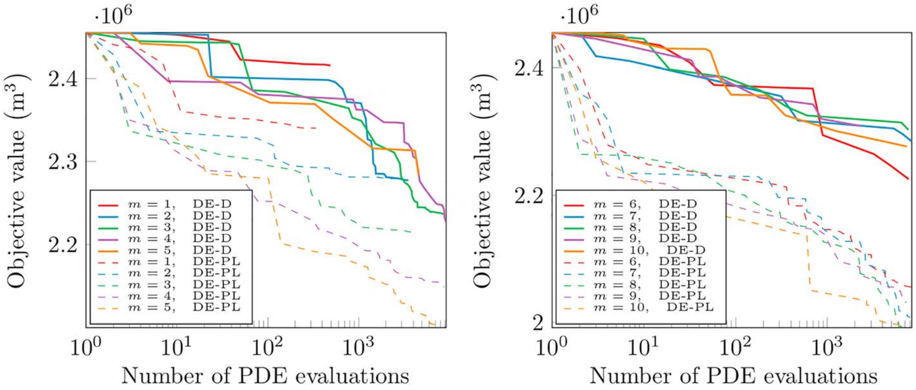

**Figure 8.** Comparison of objective versus number of partial differential equation evaluations for the Optimal Flood Mitigation Problem in section 4.4.4, using DE-D and DE-PL for configurations of 1 through 10 revegetation projects. PDE = partial differential equation.

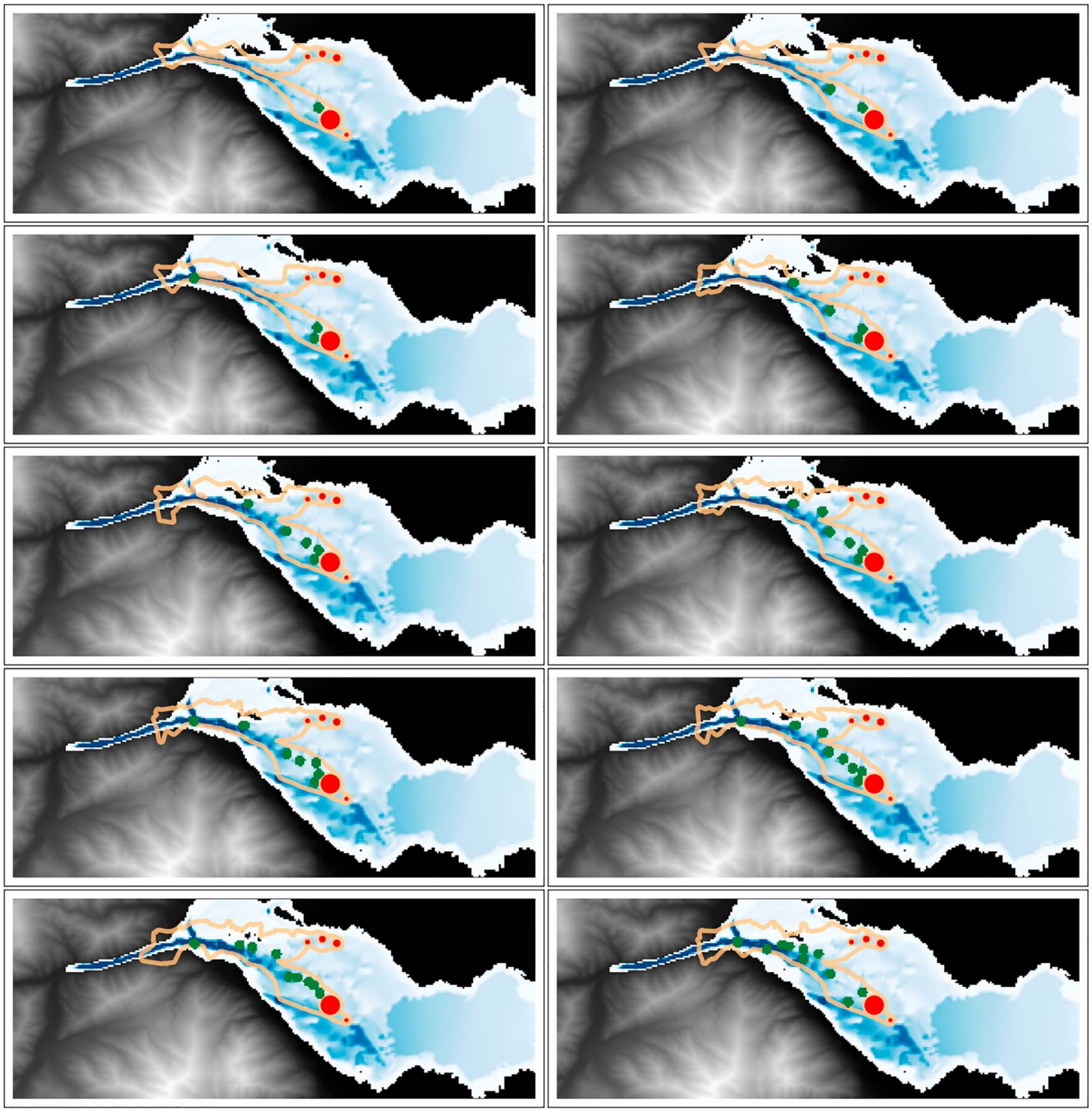

**Figure 9.** Revegetation locations and maximum depths for configurations of 1 through 10 projects for the ICOLD 2013 dam failure scenario using DE-PL. Darker blue corresponds to larger maximum depths; gray corresponds to the initial topographic elevation field; green corresponds to the placement of revegetation projects; and red corresponds to asset locations. Orange lines represent the exteriors of the restricted positional sets $\tilde{\mathcal{P}}$ initialized in algorithm 1.

approaches cannot. However, for $m > 3$, DE-PL-S performs most favorably, indicating a combination of pathline-based and sequential approaches is needed to solve challenging problems.

### 4.4.4. Proof of Concept for Soft Structural Mitigation Measures

Sections 4.3 through 4.4.3 focus on OFMPs designed to configure the placement of *hard* structural mitigation measures (i.e., $\bar{b}_i > 0$ and $\bar{n}_i = 0$). However, it is important to emphasize that the problem formulations and techniques described throughout sections 2, 3, and 4.1 are not limited to such measures. To exemplify this, a proof of concept employing only soft structural measures is assessed. In particular, an OFMP taking the form of equations (21a) through (21g) is proposed that optimizes the configuration of $m$ revegetation projects (i.e., $\bar{n}_i > 0$ and $\bar{b}_i = 0$).

Using the ICOLD 2013 scenario, the above problem was constructed for a number of revegetation projects ranging from 1 to 10. Each revegetation project was assumed to have a radius of 250 m and increased the Manning's roughness coefficient in the project region from 0.035 to 0.123. An experimental setting equivalent to that described in section 4.2 was used. However, in these experiments, only the DE and DE-PL solvers were compared. Furthermore, only a single random seed was used.

In Figure 8, for each experiment, the objective behavior is plotted against the number of PDE evaluations required to reach that objective. The DE-PL solver was generally able to improve upon solutions more rapidly, especially for configurations involving larger numbers of revegetation projects. These results mimic the behaviors of Figure 2 and Tables 1  and 2. That is, for smaller numbers of projects, the direct algorithm is sufficient, but for larger numbers of projects, the pathline-based algorithm is needed to obtain meaningful solutions.

Finally, in Figure 9, the configurations using DE-PL are displayed pictorially for all pairs of scenarios and numbers of projects. The results are highly intuitive upon greater inspection. First, many of the projects appear to be placed in locations that interdict the initial flood wave. More interestingly, many are located along the primary channels of the scenario domain, where larger velocities would occur. This makes sense, as the bed shear stress source terms are proportional to the square of velocity; measures that increase roughness are thus highly beneficial in these regions.

## 5. Conclusion

This study addressed the difficult problem of designing structural FRM strategies for use within the risk assessment process. To this end, an optimization-based decision support approach was proposed for designing mitigation strategies. A number of numerical methodologies were developed that generally function through modifying the bed slope and bed shear stress source terms of the 2-D shallow water equations. However, the methodologies are sufficiently general to modify other source terms (e.g., adjustment of soil properties that affect $\mathbf{S}_R$ via infiltration) or even supplant the shallow water equations with a different physical model.

To formalize the mitigation task, the OFMP was introduced. To solve practical problems of this type, a time-limited search-based optimization algorithm was developed. Within this algorithm, three approaches to generate solutions were explored: a direct approach using only derivative-free optimization, an augmented approach using pathlines to restrict the search space, and a sequential optimization approach. The latter two were largely successful, depending on the number of mitigation measures defined in the OFMP. Overall, the nonsequential and sequential pathline-based differential evolution approaches provided average improvements of 59% and 65% over their direct counterpart, respectively. Results illustrate the first meaningful solutions to large-scale optimization problems of this type.

Future work should seek to increase and prove the applicability of the approach to realistic flood scenarios. First, it should seek to generalize the approach by benchmarking performance on a greater number of real-world flood scenarios. Second, it should address the inherent uncertainty in flood scenario parameterizations (e.g., topographic elevation, dam breach parameterization, and bed friction). To this end, a stochastic optimization approach should be developed to ensure solutions are distributionally robust from a planning perspective. Third, a human behavioral study should be conducted to compare the utility of the optimization approach presented herein with the typically manual process used in simulation-based mitigation design. Fourth, algorithmic enhancements should be made to increase the realism of mitigation designs. For example, flood walls used in the numerical experiments were overtoppable. This may not be realistic from a flood risk management perspective. Such realism can be embedded within the optimization problem in the form of additional penalties (e.g., when walls are overtopped, a penalty is introduced) or additional constraints. Finally, the approach should be extended to solve OFMPs for scenarios that require modeling at finer spatial resolutions. To accomplish this, a multiresolution approach should be developed, where the spatial resolution of a flood scenario is iteratively refined as optimization progresses. This work would be valuable for realistic scenarios, where fine resolution details are sometimes necessary to accurately predict flooding behavior.

## Appendix A: ComputePathline(U, $x_0, y_0$)

In algorithm 3, line 2, the pathline and current pathline segment lengths, $L$ and $\ell$, are initialized to zero, and the pathline-describing point set $\mathcal{L}$ is initialized. In line 3, the integration loop is defined. Integration halts once the total pathline length is greater than some predefined threshold, $L_{\max}$, or the time falls outside the interval of interest, $\left[t_0, t_{\text{wet}}\right]$, where $t_{\text{wet}}$ is calculated as per equation (17). In line 4, the discrete solution indices are obtained. Here GetIndex$(x, y)$ is a function that maps the spatial coordinates $(x, y)$ to the corresponding spatial index on the rectangular solution grid $G$, $(i, j)$. Similarly, the time index $k$ is obtained by computing the index of the ordered time stamp set $\mathcal{T}$ corresponding to the least absolute difference with the current integration time $t$. In lines 5 through 7, the loop is terminated if the current speed or depth is smaller than some arbitrarily small constant $\epsilon_m$.

**Algorithm 3** ComputePathline: Approximates a pathline emanating to some point

```
 1: function ComputePathline $(\mathbf{U}, x_0, y_0)$
 2:     $L \leftarrow 0, \ell \leftarrow 0, x \leftarrow x_0, y \leftarrow y_0, t \leftarrow t_{\text{wet}}(x_0, y_0), \mathcal{L} \leftarrow \{(x_0, y_0)\}$
 3:     while $L < L_{\max}$ and $t \in \left[t_0, t_{\text{wet}}(x_0, y_0)\right]$ do
 4:         $(i, j) \leftarrow \text{GetIndex}(x, y), k \leftarrow \operatorname{argmin}\{\tau \in \mathcal{T}(\mathbf{U}) : |t - \tau|\}$
 5:         if $\sqrt{u_{ijk}^2 + v_{ijk}^2} \leq \epsilon_m$ or $h_{ijk} \leq \epsilon_m$ then
 6:             break
 7:         end if
 8:         $\Delta t \leftarrow -\frac{1}{3} \min\left(\frac{\Delta x}{|u_{ijk}|}, \frac{\Delta y}{|v_{ijk}|}\right)$
 9:         $x_* \leftarrow x + u_{ijk}\Delta t, y_* \leftarrow y + v_{ijk}\Delta t, t_* = t + \Delta t$
10:         if $(x_*, y_*) \notin D(\mathbf{U})$ then
11:             break
12:         end if
13:         $(i_*, j_*) \leftarrow \text{GetIndex}(x_*, y_*), k_* \leftarrow \operatorname{argmin}\left\{\tau \in \mathcal{T}(\mathbf{U}) : |t_* - \tau|\right\}$
14:         if $h_{i_*,j_*,k_*} \leq \epsilon_m$ then
15:             break
16:         end if
17:         $x_n \leftarrow x + \frac{1}{2}\Delta t\left(u_{ijk} + u_{i_*,j_*,k_*}\right), y_n \leftarrow y + \frac{1}{2}\Delta t\left(u_{ijk} + u_{i_*,j_*,k_*}\right)$
18:         $\Delta s \leftarrow \sqrt{(x_n - x)^2 + (y_n - y)^2}$
19:         if $(x_n, y_n) \notin D(\mathbf{U})$ or $\Delta s \leq \epsilon_m$ or $\Delta s > 2\alpha$ then
20:             break
21:         end if
22:         $L \leftarrow L + \Delta s, \ell \leftarrow \ell + \Delta s$
23:         $x \leftarrow x_n, y \leftarrow y_n, t \leftarrow t_*$
24:         if $\ell \geq \frac{1}{2}(\Delta x + \Delta y)$ then
25:             $\ell \leftarrow 0, \mathcal{L} \leftarrow \mathcal{L} \cup \{(x_n, y_n)\}$
26:         end if
27:     end while
28:     return $\mathcal{L}$
29: end function
```

In line 8, the time step is computed to (approximately) ensure the integrated distance will not be greater than one third the length of a grid cell. In line 9, the first step of second-order Runge-Kutta integration is performed. In lines 10 through 12, the loop is terminated if the point suggested by the previous integration step falls outside the flood scenario's spatial domain, denoted as $D(\mathbf{U})$. In line 13, the discrete indices of the proposed solution are obtained. In lines 14 through 16, the loop is terminated if the depth at the proposed index is too small. In line 17, the second Runge-Kutta integration step is performed. In lines 19 through 21, the loop is terminated if the integrated point falls outside $D(\mathbf{U})$, if the change was small, or if the change was very large (where $\alpha$ is some predefined fixed distance). In line 22, the total pathline and temporary segment lengths are updated using the most recent integration distance. In line 23, the relevant variables are integrated. In lines 24 through 26, the temporary segment length is reset to zero, and the pathline approximation is updated if the segment length is greater than or equal to the mean grid cell spacing.

## Appendix B: AlphaShape$(Q, \alpha)$

In algorithm 4, Delaunay$(Q)$ is a function that computes the *Delaunay triangulation* for a set $Q$ of discrete points. A Delaunay triangulation is a set of triangles such that no point in $Q$ is contained within the circumscribed circle of any triangle. A number of algorithms exist to compute this triangulation; herein, that of Barber et al. (1996) is used.

In line 2 of algorithm 4, the set of Delaunay triangles $\mathcal{D}$ is computed for the point set $Q$, and the set $\mathcal{B}$ comprising the triangular regions of the alpha shape is initialized as the empty set. In line 4, the function GetVertices($\Delta$) is used to obtain the vertex positions of the triangle $\Delta$. In line 5, the Euclidean edge distances are computed for the triangle $\Delta$. In line 6, the semiperimeter $s$ of the triangle $\Delta$ is computed. In line 7, the area of the triangle $\Delta$ is computed via Heron's formula. In line 11, if the circumscribed radius of the triangle is less than the constant $\alpha$, the triangle is unioned with the set $\mathcal{B}$ describing the alpha shape. In this paper, $\alpha$ is always taken to be $5(\Delta x + \Delta y)/2$, where $\Delta x$ and $\Delta y$ are the grid cell spacings used to discretize the spatial domain in the $x$ and $y$ directions, respectively.

**Algorithm 4** AlphaShape: Computes an alpha shape from a discrete set of points $Q$

```
 1: function AlphaShape($(Q, \alpha)$)
 2:     $\mathcal{D} \leftarrow$ Delaunay($Q$), $\mathcal{B} \leftarrow \emptyset$
 3:     for $\Delta \in \mathcal{D}$ do
 4:         $(a, b, c) \leftarrow$ GetVertices($\Delta$)
 5:         $d_a \leftarrow \|a - b\|, d_b \leftarrow \|b - c\|, d_c \leftarrow \|c - a\|$
 6:         $s \leftarrow \frac{1}{2}(d_a + d_b + d_c)$
 7:         $A \leftarrow \sqrt{s(s - d_a)(s - d_b)(s - d_c)}$
 8:         if $A = 0$ then
 9:             continue
10:         else if $\frac{d_a d_b d_c}{4A} < \alpha$ then
11:             $\mathcal{B} \leftarrow \mathcal{B} \cup \Delta$
12:         end if
13:     end for
14:     return $\mathcal{B}$
15: end function
```

**Acknowledgments**

All primary input and output data for the experiments described in this study are publicly available at http://nuflood.com/documents/resources/tasseff-2019a.tar.gz. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under grant DGE-1256260 and under the auspices of the National Nuclear Security Administration of the U.S. Department of Energy at Los Alamos National Laboratory under contract 89233218CNA000001.

## References

Barber, C. B., Dobkin, D. P., & Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, *22*(4), 469–483. https://doi.org/10.1145/235815.235821

Brekelmans, R., den Hertog, D., Roos, K., & Eijgenraam, C. (2012). Safe dike heights at minimal costs: The nonhomogeneous case. *Operations Research*, *60*(6), 1342–1355. https://doi.org/10.1287/opre.1110.1028

Brodtkorb, A. R., Sætra, M. L., & Altinakar, M. (2012). Efficient shallow water simulations on GPUs: Implementation, visualization, verification, and validation. *Computers & Fluids*, *55*, 1–12. https://doi.org/10.1016/j.compfluid.2011.10.012

Che, D., & Mays, L. W. (2015). Development of an optimization/simulation model for real-time flood-control operation of river-reservoirs systems. *Water Resources Management*, *29*(11), 3987–4005. https://doi.org/10.1007/s11269-015-1041-8

Chertock, A., Cui, S., Kurganov, A., & Wu, T. (2015). Well-balanced positivity preserving central-upwind scheme for the shallow water system with friction terms. *International Journal for Numerical Methods in Fluids*, *78*(6), 355–383. https://doi.org/10.1002/fld.4023

Colombo, R. M., Guerra, G., Herty, M., & Schleper, V. (2009). Optimal control in networks of pipes and canals. *SIAM Journal on Control and Optimization*, *48*(3), 2032–2050. https://doi.org/10.1137/080716372

Costa, A., & Nannicini, G. (2018). RBFOpt: An open-source library for black-box optimization with costly function evaluations. *Mathematical Programming Computation*, *10*(4), 597–629. https://doi.org/10.1007/s12532-018-0144-7

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, *186*(2), 311–338. https://doi.org/10.1016/S0045-7825(99)00389-8

Edelsbrunner, H., Kirkpatrick, D., & Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, *29*(4), 551–559. https://doi.org/10.1109/TIT.1983.1056714

Fischer, K. (2000). Introduction to alpha shapes. https://graphics.stanford.edu/courses/cs268-11-spring/handouts/AlphaShapes/as_fisher.pdf Accessed: 2017-09-19.

Judi, D. R., Pasqualini, D., & Arnold, J. D. (2014). Computational challenges in consequence estimation for risk assessment-numerical modelling, uncertainty quantification, and communication of results. United States: Los Alamos National Laboratory (LANL).

Judi, D., Tasseff, B., Bent, R., & Pan, F. (2014). LA-UR 14-21247: Topography-based flood planning and optimization capability development report. United States: Los Alamos National Laboratory. https://doi.org/10.2172/1122029

Kurganov, A., & Petrova, G. (2007). A second-order well-balanced positivity preserving central-upwind scheme for the Saint-Venant system. *Communications in Mathematical Sciences*, *5*(1), 133–160. https://doi.org/10.4310/CMS.2007.v5.n1.a6

Sayers, P., Yuanyuan, L., Galloway, G., Penning-Rowsell, E., Fuxin, S., Kang, W., et al. (2013). *Flood risk management: A strategic approach*. United States: Asian Development Bank, GIWP, UNESCO, and WWF-UK.

Storn, R., & Price, K. (1997). Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341–359. https://doi.org/10.1023/A:1008202821328

Tasseff, B., & Judi, D. (2016). *Nuflood, Version 1.x*. Computer Software. USDOE. https://doi.org/10.11578/dc.20171025.1807

Tasseff, B., Bent, R., & Van Hentenryck, P. (2016). Optimal flood mitigation over flood propagation approximations. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (pp. 358–373). Springer. https://doi.org/10.1007/978-3-319-33954-2_26

Telea, A. C. (2014). *Data visualization: Principles and practice*. MA, USA: CRC Press. https://doi.org/10.1201/b10679