Response by the authors to the comments of reviewer 1

Date: 20/07/2017

We took the original text of the review and divided it up into smaller pieces, which we answered. The original text by reviewer 1 is presented in blue, italic text, while our responses are numbered and in black.

Any references to lines in the manuscript are made to the manuscript which has the visible changes compared to the old manuscript. The manuscript with the visible changes can be found at the end of this document.

The paper provides a coherent narrative and is clearly within the scope of NHESS. It also provides a scientific background to how engineers can systematically explore a multi-dimensional space for optimal solutions using a method unknown to many. As such it is publishable.

My review is based on my background knowledge which is more related to the traditional cost-benefit analyses in relation to risk-based design. So please bear with me if there are things I have misunderstood. On the other hand I have done exactly the same as the authors using traditional economic tools in relation to risk-based design.
I think the paper should be rewritten to improve clarity. Therefore I only have overall comments.

The paper, and in particular the Introduction section, is not very well written for the reader not already familiar with the thinking of the authors. Assumptions about prior knowledge on Dutch design criteria are very high, previous work is not introduced as more than a reference (sometimes even to studies in Dutch).

1. Thank you for this comment. We have expanded the introduction to include a better introduction of the economic optimisation of flood defences as it used in the Netherlands. Specifically:
    - The addition of Figures 1 and 2, the expansion of Eq.1 and the addition of Eq.2
    - The additions/changes on Page 1, lines 15-23 and Page 2, lines 1-15
    - The study in Dutch is now accompanied with a reference to a chapter from a PhD thesis, which is in English (page 4, line 8)
    - A more complete overview of relevant literature (e.g. page 3 lines 17-22, page 4 lines 1-8)

The authors seem to use the term risk to characterize probabilities and economic loss interchangeably. Please define and use a clear notation. This could be done in relation to Equation 1 which in poorly defined. The problem is encapsulated in the sentence on page 2, line 17.

2. Thank you for this comment. We have clarified our definition of risk/probabilities/economic loss.
    - Specifically, see Eq.1 and Eq.2 and the description of these equations on page 1 lines 15-23.
    - The introduction of using AED, or Annual Expected Damage as an alternative for the annual expected risk cost (page 1, lines 19-23, now used throughout the paper).

The authors rightly state (e.g. page 3 line 5) that the major work in relation to risk-based design is calculation of the residual risk (in monetary terms) by a complex procedure involving complex hydrological and hydraulic calculations and subsequent calculation of loss of vulnerable assets. However, I cannot see how calculation of the edges as outlined on e.g. page 4 line 5 can be done without such calculations. Indeed this is also stated on page 7, line 5. So I see a reduction in the required number of calculations in comparison to LP, but at least as computationally demanding than traditional Cost- Benefit Analyses. It is not difficult to set up the mathematical framework for optimization within economics that can identify

economically optimal solutions if the risk can be formulated in a simple equation such as the authors do in their examples (e.g. Eq 3).

3. Thank you for this comment. We tried to answer it in three parts:
    - Edge calculations: It is correct that each edge is associated with potential risk calculations. The main challenge is there to reduce the required number of edges "visited", to save valuable computation time. The reduction in number of calculations is indeed compared to I(L)P. We have tried to make this clear throughout the paper, particularly in:
        - Page 4, lines 9-15
        - The introduction of section 3 (page 12, lines 4-12)
        - Section 3.3 on page 14-15 and Figures 14, 15
        - By adding the percentage of actual executed calculations to the examples in Sections 4.1-4.3
        - In the first paragraph of the Discussion on Page 21-22.
        - By focusing on the reduction of cost calculations in the first paragraph of the Conclusions on page 21
    - Traditional C/B analyses: we assume that by traditional C/B analyses, the reviewer refers to marginal C/B analyses (i.e. those that optimise flood defences separately and independently of hydrodynamic interactions in a larger system of flood defences). We have clarified that our proposed method of looking at the whole flood defence system makes sense if hydrodynamic interactions are expected to lead to significantly different flood risk estimates. If the flood risk estimates are approximately the same with and without hydrodynamic interactions, the economic optimisation might be done just as well (and possibly more efficiently) by looking at each flood defence independently. See also:
        - Equation 3 on page 6, and the description on page 6, lines 13-19 and page 7 lines 1-12.
    - Simple equations: We purposefully chose these simple equations in order to focus on the approach and not on the examples. However, these simple equations should not be seen as representative of the computational costly risk calculations we have in mind. In follow-up research, we have a more complex case study in which hydrodynamic interactions are explicitly modelled. However, if we were to include that (or a similar) case study in this paper it would need to have either a lengthy description (muddying the focus of the paper) or a reference to future (unpublished) work which would make the case hard to reproduce. Nevertheless, we have added a description that these equations are simplified for the purpose of this paper, and can be replaced by (for example) more complex hydrodynamic models and probabilistic computation methods.
        - Specifically, see Page 17, lines 2-6.

I would prefer if the extension involving several dikes heights to be optimized simultaneously were introduced using multiple dimensions. Since the paper only discusses two dimensions it should be straight forward also to show graphically. It will make comparison to marginal economic studies on efficiency of alternative measures quite apparent. Still, the visualization and structured approach to identify optimal trajectories makes the approach valuable.

4. Thank you for this comment. We interpreted this comment as that we need to explicitly show and describe in that the approach is applicable to more than two lines of defence.
    - We have done so by adding Equation 3 (page 6) and the description on Page 6 (lines 13-19) and page 7 (lines 1-12).

Response by the authors to the comments of reviewer 2

Date: 20/07/2017

We took the original text of the review and divided it up into smaller pieces, which we answered. The original text by reviewer 2 is presented in blue, italic text, while our responses are numbered and in black. The headings ("General comment" and "Specific comments") are retained from the original review text.

Any references to lines in the manuscript are made to the manuscript which has the visible changes compared to the old manuscript. The manuscript with the visible changes can be found at the end of this document.

General comment:

Before reading this referee comment, the reader must be aware of the fact that the authors of this paper actively asked me to referee their paper. I thank them for this opportunity and making me aware of this and previous papers. I have had a meeting with two authors of this paper to discuss my first impressions. This referee comment benefitted from the insight the authors provided me in this meeting.

In this general comment I will state that this paper copies earlier work. No proper references are made to this work. Hence, this paper doesn't meet minimal scientific standards. I will provide references to plenty published reports and articles to support my claim. In the Netherlands, many involved experts, including many full professors at various Dutch universities (which wrote and published referee reports on this earlier work or supported the development of earlier work) can be asked to confirm my claim.

The main claim by the authors that a shortest path/dynamic programming approach (previously presented and discusses by other authors) to solve economic optimal dike heightening is 'advantageous' needs to be elaborated a lot more. Many previously stated and published arguments against dynamic programming are not mentioned. Moreover, the scientific ambition of this paper is not clear. Furthermore, no calculations are presented by Dupuits et al. (2017) to support their claim. I will provide arguments for this claim.

We thank the reviewer for taking time to sit down with us.

We do not comment on these first three paragraphs, as it seems that all the mentioned issues return in a more detailed form in the remainder of the review document.

This paper copies the approach by Zwaneveld & Verweij (2014a) for finding an optimal configuration for interdependent lines of flood defences. In Zwaneveld (2012; section 1.1; in 2014 provided to the authors by email) several approaches are discusses to solve this model. This modelling approach by Zwaneveld & Verweij (2014a, 2014b) including graph-based (shortest path or minimum cost flow problems) solution approaches and the preferred ILP approach was earlier copied and described by Yuceoglu (2015, Chapter 5: Safe Dike heights in the Netherlands). This PhD-thesis builds on the work by Zwaneveld and Verweij (2014a , 2014b) and discusses graph –based algorithms to solve the so-called Diqe-Opt model (see later for a discussion of this model). Zwaneveld & Verweij (2014a) identify several algorithms to solve the problem both to proven optimality and to solve the problem heuristically (with the advantage that computing times remain limited). Zwaneveld & verweij92014a0 aplly their model to ral world problem instances to support crucial Cabinet decisions for the Netherlands.

1) We share a similar approach as Zwaneveld & Verweij (2014a), by using graphs to model the problem. However, contrary to the approach of Zwaneveld & Verweij (2014a), we use a greedy algorithm to solve the shortest path problem instead of modelling it as an I(L)P model. The reason for using a greedy algorithm is, among other reasons, an attempt to reduce the number of risk calculations (see also answer 3 for a more detailed answer). Therefore, we disagree with the use of the word 'copy'. Nevertheless, we did miss that in the appendix of Zwaneveld & Verweij (2014a) the problem was already identified as a graph problem. We therefore improved the paper:
    - The entire introduction (pages 1-5) has been rewritten with help of suggestions from reviewer 2, and should now contain a more complete overview of related publications
    - Acknowledgements now also express gratitude to Zwaneveld & Verweij for sharing their model

I apply and explain an algorithm to solve the problem to proven optimality. Their algorithms requires hardyly any programmng efforts and little solution time ('less than one minute or so'). The ideas to solve the problem heuristically were not implemented in practice due to the fact that the algorithm to solve the problem to proven optimality was superior according to Zwaneveld and Verweij (2014a).

2) Based on our own experience, finding the shortest path in a graph with a greedy algorithm can be explained intuitively for most engineers working in the field of flood risk. This explanation is one of the motivations of writing Section 2. It is the opinion of the authors that an IP model (specifically the model as proposed by Zwaneveld and Verweij (2014a)) requires at least a basic knowledge of integer programming models. Even if the model code of the IP model is available to a user, the model code will need to be expanded if that user wants to add an additional line of defence. Granted, the extension is relatively straightforward (provided the user understands the linear programming model), but it is not a "blind copy-paste action". Contrary to this, our approach builds (and solves) the graph automatically for an arbitrary number of lines of defence and, given the model code, only needs inputs. This has been further explained in the paper:
    - The introduction, where this issue is now introduced (page 4, lines 9-24 & page 5 lines 1-5)
    - Equation 3 on page 6, which is the basis for automatically building the graph, and Sections 3.1 and 3.2 which describe how the entire graph can be represented with only a fraction of the entire graph. Note that we expanded the description of Section 3 on page 12 with a clearer separation between the graph in-memory representation (3.1 and 3.2) and the reduction in risk cost calculations. (3.3)

3) Furthermore, we clearly state in our aim that we want to achieve computational efficiency by means of reducing the number of risk calculations. In the IP approach by Zwaneveld and Verweij (2014a), the flood risk calculations are not considered as a part of the solving time. Instead, the risk calculations are assumed to have be carried out beforehand in their approach. If all flood risk calculations (and other calculations) are done beforehand, and the only issue at hand is the solving time of the algorithm, the IP approach will be more efficient than the greedy algorithm. We don't contest this in the paper, because we do not consider the solving time of the graph algorithm as dominant. We think there are plenty of scenarios where the flood risk calculation time is dominant (and even limiting). In that case, we have shown that a greedy algorithm (i.e. one that does not necessarily visits all vertices) coupled with an efficient evaluation of the risk estimates (i.e. only calculate the risk if a vertex is actually visited, as opposed to calculating the risk for all existing vertices), is expected to result in fewer risk calculations for most situations. How many calculations will be saved depends wholly on the case study characteristics and discretization. This has been further explained in the paper:
    - We rewrote the approach in the introduction (Page 5, lines 13-22) to make sure that our definition of computational efficiency is a reduced number of risk cost calculations, not the efficiency of the algorithm itself. Similarly, the conclusions have been updated (page 23-24).
    - A similar addition as mentioned in the previous point has been made to the abstract

- Risk cost calculations are assumed to have a (much) higher computational burden than the optimisation algorithm, which is now stated multiple times in the paper (e.g. see page 4 lines 9-15).

In line with Brekelmans et al (2012), Eijgenraam et al, (2010; in revised version published as: Eijgenraam et al. 2016) a dynamic programming (read: shortest path approach) is identified in Zwaneveld (2012) as one of the options to solve the model. Zwaneveld and Verweij (2014a, Annex A, Figure A; 2014b) and Zwaneveld (2012) contribute to these earlier papers by identifying that the dike optimization problem can be seen as a graph based problem. For example, Zwaneveld and Verweij (2014b, p.29) state that the the dike optimization model 'satisfies the most fundamental of all network flow problems (Ahuja et al., 1993), namely the minimum cost flow model'. This point was missed by earlier published work and also by Dupuits et al. 2017. Dupuits et al. (2017) present a graph based representation which is identical to the graph representation of Zwaneveld and Verweij (2014a) and Zwaneveld and Verweij (2014b). For example, compare Figure A in Zwaneveld and Verweij (2014b, p. 29) with the figures of Dupuits et al. 2017. They are clearly (almost) identical.

4) Regarding the figures containing graphs being almost identical: we think that graphs with this kind of structure always look almost identical. Nevertheless, as already mentioned in 1), we added references to Zwaneveld & Verweij regarding seeing the flood defences optimisation problem as a graph.
   Regarding the dynamic programming approach in the listed papers (Eijgenraam et al, (2010), Eijgenraam et al, (2016)): dynamic programming is mentioned there without specifying which dynamic programming algorithm is actually used. Dynamic programming can actually entail a number of algorithms (Cormen (2009)). Furthermore, in Cormen (2009), a clear distinction is made between dynamic programming and greedy algorithms. We explicitly mention that we use a greedy algorithm. Therefore, (part) of our contribution is to use a greedy algorithm instead of a dynamic programming approach. This is further elaborated upon in the paper:
   - See page 5, line 31 for an additional reference in Section 2.
   - See page 8, line 17-24 for the discussion regarding dynamic programming.

Unfortunately, this paper by Dupuits et al. (2017) does not clearly refer to these previous papers and reports which they copy and build upon. In my opinion, this paper needs a thorough revision to correctly and clearly refer to the work of previous mentioned authors to meet minimal scientific standards.

5) see 1), where we address these citation issues.

This paper states in the introduction that "However, existing cost-benefit analyses tend to focus on flood defences with independent lines of (Kind 2014), or are not readily generically applicable (e.g. Zwaneveld and Verweij (2014a). Therefore, the aim of this paper is to find general, computationally efficient approach… with arbitrary number of lines"
The authors do not mention the fact that Zwaneveld and Verweij (2014a, including background papers), Bos and Zwaneveld (2012) and Zwaneveld and Verweij (2016, UK CPB discussion paper on previous Dutch reports) do present for the first time a generic , computationally approach to assess dependent flood defense systems whit arbitrary number of flood defense lines.

6) We choose the words "readily generically applicable" with the argumentation of 2) in mind. In 2), we acknowledge that the IP model is extensible, but extending it requires at least some editing. Furthermore, see 1) for citation issues.

Moreover, these authors do apply their approach in a real world environment and under time pressure to obtain economic optimal flood protection policy measure for the Lake IJssel region (including many dependent dike rings and barrier dams). Dupuits et al. (2017) are aware of this approach and solution method since they apply it in section 4.3. Zwaneveld and Verweij kindly provided Dupuits et al. (2017) with their programming code and data to allow scientific reuse of their earlier work.
Although this approach is not yet published in in a UK written scientific journal (the authors are working on it, see Zwaneveld and Verweij , 2016)), the scientific quality had been assessed by two different committees with professors and other experts (see Donders et al., 2013; Van Ierland et al., 2014). This was due to the fact that very important hydrological and economic policy decisions are based upon the application of the Diqe-Opt model (in Bos and Zwaneveld 2012; Zwaneveld and Verweij 2014a). The Ministry of Infrastructure and Environment had to be sure about the quality of the Diqe-Opt model and the two reports. Documents are published on the UK and Dutch based CPB-website. A few documents are also presented to the Dutch Parliament: they can also be found at the website of the Dutch Parliament.
The latter 2014-committee of four professors at Dutch universities conclude in Dutch: "Het CPB heeft met deze studie een belangrijke stap vooruit gezet in het onderzoek naar waterveiligheid. Het is een indrukwekkende studie waarin zeer veel hydrologische en economische kennis op een prachtige manier wordt samengebracht. Met name het meenemen van afhankelijkheden in de overstromingskansen van dijken is een belangrijke innovatie. Het ontwikkelde model Diqe-Opt is een vernieuwend en zeer nuttig instrument" [UK translation: "CPB has made with this study an important step forward in the search for water safety. It is an impressive study in which very many hydrological and economic knowledge is combined in a wonderful way. In particular, the inclusion of dependencies in the flood dikes of opportunities is an important innovation. The developed model Diqe-Opt is an innovative and very useful tool"". Note that Zwaneveld and Verweij (2014) name their 3 generally applicable method: Diqe-Opt. Due to the generally applicable of the Diqe-Opt approach the model is by request from the Ministry of Infrastructure and the Environment being transferred to hydrological consultancy company Deltares. Deltares can use the model as long as proper references are made to earlier CPB-work by Zwaneveld and Verweij. The Dutch institutions setting of the CPB (employer of Zwaneveld and Verweij) prohibit these activities by CPB. This innovative Diqe-Opt approach was also recognized by a recent Dutch handbook on water safety (ENW, 2016, Literature list to Chapter 4).
Hence, Dupuits et al. (2017) should state that they copy the Diqe-Opt model by Zwaneveld and Verweij (2014a , 2014b) instead that the "aim to find an … approach". Proper and clear references are missing towards this earlier work in the starting sections of this paper.

7) We are thankful for sharing the model with us. We already referred to using it in Section 4.3, and we have extended our gratitude to the acknowledgements. Regarding the suggested change for the aim, missing references and the usage of 'copy': see 1) where we address the same comment.

Dupuits et al (2017) present a shortest path algorithm to the problem definition as presented by Zwaneveld and Verweij (2014a).

8) We interpreted this as a similar comment (regarding the problem definition/approach) as one that is already answered by our answer in 1). Therefore, see 1) for our answer.

For the cases presented in section 4.1 and 4,2, almost identical and probably more efficient dynamic programming approaches (shortest path approaches) are presented in Eijgenraam et al. (2010) and Brekelmans et al. (2012). Proper references should be made to this earlier work.

9) According Cormen et al (2009), greedy algorithms typically need less computational time than dynamic programming approaches. From that perspective, we do not see how dynamic programming can be more efficient than greedy algorithms. We have clarified this in the paper; see also 4).

Furthermore, we do not think we need to show an extensive comparison with existing solutions, as we do not present our approach as a competitor to these existing methods. See also our answer in 16) where we answer the same comment.

I do not see the value added by the shortest path approach of section 2 and section3 in addition to these two papers.

10) Because we believe we use a different algorithm to solve the shortest path approach, see also 4) where we mention the difference between dynamic programming and greedy algorithms.
11) The purpose of section 2 is to show the basic functioning of the greedy algorithm with respect to a (very) simplified problem, in combination with a graph. We believe this helps to create a fundamental understanding for engineers who are not familiar with the concept; as the intended target audience might not be as familiar with graph algorithms as an operations research audience. Furthermore, understanding how a greedy algorithm 'moves' through a graph (along with lazy evaluation) is essential for achieving the reduction in risk cost calculations.
    - Specifically, see page 8 lines 22-24
12) Section 2.4 contains references which are relevant for the used greedy algorithm, and provides context to whether or not the shortest path (optimal solution) of a graph with non-negative edge weights is found, which is important for any optimisation algorithm. Furthermore, Section 3.3 explains (based on section 2) how the risk cost calculations are reduced.
    - See also the additions on page 15, lines 2-11, Figure 14 and Figure 15.

Zwaneveld and Verweij (2014b , Annex A) present an alternative approach for these two cases in section 4.1 and 4,2 based upon an ILP-model. Applying this approach requires no programming effort whatsoever since user friendly software can be used to model the problem.

13) We addressed the implementation of I(L)P models in 2).

No efforts are required to solve the stated model since standard LP/IP solvers can be used. Note that an LP-solver is at present a plug in tool in Microsoft Excel and modelling languages as CPLEX, GAMS and AIMMS are easily available. Free and easy to use solvers are easily available.

14) CPLEX, GAMS and AIMMS are all tools that are geared towards applications with, for example, I(L)P models such as used in Zwaneveld and Verweij (2014). This requires at least basic knowledge of this kind of model, and knowledge of the specific tools. See also 2). This is now also mentioned in the paper on page 4, line 16-20.

15) GAMS and AIMMS are algebraic modelling systems with proprietary licenses. CPLEX is a solver with a proprietary license. Free solvers, at least at the time of writing, don't scale well with thousands of decision variables and will have a hard time to solve the problems as discussed in Zwaneveld and Verweij (2014). On the contrary, our approach can be used in any general-purpose programming language, such as for example the freely available open-source languages Python and Julia. Inherently, this means that threshold for application and use is lower than with proprietary licenses.
    We did not think it was necessary to include the above in the paper, as our primary

goal is an efficient computation time (by reducing the amount of risk cost calculations), and not presenting a direct competitor for the Zwaneveld and Verweij (2014) model; see also the changes mentioned in 3).

The authors should mention in section 4.1 and 4.2 the use of these competitive and in some cases almost identical approach and should compare it with their approach. This comparison was already presented in Zwaneveld and Verweij (2014b, Annex A). They provide arguments and conclude that an ILP-approach is by far more preferable than a dynamic programming approach. Dupuits et al. (2017) should – as a minimum- discuss this work by Zwaneveld and Verweij (2014b).

16) The purpose of the (simplified) examples in section 4 is to show that the greedy algorithm works in terms of reducing risk cost calculations; not to show a comparison between various shortest path algorithms or I(L)P models. See also 3) for the context of our method. We measure computational efficiency in the number of risk cost calculations actually evaluated, versus the total number of possible risk cost calculations in a graph. This has been further elaborated upon in the paper:
    - In the abstract and in the introduction, specifically in the approach (page 5)
    - Section 3.3, particularly page 15 lines 2-11
    - In the examples of section 4.1-4.3 (page 18 lines 6-9, page 19 lines 17-20, page 21 lines 7-10)

Furthermore, it is unclear why Dupuits et al 2017 conclude that an dynamical programming approach is 'relatively easy'. From a discussion with the authors, I learned that their intention and scientific ambition is to present a heuristic approach to solve the model by Zwaneveld & Verweij (2014a). Although heuristic approaches are presented in Zwaneveld & Verweij (2014a, 2016), these were not yet implemented. The advantages of this heuristic approach is to reduce calculation time with the disadvantage that a non-optimal solution is found. Furthermore, to help the authors, some persons may prefer a dynamic programming approach over an ILP approach. I also learned that a dynamic programming approach is easier to understand for many people than an ILP-approach. Therefore, Zwaneveld & Verweij(2014a, 2014b) always present their model as a graph problem and then introduce that they prefer to solve this graph problem to optimality by using an ILP-approach.

17) Our aim is to use a greedy algorithm in order to reduce the number of risk cost estimates actually executed (see also 4 for the difference between dynamic programming and greedy algorithms). We think that the UCS algorithm, as discussed in for example Fellner (2011), is easy to implement because the fundamental concepts related to UCS area taught in basic algorithm classes (Fellner, 2011). Furthermore, we consider the actual implementation in code as easy because of the brevity of the algorithm (see Fellner (2011), about 10 lines of pseudo code), coupled with the fact that the target audience (civil engineers) is more likely to be familiar with general purpose programming languages (see also 15)) than IP models (see also 14)). The optimal path is found for a graph using a greedy algorithm, if implemented as explained in Section 2.
    - Figure 10 already described explicitly that the user only needs to provide the inputs, provided that the greedy algorithm and automated graph generation have been implemented and shared with the user.

However, the ambition by the authors, as I learned from personal communication with them, doesn't meet their statement on page 5 of Dupuits et al. (2017) : "For our applications, we did not come up with a heuristic

function, which reduces the choice of a graph alfgorithm to either Dijkstra or UCS". Hence, this requires more explanation by the authors. I cannot see why both claims are valid.

We interpreted this comment as that the reviewer sees a greedy algorithm as a heuristic. This interpretation is answered in 18). However, we meant to discuss greedy algorithms with an optional heuristic function, which is clarified in 19).

18) An attribute of greedy algorithms is that, while efficient, they sometimes don't find the shortest path in a problem. We assume this is what the reviewer means with a heuristic. However, in Section 2.4 (see also 12)) we think we provide some references containing arguments for the greedy algorithm that it does find the optimal path for the discussed application area. Therefore, we refrained from calling the greedy algorithm a heuristic.
19) A heuristic function, as mentioned in the paper, relates to a subset of greedy algorithms. Heuristic functions can be used with a greedy algorithm in order to give additional information in an attempt to speed up the finding of the shortest path. A graph algorithm which can use such a heuristic is the A* algorithm. If the heuristic function always returns zero (i.e. what we called no heuristic function), the A* algorithm reduces to the Dijkstra algorithm or the closely related UCS algorithm:
    - Ultimately, we thought this was a non-essential detail and removed it to prevent any further confusion (page 8, lines 7-8).

For the case presented in section 4.3 proper references should be made that this is a simplified version of Zwaneveld and Verweij (2014a).

20) We do not see how a case as simple and general as shown in section 4.3, which in this general form can be found in many coastal areas around the world, should be referred to as a simplified version of the case presented in Zwaneveld and Verweij (2014a). We think that the fact that we use (and refer to) the method of Zwaneveld and Verweij (2014a) as a benchmark for the correct answer implies that the method of Zwaneveld and Verweij (2014a) can be applied to the case study as well. In addition, see our reply in 1).

Again, an explicit discussion of the pro's and cons of solving the model by Zwaneveld and Verweij (2014b) by using an ILP-approach and their approach should be presented. Zwaneveld and Verweij (2014a) and Zwaneveld (2012) do present such a comparison and they conclude that the ILP-solution approach is superior to dynamic programming (or: shortest path) for real-life applications. This earlier assessment should be presented. Why do Dupuits et al. (2017) conclude the opposite?

21) After careful re-reading of our own words, we cannot find any evidence that suggests we even compare the performance of ILP model to our greedy algorithm, let alone conclude that the greedy algorithm is superior. The only qualifications we make is that the outcomes are equal, with the Zwaneveld and Verweij (2014a) method being the benchmark, and that less risk calculations are done. See also 3) and 16).

Why do the authors present only 'toy problems instances'which can easily be solved using existing approaches?

22) Because a more complex case study would shift focus from the topic, which is to explain the application of graphs in combination with a greedy algorithm in the context of computationally expensive risk calculations (A more complex case study would require a significant amount of introduction regarding the assumed hydrodynamic interaction between multiple lines of defence, for example). The main point of the simplified examples is to show, in conjunction with earlier mentioned points in 12) and 18) and section 2.4, that the optimal path is found and that risk cost calculations are saved. A more complex case study will be an integral part of a follow-up research.
    - We have improved this description in the paper on page 17, lines 2-6.

Zwaneveld and Verweij (2014a) and Bos and Zwaneveld (2012) were capable of solving very large real-time problem instances given very short research leadtime and research capacity.

23) Our approach has different goals than the mentioned approaches. This was already touched upon in 2), 3), 14) and 15). Primarily: the computational cost of risk calculations is an issue, secondary: knowledge of IP models (and specifically the model by Zwaneveld and Verweij) is not commonly present among the target audience. For our intended use (see also our re-phrased aim of the paper) and the intended target audience, we consider the replies of 2), 3), 14) and 15) highly relevant.

Moreover, setting up a dynamic programming algorithm is requires very substantial programming efforts as is clear from section 2 and 3 from this paper. The approach by Zwaneveld and Verweij (2014a) requires only the code "SOLVE DIQE-OPT MODEL USING CPLEX" to obtain the proven optimal solution. Hence, the claim that a dynamic programming is 'more easy' than a ILP-approach by Zwaneveld and Verweij (2014a) is not valid or – at best - not properly motivated in my opinion.

24) We do not agree with this statement. See also 23) for a summary about the relevance of our approach. From our own experience, re-creating the model of Zwaneveld and Verweij (2014a) took a roughly equal amount of programming (We had to recreate the model based on the model description/formulas due to an absence of the necessary proprietary licenses for GAMS and CPLEX). This excludes the amount of time it took to familiarize ourselves with the IP model by Zwaneveld and Verweij and its inner workings. Furthermore, even though our approach did require a programming effort, we can share this code with anyone who has access to a computer, because it was coded using a freely available open-source language. Therefore, in principle, no additional coding effort is required by third parties regarding implementation of graphs and the greedy algorithm. See also 17).

Specific comments

Section 2.1:

The representation of the problems copies the approach by Zwaneveld and Verweij (2014a) and Zwaneveld and Verweij (2014b). Especially the graphs in this section are strikingly identical to Figure A from Zwaneveld and Verweij (2014a) See also almost identical figures in Yuceoglu (2015). Also references should also be made to Dynamic programming approach by Eijgenraam et al. (2010) and Brekelman et al. (2012) which seems to be mathematically identical. Proper references are missing to this earlier work. Dupuits et al. (2017) should clearly state that they copy previous work.

The cases presented in paragraph 4.1 (single flood defense) and 4.2 (independent lines of defences) can be solved by the dynamic programming (or shortest path approach) which is extensively discussed in Eijgenraam et al (2010) (a revised version of this paper was published as Eijgenraam et al 2016) and briefly discussed in Brekelmans et al. (2012). Zwaneveld & Verweij (2014b, 'paper under revise and resubmit') make the point in Annex A that these shortest path problems can be much easier solved to proven optimality using LP-relaxation or IP-model formulation. All this should be mentioned.

25) This remark seems to repeat a number of earlier made remarks, which we answered in our earlier replies. Addendum: Our focus is not to find the most efficient graph algorithm, our focus is to find a graph algorithm which handles risk calculations efficiently. In our opinion, this makes the requested addition of earlier made comparisons between dynamic programming and IP models non-relevant (at least not relevant for this paper). See also 3).

Section 2 and 3:
The presented approach is basically the well known shortest path algorithm. The discussion should can be deleted or removed to an electronic companion . I do not see any scientific added value in comparison with earlier work by Brekelmans et al (2012), Eijgenraam et al (2010) and the large literature of shortest path problems and dynamic programming. I personally prefer to refer to the well-written UK –based Wikipedia discussion of the subject(see Zwaneveld, 2012).

26) As mentioned in previous points:
    - In our opinion we don't use dynamic programming (we use a greedy algorithm).
    - Furthermore, we think section 2 serves as an essential introduction for non-experts in the domain of graph optimization, which we think is relevant given the journal's audience. Furthermore, it serves as a foundation for explaining how the number of risk cost calculations can be reduced. See also 10, 11, 12.

The claim that repetitiveness of vertices is in most cases incorrect. Note that vertex 12 represent a later year than vertex 7 (see Figure 9) . Due to yearly increases of economic growth and flood probabilities al risk calculation has to be calculated again. Hence, vertex 7 and 12 are not identical and no calculation time is saved.

27) We disagree with this comment. Risk calculations are not mentioned in section 3.1. We use the repetitive characteristics to reduce the size of the data structures belonging to the vertices and edges. This has been improved in the paper:
    - Section 3, particularly on page 12 lines 9-12.

Note that Brekelmans et al. (2012, p.1343) state that a simple 'homogenous case can be conveniently solved using dynamic programming. Unfortunately, this is not possible for the nonhomogenous cases, because the state space explodes…. We show how the nonhomogeneous diek height problem can be solved as a MINLP-problem.' A more or less similar statement by Brekelmans et al. (2012, p. 1345): "Unfortunately, the state space grows too large ….which implies that the dynamic programming approach is not applicable". This is the – very good- reason why Brekelmans et al. (2012) prefer their MINLP approach.

Dupuits et al (2017) do not properly discuss this exploding problem, i.e. exploding state spaces and exponential calculation time of all sorts in the problem size. Nor do they refer to these previously mentioned authors which did identify this problem before.

28) We do not use dynamic programming, we use a greedy algorithm. Furthermore, we believe the state space explosion can be partly negated with the help of 3.2 and with section 3.4 (if 3.4 is applicable), because these techniques can reduce the amount of edges and vertices (and therefore the computational time). Furthermore, state space explosion is inevitable if all flood defences are assumed to depend on each other (assumption repeated throughout the paper, see for example page 6, lines 14-16.)

From the paper I have got the impression that they apply a shortest path algorithm to solve the problem to proven optimality given – theoretically- computing time which are exponential in the problem size. From personal communication with the authors, I did get a different impression, namely that they aim to present a non-optimal solution approach given limited computing time. The authors should clarify that ambition.

A more or less similar remark holds for the algorithm. From the paper I get the impression that they implemented the algorithm themselves to find a proven optimal solution. From personal communication, I did get the impression that they use standard plug-in heuristic procedures to solve the graph. Hence, no programming effort whatsoever is required. The latter would make their approach of course more easy to use but also make their algorithm less innovative. The authors should clarify their ambition.

29) Unfortunately, it seems like we didn't make this sufficiently clear during our personal communication. We use an existing greedy shortest path algorithm, which we tried to implement in an efficient manner for the particular problem of economic optimization of multiple lines of defence. Efficiency was sought primarily by reducing the number of risk cost calculations (i.e. Section 3.3). See many of our earlier replies, for example 12), 18), 19) and 22), for further clarifications and replies.

Section 5:

The authors state that Kind (2014) proposes an linear programming approach. This is incorrect. Kind (2014) doesn't propose any method. He uses the approach by Brekelmans et al. (2012), which is an MINLP-approach. The IP-approach was proposed by Zwaneveld and Verweij (2014b, 'paper in revise and resubmit' to an academic journal). An IP-approach is not identical to a linear programming approach.

30) Thank you for these suggestions. We have improved the introduction, see also 1).

The claim that the application area is roughly similar to Zwaneveld and Verweij (2014a) is incorrect. The application area is completely identical and copied from Zwaneveld and Verweij (2014a). Furthermore, reference should be made that dynamic programming/shortest approaches of cases in section 4.1 and 4.2 to Eijgenraam et al (2010) and Brekelmans et al. (2012). And to heuristic ideas (and some attempts) to solve the dike height problem in previous work by Eijgenraam, Brekelmans and Den Hertog and Zwaneveld & Verweij (2014a, 2014b, 2016)

31) We refer to our answer of 1) regarding the suggested use of the word 'copy'. See also our answer in 3) for our aim, which (although similar) emphasizes different aspects than Zwaneveld and Verweij. Therefore, we refrained from using words such as 'identical' and 'copy'. We believe our proposed approach is complementary to the approach by Zwaneveld and Verweij (and other earlier proposed methods); it is not meant as a replacement.

Lines 17-22 Page 17: The authors should mentioned that fact that the approach by Zwaneveld and Verweij (2014b) was especially develop to include other flood defence systems than height-dependent dikes.

32) Thank you for this suggestion. See page 22, line 17-19 for the relevant addition to the discussion.

The fact that Dupuits et al (2017) can also include these approaches is a direct consequence of the fact that they copy the approach by Zwaneveld and Verweij (2014a, 2014b) and, therefore, both have identical application areas. The Diqe-Opt model was already used to asses many of these alternative flood defence systems in Bos and Zwaneveld (2012) and Zwaneveld and Verweij (2014a). See also Donders et al. (2013) and van Ierland et al. (2014)

33) Regarding the use of qualifications like 'copy' and 'identical application areas': See 1) and 3). See also 32) for properly referring to using alternative flood defence systems with Zwaneveld and Verweij (2014).

Section 6

The authors claim that it is an advantage that 'their approach do not need pre-calculate risk which linear programming approaches do'. However, the IP-approach by Zwaneveld and Verweij(2014a) – again this is NOT a linear programming approach – indeed does require risk estimates in a pre-processing step. In addition, stating that risk calculation can be performed 'on the fly' is complete impractical in a real-world setting of Zwaneveld & Verweij (2014a), Bos and Zwaneveld (2012), Brekelmans et al (2012) and Eijgenraam et al. (2016), since it requires in general running hydrological models. Hence, the approach by Dupuits et al. (2017) requires in each iteration to consult a hydrological experts to run their model and to report the result back. Doing these calculations in a pre-processing step as advocated by Zwaneveld and Verweij (2014a and 2014b) and Bos and Zwaneveld (2012) has very significant practical advantages. For real-world instances, risk calculation were no problem whatsoever in the approach by Zwaneveld & Verweij (2014a, 2014b), Brekelmans et al. (2012) and Eijgenraam et al. (2016).This argumentation is missing in this section.

34) We disagree that each iteration requires consulting a hydrological expert. In follow-up research, we are calculating risks 'on the fly' in a case study. We do not know the details of the referred to risk calculations, but we can predict that in the setting of this paper (multiple lines of defence of which the risk of a downstream defence is assumed to be dependent on all upstream defences), risk calculations will not be computationally cheap. This has been emphasized in the paper:
    - Page 4, lines 9-15
    - Page 4 lines 21-23 and page 5 lines 1-5

Finally, the claim by Depuits et al. (2017) that their approach requires less risk calculation than the graph-based ILP—approach by Zwaneveld and Verweij(2014a) is not supported by calculations.

35) See 3) where we argue why we cannot predict the amount of savings (depends on the case). We did mention the savings of the Section 2 example, specifically in figure 12 of section 3.3. This has been further expanded:
    - An expanded description of the number of saved risk cost calculations on page 15, lines 2-11
    - The number of risk calculations have been added to the examples of Section 4 (see also 16).

Using graphs to find economically optimal safety targets for multiple lines of flood defences

Egidius Johanna Cassianus Dupuits[1], Ferdinand Lennaert Machiel Diermanse[2], and Matthijs Kok[1]

[1]Delft University of Technology, Faculty of Civil Engineering and Geosciences, P.O. Box 5048, 2600 GA Delft, Netherlands
[2]Deltares Unit Inland Water Systems, department of Flood Risk Management, P.O. Box 177, 2600 MH Delft, Netherlands

Correspondence to: E.J.C. Dupuits (e.j.c.dupuits@tudelft.nl)

Abstract. Flood defences can be designed as multiple lines of defence. This paper presents an approach for finding an optimal configuration for flood defence systems, based on an economic cost-benefit analysis with an arbitrary number of interdependent lines of defence. The proposed approach is based on a graph algorithm and is, thanks to some beneficial properties of the application, able to traverse large problems. Furthermore, computational efficiency is achieved by delaying cost calculations until they are actually needed by the graph algorithm. A number of case studies were carried out to compare the optimal paths found by the proposed approach with the results of competing methods, and were found to generate ~~(near)~~ identical results. The work presented here makes cost-benefit analyses of complex flood defence systems with interdependent multiple lines of defence both easier and applicable to a broad range of flood defence systems with multiple lines of defence.

1 Introduction

Concerns regarding the safety of people and assets in flood prone areas has led to the construction of flood defence systems all around the world. Some flood prone areas, for example a large part of the Netherlands, face huge potential loss of life and economic value in case heavy flooding occurs. This has led to extensive research regarding estimating the flood risk of flood prone areas. Coupled to this quantification of the flood risk, is the question of 'how safe' a flood prone area should be. An often used approach to help answer this question is a cost-benefit analysis.

Economic ~~optimization~~ optimisation of flood defences, as applied in the Netherlands, is a cost-benefit analysis of the flood risk cost reduction balanced against the investment costs for flood defences. This type of cost-benefit analysis was already developed in the 1950's by Van Dantzig (1956), and is still used and discussed to this day (Eijgenraam, 2006; Kind, 2014). The basic principle behind the economic ~~optimization~~ optimisation of flood defences is finding the minimum of the total costs ~~; the~~ and is illustrated in Figure 1. The total costs (TC, Eq. 1) are the sum of ~~risk (R_{cost}~~ the annual risk costs ($\sum R$) and investment costs (~~I_{cost}) , as shown in Eq. 1. The risk cost is~~ $\sum I$) for a given time period. An annual risk cost (R) is in Eq. 2 defined as the annual probability of flooding ~~times the~~ (P_{flood}) times the annual expected loss incurred due to flooding ~~.~~ (D_{flood}). An alternative term for the annual risk cost is the Annual Expected Damage, or AED. Generally speaking, a larger investment will lead to ~~lower risk~~ a lower AED and this is where the economic ~~optimization~~ optimisation tries to find an optimal ~~(~~solution

(i.e. the lowest total cost)~~situation. This optimal situation can be related to an optimal investment scheme over time (e.g. see Eijgenraam (2006); Kind (2014)).~~ .

$$TC = R_{cost} + I_{cost}$$

$$TC = \sum R + \sum I \tag{1}$$

$$R = P_{flood} \cdot D_{flood} \tag{2}$$

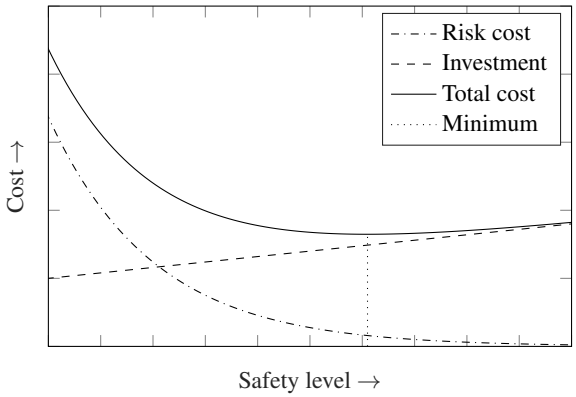

Figure 1. Schematic view of an economic cost-benefit analysis for a flood defence. The total costs are the sum of the risk and investment costs, and the optimum can be found at the minimum of the total costs.

Recent publications regarding economically optimal safety targets, for The Netherlands, can be found in Eijgenraam (2006); Brekelmans et al. (2012); Zwaneveld and Verweij (2014b, a). In Eijgenraam (2006) & Eijgenraam et al. (2016), a set of equations was derived which describe the economically optimal safety target for a single homogeneous flood defence system (dike ring). ~~These equations~~ Because of the incorporated influence of time-dependent parameters such as economic growth or climate model parameters, these equations also describe the quantity of (repeated) investments, as well as the optimal time between these investments. ~~The equations~~ These repeated investments are necessary to 'repair' the effect of for example economic growth (i.e. a higher expected losses in case of a flood) or subsidence (i.e. a higher flood probability). A schematic view of the result of such an economic optimisation with time-dependent parameters is shown in Figure 2. This figure shows that the safety level goes down over time, and recurring investments are needed to repair the effects over time of time-dependent parameters such as economic growth or climate model parameters.

The equations described in Eijgenraam et al. (2016) are analytically solvable, and the method resulted in a global minimum of the total costs for relatively simple homogeneous systems. However, ~~because~~ dike rings in the Netherlands often consist of

mutually different, non-homogeneous sections~~,~~. This meant the homogeneous case needed to be extended. In Brekelmans et al. (2012), a possible, heuristic solution is given by modelling the problem as a mixed-integer nonlinear programming (MINLP) problem. ~~Zwaneveld and Verweij (2014a)~~ Zwaneveld and Verweij (2014b) improved on this method by ~~rewriting the problem as an integer linear programming (ILP) problem. Zwaneveld and Verweij (2014b) further extended their ILP problem to be applied to a case study with three lines of defence.~~ developing a graph based modelling approach to solve the non-homogeneous case to proven optimality.

~~The~~

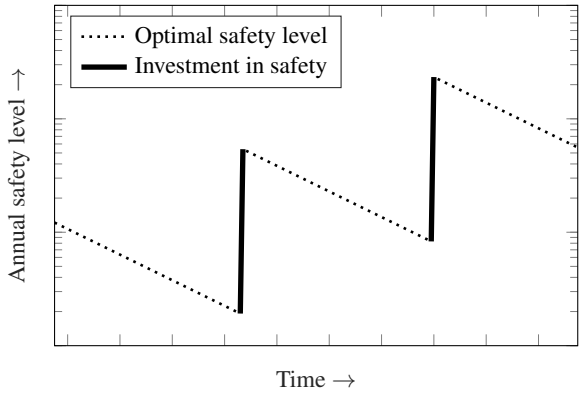

Figure 2. Schematic view of an economic cost-benefit analysis for a flood defence, with time-dependent parameters. Because of these time-dependent parameters (e.g. economic growth or subsidence), recurring investments in safety are needed.

Eijgenraam (2006); Brekelmans et al. (2012); Zwaneveld and Verweij (2014b) assess independent flood prone areas in which individual flood defences within a dike ring area fail under identical circumstances. No interdependencies exists in their modelling approaches. However, the notion of multiple lines of flood defences expresses that failure of one flood defence might alter the ~~flood risk~~ AED of other components. Moreover, a dike ring may fail under different circumstances. An example of a flood defence system with multiple lines of defence which can be found in practice is shown in Figure 3. This notion of multiple lines of flood defences has been, from a flood risk perspective, the main topic in a number of recent papers (e.g. Vorogushyn et al. (2010, 2012); Courage et al. (2013); De Bruijn et al. (2014)). All of these papers showed that viewing the flood defence system as a whole, with multiples lines of defence, resulted in different ~~risk~~ AED estimates than viewing the flood defences as separate, independent elements.

As the ~~flood risk changes, so will the associated (flood) risk costs. This in turn will affect the economic optimization~~AED changes, the economic optimisation will also be affected. Therefore, it makes sense to explicitly integrate the effect of multiple lines of defence on the ~~flood risk~~ AED in the economic ~~optimization routines. However, existing cost-benefit analyses tend to focus on flood defence systems with independent lines of defence (Kind, 2014), or are not (readily) generically applicable (e. g. Zwaneveld and Verweij (2014a)). Therefore, the~~ optimisation routines. A method to provide a modelling approach to the economic optimisation of a flood defence system with multiple dependent and independent dikes was first

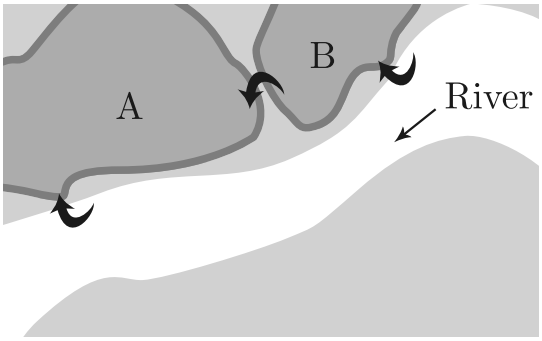

Figure 3. A hypothetical example of a system with multiple lines of defence. Area A has, aside from its own flood defences, an additional flood defence layer in the form of area B and its defences; this is because breaches (indicated by the curved arrows) at area B can impact the ~~flood risk~~ AED (Annual Expected Damage) of area A.

presented in Zwaneveld and Verweij (2014a). In Zwaneveld and Verweij (2014a) (in Dutch), a graph based modelling approach is used to obtain economically optimal safety norms and heights for multiple lines of flood defences. Furthermore, they mentioned that the economic optimisation problem can be formulated in the form of a minimal cost flow graph or a shortest path problem. Three approaches to solve economic optimal safety problems for multiple flood defences were identified by Zwaneveld and Verweij (2014a) & Verweij (2014): a heuristic approach based on closed form formulas, a dynamic programming/shortest path approach (as also used in Eijgenraam et al. (2016)) and a branch-and-cut/ILP approach. In Zwaneveld and Verweij (2014a) the branch-and-cut/ILP-approach is preferred and applied. An English description of of the model in Zwaneveld and Verweij (2014a) can be found in (Yüceoglu, 2015, Chapter 5).

However, a consequence of using an ILP approach in Zwaneveld and Verweij (2014a) is that, prior to starting the optimisation routine, all AED estimates for each and every possible combination of flood defences in time need to be computed. Generally speaking, finding AED estimates for a number of these combinations will not be necessary. For example, it is unlikely that is economically optimal to keep all flood defences at their lowest level for the next 300 years. Calculating these AED estimates can be costly, especially if hydrodynamic interactions are included: acquiring a single AED estimate can take hours (De Bruijn et al., 2014) or even days (Courage et al., 2013). In these cases, computationally efficiency will be largely determined by the time it takes to compute AED estimates.

Furthermore, the method of Zwaneveld and Verweij (2014a) is modelled in GAMS and solved using CPLEX. While the method of Zwaneveld and Verweij (2014a) is in principle applicable to an arbitrary number of lines of defence, in practice this means manually extending the model code with new equations that implement any additional lines of defence. While these extensions are trivial for anyone with experience in integer programming and GAMS, we believe that by automating these steps the threshold for using and applying these models can be lowered.

Reducing the number of AED estimates that will be computed can be done based on the principle of 'lazy evaluation', which delays calculations until they are actually required. However, 'lazy evaluation' requires a tight coupling between the AED estimation and the economic optimisation routine. This tight coupling needs to be technically and organisationally possible.

Organisationally, it is possible for projects which are carried out by a single team combining all relevant disciplines; in the remainder of this research we assume that the organisational requirement is fulfilled. Technically, the economic optimisation routine needs to be able to dynamically call the AED estimation function during its optimisation process. However, optimisation routines typically expect a pre-calculated set of data, which means an optimisation routine will need to be modified in order to support 'lazy evaluation'.

Therefore, we intend to further investigate the shortest path based approach in order to solve the problem of an economic optimisation for multiple lines of flood defences. The aim of this paper is to find a generic, computationally efficient approach for finding the economically optimal configuration of a flood defence system with an arbitrary number of defence lines. These multiple lines of defence can be dependent on each other (i.e. influence each other~~'s risk). We intend to~~ 's AED). The reliability and performance (in terms of number of AED calculations) of finding economically optimal targets will be tested by comparing the results of the proposed method with a number of benchmark problems. We will accomplish this aim with the following approach:

- ~~A generically applicable, flexible representation of the problem space to being able to use an arbitrary number of defences.~~
- Computational efficiency will be primarily obtained by minimising the number of (time-consuming) ~~risk~~ Annual Expected Damage (AED) computations in the algorithm until they are actually required ~~. Risk computations, especially those with multiple lines of defence, can be demanding because of the involved hydrodynamic and flooding simulations (e.g. see Courage et al. (2013) or De Bruijn et al. (2014)) .~~ (i.e. 'lazy evaluation')
- ~~The reliability of finding economically optimal targets will be tested by comparing the results of the proposed method with a number of benchmark problems.~~ A generically applicable, flexible representation of the problem space will be presented which is able to use an arbitrary number of defences. Specifically, this entails generating a graph in an automated way based on an arbitrary number of lines of defence

Section 2 starts with a description of the application and a description of the applied algorithm. Implementation details, focusing on the computational efficiency of the algorithm, are discussed in Section 3, as well as a list of potential future improvements to the algorithm. Next, the proposed approach is applied to some simplified case studies in Section 4, and is followed by a discussion (Section 5) regarding the relevance of the proposed approach. Finally, the results and experiences are concluded in Section 6.

2 An algorithm for flood defence systems with multiple lines of defence

2.1 Programmatic representation of the solution space

A common choice to present optimisation problems is to use graph algorithms (Cormen, 2009). Regarding the economic optimisation of flood defences, this choice was also made in Zwaneveld and Verweij (2014b). An example of ~~such~~ a graph for

a single flood defence is shown in Figure 4. The graph shows the possible investments over time for a single flood defence. In this graph the vertices (dots) are the possible heights the flood defence can have at a certain point in time. In order to go the next point in time, edges are drawn which connect a vertex to all the possible vertices in the next point of time.

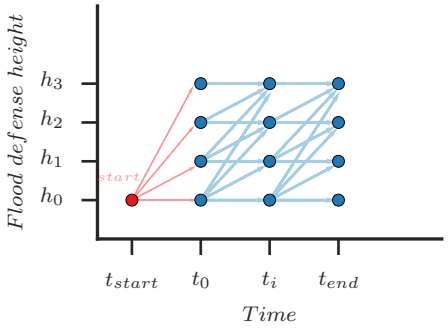

Figure 4. Graph where the vertices (dots) at each time step are connected via edges (arrows) to the next time step.

These points in time are not fixed; the amount and position can be altered to the needs of a particular problem. In practice, these points in time can be related to the (political) decision process of a particular problem: if the relevant flood defences are reviewed and (if necessary) reinforced every five years, it would make sense to have a graph that corresponds to these points in time.

Generally speaking, edges in a graph can be directed or undirected. However, steps backwards in time do not make sense for investment schemes. Therefore, only edges directed forward in time are used. The edge cost (or weight) of an edge is the total cost (~~risk cost~~ AED plus investment cost) of moving between the connected vertices. Furthermore, it is assumed that flood defences will not be intentionally decreased to a lower level, which is why, for example, there are no edges running from h_1 to h_0. The starting point of the graph is denoted with start in Figure 4 at time t_{start} at a height equal to the current height (h_0).

In case of multiple lines of defence, our method takes into account that flood defences can be interdependent ~~with regard to the risk of the area protected by the flood defences . This~~ and interact with each other hydrodynamically. This means that the AED of the system of defences can potentially be influenced by each defence, which also means that each combination of flood defence levels has to be considered relevant. ~~These combinations can be found~~ For a graph with multiple interdependent flood defences, these combinations replace the height of a single flood defence on the y-axis in Figure 4. These combinations of heights for multiple flood defences can be obtained by computing the Cartesian product of the flood defence levels of all the involved defences. ~~If~~ For n flood defences, the Cartesian product equation for determining the combinations is shown in Eq. 3:

$$\prod_{i=1}^{i=n} \boldsymbol{X}_i = \boldsymbol{X}_1 \times \ldots \times \boldsymbol{X}_n$$

$$= \{(x_1, \ldots, x_n) | x_1 \in \boldsymbol{X}_1, \ldots, x_n \in \boldsymbol{X}_n\} \quad (3)$$

where $\boldsymbol{X}_i$ is a vector containing all the flood defence levels of flood defence i, and x_i is a realisation of vector $\boldsymbol{X}_i$ (i.e. a flood defence level for flood defence i). If all vectors $\boldsymbol{X}_i$ are of the same length y, the total number of combinations will be y^n.

The number of relevant system combinations reduces significantly if each flood defence can be optimised independently of the other flood defences in the system. The assumption of independence can be made if none of the flood defences in a system have a (significant) influence on the AED estimates of the other flood defences. The total number of system configurations under the independence assumption is $n \cdot y$, as each flood defence could then be optimised separately (e.g. using a graph per flood defence similar to the graph shown in Figure 4. If only some defences are independent from other flood defences in the system, this is considered a special case of our approach. In that case, our method of using the Cartesian product is still valid and applicable~~. However, in case of independence it can be worthwhile to make adaptations to the method for computational efficiency, by making use of the attractive features of independence. Such an approach will be~~, although it will result in larger than necessary graph. In case of some independent elements, a possible approach to reduce the size of the graph is discussed in Section 3.4.

Figure 5 shows an example of the Cartesian product for two flood defences where each flood defence has two possible heights. The graph in Figure 5 resembles the graph in Figure 4 for a single flood defence. Similar to Figure 4, edges in Figure 5 are only drawn to vertices containing sets of heights equal or greater than the set of heights in the vertex at the origin of the edge. However, because Figure 5 has two defences instead of one, the outgoing edges are slightly different when compared to Figure 4. For example, the height combination h_{A0}, h_{B1} is never connected to h_{A1}, h_{B0} (since that would correspond to a reduction in height for defence B).

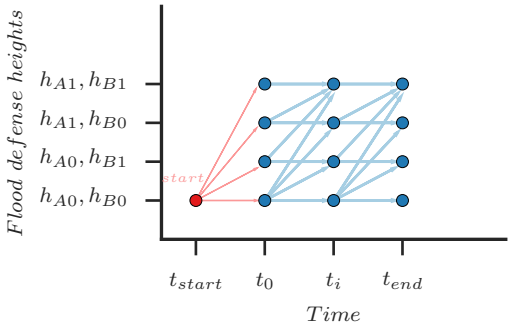

Figure 5. Graph with vertices (dots) and edges (arrows) for two defences (A and B). Each defence has two possible heights.

2.2 Implementation of a graph algorithm

In general terms, a graph algorithm will iterate over vertices in a graph in an effort to find the path with the lowest costs between a given start and end vertex. However, in the graphs of Figure 4 and Figure 5 t_{end} contains a number of possible end points, which means that the algorithm will need to find as many optimal paths as there are end points in the graph. In order to only have to run the algorithm once, a stop vertex is added; the graph of Figure 5 with an additional stop vertex is shown in Figure 6.

The edges running towards this stop vertex are all given a weight of zero. Now, the algorithm only has to find a single optimal path between t_{start} and t_{stop}. Why this is an efficient contribution is illustrated in Section 2.4.

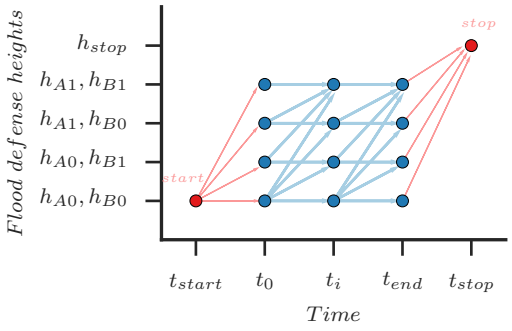

Figure 6. The graph of Figure 5 with an additional stop vertex.

The graph as shown in Figure 6 is a graph with directed non-negative edges. For this kind of graph, a number of algorithms can be used to find the shortest (optimal) path in a graph, for example: the Dijkstra algorithm (Dijkstra, 1959), the A* algorithm (Hart et al., 1968), and the Uniform Cost Search (e.g. (Verwer et al., 1989)). All three can be considered to be part of the family of best-first search algorithms, where both the Dijkstra and the Uniform Cost Search (UCS) algorithms can be seen as a special case of the A* algorithm~~(i. e. A* without the use of a heuristic graph function). For our application, we did not come up with a heuristic function, which reduces the choice of a graph algorithm to either Dijkstra or UCS.~~ .

Typically, the best-first search algorithms are implemented with a min-priority queue. A min-priority queue holds a sorted list of vertices, where the sorting is based on the cost of reaching that vertex from the start vertex; the vertex with the lowest cost is at the top of the queue. This list of vertices in the priority queue constitutes of, depending on the implementation, either all vertices in the graph (Dijkstra as implemented in Cormen (2009)), or only the vertices already visited by the graph algorithm (UCS). A comparison between the two algorithms can be found in Felner (2011), where the priority queue as implemented by UCS was found to be faster and using less memory. We consider this a relevant advantage, as the number of vertices can be large when using the Cartesian product of flood defence levels (Section 2.1). For this reason, we chose to implement the UCS algorithm.

In Eijgenraam et al. (2016), a dynamic programming approach was used, which is related to the shortest-path algorithms discussed thus far. However, the Dijkstra algorithm (and by extension the UCS and A* algorithms) are seen in Cormen (2009) as a part of the *greedy* shortest-path algorithms family, which in Cormen (2009) is clearly defined as a different type of algorithm than dynamic programming. Greedy algorithms are typically much faster than dynamic programming approaches, at the expense of not always finding the optimal solution (because less possible solutions are considered). The optimality condition is further discussed in Section 2.4. Nevertheless, because less possible solutions are considered in a greedy algorithm, this can also lead to a part of the graph never being visited by a greedy algorithm. Combined with 'lazy evaluation', this can lead to a significant reduction in the number of AED calculations which are actually executed; see also Section 3.3. Applying the UCS

algorithm to a graph such as shown in Figure 6 begins with creating a priority queue which only contains the start vertex. After this initialization, the iteration process is started. Each iteration starts with taking out the vertex with the lowest cost known thus far from the priority queue (which is the top entry in the queue). Taking out means the optimal route (lowest cost) from the start vertex to this vertex now known. The vertex that has just been taken out of the priority queue is then queried in the graph to find all the connecting vertices in the next time step. Each connecting vertex is added to the priority queue if the vertex is not already in the queue. If the vertex already exists in the queue, the weight is only updated if the newly proposed cost is lower than the known cost so-far. Iteration continues until at the start of an iteration the stop vertex is the top entry in the priority queue. An actual example of the application of this algorithm will be elaborated in Section 2.3.

2.3 Example application of the algorithm in an economic ~~optimization~~optimisation

This section shows a simple example of an economic ~~optimization~~ optimisation for a single flood defence. While this example uses a single flood defence for simplicity, the same principles apply for multiple flood defences. Regarding the investment ~~and risk costs~~ costs and AED estimates, if a vertex at t_1 is connected to another vertex with a larger height at t_2, it is assumed that the actual heightening occurs at t_1. This leads to a slightly different graph than the conceptual implementation shown in Section 2.1 & 2.2, and is emphasized by drawing the edges of the figures in this example (i.e. Figures 7, 8 & 9) in a way which is visually more consistent with the timing of the investment decision.

The result of the first two iterations is shown in Figure 7, where the start vertex is labelled with the number 1. In this example, the start vertex is associated with a height of 4.25 meter and starts at $t = 0$, identical to vertex 2. Because the path to vertex 2 is the only possible path, vertex 2 is the only addition to the priority queue. In the next iteration, vertex 2 is taken out of the priority queue as it is the vertex with the lowest total cost. The total cost to reach vertex 2 is 0, because there was no heightening (height remains at 4.25 meter) and no time expired ($t_{start} = 0$); the ~~risk cost~~ AED is zero because time needs to expire for risk to occur. From vertex 2, the number of possible next steps and associated total costs are computed and added to the priority queue, as illustrated in Figure 7. Note that the total costs to reach for example vertex 22 consists of the total cost from t_{start} to $t = 100$, not just the cost from $t = 50$ to $t = 100$.

The algorithm will continue for a while, until the situation of Figure 8 is reached where vertex 24 is taken out of the priority queue. The new found total costs for vertex 29, 30 and 31 are not lower than the total cost for vertex 25, which means the algorithm takes a step back and continues from vertex 25. From vertex 25, vertex 30 and 31 are re-evaluated in Figure 9, where only vertex 30 results in lower costs than the existing options. This means that only vertex 30 is updated with the new, lower, total cost in the priority queue. Additionally, if hypothetically vertex 31 is the vertex with the lowest cost, the optimal path would revert back to using vertex 24 instead of vertex 25 (because the path from vertex 25 to 31 has higher costs than the path from vertex 24 to 31).

2.4 Global optimal solution

The UCS algorithm finds the shortest path in a graph, see for example Felner (2011) for a recent elaboration regarding the 'correctness' of the UCS algorithm or Gelperin (1977) for a proof regarding A* (UCS can be considered a special case of A*).

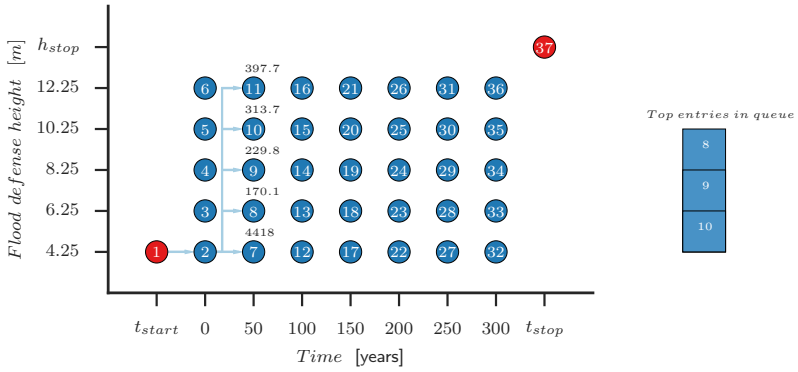

Figure 7. The first two iterations of the graph algorithm with a min-priority queue. The vertices available in the priority queue are those which have total costs above their respective vertices, and the first three entries are shown in the column on the right. Note that because the choice was made to connect the start vertex to vertex 2, vertices 3 - 6 will not be visited.

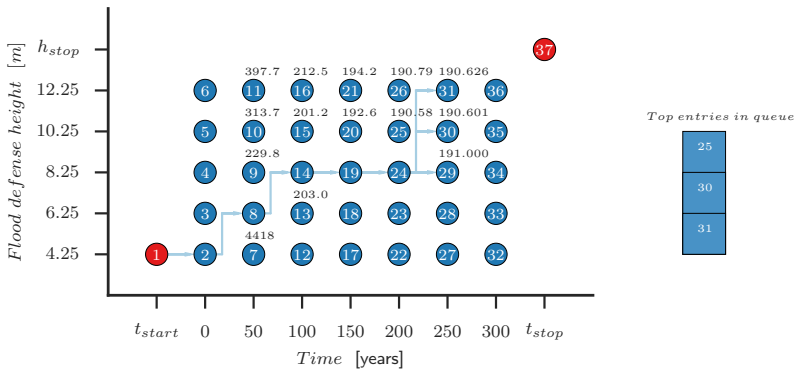

Figure 8. After six iterations with the graph algorithm, vertices 29, 30 and 31 are added to the priority queue. However, in this case the algorithm makes a step back in time, because vertex 25 is the item with the lowest total cost in the priority queue.

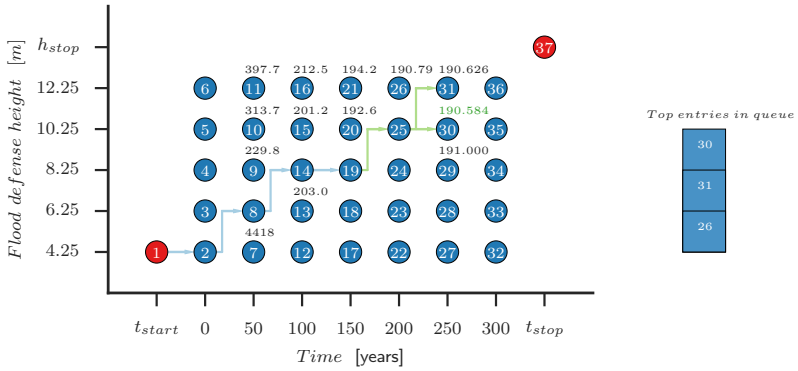

Figure 9. During iteration seven, the old path is abandoned, and an alternative path with vertex 25 instead of vertex 24 is taken. Vertices 30 and 31 are already in the priority queue (calculated from vertex 24), and will only get updated if the total costs from vertex 25 are lower (which is the case for vertex 30).

What remains is whether the additional stop vertex of Section 2.2 leads to a potential heuristic solution or still to the optimal path. However, assuming that the optimal path towards the stop vertex is found, whichever vertex at t_{end} is part of that optimal path has to be the optimal choice. Otherwise, the path towards the end vertex is not optimal, which contradicts the earlier mentioned proofs. In order to further test the performance of the proposed method, Section 4 will compare numeric results from our proposed method to other approaches. These approaches are known to give global optimal results~~, or at least very close to the global optimum~~.

2.5 Overview of the approach

A general overview of the approach discussed in the previous sections is shown in Figure 10. The method is composed of four steps: input, pre-processing, processing and post-processing. Of these steps, user interaction is only required at the input step. The rest of the steps run automatically. Specifically, the user needs to supply vectors of flood defence levels per flood defence, a time vector and a function which can calculate the cost of an edge in the graph. In the following steps, the graph is created (pre-process), the optimal path is found (process), and the optimal path is shown (post-process).

~~The first two iterations of the graph algorithm with a min-priority queue. The vertices available in the priority queue are those which have total costs above their respective vertices, and the first three entries are shown in the column on the right. Note that because the choice was made to connect the start vertex to vertex 2, vertices 3 - 6 will not be visited. After six iterations with the graph algorithm, vertices 29, 30 and 31 are added to the priority queue. However, in this case the algorithm makes a step back in time, because vertex 25 is the item with the lowest total cost in the priority queue. During iteration seven, the old path is abandoned, and an alternative path with vertex 25 instead of vertex 24 is taken. Vertices 30 and 31 are already in the priority~~

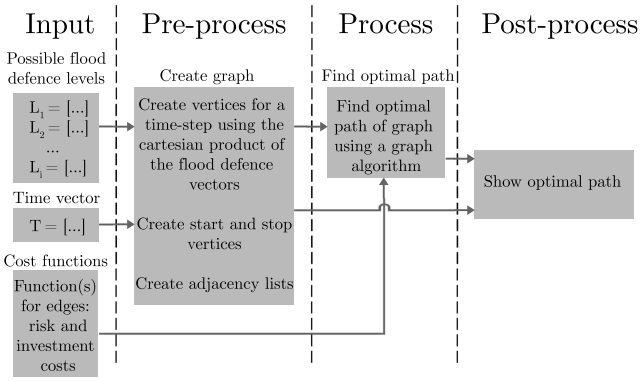

Figure 10. Overview of the approach using a graph and graph algorithm. In our approach, the graph algorithm is the UCS algorithm. The input column is the only part what the user should provide, the other steps run automatically.

~~queue (calculated from vertex 24), and will only get updated if the total costs from vertex 25 are lower (which is the case for vertex 30).~~

3 Efficiency improvements

The economic ~~optimization~~ optimisation of multiple lines of defence, implemented in a graph using Section 2, can potentially lead to large numbers of vertices and even ~~large~~ larger numbers of edges. For example, for eight lines of defences with six possible heights the number of vertices per time step is approximately 1.68 million (6^8), while the number of edges per time step is even larger at approximately 35 billion. For large problems such as these, storing all the possible vertices and edges would lead to huge data structures ~~. Fortunately, the graphrepresentation as presented in Section 2.1 has repetitive properties which can be used reduce the size of the data structures.~~ and a huge number of AED calculations. This requires both an efficient implementation of the graph, and an efficient evaluation of AED calculations (i.e. as few as possible). An efficient graph implementation is discussed in Sections 3.1 & 3.2, while the efficient evaluation of AED calculations is discussed in Section 3.3. Potential further efficiency improvements are discussed in Section 3.4.

3.1 Repetitiveness in lists of vertices

Even though the graphs of Section 2.1 can be classified as sparse graphs (number of edges is much smaller than the number of vertices squared, Cormen (2009)), the number of edges is still much larger than the number of vertices. Therefore, we first focused on data structures related to the edges of a graph. For sparse graphs, these are the adjacency lists: a group of vertices connected via edges stemming from a source vertex in a previous time step. In these adjacency lists, repetitiveness can be found with respect to two aspects.

The first repetitive aspect is the similarity of adjacency lists for the same combination of flood defence levels at different time steps (except for the adjacency lists at t_{end}). In Figure 7, vertices with the same combination of flood defence levels at different time steps are for example vertices 2, 7 and 12. The adjacency lists for these vertices are shown in Figure 8, where it is apparent that the adjacency list for the next time step can be found by adding an offset to the elements of the adjacency list of the current time step. For example, the adjacency list of vertex 2 can be turned into the adjacency list of vertex 7 by adding the total number of combinations in each time step (which is five in Figure 11)

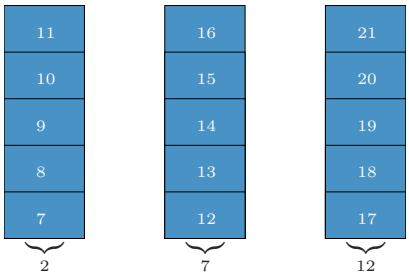

Figure 11. The adjacency lists for vertices 7 and 12 of Figure 7 can be obtained by adding an offset to the adjacency list of vertex 2.

The second repetitive aspect is for adjacency lists between vertices in the same time step. Because the lowest vertex in each time step (e.g. vertices 2, 7, 12, 17, 22 and 27 in Figure 7) has outgoing edges running to each and every vertex in the next time step, higher vertices (e.g. in Figure 7, vertex 8 is 'higher' than vertex 7) contain a subset of the adjacency list of the lowest vertex. In other words, outgoing edge lists in a single time step can be generated dynamically by shrinking the adjacency list of the lowest vertex in a time step. This is shown in Figure 12.

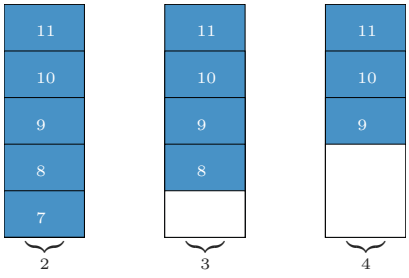

Figure 12. The adjacency lists for vertices 3 and 4 of Figure 7 are reduced sets of the adjacency list for vertex 2.

The combination of these two repetitive characteristics results in that only a single adjacency list needs to be stored in memory (i.e. the adjacency list of the lowest vertex in the first time step). This single adjacency list can be adapted to most

vertices in the graph by means of offsetting and shrinking the stored adjacency list. Notable exceptions are the adjacency lists for the vertices at t_{end}, but the adjacency lists for these vertices are already known and only contain the stop vertex.

3.2 Conditionally removing edge connections

Besides reducing the size of the data structures associated with a graph, the adjacency list associated with a vertex can also be reduced under certain conditions. Typically, the time between improvements in flood defences is large (in the order of 50 years), due to either high (fixed and variable) costs associated with investments in flood defences, or long planning periods (Zwaneveld and Verweij, 2014b). Therefore, if one or multiple flood defences have been strengthened recently, the adjacency list can be reduced to only contain vertices that keep the recently strengthened flood defence(s) at the current level(s). However, this so called 'waiting time' before new investments are considered has to be chosen with care, because the waiting time should not influence the optimal time between investments. Nevertheless, a correctly chosen waiting time can greatly improve the run time of the algorithm, because of the significant reduction in number of edges that need to be evaluated. This reduction is shown in Figure 13, where the total number of visited vertices is plotted as a function of the 'waiting time'; the underlying problem that is solved by the algorithm is the same problem as shown in Section 4.3.

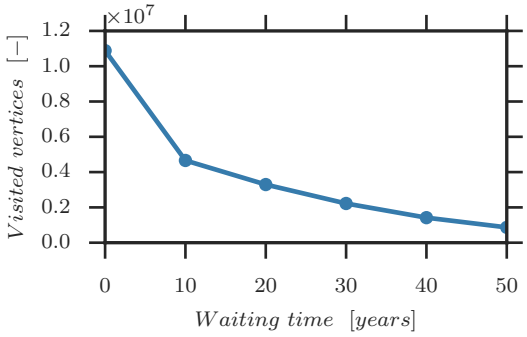

Figure 13. ~~Reduction in the total~~ Total number of visited vertices as a function of the waiting time for the example of Section 4.3.

3.3 Reducing the number of ~~risk~~ AED calculations

In the overview of Figure 10 it is implied that the ~~risk~~ AED calculations belonging to an edge are only carried out when that edge is visited by the graph algorithm. Provided that a graph algorithm does not visit all vertices, delaying ~~(risk)~~ AED calculations belonging to an edge until that edge is visited leads to less ~~risk~~ AED calculations than the total number of ~~edges~~possible AED calculations in a particular graph. In contrast, if ~~(risk~~AED (or more generally, cost) calculations are done before a graph algorithm is ~~initialized, all edges~~ initialised, all possible AED calculations need to be calculated beforehand.

As an example, Figure 14 shows the number of times each vertex is visited is in the example of Section 2.2. The majority of the vertices in Figure 14 get visited once, but a significant proportion is never visited by the graph algorithm; these vertices have a zero above their indices. A small proportion of the vertices, specifically vertices 30 and 31, are visited twice; the reason for

this re-visiting can be seen in Figure 9. To avoid completely re-doing ~~risk~~ cost calculations upon a revisit, parts of a calculation can be cached in order to reduce the computational penalty incurred by revisiting vertices. The total number of possible AED calculations is the number of years multiplied with the number of options on the y-axis; for Figure 14 this leads to a total number of AED calculations of 1505 (or $301 \cdot 5$). Because a number of vertices do not get visited by the algorithm, the number of actual executed AED calculations goes down to 1000, or approximately 66% of all possible AED calculations.

Furthermore, the number of AED calculations can be further reduced by using the 'waiting time' of Section 3.2. In Section 3.2, it was found that a minimum waiting time between investments will lead to less edges being evaluated by the algorithm. This also implies that less AED calculations will be executed. Using the same example as in Figure 13, the reduction in the percentage of actual executed AED calculations is given as a function of the waiting time in Figure 15. Between using a waiting of 0 years (i.e. no minimum waiting time at all) and a waiting time of 50 years the number of AED calculations goes down from approximately 60% to 40% for the example of Section 4.3.

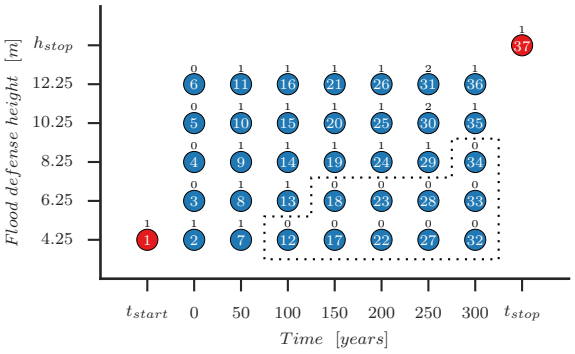

Figure 14. Number of times each vertex is visited by the algorithm for the example in Section 2.2. The dotted area emphasizes that a part of the graph is never visited, while vertex 30 and 31 get visited twice.

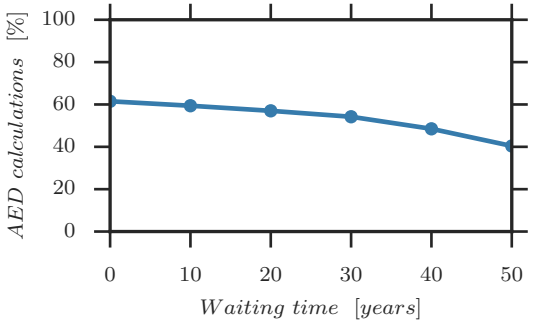

Figure 15. Percentage of actual executed AED calculations as a function of the waiting time for the example of Section 4.3.

3.4 Potential improvements and special cases

Further improvements can be made both to the graph implementation and to the implementation of the algorithm. The algorithm was implemented as a single process; a performance improvement might be found by utilizing parallel programming. The first place where parallel programming could be beneficial is the loop over an adjacency list. This is because the ~~risk cost calculations (i.e. edge weights) are potentially expensive to compute, therefore parallelizing~~ potentially expensive AED calculations are done as part of determining an edge weight. Therefore, parallelising the loop over an adjacency list over multiple computational nodes can lead to significant performance improvements.

Furthermore, regarding the graph implementation, a special case is a flood defence system which has independent flood defences. Section 2.1 uses the Cartesian product of flood defence options, which has the underlying notion that all flood defence lines are interdependent. If some flood defences are independent (i.e. the defences protect different, independent areas), this leads to an inefficient graph. The independency of flood defences can be used in an adapted graph representation in order to get an efficient graph. While we did not implement this, a way to solve this for the system in Figure 16 is shown in Figure 17, which uses 'subgraphs' to reduce the number of combinations.

These subgraphs are small graphs which only contain the number of strengthening options for a single defence for a single time period (e.g. in Figure 17, from t_{i-1} to t_i). Additionally, the subgraphs take into account what the level is of the influential defences (e.g. in Figure 17, the front defence B is the only influential defence for the rear defences). The use of subgraphs leads to a smaller number of combinations, as the Cartesian product would have resulted in a total number of combinations (or vertices per time step) of 5120 (~~$5*4^5$~~$5 \cdot 4^5$). With subgraphs, the number of combinations is reduced to 100 (~~$5*5*4$~~$5 \cdot 5 \cdot 4$).

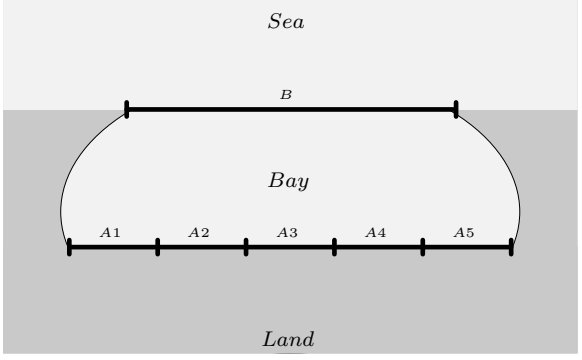

Figure 16. A top view of a system with a front line defence (B, five possible safety levels) and five rear defences ($A1 - A5$, each has four possible safety levels). The front defence influences the rear defences, but the rear defences do not influence each other.

4 Results for simplified flood defence systems

In order to test the performance of the proposed algorithm versus some existing approaches, three cases are investigated. For simplicity, these three cases will be based upon a common set of investment and ~~risk~~ AED relations, as well as a common set of

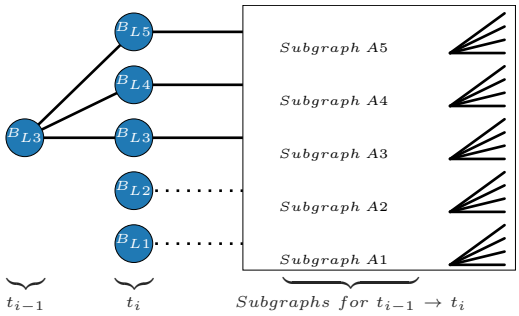

Figure 17. Part of the graph belonging to the system of Figure 16 for the period t_{i-1} to t_i. Because the rear defences do not influence each other, subgraphs are used for the rear defences.

input values. The values and symbols used in this section are largely copied from (Eijgenraam, 2006, page 34) and reproduced in Table 1, with only minimal changes. These AED and investment cost relations consist of simple formulations which were specifically chosen for exhibiting the approach, for ease of reproducibility, and for showing the efficiency regarding the number of AED calculations. In practice, AED estimates can be quite complex and/or have a high computational burden, especially when flood defences are modelled to have hydrodynamic interactions with each other. For example, a single AED estimate for a complex flood defence system with interdependencies can take hours (Klerk et al., 2014) or even days (Courage et al., 2013).

The common set of investment (I) and ~~risk (~~AED (or flood risk cost, R) relations are similar to the relations used with the data of Table 1 in Eijgenraam (2006). The sum of the investment ~~and risk~~ cost and AED is the total cost, which needs to be ~~minimized~~ minimised in order to get economically optimal safety targets:

$$\text{Total Cost} = \int_0^{\infty} R(t)dt + \sum_{t=0}^{\infty} I(t) \tag{4}$$

$$R(t) = P_0 e^{-\alpha(H_t - H_0 - \eta t)} V_0 e^{\gamma t} e^{-\delta t} \tag{5}$$

$$I(t) = (C_v u_t + C_f \text{sign}(u_t)) e^{-\delta t} \tag{6}$$

where sign(u_t) is used to prevent fixed costs in case there is no heightening u_t. This sign(u_t) function returns zero if the heightening u_t is equal to zero, and returns one when the heightening u_t is larger than zero.

4.1 Single flood defence

For a single flood defence, with the values of Table 1, an analytical solution can be found in (Eijgenraam, 2006, page 35). This solution consists out of an initial dike height increase coupled with a periodical, constant dike increase over an infinite time horizon. The initial increase was found to be 236 centimetres, with a periodical increase of 129 centimetres every 73 years.

Table 1. Variables and values taken from Eijgenraam (2006) for the ~~risk~~ AED and investment equations in this section. NLG refers to the currency used in the Netherlands prior to the euro.

Name	Unit	Symbol	Value
Height above mean sea level, base	cm	H_0	425
Annual exceedance probability belonging to H_0	-	P_0	0.0038
Parameter exponential distribution water level	1/cm	α	0.026
Increase water level	cm/year	η	1
Damage by flooding in 1953	10^6 NLG	V_0	20000
Economic growth	1/year	γ	0.02
Rate of interest (real)	1/year	δ	0.04
Variable costs of investment	10^6 NLG/cm	C_v	0.42
Fixed costs of investment	10^6 NLG	C_f	61.7
Heightening of the flood defence at time t	cm	u_t	-
Height of the flood defence at time t	cm	H_t	-

Because the approach introduced in this paper is a numerical approach, a finite time period has to be used instead of an infinite time horizon. Similar to Zwaneveld and Verweij (2014b), we choose to use a time period of 300 years with, for this application, steps of one year. The possible heights were discretized using a range starting from 425 to 1225 cm, with steps of one centimetre. Note that these step sizes (and dimensions) were deliberately chosen to be on par with the accuracy level of the analytical solution. In practice, these step sizes would probably be too detailed for the practical attainable accuracy in flood defence construction (see also Zwaneveld and Verweij (2014b)). Furthermore, the total number of possible AED calculations in this problem is 241,101 (or $801 \cdot 301$). Of these, 137,971 were actually executed by the UCS algorithm, which corresponds to using only 57% of all possible AED calculations. If a 'waiting time' of 50 years is used, the solution is unaffected but the percentage of executed AED calculations goes down to 43%.

A comparison of the results found using the algorithm and the analytical solution is shown in Figure 18. The algorithm found an initial increase of 235 cm, with three additional increases in height at 73 years apart. These three were found to be 129 cm, 130 cm, and 132 cm. The last increase is different from the analytical solution, and can be attributed to being close to the end of the time horizon. A finite time horizon implies that there is no ~~risk~~ AED beyond the time horizon, which explains why there

is no investment found by the algorithm in year 292. To compensate for the lack of an investment in year 292, the investment in year 219 is slightly larger. This explanation is supported by results with a time horizon of 400 years, where the heightening in year 219 changes to an expected 129 centimetres. These deviations near to the time horizon underline that if a certain point in time is considered relevant, the used time horizon should stretch significantly beyond that point in time. However, this is a general problem with all numerical methods, because of the required finite time horizon, and not a specific issue related to the approach proposed in this paper. Furthermore, in practice this problem can be circumvented by setting the time horizon used in the algorithm to sufficiently exceed the practically required time horizon.

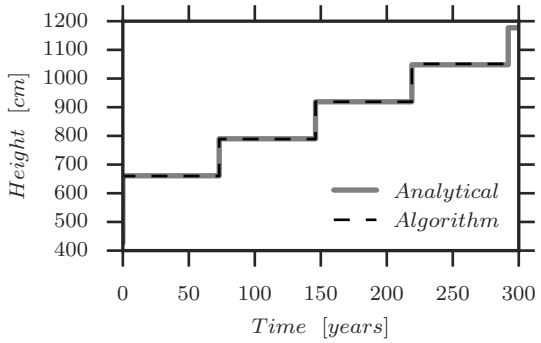

Figure 18. The investment scheme found using the algorithm is almost identical to the analytical solution.

4.2 Two independent lines of defence

In the next example two defences are investigated using the graph algorithm, both with the same characteristics as the single flood defence in the previous section. However, the step size for the heights is increased to 20 cm in order to test the response of the algorithm to larger step sizes. Expected is that, despite the less detailed step size, the investment scheme for both defences should be identical to each other and close to the analytical solution provided in the previous section.

Indeed, the results of the algorithm, illustrated in Figure 19, show that both defences are initially increased with 240 cm, while in both year 75 and 143 the defences are increased with 120 cm, and finally in year 212 with 140 cm. Clearly, the larger step size in height leads to larger differences when compared to the analytical solution. Nevertheless, any overshoot/undershoot of the height is 'repaired' in the duration between investments, keeping the solution of the optimal path stable and close to the analytical solution. Furthermore, the total number of possible AED calculations in this problem is 24,682 (or $2 \cdot 41 \cdot 301$). Of these, 14,510 were actually executed by the UCS algorithm, which corresponds to using only 59% of all possible AED calculations. If a 'waiting time' of 50 years is used, the solution is unaffected but the percentage of executed AED calculations goes down to 48%.

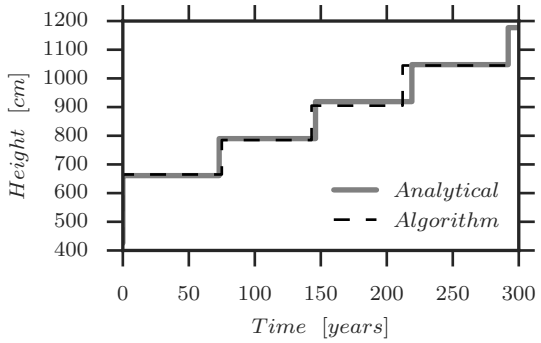

Figure 19. The two (independent) lines of defence have an identical solution with the approach proposed in this paper, and are (even with the usage of larger step sizes) good approximations of the known analytical solution.

4.3 Two dependent lines of defence

The final case is similar to the case with two independent lines of defence, however the second defence is now dependent on the performance of the first defence. This dependency is illustrated in Figure 16, and is a simplified version of the case discussed in Dupuits et al. (2016).

Figure 20. A coastal system with two lines of defence. This figure is an adaption from an illustration found in Dupuits et al. (2016).

The dependency between the defences in Figure 20 is implemented by adapting the ~~risk~~ AED equation of Eq. 5 as follows:

$$R(t) = \left(P_1 P_{2|1} + (1 - P_1) P_{2|\bar{1}}\right) V_0 e^{\gamma t} e^{-\delta t} \tag{7}$$

$$P_i = P_0 e^{-\alpha_i (H_{i,t} - H_0 - \eta t)} \tag{8}$$

where P_i is a generic formulation used for the failure probabilities P_1, $P_{2|1}$ and $P_{2|\bar{1}}$. The probabilities and are the failure probabilities of the second defence, dependent on the failure ($P_{2|1}$) or non-failure ($P_{2|\bar{1}}$) of the first defence, where the failure probability of the first defence is denoted by P_1. Similarly, the investment equation in Eq. 6 is expanded to include different costs for the two lines of defence:

$$I(t) = \left(C_{v1} u_1 + C_f \text{sign}(u_1) C_{v2} u_2 + C_f \text{sign}(u_2)\right) e^{-\delta t}$$

$$I(t) = (C_{v1}u_1 + C_f \text{sign}(u_1))\, e^{-\delta t}$$

$$+ (C_{v2}u_2 + C_f \text{sign}(u_2))\, e^{-\delta t} \quad (9)$$

The new variables used in Eqs. 7, 8 & 9 are listed in Table 2. The solution found with the approach proposed in this paper was checked with the method proposed in Zwaneveld and Verweij (2014a); the outcomes of both methods were found to be identical and are shown in Figure 21. Furthermore, the total number of possible AED calculations in this problem is 505,981 (or $41^2 \cdot 301$). Of these, 311,190 were actually executed by the UCS algorithm, which corresponds to using only 62% of all possible AED calculations. If a 'waiting time' of 50 years is used, the solution is unaffected but the percentage of executed AED calculations goes down to 40%.

Table 2. Additional variables used in Eqs. 7, 8 & 9, complementary to Table 1.

Name	Unit	Symbol	Value
Annual exceedance probability belonging to H_o	-	P_0	0.01
Exponential parameter for defence 1	1/cm	α_1	0.026
Exponential parameter for defence 2 for $P_{2\|\overline{1}}$	1/cm	$\alpha_{1\|\overline{2}}$	0.052
Exponential parameter for defence 2 for $P_{2\|1}$	1/cm	$\alpha_{1\|2}$	0.026
Variable costs of investment for defence 1	10^6 NLG/cm	C_{v1}	0.21
Variable costs of investment for defence 2	10^6 NLG/cm	C_{v2}	0.42

5 Discussion

The proposed approach (see also Figure 10) in this paper is based on a best-first graph algorithm, which is relatively easy to implement in most general or scientific programming languages. In our opinion, this is a significant advantage over linear programming type algorithms, especially for those who are not familiar with the implementations of linear programming as proposed by ~~Kind (2014); Zwaneveld and Verweij (2014a). The~~ Zwaneveld and Verweij (2014a). Although the application area is ~~roughly similar as for Kind (2014); Zwaneveld and Verweij (2014a), with the notable difference that our approach~~ the same as for Zwaneveld and Verweij (2014a), notable differences are present between the two approaches. The approach

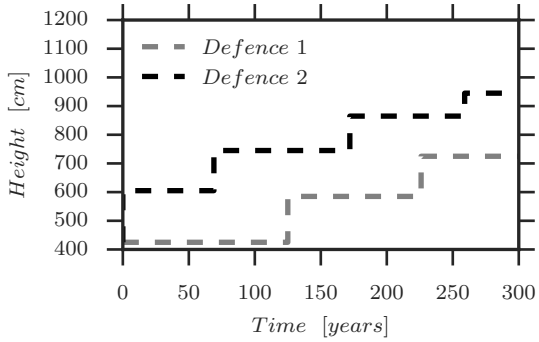

Figure 21. Optimal investment schemes for the case with two dependent lines of defence.

of Zwaneveld and Verweij (2014a) is capable of including both interdependent & independent lines of defence and focused on finding the proven economically optimal solution quickly given pre-calculated AED estimates and investment costs. Our approach focuses on flood defence systems with ~~multiple (interdependent)~~ mostly multiple interdependent lines of defence ~~,~~ ~~while their applications seem to be more focused towards systems with solely (Kind, 2014) or a majority (Zwaneveld and Verweij, 2014a) o~~ ~~independent elements. In~~ (though Section 3.4 ~~, a possible improvement is mentioned which can make our approach more applicable towards systems with independent elements~~does discuss a possible efficient extension to mostly independent lines of defence) and computational costly AED calculations. Therefore, the focus of our approach is on reducing the number of actually executed AED calculations (compared to pre-calculating all possible AED estimates).

An inherent problem of working with flood defence systems where most, if not all, elements are dependent on each other, is that the number of system combinations grows exponentially with the number of lines of defence. The sheer number of combinations means that the total number of lines of defence should probably be kept below ten. This is nothing more than a rule of thumb based on our experience running the best-first graph algorithm on a consumer laptop. The true maximum depends on a number of factors: the number of height options per defence, the performance of the particular implementation of the proposed approach, the computational cost of the associated ~~risk~~ AED functions and the computational power of the used computer.

Even though all examples in this research make use of flood defence heights, this was only done to illustrate the approach. Other measures besides flood defences can be incorporated as well ~~.~~ in the graph that is used to find the optimal solution. While this is not a unique feature of our approach (i.e. any graph-based approach can do this), it is a relevant point for the viability of practical applications. For example, if a retention area is considered (as illustrated in Figure 22), a list with possible sizes of the retention area could also be used in the approach of Figure 10; as long as a measure has a number of options or levels in increasing order that can be quantified and ~~monetized~~monetised, it can be included in the approach. This makes the actual application range much wider than flood defence systems with only height-dependent flood defences such as levees or (storm surge) barriers.

The proposed approach works best if the type of flood defence for each line is known and singular. In the case that a number of different flood defence types are considered at the same line of defence, it would be better to do an ~~optimization~~ optimisation run per system type configuration. An example of this would be the choice between a closure dam or a storm surge barrier at the same location. In this case, the algorithm should be run twice, first with a closure dam and then with a storm surge barrier. This should result in two optimal configurations (one with a closure dam, the other with a storm surge barrier), which can then be compared using some metric, for example their benefit-cost ratios.

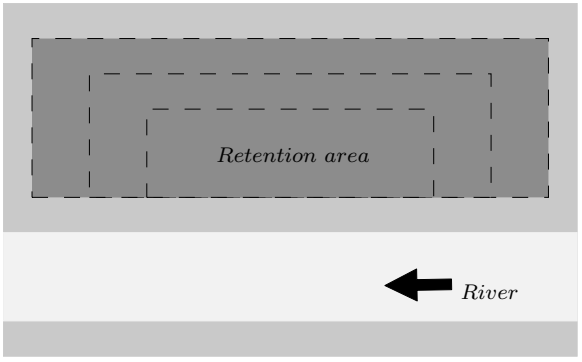

Figure 22. A retention area can also be ~~optimized~~ optimised using the approach proposed in this paper. In this example, the surface area of the retention basin is used instead of the height of a flood defence.

6 Conclusions

This paper presented ~~an easy to use, flexible~~ a generic, computationally efficient approach for finding the economically optimal ~~investment scheme for flood defence systems with multiple lines of defence . We consider this approach to be easy to implement because it uses a~~ configuration of a flood defence system with an arbitrary number of defence lines. The computational efficiency was achieved by delaying AED (flood risk cost) calculations until they are actually needed in an optimisation routine (i.e. 'lazy evaluation'), which leads to a reduction in the number of AED calculations that need to be done. In the few examples shown in this research, the reduction in number of AED calculations was at least 40%. This is a significant and relevant reduction, as the AED calculations relevant for this approach often have a high computational cost. This is especially the case when multiple flood defences interact with each other hydrodynamically in a larger flood defence system.

The proposed approach is implemented using a best-first graph algorithm, which itself is a simple algorithm. ~~It is flexible~~ Furthermore, the approach is flexible towards the number and type of flood defences because the graph representation shown in this paper can trivially accommodate an arbitrary number of ~~lines of defence. Furthermore, the approach proposed does not need to pre-calculate risk estimates prior to running the best-first graph algorithm (which linear programming solutions do). Especially in cases where the risk estimates impose a significant computational cost, this can be an advantageous property.~~

interdependent flood defences. The proposed approach utilizes the repetitive properties of the graphs in order to efficiently ~~compute problems containing roughly up to ten interdependent lines of defence on a consumer laptop~~store the representation of the graph. In case independent flood defences are present in a system, the proposed approach ~~can easily~~ of generating a graph can be adapted to a more efficient method which makes use of the attractive properties of independence. To that end, an improvement in the generation of the graphs has been proposed.

Assuming that the graph and combinations of flood defences are portrayed correctly, the best-first graph algorithm has been proven in literature to return the shortest (or optimal) path in a graph. To ~~address this~~ corroborate this for our implementation and intended application, the method was tested on a number of benchmark problems with known solutions. The tests show that indeed the optimal path is found with the approach proposed in this paper, which ~~indicates that both the generated graphs and the algorithm work as expected~~shows that the implementation was done correctly.

Author contributions. TEXT

Competing interests. TEXT

Disclaimer. TEXT

Acknowledgements. We are grateful for the financial support of the Dutch Technology Foundation STW, which is part of the Netherlands Organization for Scientific Research, and which is partly funded by the Ministry of Economic Affairs. Furthermore, we are grateful to Peter Zwaneveld and Gerard Verweij of the CPB for sharing their ILP model with us.

References

Brekelmans, R., Den Hertog, D., Roos, K., and Eijgenraam, C.: Safe Dike Heights at Minimal Costs: The Nonhomogeneous Case, Operations Research, 60, 1342–1355, doi:10.1287/opre.1110.1028, 2012.

Cormen, T. H.: Introduction to Algorithms, 3rd Edition:, MIT Press, 2009.

Courage, W., Vrouwenvelder, T., van Mierlo, T., and Schweckendiek, T.: System behaviour in flood risk calculations, Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards, 7, 62–76, doi:10.1080/17499518.2013.790732, http://dx.doi.org/10.1080/17499518.2013.790732, 2013.

De Bruijn, K. M., Diermanse, F. L. M., and Beckers, J. V. L.: An advanced method for flood risk analysis in river deltas, applied to societal flood fatality risks in the Netherlands, Natural Hazards and Earth System Sciences Discussions, 2, 1637–1670, doi:10.5194/nhessd-2-1637-2014, 2014.

Dijkstra, E. W.: A Note on Two Probles in Connexion with Graphs, Numerische Mathematik, 1, 269–271, doi:10.1007/BF01386390, 1959.

Dupuits, E., Schweckendiek, T., and Kok, M.: Economic Optimization of Coastal Flood Defense Systems, Reliability Engineering & System Safety, 159, 2016.

Eijgenraam, C.: Optimal safety standards for dike-ring areas, Tech. Rep. 62, CPB, The Hague, 2006.

Eijgenraam, C., Brekelmans, R., den Hertog, D., and Roos, K.: Optimal Strategies for Flood Prevention, Management Science, doi:10.1287/mnsc.2015.2395, http://dx.doi.org/10.1287/mnsc.2015.2395, 2016.

Felner, A.: Position paper: Dijkstra's algorithm versus uniform cost search or a case against dijkstra's algorithm, in: Fourth Annual Symposium on Combinatorial Search, 2011.

Gelperin, D.: On the optimality of A*, Artificial Intelligence, 8, 69–76, 1977.

Hart, P., Nilsson, N., and Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics, 4, 100–107, doi:10.1109/TSSC.1968.300136, 1968.

Kind, J.: Economically efficient flood protection standards for the Netherlands, Journal of Flood Risk Management, 7, 103–117, doi:10.1111/jfr3.12026, http://doi.wiley.com/10.1111/jfr3.12026, 2014.

Klerk, W., Kok, M., de Bruijn, K., Jonkman, S., and van Overloop, P.: Influence of load interdependencies of flood defences on probabilities and risks at the Bovenrijn/IJssel area, The Netherlands, in: Proceeding of the 6th international conference on flood management - ICFM6, 1-13., Brazilian Water Resources Association and Acquacon Consultoria, 2014.

Van Dantzig, D.: Economic Decision Problems for Flood Prevention, Econometrica, 24, 276–287, 1956.

Verweij, G.: Safe dike heights at minimal costs - an integer programming approach, http://edepot.wur.nl/314792, 2014.

Verwer, B. J. H., Verbeek, P. W., and Dekker, S. T.: An Efficient Uniform Cost Algorithm Applied to Distance Transforms, IEEE Transactions on Pattern Analysis and Machine Intelligence, 11, 425–429, 1989.

Vorogushyn, S., Merz, B., Lindenschmidt, K.-E., and Apel, H.: A new methodology for flood hazard assessment considering dike breaches, Water Resources Research, 46, doi:10.1029/2009WR008475, http://dx.doi.org/10.1029/2009WR008475, 2010.

Vorogushyn, S., Lindenschmidt, K.-E., Kreibich, H., Apel, H., and Merz, B.: Analysis of a detention basin impact on dike failure probabilities and flood risk for a channel-dike-floodplain system along the river Elbe, Germany, Journal of Hydrology, 436-437, 120–131, doi:10.1016/j.jhydrol.2012.03.006, http://www.sciencedirect.com/science/article/pii/S0022169412001928, 2012.

Yüceoglu, B.: Branch-and-cut algorithms for graph problems, Ph.D. thesis, Maastricht University, 2015.

Zwaneveld, P. and Verweij, G.: Economisch optimale waterveiligheid in het IJsselmeergebied, Tech. Rep. 10, CPB, The Hague, 2014a.

Zwaneveld, P. J. and Verweij, G.: Safe Dike Heights at Minimal Costs, Tech. rep., CPB, The Hague, 2014b.