Automating Model Builds for Sequence Optimization of Flood Mitigation Investment Phases

Adam S. Erispaha¹ and Christine N. Brown²

¹Sci-Tek Consultants, Philadelphia, Pennsylvania; ²Philadelphia Water Department, Philadelphia, Pennsylvania.

Abstract

A process and a set of tools are proposed to automate the creation of an exhaustive set of flood mitigation scenarios modeled by the United States Environmental Protection Agency's Stormwater Management Model (SWMM), as well as to create a strategic planning framework that takes advantage of the exhaustive model scope. Using the proposed method, models are generated for each logical combination of independent flood mitigation alternatives at each possible phase of implementation (i.e. at every practical phase of investment). Cost estimates and flood risk reduction values are calculated for each flood mitigation alternative and organized into all possible sequences of implementation. The exhaustive set of progressive implementation scenarios is then used as a decision-making tool to provide flexibility in long term planning. Because each phase of investment can be implemented and function independently, large capital improvement programs can be constructed progressively without having to commit (politically and financially) to the cost of full implementation of the program. With the proposed process and the data produced, it is shown that strategic planning can be adaptive and may be adjusted to favour short term cost effectiveness, or optimized within a long term budget while being able to end investment at any phase in the program.

1 Introduction

1.1 SWMM and Strategic Planning Frameworks

The United States Environmental Protection Agency's (USEPA) Stormwater Management Model (SWMM) is a dynamic hydrologic-hydraulic (H&H) model used to simulate water runoff and drainage to assist in the design, analysis and development of urban drainage systems. SWMM applications are useful across various project types including combined sewers, sanitary sewers, storage units, subcatchments and, most recently, low impact development (LID) projects (Rossman 2015). For the purpose of urban flood mitigation, these last types of projects must be evaluated in tandem to allow for an exhaustive and flexible planning strategy. SWMM has laid the foundation for H&H modeling processes; however, with an ever-changing demand for additional modeling capabilities, some limitations of SWMM have become apparent.

Various tools have been developed to increase the functionality of SWMM. Proprietary software packages like PCSWMM and InfoSWMM have enhanced project management, data processing, and visualization capabilities beyond the basic offerings of SWMM. Scenario management, import-export file extension exchange, data auditing tools, better output visualizations (tabular and graphical), geographical information systems (GIS) and geoprocessing tool integration, calibration parameter estimation,

dry weather flow simulation, and integrated 1D/2D modeling are among the additional features provided.

Because SWMM is available as open source software and due to the continuing growth of the online open source community, new methods have been developed in the form of individual packages for specific applications. Tim Cera developed *swmmtoolbox*, a Python based tool used to read SWMM binary output files that allows users to extract time series results from individual SWMM elements. Parameter modification methods have been extended via tools such as MatSWMM and PySWMM, which allow Pythonic access to SWMM data models during simulation runs. MatSWMM works as a co-simulation engine, allowing the modification of attributes such as actuator settings and control rules to improve design optimization capabilities. Similarly, PySWMM was created with a user interface for streaming simulation results, allowing the user to improve control rules during simulation runs and abort model runs when simulation results show poor model performance.

Thus far, individual package improvements have given users the capability to optimize various aspects of drainage system design. For large scale sewer system planning endeavours, such as those in urban flood relief programs, the capability to generate a number of project scenarios in an automated fashion has yet to be developed. The closest approach to satisfying this need is a SWMM5-EA educational tool, which permits users

to utilise genetic algorithm based methods to take in existing SWMM models, optimize any parameter, and create a suite of new SWMM models for all values of that parameter.

The urban drainage models produced by these types of optimization tools can act as the bases for large scale capital improvement programs (CIP). But finding the most efficient implementation sequence in this type of renewal strategy is often challenging during urban drainage planning. As Halfawy notes, commercial off-the-shelf (COTS) software systems with decision support capabilities have been developed for long term asset management planning, particularly RIVA (real time asset valuation analysis) which assists the prioritization of maintenance and rehabilitation by using sewer valuation, condition assessment and remaining service life (RSL) estimation capabilities (Halfawy et al. 2006). However, asset management functions that are fundamental to renewal strategies, such as performance modeling, have limited support in existing proprietary software systems (Halfawy et al. 2006). For asset renewal strategies that focus on flood risk reduction, performance modeling capabilities are vital for selecting implementation strategies that provide the highest level of relief. With the development of new open source Python packages (Python Software Foundation 2017), it has become possible to use performance modeling to find the most efficient, or highest performing, implementation sequence within a CIP.

This study highlights a new set of tools that take performance data from an urban drainage model and create a full suite of implementation scenarios at every logical phase of implementation (or capital investment). Though scenario development is possible with InfoSWMM and PCSWMM, it requires the manual creation of scenarios, the addition, removal or modification of model elements, and model run initiation. From a planning perspective, the ability to automatically create an exhaustive set of scenarios for alternative comparisons greatly enhances an agent's ability to develop a strategic planning framework. It allows users to view model results for independently functioning phases of investment. With the proposed process, strategic planning can be adaptive and may be adjusted to account for the realistic risk of partial solution abandonment.

1.2 Objective

The objective of this study is to demonstrate the practicality of using two Python packages, *swmmio* and *pyplan*, in the development of a flexible strategic planning framework. Initially, a SWMM model was developed to examine a set of infrastructure improvements to initiate a utility's flood risk management program. This model acts as the full implementation scenario (FIS) from which possible partial implementation scenarios may be developed. First, the FIS is broken down into relief sewer networks (RSN) with common drainage points and then further divided into individual sewer segments that represent independently functioning progressive implementation phases (IPs). Next, automation routines are applied using *swmmio*, utilizing parallel processing and model version control techniques, to generate scenario models for every

logical combination of sewer segments. The scenario models lay the groundwork for *pyplan*, which illustrates the cost-benefit dynamics of subprojects and finds the highest performing implementation path, within capital budgetary constraints.

2 Methodology

2.1 Baseline, Full Implementation and Segmented SWMM Model Development

After identifying a study area in which flood risk reduction is sought, a baseline SWMM model that represents the existing conditions must be developed and calibrated. Next, an infrastructure plan must be developed and modeled. The model represents the upper bound flood relief scenario in which infrastructure is sized and aligned to provide the greatest feasible flood risk reduction benefit, regardless of budgetary and other constraints. This scenario constitutes a flood risk reduction program's FIS, with the understanding that the total cost and scope of construction may be prohibitive within the constraints of a sewer utility's budget and sewer construction capacity. The FIS should be considered a state of infrastructure that may only be realized after decades of incremental construction phases. Note that baseline model development, calibration and validation, as well as FIS relief sewer sizing and siting, is not discussed in detail at this point.

Next, the FIS is conceptually divided into hydraulically distinct RSNs in which proposed sewers share a common downstream point (e.g. outfall). Within each RSN, the relief sewers are split into segments that represent independent, progressive construction phases that extend upstream from the furthest downstream point. The length of each segment is identified by determining the logical stopping points of construction: segments should span from the most downstream point to the nearest upstream point along the RSN at which a proposed relief sewer can connect to existing infrastructure and provide incremental flood relief. It is emphasized that each segment, as identified above, should be considered a potential implementation stopping point beyond which no additional infrastructure is required for the segment to function; this is critical for providing planning flexibility and optimization, as will be discussed below.

With a conceptual plan of how the FIS is organized into independent and segmented RSNs, SWMM models are manually developed for each progressive IP along each RSN. Because each segment is hydraulically dependent upon all downstream segments within a given RSN, models are developed for each IP representing the union of each segment and the set of downstream segments:

$$IP_i^j = \bigcup_{i=1}^n S_i^j \quad (1)$$

where:

$$\begin{aligned} IP_i^j &= \text{the } i\text{th IP of RSN } j, \text{ and} \\ S_i^j &= \text{the } i\text{th segment of RSN } j. \end{aligned}$$

For example, assuming an RSN, R , can be segmented into three independent construction phases, three IPs will be modeled: the first phase, IP_1^R , will include (and be equivalent to) the first segment, S_1^R ; the second phase, IP_2^R , will represent the union of the first and second segments, $S_1^R \cup S_2^R$; and the third phase, IP_3^R , will represent the union of the first, second and third segments, $S_1^R \cup S_2^R \cup S_3^R$.

SWMM models are manually developed for each IP by changing the baseline model to incorporate each applicable segment, as modeled in the FIS. Models are organized in a directory structure with a directory established for the baseline model and a sibling directory established for the RSNs. Within the RSN directory, subdirectories are created for each RSN that contain child directories for each associated IP. SWMM input (*inp*) files for each IP are organized within the appropriate IP child directory. For example, for a project including two RSNs, network A and network B, each having two IPs (i.e. A1, A2, B1, B2), the following directory structure is created (Figure 1).

Figure 1 Example of model directory structure for a project with two RSNs, A and B, each with two IPs.

This directory structure is established to organize the project and to facilitate the automation tools, described below. The IP SWMM models that are manually developed constitute parent models which are used as building blocks to generate a suite of child model scenarios for all logical combinations of IPs across each RSN.

2.2 Identifying the Set of Scenarios

When the parent SWMM models have been developed, automation routines are invoked to develop an exhaustive suite of model scenarios that represent each combination of IPs between all RSNs. The set containing all combinations of parent IP scenarios, P , is determined by computing the Cartesian product of the sets of IPs within each RSN:

$$P = A_1 \times A_2 \times A_3 \times \dots \times A_n \quad (2)$$

where:

P = set of all relief sewer scenarios, and

A_n = set of IPs in the n th RSN.

Here, the set A_n includes each phase of implementation along RSN n , including the zeroth phase (i.e. the scenario where

nothing is implemented). For example, two RSNs, A and B, each containing one IP are represented by the following sets:

$$A = \{0, a\} \quad (3)$$

$$B = \{0, b\} \quad (4)$$

where:

a = first and only IP of A, and

b = first and only IP of B.

This yields the following scenarios:

$$A \times B = \{(0,0), (a,0), (0,b), (a,b)\} \quad (5)$$

which is then simplified by removing all the zeroth phases:

$$A \times B = \{a,b, (a,b)\} \quad (6)$$

The cardinality of the sets produced (i.e. the number of scenarios generated) is equal to the product of the cardinalities of all IP sets, including the zeroth phase, across all RSNs. Logically, the set containing only the zeroth phase of each RSN (i.e. the scenario with no implementation along any RSN) is equivalent to the baseline scenario and is not considered in the solution space. Therefore, the total number of scenarios (including the manually developed parent models) is defined by:

$$P = [|A_1| \times |A_2| \times \dots \times |A_n|] - 1 \quad (7)$$

where:

$|P|$ = the number of all relief sewer scenarios, and

$|A_n|$ = the number of IPs in the n th RSN, including the zeroth phase.

For example, 124 total scenarios would be created given three RSNs each having 5 IPs, $([5 \times 5 \times 5] - 1)$. And, given RSNs A and B, three scenarios would be produced, $([2 \times 2] - 1)$, as shown in Equation 6.

2.3 Automating Model Builds

To automatically develop SWMM models for all child scenarios, *swmmio* first produces a patch file for each parent IP model that concisely defines how the IP model differs from the baseline condition. The role of the patch file is conceptually the same as information produced by the *diff* algorithm (Hunt et al. 1976) and similar utilities that provide a list of changes necessary to bring two text files into agreement with each other. In this context, the patch file provides the instructions necessary to recreate a given SWMM model with respect to a baseline SWMM model. SWMM *inp* files have unique characteristics, however, that make the use of existing *diff* utilities less effective; *inp* files for two equivalent models may be represented with sections (and elements therein) arranged in any order. Additionally, modeling workflows typically combine several processes (e.g. working GIS tools, spreadsheet computations, and editing within the SWMM graphical user interface, GUI) that make it difficult to ensure *inp* file contents possess consistent line-by-line formatting.

To address this complication, *swmmio* organizes each element within each section in an *inp* file into two-dimensional, indexed tables (i.e. Dataframe objects as defined within the Pandas package) where each SWMM element is assigned an index equal to its unique name property. Data within each row is organized into columns based on the specific parameters present within each section, with extraneous whitespace and comments being removed. Differences between *inp* files can then be identified by comparing the data element-by-element (using the index of each element as a key), making the comparison agnostic of element order. Similarly, because data is organized into structured tables, differences in whitespace and end-of-line characters (often introduced by various pre-processing techniques) are ignored. This comparison is encapsulated in an *ElementDiff* object that is instantiated with a reference to two SWMM models and a SWMM element section header to identify the section in which to perform the comparison.

To define all differences between two SWMM models, an *ElementDiff* object is instantiated for each section within the models and written to a patch file. Information therein defines how *model2* can be reconstructed given *model1*; for each section, each element that must be added, removed, or altered is defined. Here, altered elements are those that exist within both models whose respective values in each column are not exactly equivalent. The patch files are human-readable and follow the formatting convention used within SWMM *inp* files such that model differences can be readily understood.

To generate a suite of child models, patch files are first created for each IP parent model and then combined, as in Equation 2, into a set of child patch files. Here, the instructions for each element section of each parent patch file are merged so that the child patch file includes the instructions of both parents. Child patch files are then incorporated in the baseline SWMM model yielding a complete child SWMM, reflecting the changes proposed in all parent IP models. For example, a child SWMM model, AB, can be generated by aggregating the patch files for the parent models A and B. The process is shown in Figure 2.

Figure 2 Conceptual illustration showing patch files of parent SWMM models, A and B, aggregated and incorporated in the baseline model to yield child model AB.

2.4 Parallel Processing Model Runs

To efficiently process the suite of models generated, *swmmio* uses the *multiprocessing* module in the Python 2.7 standard library to distribute model runs across the available cores on the user's machine. This is especially useful as the number of child models that are generated grows.

The *start_pool* module within *swmmio* can be invoked as a command line tool to initiate a pool of model runs in parallel on a given machine:

```
$ python -m swmmio --start_pool Model_Directory
```

where *Model_Directory* is a directory containing SWMM *inp* files. The *start_pool* module scans the *Model_Directory* for *inp* files and adds each to a multiprocessing pool to be run on all available computational resources. Optionally, the number of processors to leave unused in the multiprocessing pool can be overridden by setting the *--cores_left* parameter; by default, 4 processors are left unused. The number of models processed concurrently is equal to the number of processors on the host machine minus the value of the *--cores_left* parameter. The *start_pool* module will also accept a list of directories to scan for models to add to the multiprocessing pool.

Each model handled by the *start_pool* module is routed through a three-step hot start sequence to adequately prime sewers within the network with baseflow:

1. The model is run without a hot start file, ignores rainfall, and produces *hotstart1*;
2. The model is run with *hotstart1*, ignores rainfall, and produces *hotstart2*; and
3. The model is run with *hotstart2* and includes rainfall.

A log file is generated in the working directory to document the progress of the pool of models being processed. For each model processed within the pool, SWMM report files (*rpt*) are saved within the directory containing each *inp* file. When a model has completed the hot start sequence, it will exit the multiprocessing pool and be replaced with an unprocessed model until all models found within the *Model_Directory* are complete.

2.5 Post-Processing Automation

Because the *rpt* files produced by SWMM are not easily imported into other programs for processing (Rajan et al. 2012), it is especially challenging to manage the data for a multitude of scenarios. To effectively make use of the breadth of data produced by a suite of SWMM models, program access is provided (via *swmmio*) to relevant data in companion *inp* and *rpt* files for a given SWMM model.

Several high level Python objects are created that encapsulate and organize data relevant for flood risk reduction optimization. At the core, a *Model* object that associates data element-wise between the *inp* and *rpt* files from a SWMM model is used. Here, relevant data related to each SWMM node is organized into a unified table (i.e. Pandas *dataframe*), pulling information from the *JUNCTIONS*, *OUTFALLS*, *STORAGE* and *COORDINATES* tables in the

inp file and from the Node Depth Summary and Node Flooding Summary tables in the *rpt* file. Similar aggregations are provided for conduit and subcatchment elements in SWMM models. Figure 3 outlines which tables are extracted from *rpt* and *inp* files and then organized into dataframes within the *Model* object. This aggregated data is then programmatically accessible and can be easily output as comma separated values (csv) files, shapefiles, and GeoJSON files for manual processing with spreadsheet and GIS tools.

Figure 3 Data extracted from SWMM *inp* and *rpt* files is organized into three unified tables within the *swmmio.Model* object.

Building on this foundation, several additional objects are created and methods are established to automate higher level post-processing tasks. A *damage* module is provided that relates flood severity (reported in the Node Flooding Summary *rpt* table), via the *Model* object, to parcels in a study area. Further, data processed by the *damage* module can be managed within reporting and scenario comparison modules, which provide a means of evaluating the performance of competing infrastructure scenarios. Detailed performance metrics, which include estimates of scenario cost, overall flood risk reduction, risk transference and spatial distribution of risk throughout a study area, are calculated automatically. By default, performance metric calculations are suitable for a CIP in Philadelphia, but may be adjusted for other conditions.

The total cost of a scenario is determined by applying unit cost estimates to the proposed infrastructure. Here, all proposed infrastructure is identified by instantiating an *ElementDiff* object with the Conduits sections of the baseline and proposed scenario SWMM models. Proposed conduits are categorized by cross section and unit costs are applied to the proposed length of each category by Equation 8.

$$C_j = \sum_{i=1}^n L_i^j U_i \quad (8)$$

where:

C_j = the total cost of scenario j ,

L_i^j = the total length of proposed infrastructure of cross section category i in scenario j , and

U_i = unit cost (\$USD/ft) of cross section category i .

By default, unit costs represent 2016 planning level costs as a function of sewer diameter if the sewer cross section is circular and the diameter is ≤ 7.5 ft (2.29 m), as illustrated in Figure 4. Per length unit cost of relief sewers with non-circular cross sections or with diameter > 7.5 ft (2.29 m) is computed by assuming a cost of \$80/ft (\$860/m) per unit area (ft², m²) of cross section. Unit costs can easily be adjusted.

Figure 4 Cost per linear foot for sewers of various sizes (2016 USD).

In addition to providing quantitative data from automated post-processing routines, several graphical methods are included. Because it is often a challenge to maneuver data between spreadsheets and GIS tools, methods are provided to create visualizations directly from the *Model* object as portable network graphics (PNG) files, as in Figure 5, Figure 7, Figure 8, and Figure 9. More sophisticated interactive visualizations are generated by exporting GeoJSON data into web mapping applications, such as Mapbox and Leaflet; examples are shown in Figures 6 and 7.

Figure 5 Visualization showing the relative magnitude of node flooding throughout a sewer network, where node radius is proportional to flood duration.

Figure 6 Visualization showing flood risk as three-dimensional extrusions above each parcel in a study area.

Figure 7 Visualization of flood risk distribution among parcels in a study area.

Figure 8 The nodes and conduits network within the South Philadelphia baseline SWMM model, highlighting the Pennsport study area; the region is bounded by the Schuylkill River (west) and the Delaware River (east).

Figure 9 An inset view of the Pennsport study area, with the FIS (in orange) split between three independent RSNs.

2.6 Automation Case Study: South Philadelphia CIP

This section describes how the proposed process is applied to the Pennsport Storm Flood Relief CIP, located in the southeast of Philadelphia, Pennsylvania as shown in Figures 8 and 9 above.

Currently, three RSNs are proposed within the Pennsport study area; they are split into 16 IPs (including zeroth phases), outlined in Table 1. Given the set of parent IPs across each RSN, the breadth of child scenarios is determined (as described above) for a total of 143 unique scenarios (including the parent models). The child models are generated automatically by creating and applying each patch file combination to the baseline model.

Table 1 Potential infrastructure phases proposed for the Pennsport study area.

Relief Sewer Network	Number of Implementation Phases	Total Length of Relief Sewer Proposed
M	4	1.10 mi (1.77 km)
R	6	1.93 mi (3.12 km)
W	6	1.35 mi (2.17 km)

The efficacy of each infrastructure scenario is estimated by quantifying the reduction of risk provided with respect to the baseline condition. It should be emphasized that the metric used to define and quantify flood risk is highly dependent upon the characteristics of the flooding issue and the design goals of the flood mitigation program. Flood risk may, for example, be estimated by quantifying expected damage costs by a comparison of the predicted flood elevations that result from a probabilistically defined design event with flood damage curves. If certain assets are deemed more critical than others, flood risk may more appropriately be quantified by spatially intersecting the predicted extents and magnitudes of flooding with categorized assets.

In this study, the prescribed design goal is to reduce the risk of basement flooding caused by sewer backups in the mainly residential urban area served by a combined sewer system. For a given parcel, the basement flood risk reduction provided by an infrastructure improvement is, therefore, conceptualized as the

difference in risk between the baseline and the proposed infrastructure scenarios:

$$\Delta R_i^j = [p(L)_i^0 - p(L)_i^j] L_i \quad (9)$$

where:

- ΔR_i^j = risk reduction of parcel i , given infrastructure scenario j ,
- $p(L)_i^j$ = probability of loss for parcel i , given infrastructure scenario j ,
- $p(L)_i^0$ = probability of loss for parcel i , given the baseline infrastructure scenario, and
- L_i = potential loss (i.e. damage) for parcel i resulting from basement flooding.

For simplicity, the value of each parcel (i.e. the potential loss due to basement flooding), L_i , is assumed to be equivalent throughout the study area and the probability of loss, $p(L)_i^j$, for parcel i is assumed to be proportional to the duration of flooding measured at an adjacent node, within the design scenario outlined in Table 2. The magnitude of L_i , therefore, irrelevant within the context of scenario optimization because the efficacy of each competing scenario would be scaled equally by a factor of L_i .

Table 2 Design scenario parameters used to assess basement flood risk.

Scenario Parameter	Description
Storm Event	10 y return period, Type II 24 h alternating block design storm
Outfall Boundary Condition	Mean high-high tide fixed water elevation

Similarly, node flood duration is used directly as a surrogate for flooding probability because its assumed probability of proportionality to flooding is sufficient within the context of scenario optimization. If, for example, the absolute likelihood of flooding at a given node is related to node flood duration, D , by a factor α , then the proportionality of risk present in competing scenarios j and k is independent of α because:

$$\frac{\alpha D_j}{\alpha D_k} = \frac{D_j}{D_k} \quad (10)$$

Regardless of how the factor α adjusts the node flooding durations to estimates of probability, the relative difference in risk between scenarios j and k is unchanged, resulting in the same optimization outcome.

At each node, basement flooding is simulated in the model when the hydraulic grade line (HGL) rises above the assumed elevation of adjacent basements. Nodes in the model are assigned values of maximum depth, d_{max} , equal to the difference (measured in ft) between their rim and invert elevations, less the assumed basement depth below ground surface, 5 ft (1.52 m):

$$d_{max} = (El_{rim} - El_{inv}) - 5 \quad (11)$$

where:

- d_{max} = maximum depth in node before SWMM reports flooding,
- El_{rim} = node rim elevation (i.e. ground surface elevation), and
- El_{inv} = node rim invert elevation.

Because the study area is flat, flooding is assumed to occur throughout the entire footprint of any subcatchment tributary to a flooding node (White et al. 2013). Parcels are assumed to experience basement flooding when the subcatchments are flooded. Therefore, the overall benefit (i.e. reduction in risk) provided by a given infrastructure scenario is estimated as the sum of the products of flood duration deltas for each node and the number parcels that are wholly or partially located in the subcatchment tributary to the node:

$$\Delta R_j = \sum_{i=1}^n [(D_{i0} - D_{ij}) p] \quad (12)$$

$$\Delta R_j = \sum_{i=1}^n \Delta D_j p \quad (13)$$

where:

- ΔR_j = risk reduction provided by scenario j (parcel-hours),
- D_{ij} = duration of flooding measured at node i , within infrastructure scenario j (h),
- D_{i0} = duration of flooding measured at node i , given the baseline infrastructure (h),
- ΔD_j = flood duration delta measured at node i , between the baseline scenario and scenario j (h), and
- p = the number of parcels located wholly or partially in the subcatchment tributary to node i (parcels).

When a parcel is located in more than one subcatchment, flood risk is conservatively based on the maximum flood duration measured at the associated nodes. Figure 10 illustrates how node flood duration is applied to parcels that spatially occupy the tributary subcatchments. In the schematic, the flood duration assigned to parcels a and c is equal to the flood duration measured at node 1 and node 2 because they respectively occupy catchments 1 and 2, tributary to each node. The flood duration in parcel b, 2 h, is taken as the maximum duration observed in catchments 1 and 2, because it occupies both areas. The total risk for this system is 5 parcel-hours.

Figure 10 Flood duration assignment for parcels occupying tributary subcatchments.

When results have been returned by SWMM for each scenario, the suite of models is batch post-processed using the reporting and comparison objects described above. In addition to generating visualizations, spreadsheets, and spatial datasets reflecting the results of each model, a summary table is generated that aggregates the cost estimate and performance (i.e. basement flood risk reduction) associated with each scenario.

2.7 Optimization of Implementation Phases

With performance data and cost estimates for each scenario abstracted into a key table, optimization of the proposed infrastructure scenario is possible by selecting whichever scenario provides the most benefit (e.g. basement flood risk reduction) per unit cost. However, because each scenario represents a combination of sequential phases of construction, optimization strategy must be applied to the sequence of construction.

Here, the optimal sequence of construction is determined by leveraging the process provided in the Python package *pyplan*, as follows:

1. First, identify which scenario has the most cost effective first IP of each RSN, where cost effectiveness is defined as the ratio of benefit provided (e.g. flood risk reduction) to capital cost; this IP should be selected as the first phase of the overall CIP.
2. Identify which RSN has the most cost effective subsequent phase, inclusive of the previous IP(s).
3. Repeat step 2 until each RSN is fully implemented.

This process yields an implementation sequence, starting with the baseline scenario to the FIS, in which each phase is the most cost effective investment strategy, given the previous implementation state. The process is shown in Figure 11.

Figure 11 Performance metrics for each scenario (blue circles) are compared and organized in a sequence (orange line) that maximizes benefits (e.g. risk reduction) provided at each implementation step.

3 Results and Discussion

3.1 Sequence Optimization Decision Support System

Developing an optimal flood relief implementation sequence, as discussed above, serves as a valuable decision making tool. By ensuring that the range of scenarios modeled and post-processed is exhaustive (given the set of IPs), final CIP decisions can be made more confidently.

Other interesting and useful implications arise from this approach to sequence optimization. For instance, certain decision points along the implementation sequence may, at times, be better served by selecting sub-optimal IPs if it is understood that a sufficient budget will be available for more efficient phases in the future. This presents a situation, for example, where a utility may choose to invest in relief sewers that offer less flood mitigation per dollar in the early phases, understanding that it will be better positioned for greater overall CIP efficiency in later phases. This indirectly points to another key strength gained by using this process as a decision support system: planning and budgetary flexibility is provided by allowing a utility to implement the components of the FIS sequentially, without having to fully commit to anything beyond one IP at a time, while maintaining confidence that each incremental investment is as cost effective as possible. However, greater long term (and overall) cost effectiveness is possible when utilities have a clear understanding of long term capital available for a given study area. Additionally, because the breadth of modeled scenarios is exhaustive, qualitative issues can be considered and adapted to. For example, a semi-optimal implementation sequence can be identified that avoids a certain implementation scenario that is determined to be politically infeasible.

A notable limitation of the proposed method results from having to ensure each IP is sized as required within the FIS. This causes proposed infrastructure within each RSN to be oversized to some degree unless the RSN is implemented fully.

3.2 Model Version Control

An additional benefit is provided by the proposed process for automatically building model scenarios by preserving patch files. Like other version control paradigms, patch files can be considered as snapshots of model states (i.e. versions). Model versions can be preserved in low footprint patch files that can readily be used to return a model to a previous state.

4 Conclusions

Flood mitigation infrastructure projects are inherently large, disruptive, and costly endeavours. To ensure that the flood risk reduction planning process respects the budgetary, political, and community impacts of major infrastructure projects, implementation should be as cost effective as possible. To support this goal, a process for rapidly developing and processing a suite of model

scenarios is provided within the Python packages *swmmio* and *pyplan*. It has been shown that these tools provide scalability for quantitatively evaluating many flood mitigation infrastructure scenarios such that implementation sequencing can be optimized while providing the flexibility to adapt to capital, political and social constraints.

Acknowledgments

We would like to recognize both the open source community, by highlighting the contributions of *pyswmm* (Open Water Analytics), *swmmtoolbox*, and *MatSWMM* to the extension of USEPA SWMM, and the H&H modelling collective, as previously referenced in this article.

References

- Halfawy M. M. R., L. A. Newton and D. J. Vanier. 2006. "Review of Commercial Municipal Infrastructure Asset Management Systems." *Journal of Information Technology in Construction* 11:211–24.
<http://www.itcon.org/2006/16>
- Hunt, J. W. and D. M. McIlroy. 1976. "An Algorithm for Different File Comparison." In *Computing Science Technical Report*. Murray Hill, NJ: Bell Laboratories.
- Rajan, R., K. Artita, J. Marge and J. Smullen. 2012. "Reformatting and Post Processing Tools for SWMM5." *Journal of Water Management Modeling* R245-19.
doi: 10.14796/JWMM.R245-19
- Rossman, L. A. 2015. *Storm Water Management Model User's Manual Version 5.1*. Cincinnati, OH: U.S. Environmental Protection Agency. EPA/600/R-14/413b.
- Python Software Foundation. 2017. *Python 2.7.13 Documentation*.
<https://docs.python.org/2.7>
- White, E., J. Knighton, G. Martens, M. Plourde and R. Rajan. 2013. "Geoprocessing Tools for Surface and Basement Flooding Analysis in SWMM." *Journal of Water Management Modeling* R246-03.
doi: 10.14796/JWMM.R246-03