Extending the Capabilities of EPANET User Interface with the Optimal Design Plugin Tool

Ehsan Madadi Kandjani¹, Lina Sela∗², Kasey Faust², Fernanda Leite², and Ben R. Hodges¹,²

¹National Center for Infrastructure Modeling and Management, The University of Texas at Austin, TX 78758
²Department of Civil, Architectural and Environmental Engineering, The University of Texas at Austin, TX 78712
*linasela@utexas.edu

ABSTRACT

This work describes the development of a plugin in the new SWMM-EPANET User Interface (UI) environment. The plugin package solves the well-studied least-cost design problem of water distribution systems, implementing genetic algorithms as the optimization approach. Simulation of water distribution systems is crucial for the technical planning and management of municipal water systems, and as such, EPANET code, and its computational engine have been extensively used in a range of design, operation, and management problems that require hydraulic and water quality simulation of pressurized pipe networks. Multiple demand conditions, planning scenarios, and various computational methods for integrating with external data sources and programming tools are common challenges to water community that are not directly addressed in the current EPANET software environment but have been (at least partially) addressed with proprietary open source and commercial software, such as Python, MATLAB, and R. Such tools and data are typically not shared with the broader community or require significant additional programming efforts to apply for a particular case. The EPA has recently focused on developing an open source modular and extensible UI, which allows plugin and scripting support, such that new applications can be integrated and shared within the EPANET environment. In this work, a new menu was created in the UI for the user to define the properties of the least-cost design problem. The developed plugin accesses the EPANET functions through the Python-based toolkit, as well as custom optimization algorithm. The paper shows the code structure and the basic functionalities of the plugin, such as user interface, communication with the EPANET solver, and the solution approach.

Keywords: EPANET, Plugin Tools, Genetic Algorithm

1 Introduction

EPANET—developed by the U.S. Environmental Protection Agency (EPA) in the early 90’—is a computerized simulation model that performs extended period simulation of hydraulic and water quality behavior within pressurized pipe networks. This model is intended to improve our understanding of flows and pressures of water and the movement and fate of drinking water constituents within water distribution networks (WDNs) [1]. Since its original release, EPANET has been used across a multitude of applications to explore diverse problems in WDN analysis, such as hydraulic model calibration [2, 3], system design and expansion [4, 5], chlorine residual analysis [6], and consumer exposure assessment [7]. In addition to a sizeable user community, the developer community
has continuously improved and extended the model capabilities. The current EPANET version allows the developers to use EPANET functionalities through the dynamic link library (DLL) — i.e. the Programmer’s Toolkit — that includes a library of functions allowing editing network data and accessing results without the user interface. The Programmer’s Toolkit also, currently, allows the execution of the hydraulic and water quality simulations from an external application for C/C++, Delphi, Pascal, and Visual Basic. In addition, numerous programming wrappers have been developed — e.g. Matlab \[8\], Python \[9\], and R \[10\] — that utilize the EPANET simulation engine.

Although EPANET is universally accepted as the reference model in WDN analysis \[11\], as community needs are evolving due to computing technological advancement, EPANET requires new editing and data processing capabilities. In recognition of these needs, the EPA has initiated the development of a UI \[12\] to re-engineer and design a modular and extensible architecture for both the EPANET \[13\] — of interest to this paper — and SWMM (a hydrology-hydraulic storm water simulation model) \[14\] software. The major functionalities of the UI is to provide modular software design, scripting, and plugin support using an open platform that employs open sources software. This, in turn, will enable the deployment of new application features created by the EPA, third-party developers, and end-users employing scripts and application “plugins” \[15\]. UI development is done using Python language due to its increasing use by scientists and software developers \[16\], support on Linux, MacOS X, and Windows, and a large pool of open source third party libraries for optimization, data processing, visualization, etc. \[17\].

As the UI version has not been formally released to the user community, the National Center for Infrastructure Modeling and Management (NCIMM) — led by the University of Texas at Austin and initiated primarily by the EPA in 2016 \[18\] — is currently coordinating the UI development and testing, as well as developing a range of functionalities. This paper highlights selected efforts by NCIMM, focusing primarily on the development and testing of the “plugin” functionality of the UI. Specifically, we present a plugin that shows a created code structure within the new SWMM-EPANET user interface (UI) in which advanced algorithms could be easily integrated and be accessible to the user and the developer. We demonstrate plugin development through the well-known WDN least-cost design problem.

Optimal WDN design is necessary for the planning and management of municipal water systems to ensure that a desired level of service is met under varying conditions. The least-cost design problem can be stated as finding the size of the system components, which minimize the capital cost and maintain the hydraulic and engineering constraints. The process of designing WDNs — whether new or adapting existing systems via upgrading, downsizing or extensions — is complex due to considerations of the many network components, unique system topology characteristics, and network hydraulics. Over the last four decades, numerous models for least-cost design of WDSs have been published in the research literature ranging from early linear, non-linear, and dynamic programming to the more recent evolutionary/meta-heuristic schemes such as genetic algorithms (GAs) or ant colony (AC) \[19, 20, 21\]. The former methods are typically limited by problem representation (e.g. continuous diameters). The latter, on the other hand, integrate simulation models to assess the quality of solutions, use penalty functions to handle violation of constraints, and require a large number of simulations. In this work, we demonstrate the development of the plugin through the implementation of the WDN least-cost design problem using genetic algorithm (GA) optimization \[19\]. In the next sections, we briefly describe the general UI architecture, the plugin development framework, and test-case application. To avoid confusion, throughout this paper ‘UI’ refers to the new SWMM-EPANET
2 Methods

2.1 Structure of the SWMM-EPANET User Interface

The UI is developed under PyQt [22], which consists of python bindings for the Qt GUI visual elements and handles user interactions with the visual elements such as GIS integration [12]. The UI (written in Python) is designed to closely match the design of the current user interface (written in Delphi) to preserve familiarity of existing users. Fig. 1 shows the new (left) and current (right) UI. Three additional capabilities include map display, scripting, and plugins. Plugins are managed and controlled via a menu item on the main form, where installed plugins are listed in the dropdown menu [12].

![Figure 1: New (left) and current (right) UIs](image)

The structure of the UI, DLL, and plugins in the UI is schematically shown in Fig. 2. The main structure includes documentation, test, and source files. The documentation (doc) folder contains the Doxygen and developers notes; the test (test) folder contains simple functionality test of the UI. The Source (src) folder contains four subfolders: (1) externals contains DLL, which are required for any hydraulic calculations, (2) core which handles the reading and writing of data through .inp and .rpt files, (3) ui that contains the UI visual layout, and (4) plugin folder, which contains all the custom source codes and packages.

2.2 Structure of the Plugin & EPANET & UI

All plugins share a common set of management options that prescribe the core program how to communicate and control the plugin functionalities upon execution. The plugins are organized in file folders and are placed in the plugin folder, where each plugin is contained in its own folder (Fig. 2). Each plugin can have unlimited level of packages or subdirectories of its own that can access main program information via class object sharing at runtime. Fig. 3 presents the main modules of the plugin and its communication with EPANET core and UI. The activation of the plugin requires communication with the UI, DLL, model input data (.inp), output data (.rpt), as well as any add-on libraries and user
functions. Reading the input file is carried out by the core UI and memory is allocated to store the model data. Plugin inherits the model data from the UI and communicates with the DLL and add-on libraries to run the user-developed algorithm. Once the algorithm is executed, post processing takes place in the plugin by writing the report and passing it back into the core UI, closing the libraries, and freeing the memory allocation.

2.3 Test Case

We demonstrate the development of the plugin through the implementation of the well-studied least-cost design problem of water distribution systems using the genetic algorithms (GA) as an optimization method [19, 20]. The problem aims to select the least-cost diameters for the network pipes, such that constraints at the consumer nodes are satisfied and hydraulic laws are maintained. Decision variables are discrete pipe diameters, constraints include minimum nodal heads, the hydraulic laws include linear mass balance and nonlinear energy conservation equations, and the goal is to minimize a nonlinear objective function [19]. Since the resulting optimization problem is nonlinear with discrete variables, many solution approaches relying on evolutionary algorithms have been proposed in the past [20]. Here, we implement the solution approach suggested by Savic and Waters [19], which couples the EPANET simulation engine with GA optimization. GA is a subset of evolutionary algorithms that can be applied to solve nonlinear discrete optimization problems. The GA optimization process initializes by generating an initial random ensemble (population) of solutions and iterates via two main steps: (1) estimate the quality of the current solutions and (2) generate new solutions that result in an improved performance. This two-step process continues until no performance improvement can be achieved or the simulation reaches the maximum number of objective function evaluations. In
each iteration, selection, mutation, and crossover processes govern how new solutions are generated. These processes have several parameters that affect the performance of the GA algorithm and, in turn, the quality of the solution. Each candidate solution — i.e. a set of potential pipe diameters — is evaluated based on the feasibility of the hydraulic constraints and the corresponding cost of the selected diameters. The hydraulic feasibility is estimated based on EPANET simulation using the DLL, which are integrated at the UI and accessible through the plugin environment. In the event that the hydraulic constraints are violated, a penalty cost is added to the cost of the selected diameters to avoid promoting solutions that are not hydraulically feasible.

The least-cost design problem formulation requires several inputs from the user including network characteristics, available pipe diameters and their corresponding costs, as well as GA parameters. Network characteristics (e.g. layout, elevations, and demands) are read from the .inp file. To set up the design and optimization data, two new menus (Fig. 4) were developed as part of the plugin to read the information regarding the pipe diameters, the corresponding costs, and the GA parameters (e.g.
population size, crossover method). Alternatively, these inputs can be directly read from the revised .inp file.

To communicate between the core UI and the plugin environment, we need to initialize access to the stored data. To initialize the EPANET inside the plugin we use the structure as follows:

```python
MyNet = pyepanet.ENepanet(inp_filename, session.status_file_name,
                          session.output_filename, session.model_path)
MyNet.ENopen(inp_filename, session.status_file_name,
             session.output_filename)
```

Once the data is linked through the plugin environment, the input data related for the optimal design is read from the UI. The hydraulic analysis are initialized and ran through a template class, as follows:

```python
frmRun = frmRunGA(MyNet, session.project, session)
session._forms.append(frmRun)
frmRun.Execute()
```

Once the hydraulic analysis is carried out by the DLL, the nodal head can be retrieved to estimate the hydraulic feasibility of the solution and compute the penalty cost:

```python
nodePressure = []
for j in range(0, numNodes):
    ID = j + 1
    pressureI = MyNet.ENgetnodevalue(ID, toolkit.EN_PRESSURE)
    nodePressure.append(pressureI)
```

We tested the developed plugin using the simple two-looped network that has seven nodes and eight pipes [23] (Fig. 1 left; full data can be found in [23]). A penalty cost is assigned to the network using a penalty factor of $50,000. Assumptions include a population size of 50, 90% cross-over rate, and a 3% mutation rate [19]. We achieve the optimal best solution reported in the literature (Fig. 5).

![Figure 5: Evolution of network cost for two-loop network over the iteration time](image-url)
3 Conclusions

The paper described the implementation of a plugin for analysis of a least-cost design problem in water distribution networks. The algorithm is integrated into the recently developed, new SWMM-EPANET user interface and is distributed using the executable file format to load its capability on Microsoft Windows operating system. The simple GA algorithm is implemented as a first step to validate the structure used to design the plugin integrated with the UI. The proposed plugin structure enables developers to port and integrate applications into the UI. Standard EPANET libraries are used for simulations while the visualization and data handling capabilities of the UI are being leveraged for plugin extension. Advanced users and developers are able to install the developer’s version of the software to extend the plugin capabilities or implement the new plugin under the proposed structure. Our future work will address the structure design, which will provide more flexibility in terms of implementing more complex plugins. In summary, the new UI will encourage users to share solutions with the broader community through the plugin mechanism. The development and maintenance of the source code will happen openly, and the source code will be distributed via GitHub for continuous integration and pull requests from developers. User interaction will happen via a discussion board and feedback will be collected using an issue tracker to facilitate interaction and maximize the impact of the project.

Acknowledgments

This publication was developed under Cooperative Agreement No. 83595001 awarded by the U.S. Environmental Protection Agency to The University of Texas at Austin. It has not been formally reviewed by EPA. The views expressed in this document are solely those of authors and do not necessarily reflect those of the Agency. EPA does not endorse any products or commercial services mentioned in this publication.

References


