INL/EXT-16-39378

Development of The PaR Tool: Modernization of the Planning and Routing Scripts

Jacob P. Lehmer, Jake P. Gentle,
Timothy R. McJunkin

August 2016

Idaho National Laboratory

INL/EXT-16-39378

Development of The PaR Tool: Modernization of the Planning and Routing Scripts

Jacob P. Lehmer, Jake P. Gentle, Timothy R. McJunkin

August 2016

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Assistant Secretary for ____, OR Office of _____
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Development of The PaR Tool: Modernization of the Planning and Routing Scripts

Jacob P. Lehmer, Jake P. Gentle, and Timothy R. McJunkin

Idaho National Laboratory, Idaho Falls, Idaho 83415, USA

The placement of an efficient route for power lines is a poorly understood. Owners of the land are often divided about whether to allow their land to have a pole placed. A planning and routing toolkit allows for the people to allow and deny access to land then instantly see the route recalculate. The original planning and routing scripts were the original proof of concept, there was a series of problems that greatly hindered the functionality of the system. The PaR Tool a fully featured application replaced the proof of concept scripts and the speed of the system was increased by 254%. It is readily extensible and far easier to use.

I. INTRODUCTION

The placement of an efficient route for power lines is a poorly understood, vital feature to the nation's infrastructure. When a new route needs to be placed there is a gap in the understanding of the general public and the team who is building the project. This lack of understanding creates a few particular problems facing placing new powerlines that can be alleviated by allowing the general public access to a tool that will give a simplified version new powerline route.

A. Motivation of a planning and routing toolkit

There are two main motivations for the development of a computerized planning toolkit. The first is public opinion and the second is the physical route that optimizes line length, construction cost, and concurrent wind cooling. The combination of these two requirements creates a complex set of problems.

When a new line needs to be placed there is an expensive and long process of determining the route for the powerline. A necessity of this process is contracting the land for the lines. Owners of the land are often divided about whether to allow their land to have a pole placed on it or to deny placement entirely. There is a lack of contiguous information available for the landowner to be able to see the different options available for the placement of the lines.

A survey lacks the needed depth in order to give an accurate depiction of the problems at hand. When a particular landholder denies access the entire route needs to be recalculated in order to find the most optimized route. This cannot happen with a single survey. A planning and routing toolkit allows for the people to allow and deny access to land then instantly see the route recalculate.

B. Scope and Purpose of Desired Tool

The tool is not meant to replace the system that is in place by the utility companies use to place routes. The tool is meant as a general public education device and information relay system. It will serve the same purpose as the PVMapper (1) tool, except instead of placement of a solar farm, this will place a powerline.

C. Usage of Tool

The tool is intended to be used through an application form that is able to serve people as easily as possible. This can take the form of a web based application or a downloaded application that has a connection to a database. There is a very definitive goal to make this application as user friendly as possible.

II. ORIGINAL PLANNING AND ROUTING SCRIPTS

The original planning and routing scripts were the original proof of concept. These scripts are the first attempts to make a system that is capable of reading terrain files and conducting wind analysis then forming a path through the land. This system is a series of four python scripts that must be executed in a particular order. Once these scripts are all ran there is an output folder that has a series of lines comprising the new powerline. This route can then be imported into a variety of different mediums.

A. Necessity of Replacement

Although the scripts did run, and did create an optimized path through the environment, there was a series of problems that greatly hindered the functionality of the system. These problems came directly from the complex interaction between the different scripts. There was also programmatic errors that hindered functionality.

1. Usage Difficulty

The usage problems exist in each of the different stages. The first of the usability problems is the configuration file. The configuration file contains twenty seven different fields that have to be entered precisely or else the system will crash without any error message, or any time to correct the issue. There is also cryptic information about how these different fields need to be entered and formatted.

The next script analyzes the terrain. The needed geometric information files need to be placed in the proper folders for this script to work properly, each one of these files need to then be listed in the configuration file. The wind analysis happens next. This script needs to have the terrain analysis script ran once and only once. There is no information that lets the user know that the system has failed or succeeded. The final script that needs to be ran implements a sophisticated path finding algorithm and thusly finds the path through the terrain.

The overarching issue that plagues the system is how the scripts handle the input and output internally. Firstly, the configuration file is re-read in every script. The scripts need to be ran individually and this means that creates a possibility that the configuration file could be changed. Also, the result files from each of the different scripts are dumped into the same place. This creates a confusing mess in the results folder. Most importantly the final result from the entire system is also placed in the results folder.

2. Speed of Operation

Python is a fantastic prototyping language for fast development or occasional use. This application is neither. The python scripts run very slowly due to the nature of a scripted language. This is a serious concern for the original purpose of the planning tool. The speed problems would case the users to lose interest in the tool.

3. Extensibility

Most importantly, there is much more functionality that needs to be added to a planning and routing tool. Functionality such as a map viewer, more layers on terrain analysis, and localized weather data needs to be added in order to give users a tool that allows for a useful depiction of powerline placement. It is not impossible to add these features directly to the python scripts. However, adding this functionality to the current scripts will compound and exacerbate the inherent problems.

III. THE PaR TOOL

The PaR Tool (Planning and Routing) Tool is a literal translation of the python scripts, built into a fully featured application written in Java. Literal translation meaning the java code was written from the python code. That is, the functionality of the Java application is derived from the python code instead of the design goals of the python scripts. This tool incorporates all four of the python scripts into one application. The development goals of the GUI is ease of use for a new person.

A. Development

The initial development of the PaR Tool is based on exactly mimicking the functionality of the python scripts. This is done in order to be absolutely sure that the resultant tool has the same features that existed in the python scripts. This resulted in interesting translational problems, and through the process subtle flaws were discovered with the python scripts that could possibly invalidate the resultant data.

1. Translation

The first problem that came up in the translation of the code is that neither Java nor Python has an in built library to deal with matrices. The library that was used in python to deal with matrices is NUMPY and the library used to deal with matrices in Java is CERN’s COLT matrix library. Both of these different libraries have the same functionality and fining the appropriate mappings in order to mimic operation in the code proved to be trivial.

The next problem was more concerning. Once again, neither Java nor python are able to read, manage, or change Geographic Information System (GIS) shapefiles. A shapefile is a particular type of file used by GIS that defines a region on a map. Python uses a lean library known as PYSHP to read these shapefiles. This library conforms to the verbose and easy to use mentality that python has declared to be its namesake. The verbosity makes reading the code quite straightforward. Java has many different possible libraries that could be used to manage the shapefiles. The library that was chosen was the openGIS geotools package.

These two libraries have similar functionality, however, the nomenclature in both of the libraries is slightly different. Once such instance is the definition of bounding box and envelope. In the Python library a bounding box is an envelope in the java library. This problem surfaced several times while translating the code.

With cautious translating the differences in the libraries did not prove to be a long term issue. Other problems in the code came from coding techniques in python that caused the data types of variables to not be known. Fortunately, these problems were fleeting.

2. Verification

Due to the differences in libraries and languages, the utmost care had to be taken in verifying that the python scripts and PaR tool had the same functionality. I used a two-step process to determine this identical functionality. The first step is in the actual translational stage, I would run the python debugger and the Java debugger at the same time for each of the methods that I was translating. Through checking that the values that were passed to the method signature in both languages were identical and the values returned were identical, it was determined that the internal workings of the python system was maintained in the Java application.

The second step was checking the output files that were dumped from each of the stages in the python script. These files were usually thousands of lines long, so in order to expedite the verification process I would check the first entry, last entry, and a random sampling of internal entries in the files dumped by the python script and the Java application. It may have been possible that some of the other entries contained in the file may have been incorrect. However, I am confident of accuracy due to the first stage of verification and the obvious detectable propagation of errors as the files are used in the other scripts.

B. Solved Problems

The translation process was able to resolve many of the problems that existed in the original python scripts. The Java application was able to remove the complex interaction by completely shielding the internal functionality from the user. The file structure was not removed for verification reasons.

1. Usage

Combining all of the different scripts together into one standalone application removed the difficulty and confusion of working with the different scripts. A single directory needs to be selected and then any issue is voiced to the user verbosely. To combat the problem of a confusing and difficult configuration file. An editor was also made that allows for straightforward editing of the configuration file without intimate knowledge of the internal workings of the system. This editor has tooltip icons on the text entry fields that have the needed information about what data needs to be inserted. An expert in human-computer interaction was consulted for the design of the user interface and the advice is currently being implemented.

2. Speed

Simply by switching languages the speed of the system was increased by 254%, see Table 1. Other than the benefit of using a compiled language rather than a scripted language, streamlining in functional design allowed for a cleaner feeling that was less daunting and easier to use. The combination of GUI design and translation solved the main concerns about the speed of the system.

3. Extensibility

Java is able to efficiently fill many roles on many different kinds of systems. If the trend is going to be towards a web or app based implementation, Java is capable of supporting a versatile web interface on any server. Otherwise if a downloaded version of the system is desired Java is capable of using modern graphical techniques to create a stunning display. This allows for a web based interface on any form of server platform that is available to be used. Apart from language advantages, the libraries that are being used for matrices and GIS are more fully featured then their python counterparts.

C. Future Developments

The future developments are prioritized by the immediate importance to the project. Implementing all of the human interaction advice is of utmost importance. Closely following this is obliterating the file structure that the python scripts use. Once the internal functionality of the system has been deemed acceptable for operation. Development towards large scale deployment will take place.

IV. CONCLUSION

In summary, there is a large gap in the general public’s knowledge of power line routing and there is great difficulty in finding reliable public opinion. A proof of concept python system was developed that gives an optimized route for a powerline. This python system has been translated into a fully featured Java application that allows for increased extensibility and user friendliness.

V. FIGURES AND TABLES

Timing Experiments for Python System And Java Application

Run	Python	Java
1	65.36s	22.13s
2	57.47s	20.51s

3	49.29s	21.48s
4	51.69s	23.05
5	56.72s	23.21s
Average	56.106s	22.076s

Table 1, Timing experiments conducted with batch script

REFERENCES

[1]J. Kuiper, D. Ames, D. Koehler, R. Lee, T. Quinby. INL/CON-13-28372 (2013)