Efficient Power Delivery Using Artificial Intelligence Planning

Dennis Sosa, Paul Fodor
Contact: dennis.sosa@stonybrook.edu
Computer Science Department, Stony Brook University, Stony Brook, NY

Introduction

The purpose of this research project is to design an efficient planning algorithm to implement power delivery in smart power grids.

For example, think about a map of the "Lower 48" states in the United States, and only four main power plants in four different states. From these four power plants, the goal is to power on a data center. Junctions and switches connect power plants and data centers.

The problem is to find the shortest sequential plan to power on a data center by turning on the least amount of power switches, which connect to junctions and to one of the main power plants.

Background

Prolog is a general purpose logic programming language associated with artificial intelligence, planning and querying. It was first developed by a group of computer scientists in Marseille, France in the early 1970s.

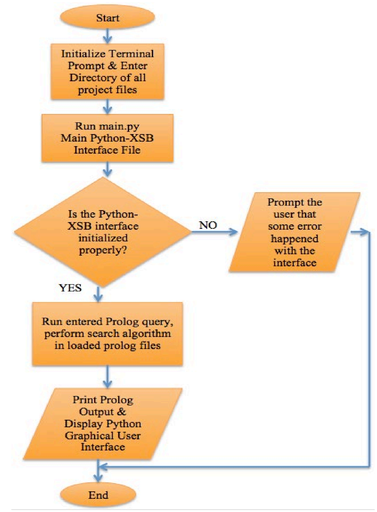
We use the Stony Brook XSB Prolog logic programming and deductive database system. XSB extends Prolog with tabled resolution, HiLog (an extension of Prolog permitting limited higher-order logic programming), and interfaces to higher level programming languages (i.e., Java and Python).

Methods

Artificial Intelligence with Prolog (Logic Programming):

- **Database/Knowledge Base** – Facts and Rules
- **Variables** – Always capitalized
- **Backtracking** – The way it searches for a pattern is called backtracking. The repeated searching for additional solutions. Goes back and tries again to find a solution.
- **Unification** – The way it matches and binds with a pattern is called unification. The binding of a variable with each name in succession, which matches query's predicate in the database
- **Recursion** – Handles the looping of the search
- **Query** – When goal is found, Prolog responds "yes", else it responds "no"

Algorithm Flow Chart →

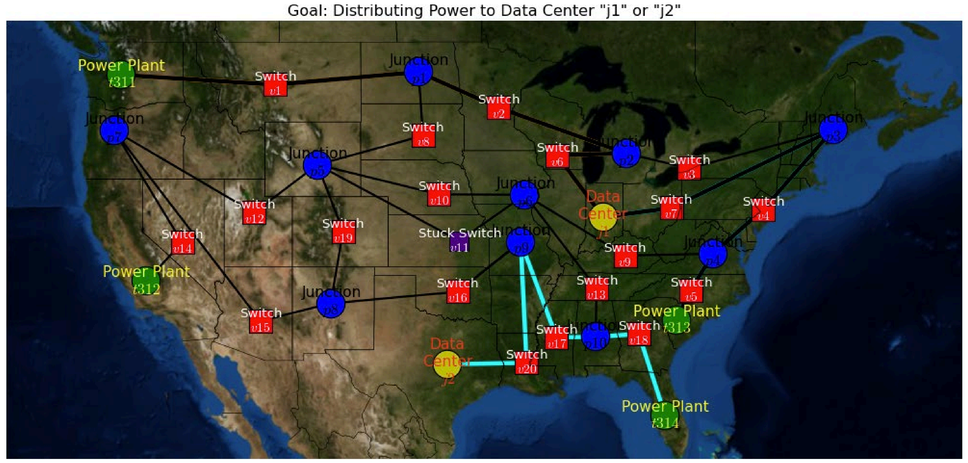


Results/Example

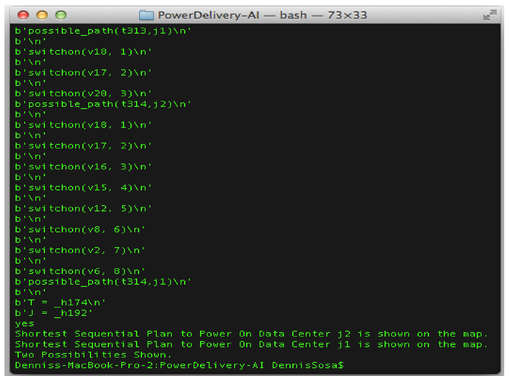
With the use of the Python packages: *os*, *sys*, and *subprocess*, a Python-XSB interface was successfully compiled in order to form communication between the Python and XSB programs.

Electric Power System Planning – Python Graphical User Interface

Goal: Distributing Power to Data Center "j1" or "j2"



Electric Power System Planning – Prolog (Artificial Intelligence) Output →



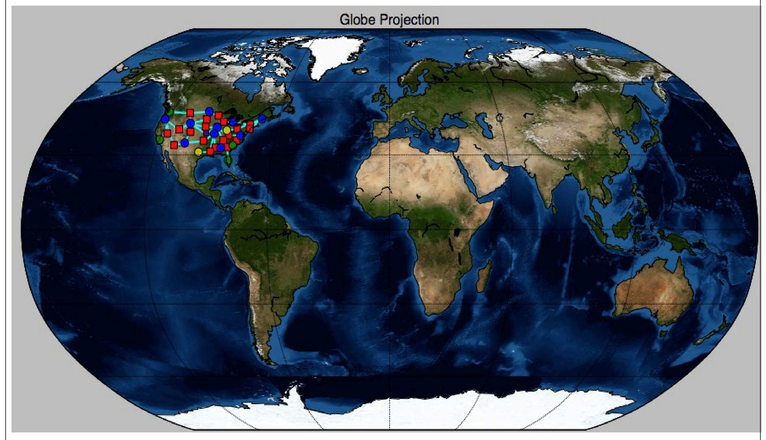
Conclusion

As shown in the illustrations, with Prolog's backtracking and unification power, the inputted database in accordance to the problem being solved, was used in parallel to another Prolog algorithm filled with predicates that allowed the automated artificially intelligent process to find the shortest path to distribute electric power to the desired data center on the map. With the implementation and communication of the Python and XSB interface, the results were successfully outputted as desired; the python graphical user-interface illustrated the overview of the Prolog database in accordance to the problem we were working with in this program, and the terminal prompt displayed the path being pursued to solve the problem on the map.

As the automated logic program searched for paths in the map, the terminal prompt listed the resulting path as the python graphical-user interface was shown to the user. The terminal prompt was able to complete both processes at the same time.

Next Steps

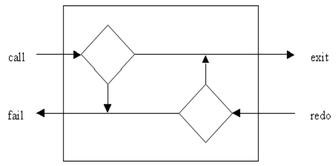
Now, lets look at the bigger picture. Imagine a management database of many electric power systems around the world. The algorithms used in this project can be used to solve a much bigger problem as well.



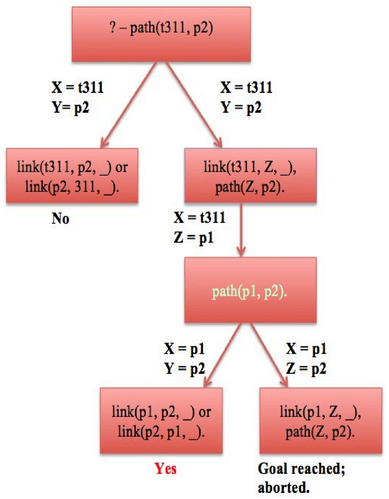
- **In continuation of this research project, Professor Fodor and I plan to develop a JavaScript algorithm for a Web distributed consumption of this efficient algorithm.**

Methods

XSB Methodology: Prolog Backtracking



Prolog Sample Tree Diagram



Consider the following Prolog rules & predicates as an example:

```
% Prolog database of connections
% links connecting graph nodes and switches
link(t311,p1,s1).
link(p1,p2,s2).

% Recursive Prolog Rules/ Program
path(X,Y) :-
   link(X,Y,_) ; link(Y,X,_).
path(X,Y) :-
   link(X,Z,_),
   path(Z, Y) .
```

References/Acknowledgements

- http://xsb.sourceforge.net
- http://python.org
- http://matplotlib.org
- http://matplotlib.org/basemap
- http://networkx.github.io

The listed links were used for Python and Prolog documentation and information used for the implemented interface.

Funding

- Public Service Enterprise Group (PSEG)