Modelling District Heating and Cooling network

Leanne Dong

June 3, 2020

Overview

District Heating and Cooling Networks (DHN)

- The most common method for heating building in cities
- Usually consist of supply and return pipes that deliver heat in form of hot water or stream

Common graph problems

- Shortest Path Problem : Breadth First Search (BFS), Dijkstra's, Bellman-Ford, Floyd-Warshall
- Connectivity/Path finding: A* algorithm, BFS, DFS
- Negative Cycle : Bellman-Ford, Floyd-Warshall
- Traveling Salesman Problem:
- Bridges: These are edges in a graph whose removal could increase the number of connected components in the graph. They are important because they represent the vulnerabilities and bottlenecks with in the graph.
- Minimum Spanning Tree: Kruskal's and Prim's algorithm
- Maximum Network Flow: Ford-Fulkerson and Edmonds Karp& Dinic's algorithms

Our graph problems

Problem 1: : Suppose there are 1000 building in a town, can we implement an algorithm that connects all the thousand building? Implement a A* algorithm.

The **Depth First Search (DFS)** is the most fundamental search algorithm used to explore nodes and edges of a graph. It runs with a time complexity of $O(V + E)$ and is often used as a building block in other algorithms.
By itself the DFS isn't all that useful, but when augmented to perform other tasks such as

- Count connected components,
- Determine connectivity,
- Find bridges/articulation point.

then DFS really shines.

General problems one may ask first

Ask yourself:

- Is the graph directed or undirected?
- Are the edges of the graph weighted?
- Is the graph we will counter likely to be sparse or dense with edges?
- Should I use an adjacency matrix (or incidence matrix), adjacency list, an edge list or other structure to represent the graph efficiently?

Nonlinear system and root finding

- A system of nonlinear equation is a set of equations

$$\begin{aligned} f_1(x_1, x_2, \cdot, x_n) &= 0, \\ f_2(x_1, x_2, \cdot, x_n) &= 0, \\ &\vdots \\ f_n(x_1, x_2, \cdot, x_n) &= 0. \end{aligned}$$

- There are three type of nonlinear system in hydraulic model. The solutions are usually found via **Newton-Raphson** or Hardy Cross Method.
- An example of nonlinear system from the DHN in Scharnhauser Park (Hassine and Eicker 2011)

$$\begin{aligned} x^2 - 2x - y + 0.5 &= 0 \\ x^2 + 4 * y^2 - 4 &= 0 \end{aligned}$$

- To find the solution (root), we would use the C++ Linear algebra library Eigen.

```
        #include <iostream>
        #include </usr/include/eigen3/Eigen/Dense>
        #include <cmath>
        #include <vector>

        void newton2d()
        {
                // F
                auto F = [](const Eigen::Vector2d &x){
                           Eigen::Vector2d res;
                        res << pow(x(0),2) -2*x(0)-x(1)+0.5, pow(x(0),2) +4*pow(x(1),2)-x(1)-4;
                        return res;
                };

                // jacobian of F
                auto DF= [] (const Eigen::Vector2d &x){
                        Eigen::Matrix2d J;
                        J << 2*x(0)-2        , -1,
                                   2*x(0)                 , 8*x(1);
                          return J;
                };

                Eigen::Vector2d x, x_ast, s;
                x << 2, 0.25; // initial value
                x_ast << 1.9007, 0.3112; // solution
                   double tol=1E-10;

                std::vector<double> errors;
                errors.push_back((x-x_ast).norm());

                do
                {
                        s = DF(x).lu().solve(F(x));
                           x = x-s; // newton iteration
                        errors.push_back((x-x_ast).norm());
                }
                while (s.norm() > tol*x.norm());

                // create eigen vector from std::vector
                unsigned int n = errors.size();
                Eigen::Map<Eigen::VectorXd> err(errors.data(), n);

                std::cout << "solution" << std::endl;
                std::cout << x << std::endl;

                std::cout << "Errors:" << std::endl;
                std::cout << err << std::endl;

#include "newton2d.hpp"

int main()
{
        newton2d();
}
//The solution is found as 1.94 and 0.39.
```

Background

Please refer to LC Evans control theory course
`https://math.berkeley.edu/~evans/control.course.pdf`

Problem

Control problem: (1) how do you adjust the flow rate in individual substations to that the return temperature has a given value (for example always 40°C)?

Assumption

- The basic assumption for the calculation is incompressible media.
- Computation of flow distributions in DHN are mainly based on the Kirchhoff law for current and voltage in circuits: The two equations describe flow rate and pressure losses in the network
- Consider the PDE describing an one dimensional flow through a horizontal pipe which can be systematically derived from the Navier-Stokes equations.

$$\frac{l}{A}\frac{d\dot{m}}{dt} + \Delta p + R|\dot{m}|\dot{m} = 0 \tag{1}$$

Note: Δp denoting the difference in pressure head between the two pipes ends and $\dot{m}$ is the mass flow rate. The variable R stands for the hydraulic resistance of the pipe element, which is postulated to be a function of the physical properties such as length, roughness and diameter.

Network Topology

The description of the heating network is based on graph theory. First we need to form the topological matrice which show the structure of the mesh in matrix form. (To be added)

Solution of network equation system

Numerical solution via finite elements

By the first law of thermodynamics, the variation of the enthalpy of one pipe element is modelled by the following hyperbolic PDE:

$$\frac{\partial(mf)}{\partial t} = \dot{H} - \dot{H}(x + \Delta x) - d\dot{Q}_l \tag{2}$$

Model Validation

Network structure

to be added

Calculations

to be added