Extension of pgRouting Contraction framework to implement Contraction By Area

Contents:

Contact Details:

Name : Ankur Shukla
Country : India
Email : work.ankurshukla@gmail.com
Phone : +918291339758
Github : www.github.com/daas-ankur-shukla
Twitter : www.twitter.com/ankurCSRE
LinkedIn : https://www.linkedin.com/in/ankur-shukla-8b73b266/

Title:

Extension of pgRouting Contraction framework to implement contraction by area

Synopsis:

The current state of the contraction framework of pgRouting supports Dead End Contraction and Linear Contraction algorithms. My work focuses on extending it by implementing contraction by area border. Contraction by area can be implemented by calculating the shortest path between the border vertices and then making the graph by taking all the shortest paths together.
For my GSOC project I propose to implement this functionality as an extension for pgRouting's contraction framework.

Benefits to the community:

Area contraction will be a valuable addition to the contraction framework of pgRouting. Area contraction enables the user to derive contracted graphs in case he wants to store large graphs which are beyond what pgRouting can support due to memory restrictions. For the case of road routing applications, this will enable users to construct hierarchies of contraction by contracting areas and then further contracting the areas formed by union of all sub-contracted areas.
Thus this addition will be valuable to the community as it increases the functionality of pgRouting both in terms of query speed for problems which demand routes passing through areas but not routing within the area and also to hold larger graphs by contracting its parts.

Deliverable:

The deliverables for the project would be:

1. Implementation of area contraction
2. Complete documentation and tests for all the components of area contraction

Future Developments:

Some of the ideas which can be implemented to extend the proposed work are:

a. Algorithm for generating layers of contracted graphs using area contraction recursively

Timeline:

A. **Community Bonding Period** (May 4-May 29, 2017): *Students get to know mentors, read documentation, get up to speed to begin working on their projects*
    a. Make a wiki page on the pgRouting repository for fortnightly tasks and report list
    b. Study the pgRouting coding architecture
    c. Study the documentation framework
    d. Study test systems for queries used in the documentation and the unit tests (understand the unit test concept)
    e. Understand pgRouting contraction framework and how to develop classes and functions to extend it
    f. Discuss the design of the functionality of *pgr_contractArea*
    g. Revisit C++ tutorials and STL videos for better understanding of STL and its functions
B. **Coding Period** (May 30-August 29, 2017): *Coding officially begins!*
    a. Coding Period Phase 1 (May 30-June 29, 2017):
        i. Week 1 (May 30-June 5):
            1. Prepare the directory structure for developing the code, tests and documentation
        ii. Week 2 & 3 (June 6-June 19):
            1. Analyse contraction framework schema to output contracted graph
            2. Extract border paths using Dijkstra
            3. Create the necessary classes to store the paths efficiently
        iii. Week 4 (June 20-June 29): *Mentors and students can begin submitting Phase 1 evaluations*
            1. Test, debug and document code

2. Prepare work for Phase 1 submission along with a brief Phase 1 report
3. Prepare Pre-Phase 2 synopsis

b. Coding Period Phase 2 (June 30-July 27, 2017):
   i. Week 5 & 6 (June 30-July 13):
      1. Analyse border paths to extract the sub paths that are shared between them and the number of paths sharing the sub path
      2. Test, debug and document code
   ii. Week 7 & 8 (July 14-July 28): *Mentors and students can begin submitting Phase 2 evaluations*
      1. Implement the structure for the output in the postgresql database
      2. Discuss the unit tests for *pgr_contractArea*
      3. Prepare elaborate documentation and tests for Phase 2 components
      4. Prepare work for Phase 2 submission
      5. Prepare Pre-Phase 3 synopsis
c. Coding Period Phase 3 (July 29-August 29, 2017):
   i. Week 9 & 10 (July 29-August 4):
      1. Implement unit tests for *pgr_contractArea*
      2. Implement documentation queries
      3. Start writing the user's documentation
   ii. Week 11 (August 5-August 18):
      1. Debug based on unit test results
      2. Finish writing the user's documentation
   iii. Week 12 (August 19-August 25): *Final week: Students submit their final work product and their final mentor evaluation*
      1. Prepare Final Phase submission along with a detailed final phase report

Proposed work, if the time permits, implement the same function to allow user to choose from

- A*
- Bidirectional Dijkstra
- Bidirectional A*
- And of course, Dijkstra

Do you understand this is a serious commitment, equivalent to a full-time paid summer internship or summer job?

I understand that GSOC is a serious commitment and I am fully prepared to carry and complete the work. I will give my best efforts and try to make worthy contribution to the pgRouting community.

Do you have any known time conflicts during the official coding period?

No, I don't have any known conflicts during the official coding time.

Studies:

What is your School and degree?

1. **Post Graduation**: M. Tech in Geoinformatics and Natural Resources Engineering, from Centre of Studies in Resources Engineering, IIT Bombay, India
2. **Graduation**: B. Tech. in Electrical Engineering, from KNIT Sultanpur, India

Would your application contribute to your ongoing studies/degree? If so, how?

My application would help me alot in my studies as my masters degree focuses on Geoinformatics, and GIS technologies. My application will enhance my skills and vision for development of geospatial technologies, which I see as my future career goal. This project will also improve my coding and team playing skills and provide me an exposure of working in an international setup.

Biography:

I am a Geoinformatics student and an open source enthusiast with a keen interest in learning and developing open source geospatial technologies. I have been involved in the backend development of a project named `SPark`, under my faculty . The project aims to build a platform for open source real-time parking information for commuters. This project introduced me to a number of geospatial technologies including pgRouting.

I, with my team presented this project idea at FOSS4G ASIA conference held at IIIT Hyderabad this year, after which I started learning pgRouting from a development point of view. With the guidance of developers at pgRouting I have also submitted a pull request, as a part of 2.4 release. I am enthusiastic about my idea for this GSOC and looking forward to a great summer of code.

Programming and GIS:

a. Computing Experience:
   i. Operating Systems: Ubuntu 16.04, Windows 7
   ii. Programming Languages: C++, Java, Python, Javascript
   iii. Frameworks: Flask, Django
   iv. API: OpenLayers 3.0
   v. Tools: osm2pgrouting

b. GIS experience as a user:

   As a part of my course in Geoinformatics I have been involved in a case study to perform an analysis of toposheets using GIS softwares. This case study involved full stack processing of provided data. I used the following softwares to perform the mentioned tasks:
   a. QGIS and ESRI ArcGIS Desktop to georeference digitized hand made base maps
   b. Perform watershed delineation and soil map classification using ArcGIS tools

c. GIS programming and other software programming:

   I have been the backend developer of a smart parking application, made with IoT components, pgRouting, OpenLayers 3.0, and Django. This application is under development to provide real time parking space availability and fastest route to commuters.

GSOC Participation:

a. **Have you participated in GSOC before?** This is my first GSOC participation.
b. **Have you submitted/will you submit another proposal for GSoC 2016 to a different org?** Yes I am trying to submit proposals to other organization as well

Detailed Proposal:

Area Contraction:

To understand area contraction it is essential to understand area. An area is a set of vertices and edges connecting them, some of the vertices from the set of vertices are defined as the border vertices. These border vertices connect an area to its exterior.
Area contraction produces a contracted graph of an area such that the graph reduces to set of shortest path from each border vertex to every other border vertex keeping the border vertex as prohibited vertices.

Approach:

In the proposed approach, the graph is going to get contracted to a set containing paths between each border vertex. This contraction will create a graph which is a representation of the complete graph suitable for making any routes from one border vertex to another.
For this I propose the following approach:

1. Calculate the shortest path between each border vertex by using many to many routing functions
2. Create a graph by taking the union of all the paths
3. Categorize each subpath of the graph by keeping the count of individual paths shared by the resultant subpath
4. Output the contracted graph

Example:

I have taken the sample data which is used by pgRouting, to demonstrate the proposed contraction method. All edges have been assumed to have unit weight.

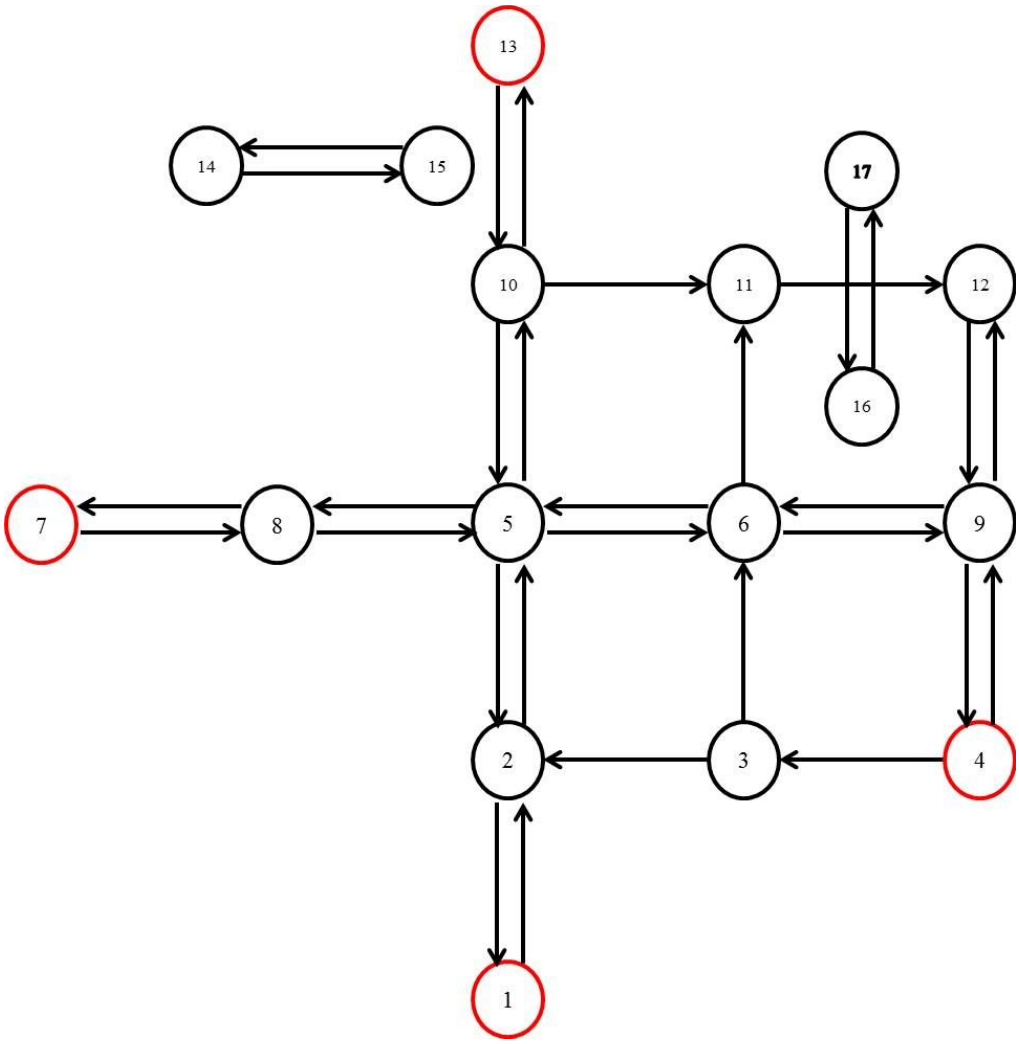

Figure 1: Complete sample data graph with red vertices as the border vertices

The graph in terms of the edges and vertices is::
E={(1,2)(2,1) (2,5)(5,2) (3,2)(3,6)(4,3) (4,9)(9,4) (5,6)(6,5)(5,8)(8,5)(5,10)(10,5) (6,9)(9,6)(6,11) (7,8)(8,7) (9,12)(12,9) (10,11)(10,13)(13,10) (11,12) (14,15)(15,14) (16,17)(17,16)}

V={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17}
|E|=30
|V|=17

Procedure:

Calculate One to Many Shortest paths for each border vertex to every other border vertex

Red Vertices : Border Vertices
Green Edge : Subpath
Blue Edge : Subpath being shared by 2 paths
Orange Edge : Subpath being shared by 3 paths
Red Edge : Subpath being shared by 4 paths
Brown Edge : Subpath being shared by 5 paths

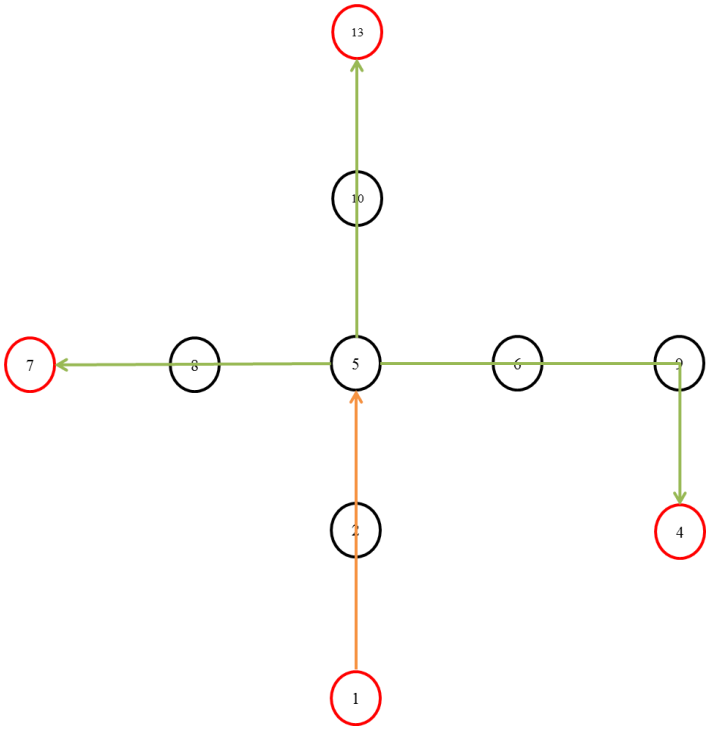

Figure 2: Paths from 1 to 4, 7, 13 with subpath from 1 to 5 being shared 3 times

Path_of(1,5) = 1→2→5
Path_of(5,4) = 5→6→9→4
Path_of(5,7) = 5→8→7
Path_of(5,13) = 5→10→13
The graph in terms of edges and vertices is:
E={(1,5) (5,7) (5,4) (5,13)}

V={1,2,5,6,7,8,4,10,13}
|E|=4
|V|=9

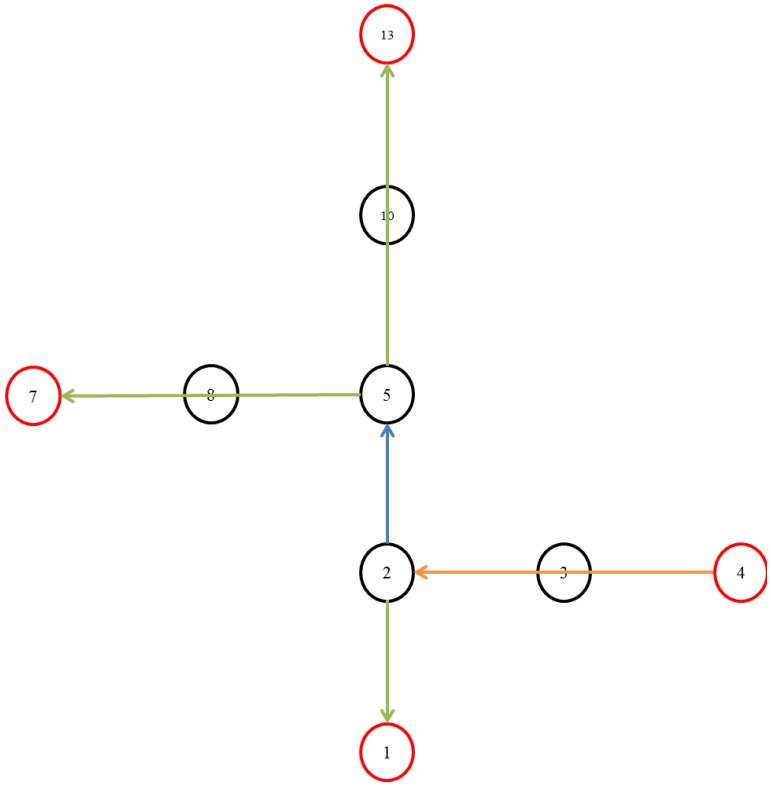

Figure 3: Paths from 4 to 1, 7, 13 with subpath from 4 to 2 being shared 3 times and subpath from 2 to 5 being shared 2 times

Path_of(2,1) = 2→1
Path_of(2,5) = 2→5
Path_of(4,2) = 4→3→2
Path_of(5,7) = 5→8→7
Path_of(5,13) = 5→10→13
The graph in terms of edges and vertices is:
E={(2,1)(2,5) (4,2) (5,7)(5,13)}
V={1,2,3,4,5,7,8,10,13}
|E|=5
|V|=9

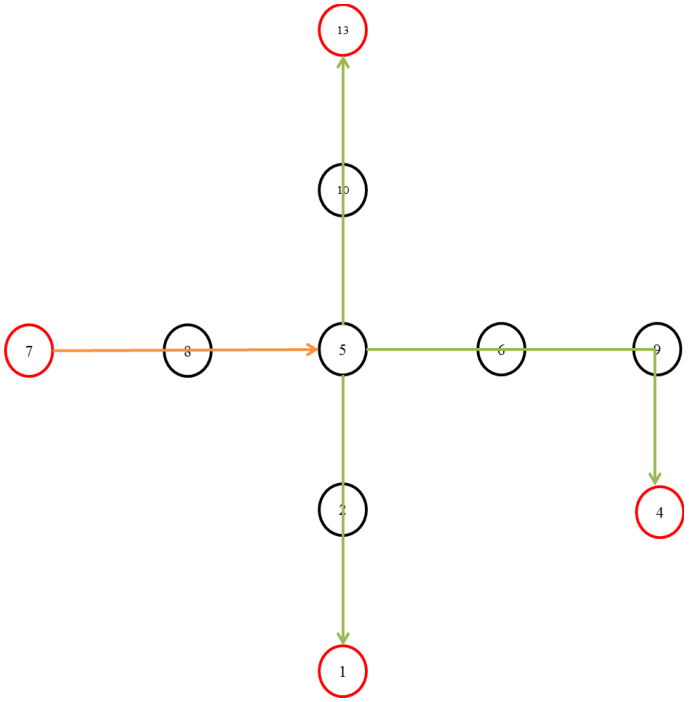

Figure 4: Paths from 7 to 1, 4, 13 with subpath from 7 to 5 being shared 3 times

Path_of(5,1) = 5→2→1
Path_of(5,4) = 5→6→9→4
Path_of(5,13) = 5→10→13
Path_of(7,5) = 7→8→5
The graph in terms of edges and vertices is:
E={(5,1)(5,4)(5,13) (7,5)}
V={1,2,4,5,6,7,8,9,10,13}
|E|=4
|V|=10

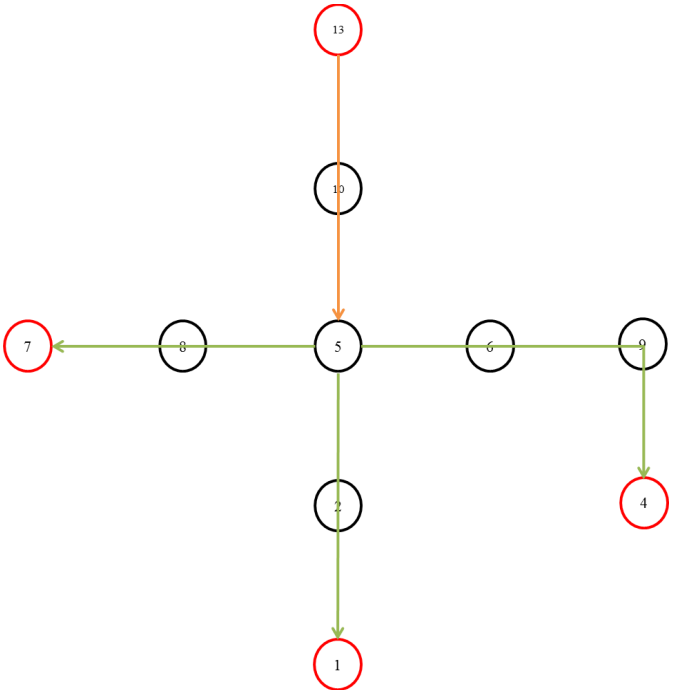

Figure 5: Paths from 13 to 1, 4, 7 with subpath from 13 to 5 being shared 3 times

Path_of(5,1) = 5→2→1
Path_of(5,4) = 5→6→9→4
Path_of(5,7) = 5→8→7
Path_of(13,5) = 13→10→5
The graph in terms of edges and vertices are:
E={(5,1)(5,4)(5,7)(13,5)}
V={1,2,4,5,6,7,8,9,10,12}
|E|=4
|V|=10

The combined contracted graph can be obtained by taking the union of all the one to many graphs. The number of times a subpath is shared changes accordingly.
E= {(1,5)(5,7)(5,4)(5,13)}
∪
{(2,1)(2,5)(4,2)(5,7)(5,13)}
∪
{(5,1)(5,4)(5,13)(7,5)}
∪
{(5,1)(5,4)(5,7)(13,5)}
E= {(1,5)(5,7)(5,4)(5,13) (2,1)(2,5)(4,2) (5,1)(7,5) (13,5)}

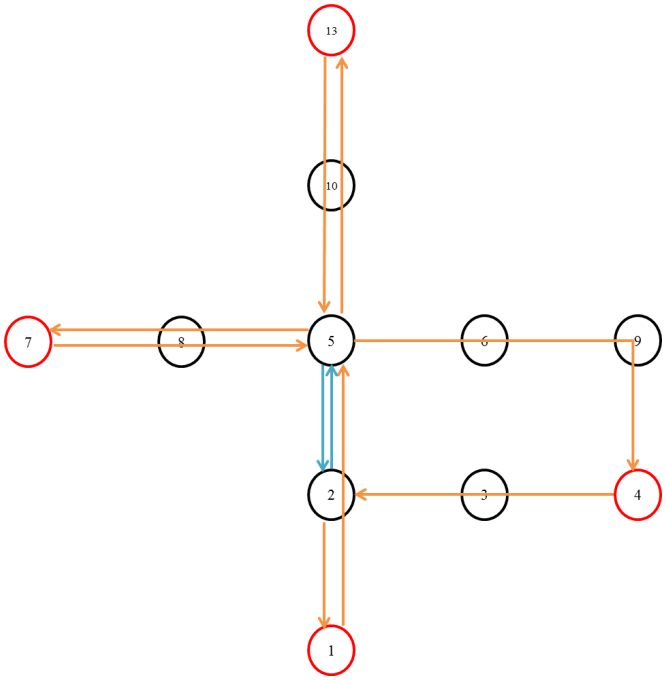

Figure 6: Combined many to many graph for all border vertices made by taking union of all one to many graphs

Path_of(1,5) = 1→2→5
Path_of(5,1) = 5→2→1
Path_of(2,1) = 2→1
Path_of(2,5) = 2→5
Path_of(4,2) = 4→3→2
Path_of(5,4) = 5→6→9→4
Path_of(5,7) = 5→8→7
Path_of(7,5) = 7→8→5
Path_of(5,13) = 5→10→13
Path_of(13,5) = 13→10→5

So this subset of edges {(1,5)(2,5)} that have the following paths:
Path_of(1,5) = 1→2→5
Path_of(2,5) = 2→5
They share a sub path, so those edges are transformed to be {(1,2)(2,5)}. And the operations to be done on the graph are as follows:
E = {(1,5)(5,7)(5,4)(5,13) (2,1)(2,5)(4,2) (5,1)(7,5) (13,5)}
E' = E - {(1,5)(2,5)} = {(5,7)(5,4)(5,13) (2,1)(4,2) (5,1)(7,5) (13,5)}
E'' = E' + {(1,2)(2,5)} = {(5,7)(5,4)(5,13) (2,1)(4,2) (5,1)(7,5) (13,5) (1,2)(2,5)}
which is the contracted graph result as shown in Figure 7.

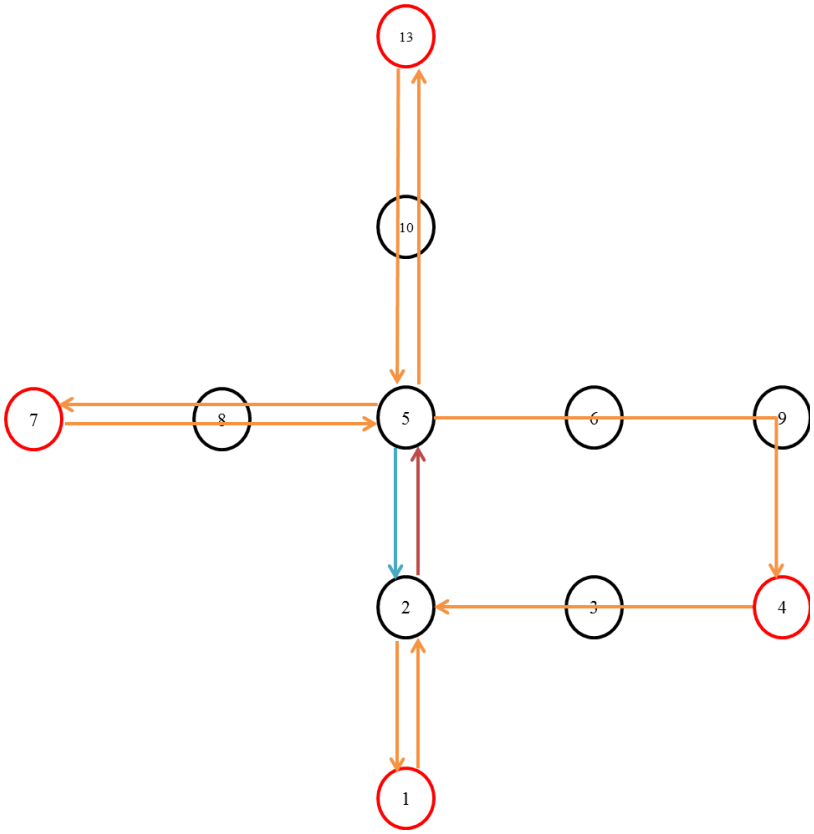

Figure 7: Visually representing the resulting graph of the Area Contraction

Proof of Concept:

As a proof of concept, that contraction of 2 adjacent areas gives the same result as contracting a large area, it can be shown as:

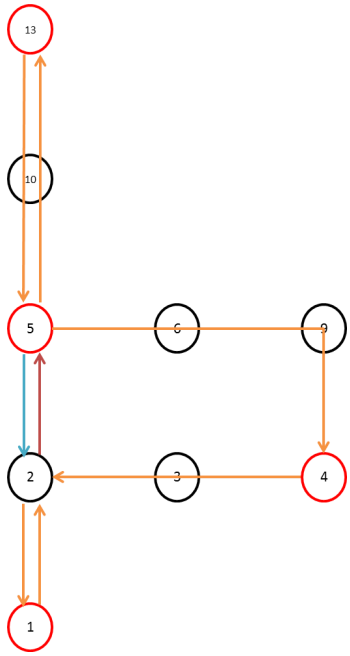

Figure 8: Contracted graph for a part of the complete sample graph

The edges for this graph are:
E1 = {(1,2)(2,1) (2,5)(5,2)(4,2) (5,4)(5,13)(13,5)}

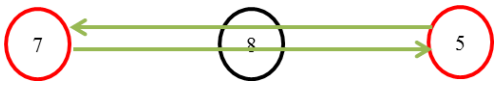

Figure 9: Graph of another part of the complete sample graph

The edges for this graph are:
E2 = {(5,7)(7,5)}

To obtain the complete graph we can take the union of the both graphs, resulting in the following edge set:
E3 = E1 ∪ E2 = {(1,2)(2,1) (2,5)(5,2)(4,2) (5,4)(5,13)(13,5) (5,7)(7,5)}

Which is equivalent to E'' from the complete graph. Thus proving that the union of two contracted graph is equivalent to contracting a complete graph.