BACKBONE DETERMINATION OF WIRELESS SENSOR NETWORKS MANUAL

ZIZHEN CHEN©

ABSTRACT. Wireless Sensor Networks (WSNs) have been the focus of intense research during the past few years because of their potential to facilitate data acquisition and scientific studies[1]. Lack of a fixed infrastructure and dynamic network topology make the routing problem one of the most challenging issues in the WSN area. One popular solution is forming a virtual backbone that forwards the packets. This algorithm engineering work graphically presents a procedure of backbone determination of various WSN geometries. The whole work is implemented via a software sketchbook Processing that is both a programming language and an Integrated Development Environment (IDE) built for visual arts[2]. This article is a manual of using the implemented graphic program.

1. INTRODUCTION

Wireless Sensor Networks (WSNs) are composed of inexpensive autonomous electronic sensors distributed over regions where the sensors communicate with each other wirelessly. Sensors are typically thrown in an unattended and random manner on the area to be monitored. Wireless communication allows the formation of flexible networks, which can be deployed rapidly over wide or inaccessible areas. Nowadays, WSNs emerge as an active research area in which challenging topics involve energy consumption, routing algorithms, selection of sensor locations, and so forth[3].

In a WSN of a given topology (as Figure 1 shows), the sensor information is usually collected then forwarded to a base station known as sink node. The need to gather data from all sensors in the network imposes constraints on the distances between sensors, so an efficient routing is required if large number of sensors are involved. Routing through only a connected subset of nodes (called "virtual backbone") in a WSN is one solution[4, 5, 6]. As only nodes in the virtual backbone (e.g. the blue and red nodes in Figure 1) forward packets or perform in-network services, other nodes can spend more time in a low-power idle mode. Therefore, a virtual backbone can simplify the routing process and reduce the overall network energy consumption. It also has other functions, such as transmission scheduling, broadcasting, and localization[7]. We use an Random Geometric Graph (RGG) and cluster-based control structure to model a WSN. RGG denoted $G(n, r)$ chooses random n locations of nodes uniformly and independently in a region and two nodes are connected by an edge if they are within a Euclidian distance less than a certain value r[8]. Cluster-based control structure allows a more efficient use of resources especially when a network consists of a large number of sensors[3]. Clustering consists in partitioning a network into a

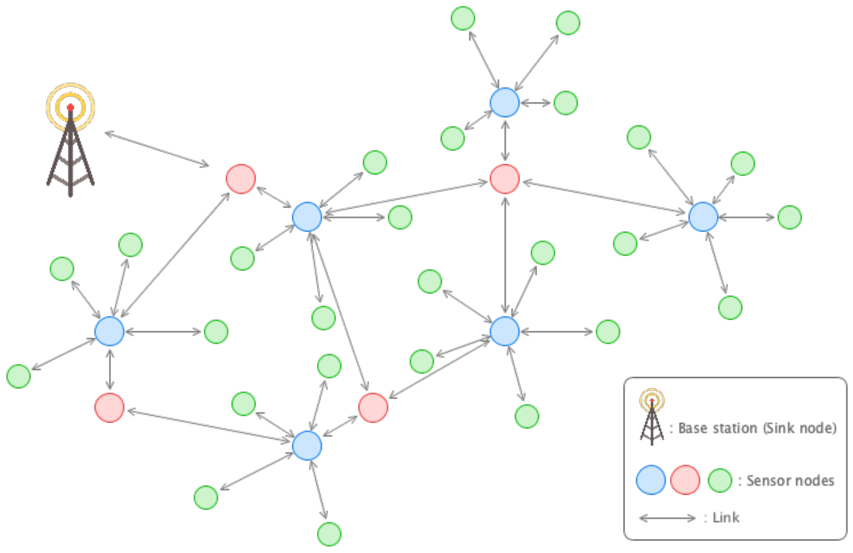


Figure 1. Wireless Sensor Network

number of clusters each is achieved by grouping nodes inside a certain transmission area (in the case of an RGG, it is a disc of radius r). A cluster is a locally connected group of nodes with a leading node (known as "cluster-head") from which each of the other nodes (known as "slave nodes") is one hop away. Forwarding information packets among cluster-heads requires a group of nodes as routing relays. We call such group of nodes a "relay set" and accordingly, the set of cluster-heads is called a "primary set". The union of a primary set and a relay set forms a virtual backbone.

Definition 1. *A **virtual backbone** of a WSN of cluster-based control structure is a connected subset of nodes composed of a primary set of cluster-heads and a relay set of routing relays. Backbone network provides a path for the exchange of information between sensor nodes and sink nodes.*

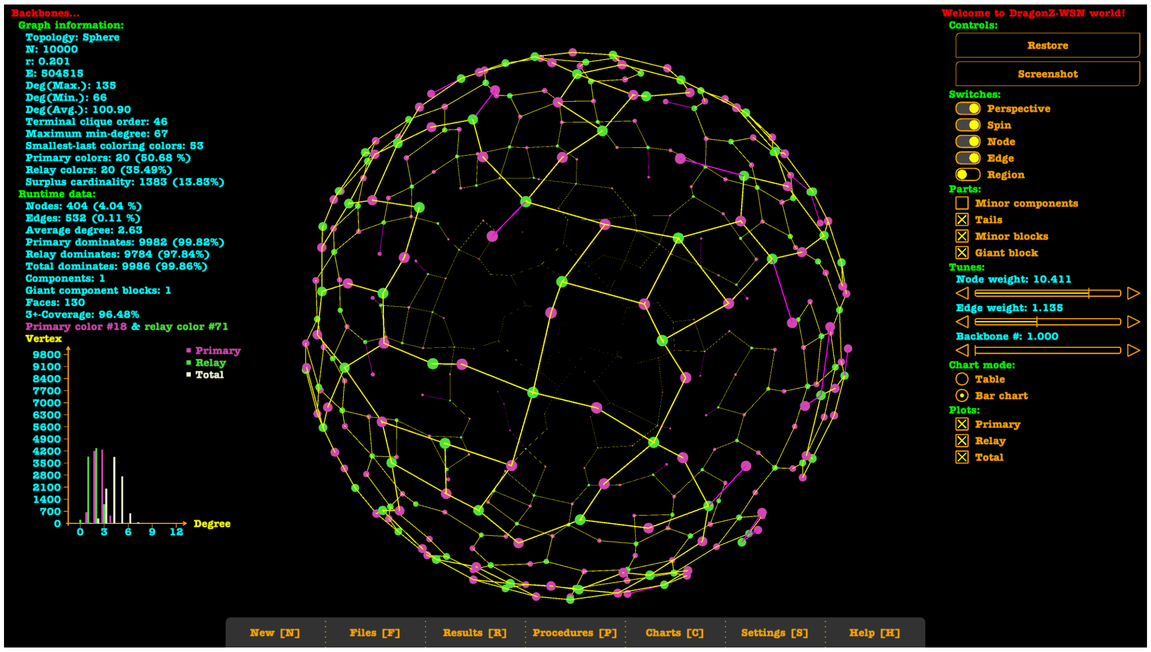
In Figure 1, each blue node is a cluster-head that controls its adjacent green nodes as slave nodes, and each red node is a routing relay to its adjacent blue nodes. The blue and red nodes form a virtual backbone routing the network's information to the base station.

This program partitions a WSN modeled as an RGG into a limited number of densely packed disjoint backbones. Each backbone is a subgraph that is composed of a pair of primary and relay set.

2. Prerequisites

2.1. **Platform.** Processing is developed based on Java, so there is no platform limitation unless some rare platform does not support Java Runtime Environment (JRE). We exported the program to Windows (64-bit), macOS and Linux. They are available at `http://lyle.smu.edu/~zizhenc/Project`. If you are using other platforms, the source code are located on GitHub: `https://github.com/zizhenc/WSN`. However, the program does require a mouse and a keyboard as input devices. It also supports single touch behavior if you have a touch screen.

2.2. **Screen Size.** Figure 2 shows a typical view of the program. There's no specific



```
Backbones...
  Graph information:
    Topology: Sphere
    N: 10000
    r: 0.201
    E: 504515
    Deg(Max.): 135
    Deg(Min.): 66
    Deg(Avg.): 100.90
    Terminal clique order: 46
    Maximum min-degree: 67
    Smallest-last coloring colors: 53
    Primary colors: 20 (50.68 %)
    Relay colors: 20 (35.49%)
    Surplus cardinality: 1383 (13.83%)
  Runtime data:
    Nodes: 404 (4.04 %)
    Edges: 532 (0.11 %)
    Average degree: 2.63
    Primary dominates: 9982 (99.82%)
    Relay dominates: 9784 (97.84%)
    Total dominates: 9986 (99.86%)
    Components: 1
    Giant component blocks: 1
    Faces: 130
    3+-Coverage: 96.48%
    Primary color #18 & relay color #71
```

```
Welcome to DragonZ-WSN world!
Controls:
Switches:
  Perspective [✓]
  Spin [✓]
  Node [✓]
  Edge [✓]
  Region
Parts:
  ☐ Minor components
  ☒ Tails
  ☒ Minor blocks
  ☒ Giant block
Tunes:
  Node weight: 10.411
  Edge weight: 1.135
  Backbone #: 1.000
Chart mode:
  ○ Table
  [✓ Bar chart]
Plots:
  ☒ Primary
  ☒ Relay
  ☒ Total
```

FIGURE 2. Program View

requirement of screen size since the window is resizable. However, it's preferred to have 16:9 window setup (We tested it in 1920x1080 window size).

3. NAVIGATION

There is always a navigation bar (as Figure 3 shows) as a dock in the middle bottom of the program window. Intuitively, the navigation bar supports both mouse control and keyboard

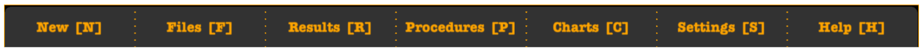


FIGURE 3. Navigation Bar

control. If a mouse cursor clicked any item of the navigation bar, the corresponding menu will pop up and waiting a further selection of an entry of further action (as Figure 4 shows). If the mouse cursor clicked somewhere else, the menu disappear. Each item of the bar or

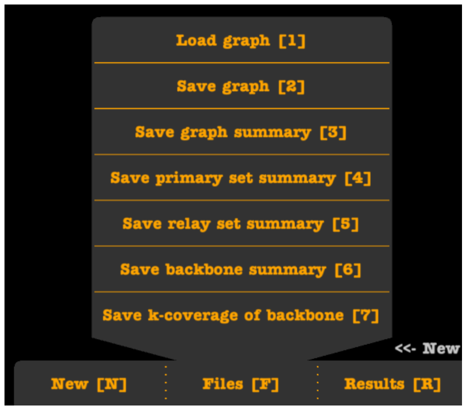

Load graph [1]
Save graph [2]
Save graph summary [3]
Save primary set summary [4]
Save relay set summary [5]
Save backbone summary [6]
Save k-coverage of backbone [7]
<<- New
New [N]
Files [F]
Results [R]

FIGURE 4. Item Menu

any menu entry has a square bracket enclosed character ([*]) as an indicator for keyboard control. The key combination of activating any item is "**ALT** + *". For example, "**ALT** + **F**" will pop up the "Files" menu and furthermore, "**ALT** + **1**" will activate "Load graph" entry and pop up a dialog to load a graph. Each menu also supports Up and Down arrow keys to highlight any entry and press Enter to activate the corresponding action.

The navigation bar contains the following items.

New: Start a new project.
Files: Load or save a file of graph information.
Results: Show graphic final results of computations.
Procedures: Several algorithm procedures of backbone determination.
Charts: Show plot charts of some results.
Settings: Configure settings of the program.
Help: Show documentation or about information.

4. Create a New Project

There are three modes of a new project as the"New" menu (Figure 5) shows.

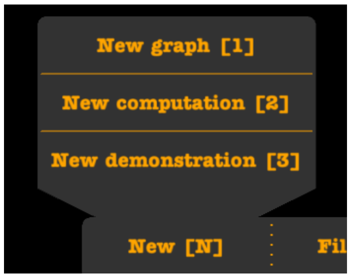
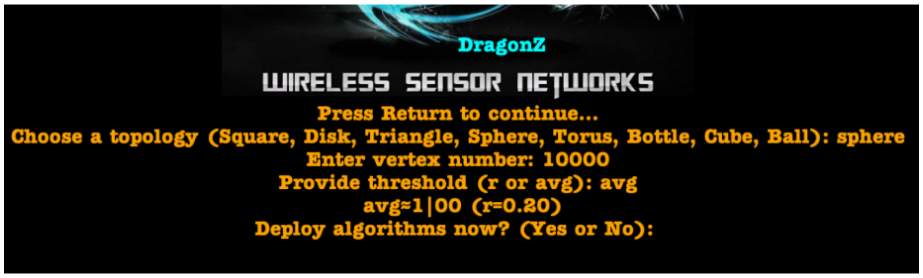


FIGURE 5. "New" Menu

(1) "New graph" starts a normal project that requires manually selecting each consecutive algorithmic procedure to partition a WSN into virtual backbones.
(2) "New computation" is designed for performing algorithms on a WSN of huge amount of nodes. It does not show the graphic procedures and directly complete all algorithmic procedures. However, you are still able to show graphic final results through the "Results" menu.
(3) "New demonstration" starts a project in a demo mode which automatically performs all consecutive algorithmic procedures on loop.

To start a project, you are required to have a keyboard to input basic information of the expected WSN. The input supports mouse and keyboard control. Mouse control is mainly for fixing input typos (use mouse to change the place of input cursor as Figure 6 shows). Keyboard control supports Backspace, Delete, Enter and arrow keys. Each information



```
Press Return to continue...
Choose a topology (Square, Disk, Triangle, Sphere, Torus, Bottle, Cube, Ball): sphere
Enter vertex number: 10000
Provide threshold (r or avg): avg
avg≈1|00 (r=0.20)
Deploy algorithms now? (Yes or No):
```

FIGURE 6. Input

needs an Enter press for confirmation.

5. GRAPHIC AND USER INTERFACE (UI) CONTROL

As Figure 2 shows, the view of a screen in the program typically can be divided into four parts:

(1) **Middle bottom** shows the navigation bar.
(2) **Left column** shows the runtime data (Screens of "Chart"s do not have left runtime data column).
(3) **Right column** is the control area.
(4) **Center** shows the main graphics.

Screens of "Results", "Procedures" and "Charts" have such view. Other screens have their own dedicated view. "Files" does not have screen.

5.1. Main Graphics Control.

Any generated networks and chart plots have 3D control features like moving, zooming and rotating supported by mouse and keyboard control (as Table 1 shows).

Table 1. 3D Control Features

Control	Mouse	Keyboard
Moving	right button pressed and drag	arrow keys
Zooming	wheel scroll	'w' and 's' keys
Rotation	left button pressed and drag[1]	'a' and 'd' keys

5.2. **UI Control.** Most UI controls are traditional mouse control parts (i.e. buttons, sliders, radios and check boxes) and majority of them support both mouse and keyboard control. The corresponding keyboard controls are pressing the letter key indicated by the bracket ([*]) directly. Some UIs are controlled by mouse only such as radios and sliders. Text field is the UI controlled by keyboard only. In summary, if an UI has a bracket ([*]), then it supports the keyboard control by pressing * key. One exception is the check box. The keyboard controls of check boxes are pressing number keys starting from 0 in the order of the list of check boxes. Keyboard controls are not influence by shift or caps lock keys.

References

[1] Geoffrey Werner-Allen, Konrad Lorincz, Mario Ruiz, Omar Marcillo, Jeff Johnson, Jonathan Lees, and Matt Welsh. Deploying a wireless sensor network on an active volcano. *IEEE internet computing*, 10(2):18–25, 2006.

[2] Casey Reas and Ben Fry. *Processing: a programming handbook for visual designers and artists*. Mit Press, 2007.

[3] Miriam Carlos-Mancilla, Ernesto López-Mellado, and Mario Siller. Wireless sensor networks formation: approaches and techniques. *Journal of Sensors*, 2016, 2016.

[4] Jamal N Al-Karaki and Ahmed E Kamal. Efficient virtual-backbone routing in mobile ad hoc networks. *Computer Networks*, 52(2):327–350, 2008.

[5] Dhia Mahjoub and David W Matula. Experimental study of independent and dominating sets in wireless sensor networks using graph coloring algorithms. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 32–42. Springer, 2009.

[6] Zizhen Chen and David W Matula. Bipartite grid partitioning of a random geometric graph. In *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 163–169. IEEE, 2017.

[7] Yu Wang, WeiZhao Wang, and Xiang-Yang Li. Distributed low-cost backbone formation for wireless ad hoc networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 2–13, 2005.

[8] Jesper Dall and Michael Christensen. Random geometric graphs. *Physical review E*, 66(1):016121, 2002.

[1]Chart plots do not support.