**Project Name:** Soukup Global Grid Routing

**Group Member:** Jose Rodriguez, Shujuan Wang

**1. Introduction of the project:**

The goal of this project is to implement the Soukup Global Grid Routing. It is a global router which develops all connections simultaneously. This router was firstly designed and implemented by Soukup[1]. The main idea for this router is to consider all nets “alive” and let them compete for the space to connect to each target. This principle, to some extent, avoids the prior connected nets destroying the connection of the following nets.

The router can be divided to two parts: Global Router (GR1) and the second Global Router (GR2), which are implemented in our project. The following lists the main ideas that implement in our project for each of the router:

1.1 GR1:

The source and target grow simultaneously and generate their own subnets. When these two subnets meet, it will be combined to be one net and sparked a connection. This process is similar to Lee-type algorithm, however, the connected net will continue to change and move to avoid blocking other unconnected subnets. The useful rule of GR1 guiding for our coding is listed:

(1) Nets which needed to be connected have higher priority than the one that has been completed

(2) The connected nets should never be broken. In our program, we need to decide which cell is the bottleneck cell and whether it is a local bottleneck or global bottleneck to avoid breaking up a connected net into disconnected areas.

(3) Alphabetically ordering is used to avoid two subnets have equal priorities.

(4) No nets can enter the pin area of another net.

1.2 GR2:

GR2 is used for the more complicated cases that GR1 can’t handle. As in some cases, the rules of GR1 will hinder the following subnets to be connected. However, GR2 enforces the individual subnets one by one to clean up the connection left unfinished by GR1. Though GR2 enforces one chosen subnet to connect, it is still global in nature.

**2. Structure of the coding:**

In the source folder of our code, we have the following source files:

grbaserouter.cpp, grgrid.cpp, grnetlist.cpp, soukups\_gr\_one.cpp, soukups\_gr\_two.cpp, problem\_object.cpp, main.cpp

The functions of these files are briefly introduced in the following (more comments are added in the code)

(1) grbaserouter.cpp: It’s used to decide whether the node is a local bottleneck. Find the routing path and retrace the path.

(2) grgrid.cpp: It’s used to construct the grid for the routing.

(3) grnetlist.cpp: It’s used to set and get the properties for the net. Such as set and get the priority and NetID for the net

(4) soukups\_gr\_one.cpp: This is used for GR1 router. Implement the priority setting and route the nets using GR1

(5) soukups\_gr\_two.cpp: This is used for GR2 router.

(6) problem\_object.cpp: This is used for getting useful information from testing json file.

**3. Test cases:**

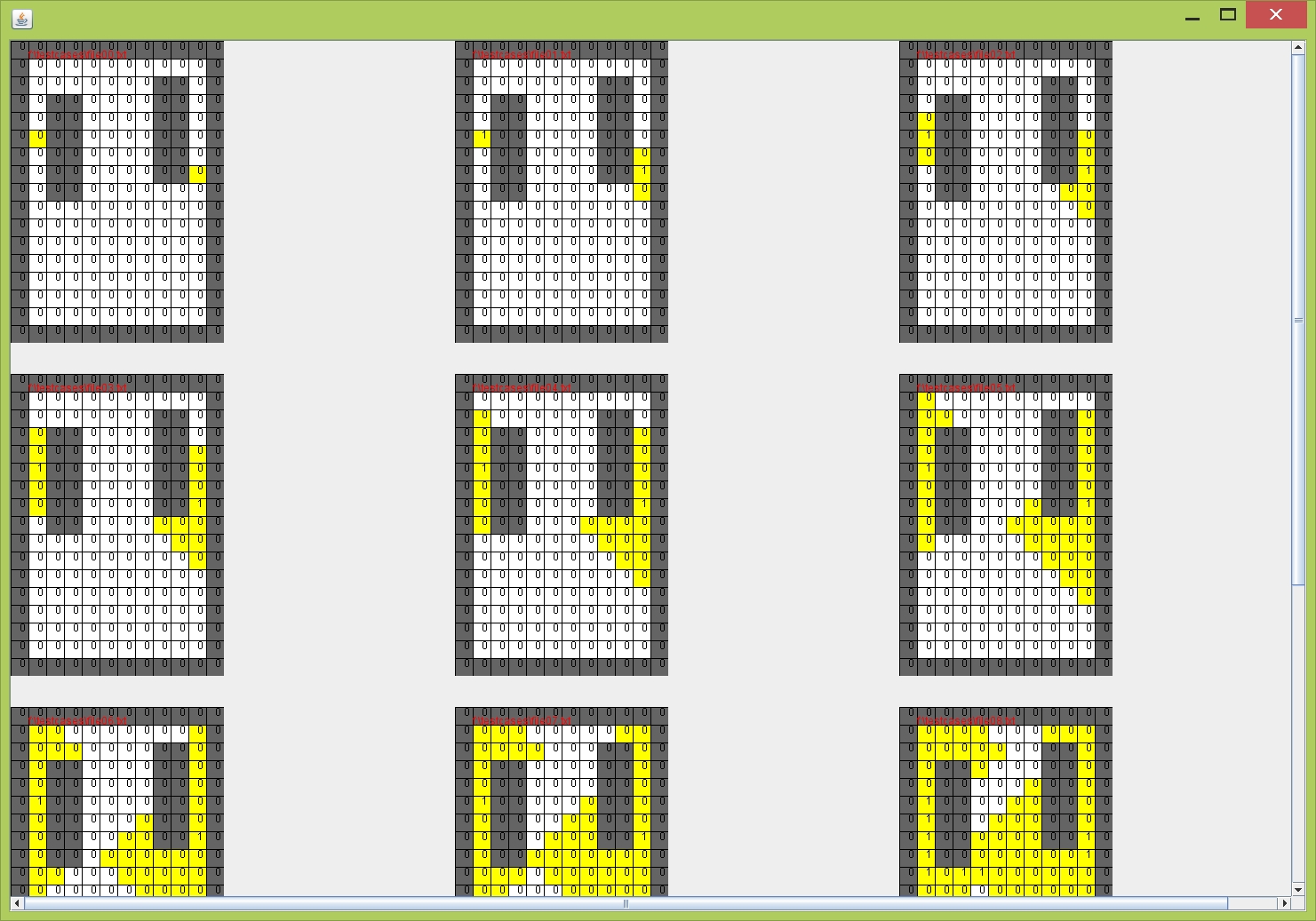
**(The following test pictures are shown by java code which displays our test results, and the java code is attached in our source code)**

**3.1 GR1**

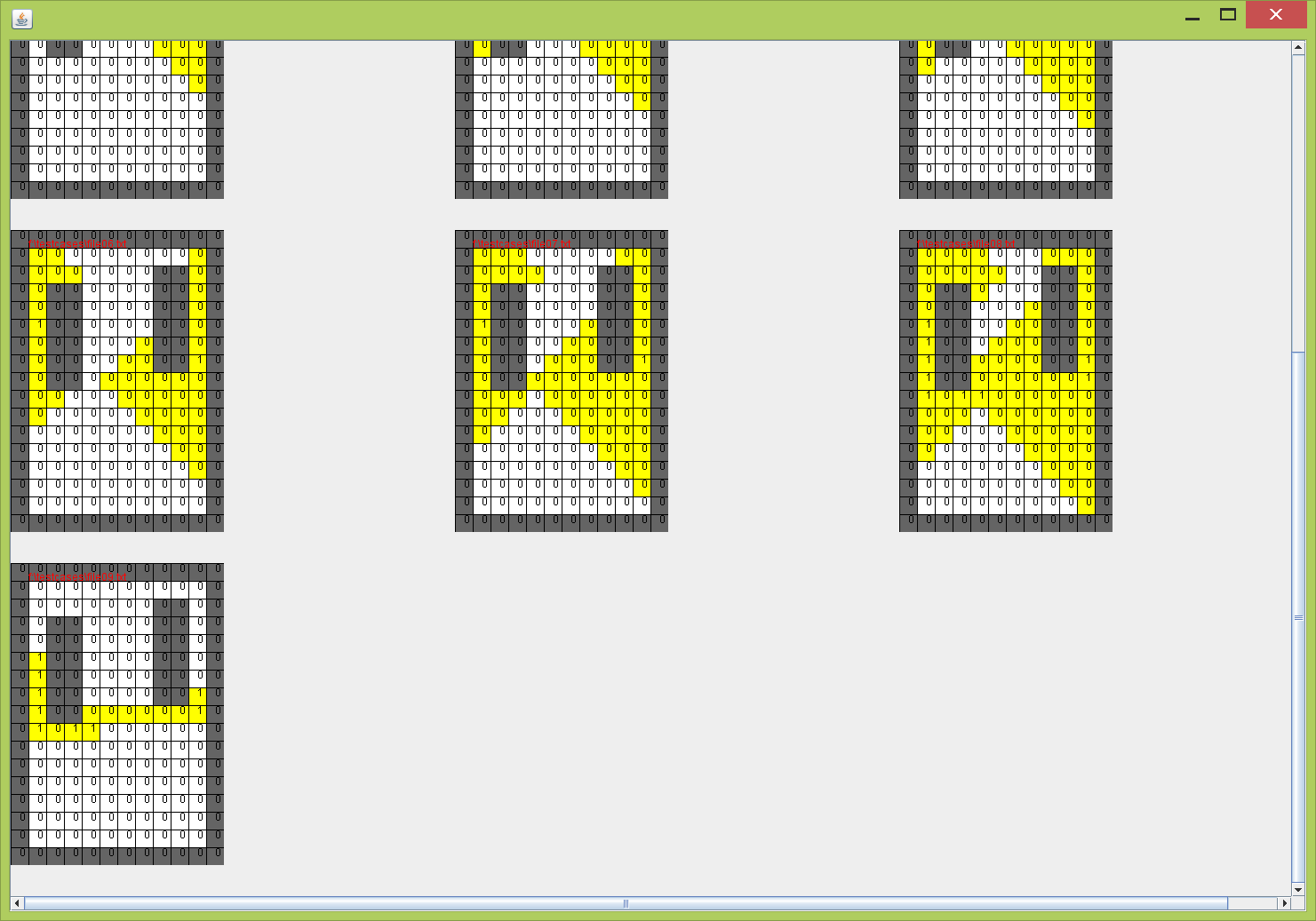
**Easy case**

**Case 1:**

Case1 starts from node (1,5) and sink at node (10,7). Figure1 shows the steps of case1 routing. The path from source to sink is found in the end shown in figure1 (b). Our code did well in this case. The number in the figure means whether the node is a bottleneck (except the number in the pins).



(a)



(b)

Figure 1 GR1 Case 1

**Case 2:**

This case has two nets. The first net starts from node (5, 4) to node (7, 3). The second net starts from node (5, 8) to node (7, 7). As figure 2 shows, both of the paths can be found quickly by our router.

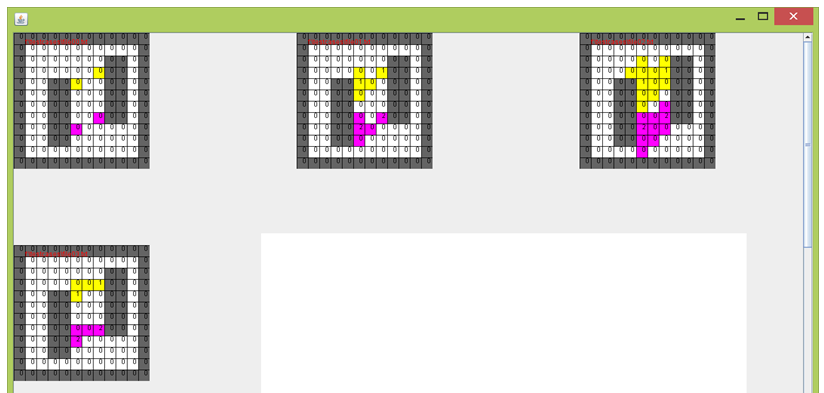
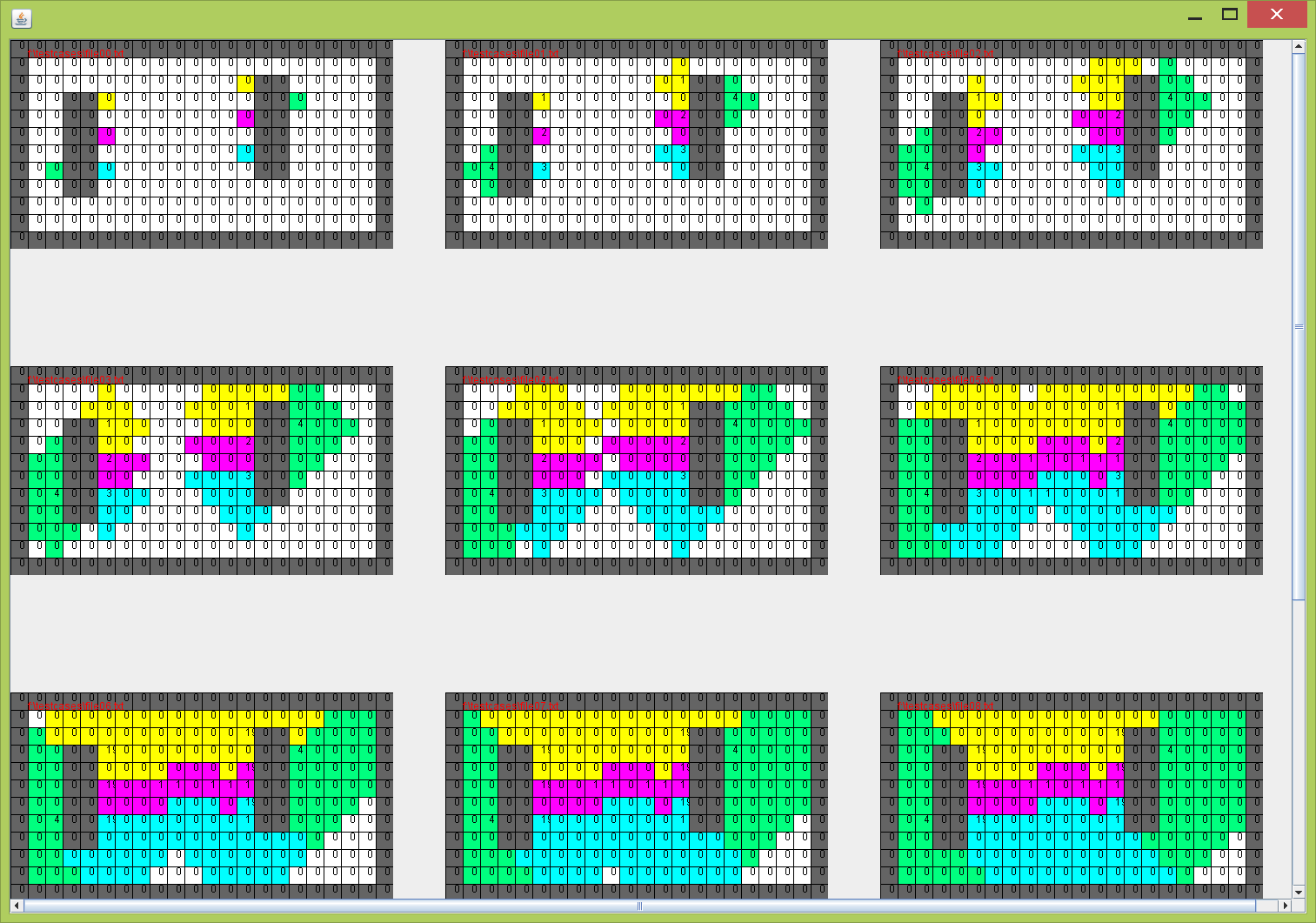


Figure2 GR1 Case2

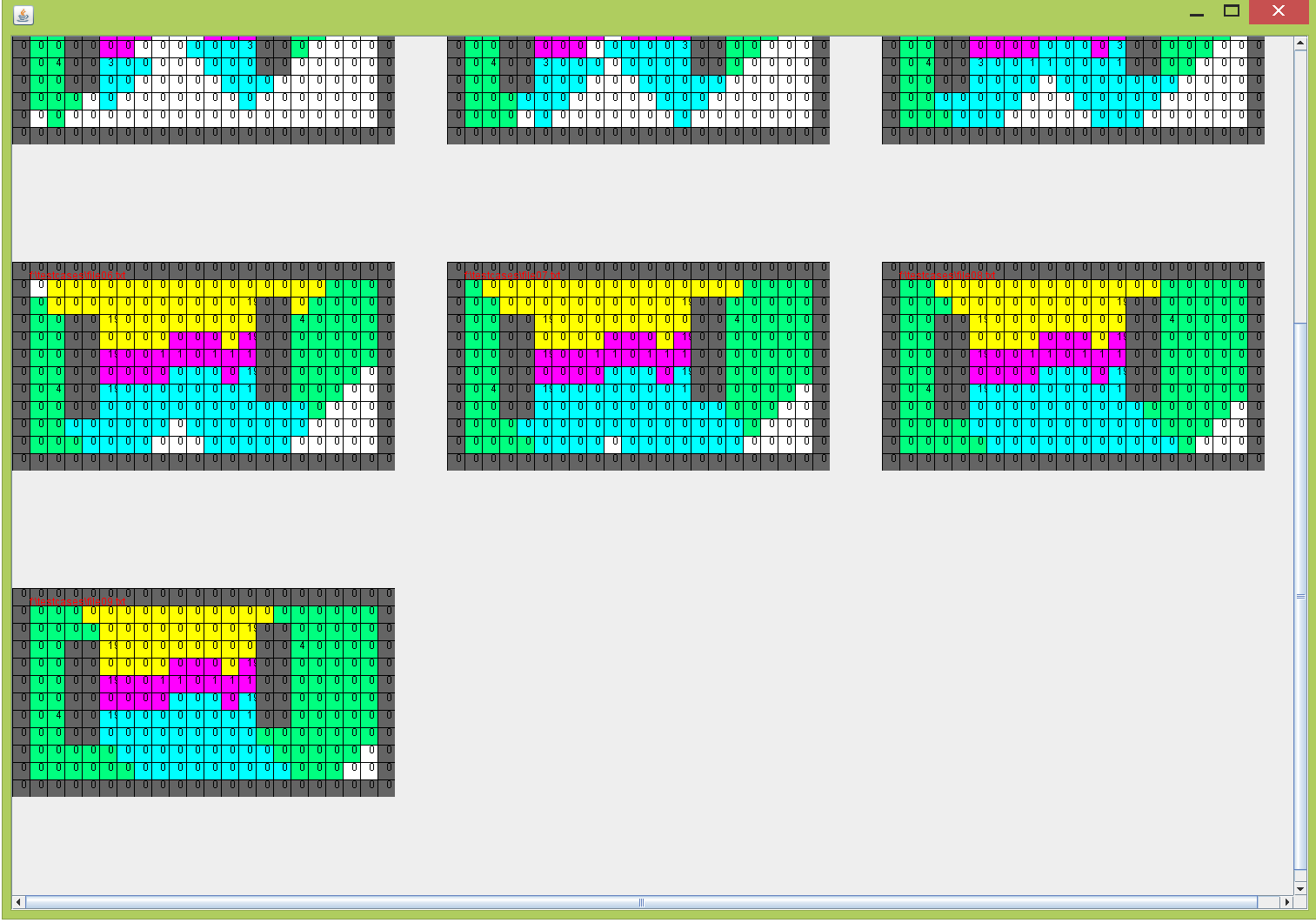
**Intermediate Case:**

**Case 3:**

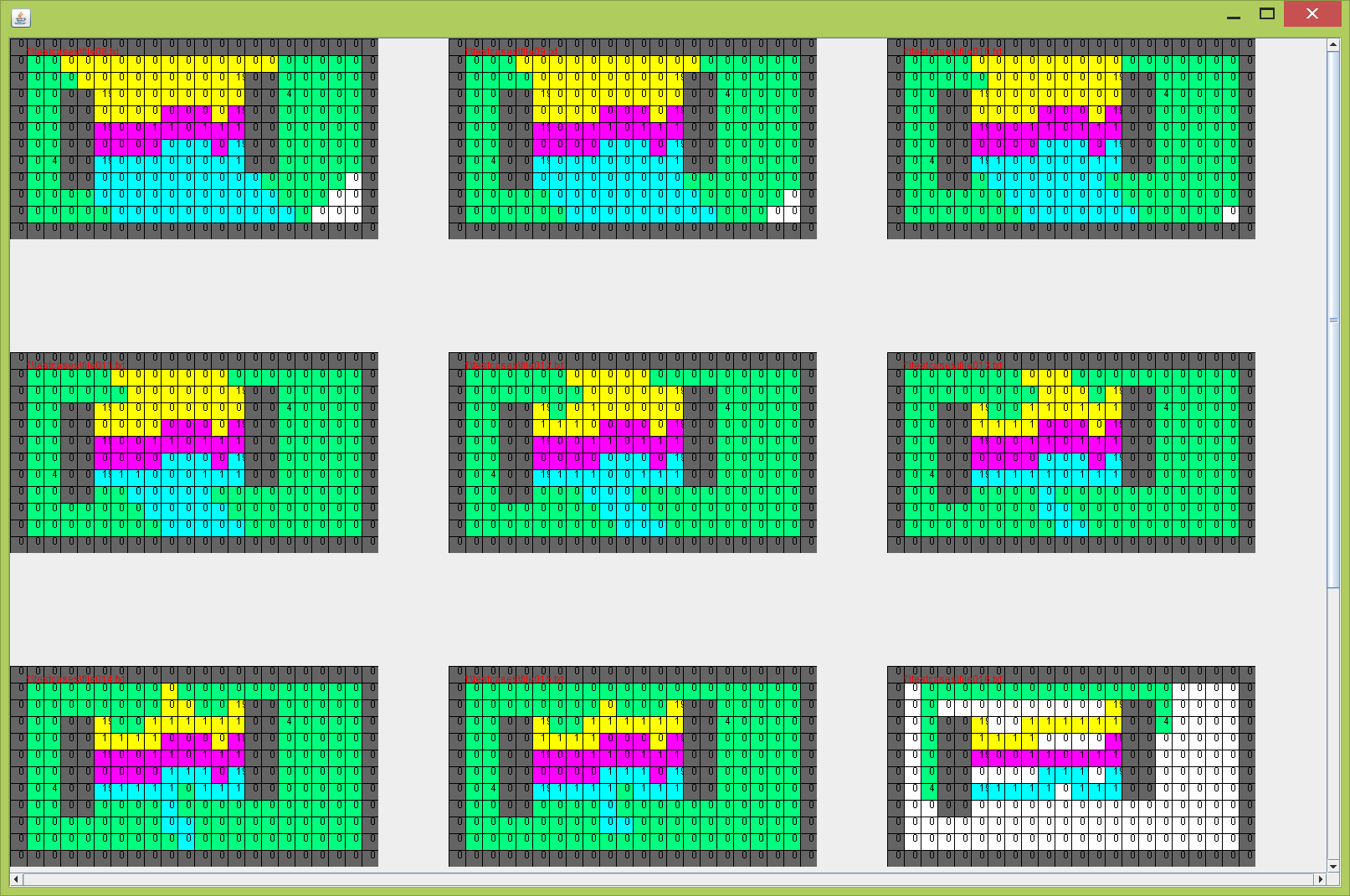
In this case, we have 4 nets, the green net starts from node (2, 7) to node (16, 3). The pink net starts from net (5, 5) to node (13, 4). The yellow net starts from node (5, 3) to node (13, 2). The blue net starts from node (5, 7) to node (13, 6). Some of the node number changes to be 1 as they become global bottleneck. The priority of the nets decreases after they are connected. The priority of unconnected nets sorts by the distance to the nearest subnet within the same net. Figure 3 shows the result of our routing. The yellow, pink and blue nets firstly finish the connection and then the green net. All the paths are finally found in figure 3(c).



(a)



(b)



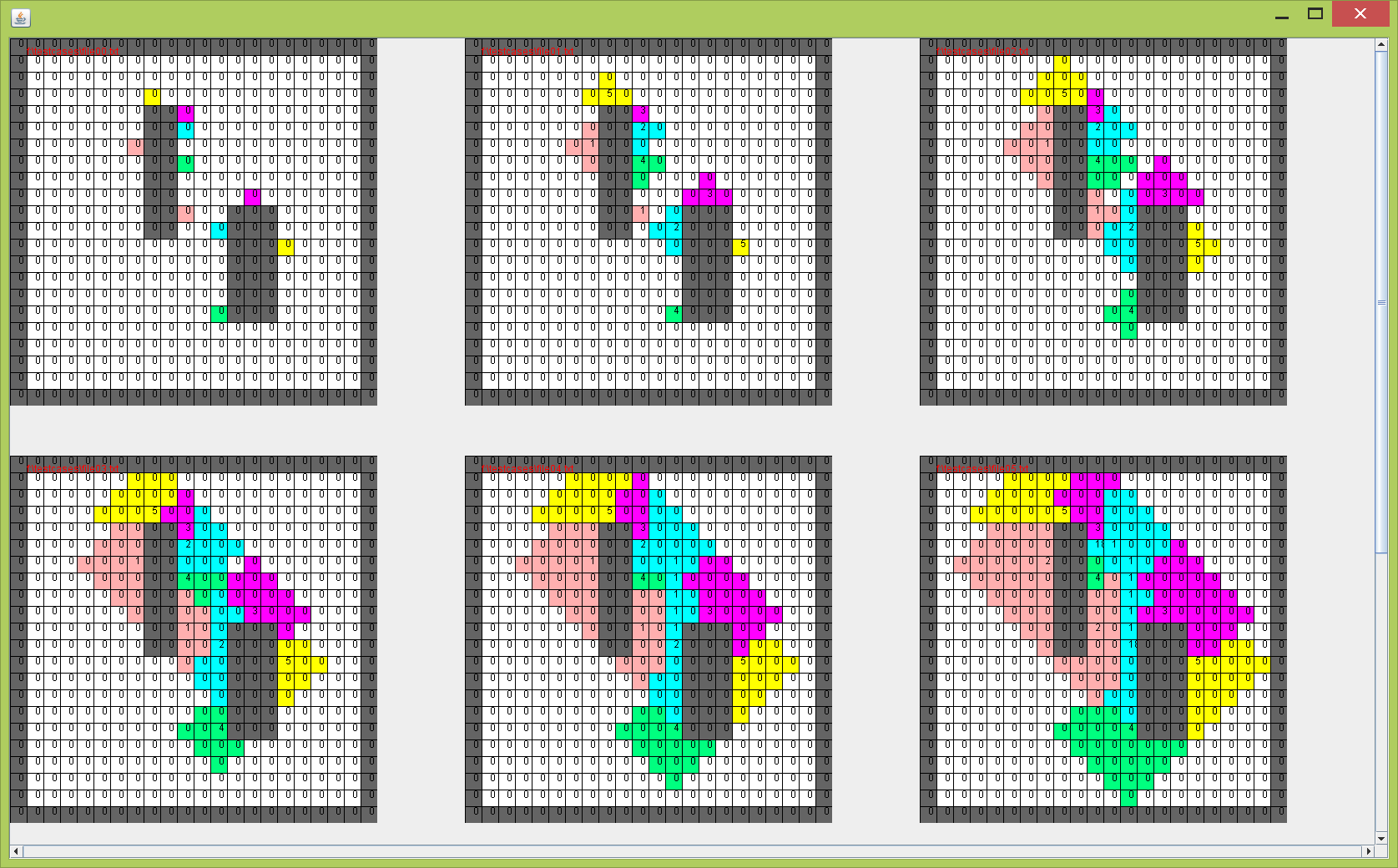
(c)

Figure 3 GR1 Case 3

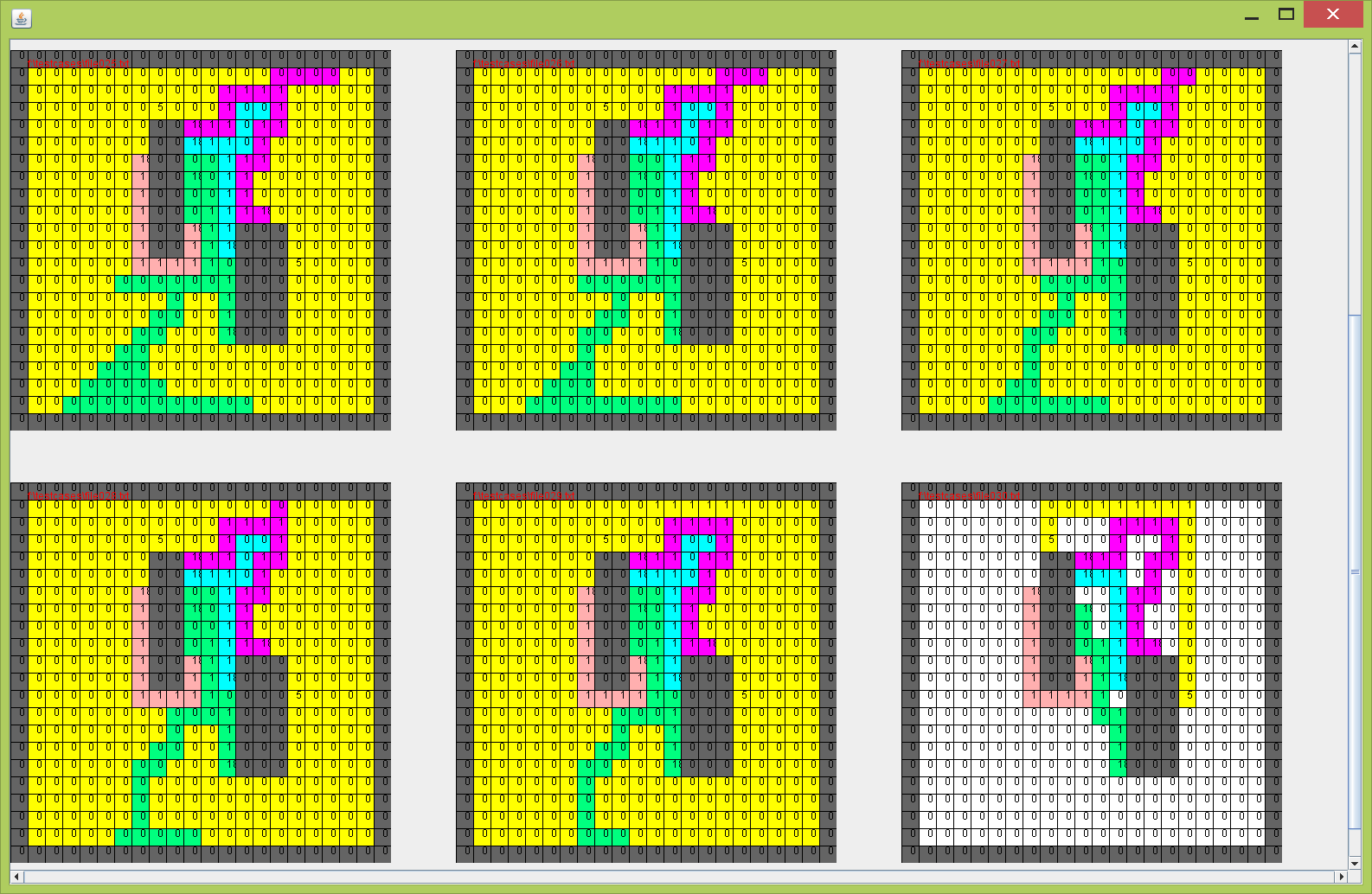
**Hard Cases**

**Case 4:**

In this case, we have 5 different nets and figured out how the nets gotten connected. The yellow net starts from node (8, 3) to node (16, 12). The pink net starts from node (10, 4) to node (14, 9). The blue net starts from node (10, 5) to node (12, 11). The green net starts from node (10, 7) to node (12, 16). The orange net starts from node (7, 6) to node (10, 10). The blue net firstly get connected and then the orange, green and pink nets, and in the end the yellow net gets connected. As the figure 4 shown, our code correctly implements the rules of GR1, which deal with the global and local bottlenecks. The connected nets will never be divided to separate subnets. The final paths are found in figure 4(b).



(a)

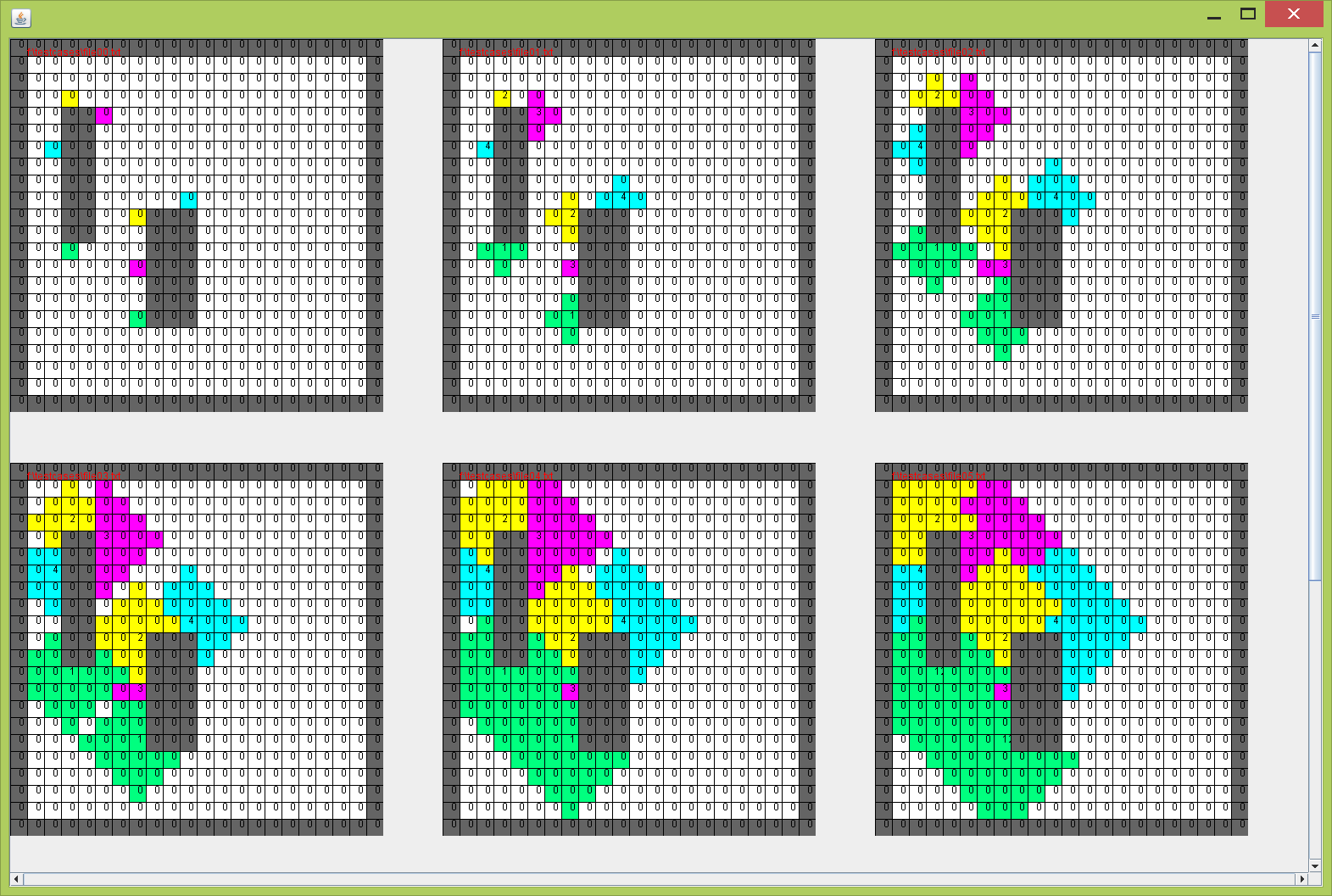


(b)

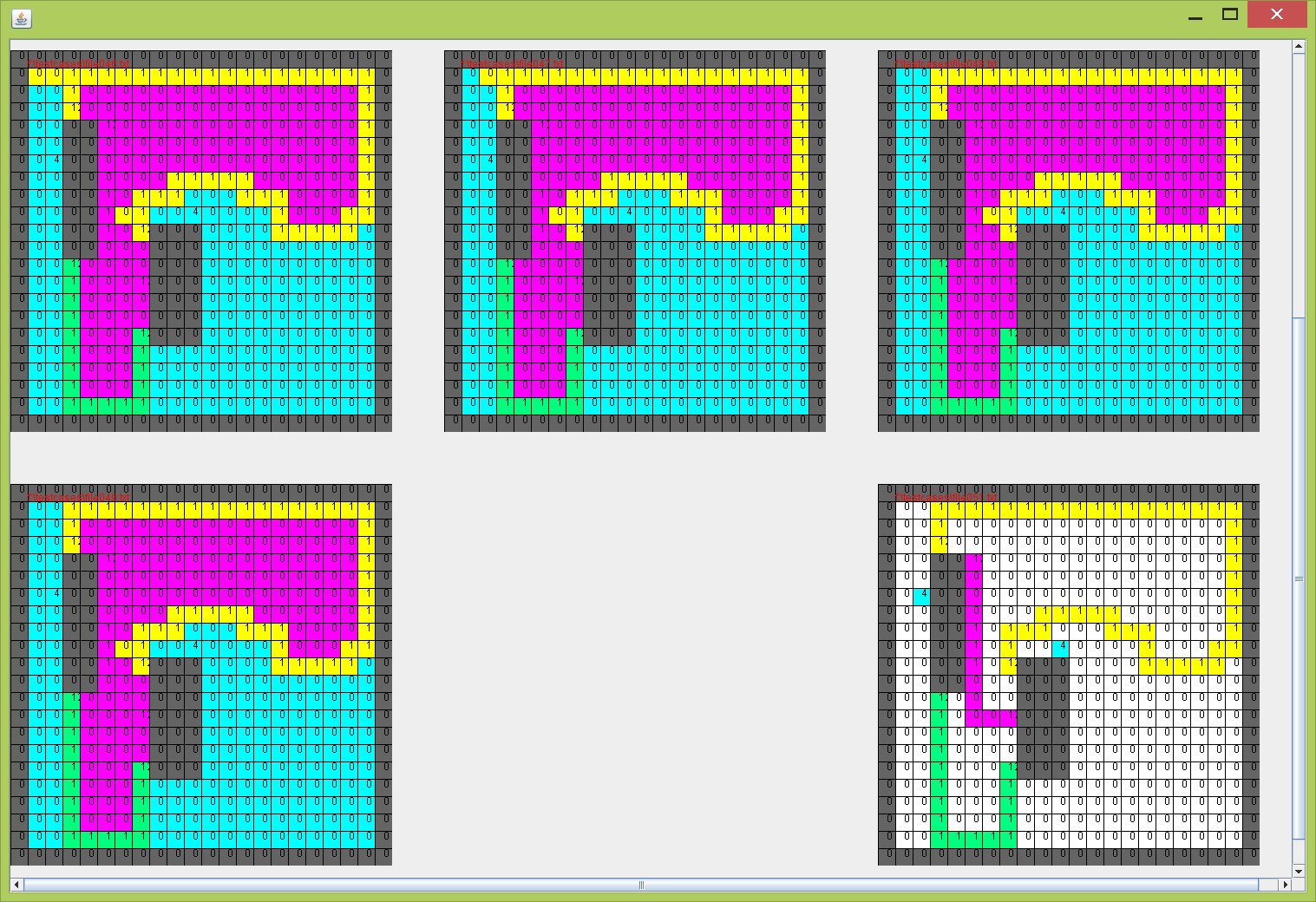
Figure 4 GR1 Case 4

**Case 5:**

This is a special case. We have five nets here. The pink net is from node (5, 4) to node (7, 13). The yellow net is from node (3, 3) to node (7, 10). The blue net is from node (2, 6) to node (10, 9). The green net is from node (3, 12) to node (7, 16). The green net firstly gets connected and then the pink and yellow nets connected. (We skip some steps here as it takes too many steps.) Then the router finds out that there is no path to connect the blue net as all the possible free nets have been blocked by the bottleneck from yellow and green nets. Thus the blue net is impossible to be connected. This case perfectly shows that our code fully implements the rules of GR1 that the connected nets can never be destroyed.



(a)



(b)

Figure5 GR1 Case 5

**3.2 GR2**

**Case 1**

In this case, we have 2 nets. The pink net starts from node (2, 2) to node (6, 5). The yellow net starts from node (2, 4) to node (6, 4). At the beginning of the routing, the yellow net is already connected and block the path for the pink net. However our GR2 Router let the yellow net make a way for the pink net and finally both nets are connected in Figure 6 (b). GR1 can’t solve this case. The number in the maze is the priority according to the rule of GR2 router.

****

(a)

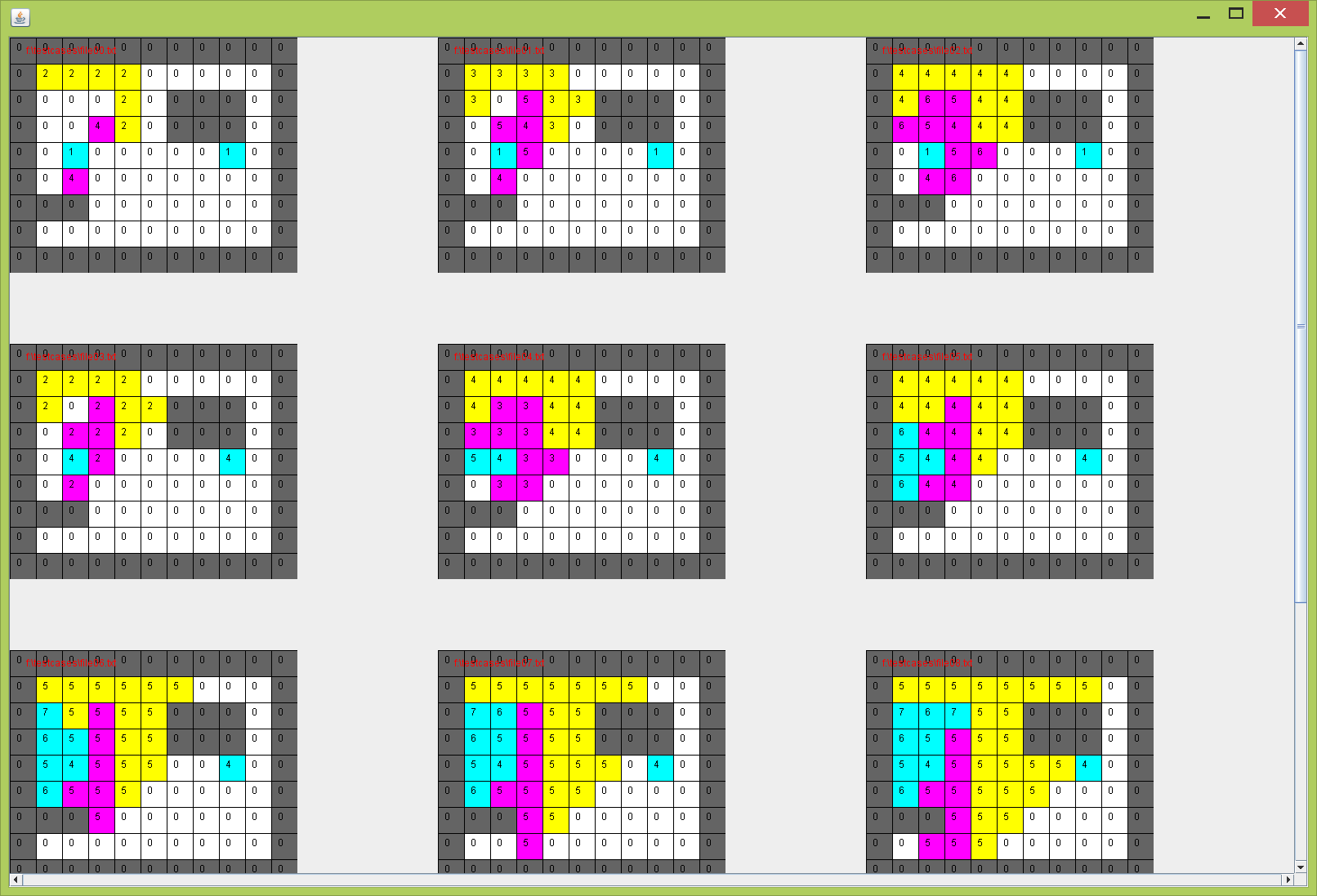
****

(b)

Figure 6 GR2 Case1

**Case 2**

This case has 3 nets. The yellow net starts from node (1, 1) to node (4, 3). The pink net starts from node (3, 3) to node (2, 5). The blue net starts from node (2, 4) to node (8, 4). At the beginning, the yellow net is already connected. Then the pink net gets connected and it blocks the way for the blue net. Then the blue nodes push the pink and yellow net and get connected finally.

****

(a)



(b)

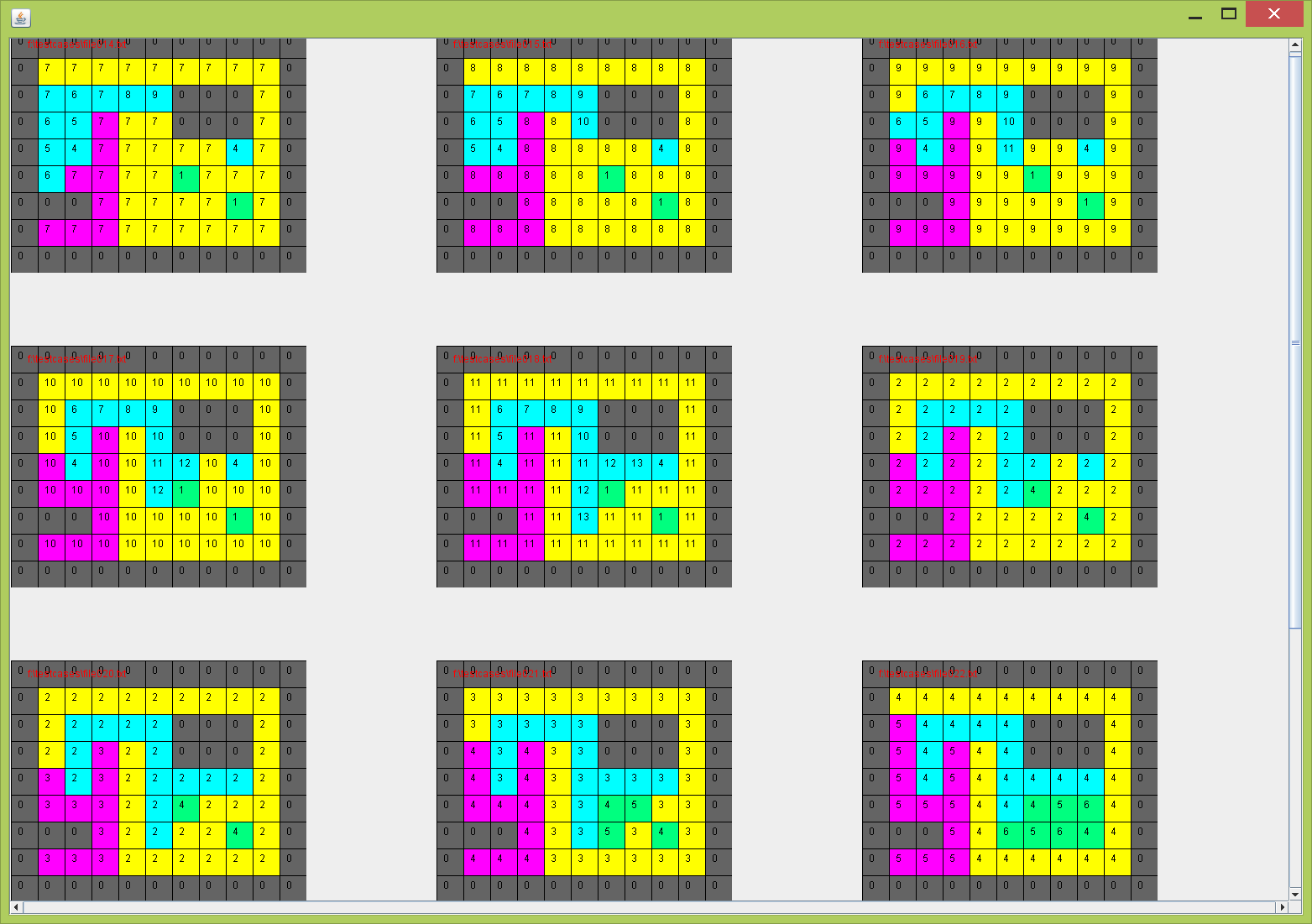
Figure 7 GR2 Case2

**Case 3**

This case is based on case 3 and it has 4 nets. The green net starts from node (6, 5) to node (8, 6). The yellow net is connected at the beginning. Then the pink net gets connected and block the path for blue net. The blue net goes around the pink net and finally gets connected. In the end, the green one gets connected. Each net gets connected one by one according to the rule of GR2.



(a)

****

(b)

Figure8 GR2 Case3

**4. Conclusion:**

In our project, both GR1 and GR2 are implemented. The codes are tested and results are listed in our report. Our GR1 test cases have been divided to 3 levels: easy, intermediate and hard. Each case, to some extent, differently shows the implementation of our code. The test results show that our code fully finished GR1 according to the rules of GR1:

(1) Nets which needed to be connected have higher priority than the one that has been completed

(2) The connected nets should never be broken. In our program, we need to decide which cell is the bottleneck cell and whether it is a local bottleneck or global bottleneck to avoid breaking up a connected net into disconnected areas.

(3) Avoid two subnets have equal priorities.

(4) No nets can enter the pin area of another net.

GR2 enforces the individual subnets one by one to clean up the connection left unfinished by GR1. We have 3 test cases and the results show that the GR2 is able to route the nets properly.

**Reference:**

1. J. Soukup: Global router. DAC 1979: 481-484