**ECE297: Communication and Design**

**Written Document #2:**

**Final Report**

cd096

Shiyao Chen, 1001792474

Margaret Riese, 1001199151

CI: Myra Bloom

April 8, 2016

**1 Introduction**

Geographic Information Systems are a vital link or bridge between people and geographical information they may need. To building this link and effectively communicate this information to people, the team created a visual representation of different urban areas in the form of a city map from existing databases on city information.

To visualize this information, we chose a grey scaled background and bold colors for main features to clearly communicate the most important elements. As you zoom in, more relevant city information becomes visible. Our special feature is a ‘Sort By’ function which organizes points of interest by user preference to help find information they are looking for faster.

Another way to communicate information faster is through technical changes in the code. The technical overview illustrates how our program works as if to a new coder taking over the project. The program has four files corresponding to each milestone. To find routes faster, we fan out from the starting intersection until the destination is found. To improve the speed of the courier function, the program travels between nearest valid delivery points to create a path and then iteratively improves the result.

Through the process of building this project, the team has encountered many challenges, especially in communication, which led to the dissolvement of the team between the current members and another member.

Due to time constraints, the team was unable to implement everything we initially wanted. If given the opportunity, we would enhance both the visualization and functionality by adding elements such as printing travel directions and outlining transportation routes.

**2 Design Overview**

In our State of Art review from our Graphics Proposal we found that the most intuitive and aesthetically pleasing Geographic Information Systems (GIS) were the ones that embraced simplicity. Building on this as inspiration, we tried to make simplicity a major theme in our project. This section will discuss the visualizations we implemented, such as the colour choices we made for various elements of the maps, and the user interface options that make our GIS unique, for example the ‘Sort By’ function which allows users to toggle on and off various types of points of interest (POI).

**2.1 Visualization Overview**

The team researched into marketing psychology and found that certain color combinations were considered more simple, pleasing, and relaxing and may therefore improve the overall user experience of our map[1][2]. The team used a grey scale for the background, the buildings, and most of the other non-interactive information. According to our research, similar colors used in this way are particularly pleasing to the eye[1][2].

The green spaces and bodies of water are green and blue respectively, as in Fig. 1, because it correlates with our existing schemas for those things[3]. This will be consistent with what users already know since it is a convention in mapping; in the same way that on a traffic light, people associate red with stop and green with go. These complementary colors will also stand out against the grey scale underneath.

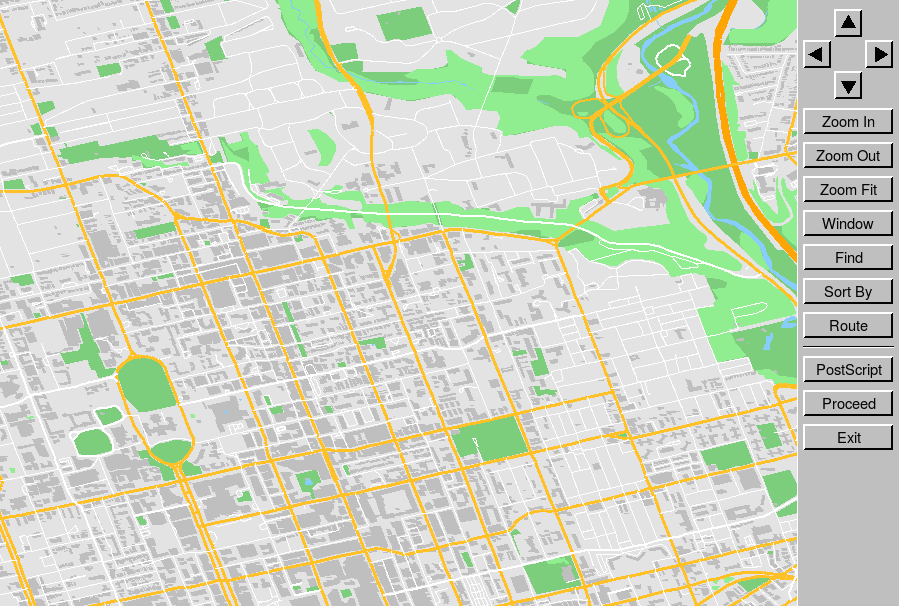


Fig. 1. Overview of the visualization components of the map

POI are marked with black and white circles with red centres, as in Fig. 2, because these colors work in the same way that a computer mouse works. The white mouse with the black outline ensure that no matter what color or surface the mouse lands on it will be visible. The bold red helps attract the user’s eye[4]. In our graphics proposal we had planned to highlight paths for our route finding function by highlighting the whole street in yellow but when we implemented it we realized that we found it relatively hard to distinguish between highlighted roads and non-highlighted roads, as seen in Fig. 3. The team reasoned that if it was difficult for us as the developers to make the distinction then it would also be hard for users, so we use red lines to guide the user down their calculated path, as in Fig. 4.

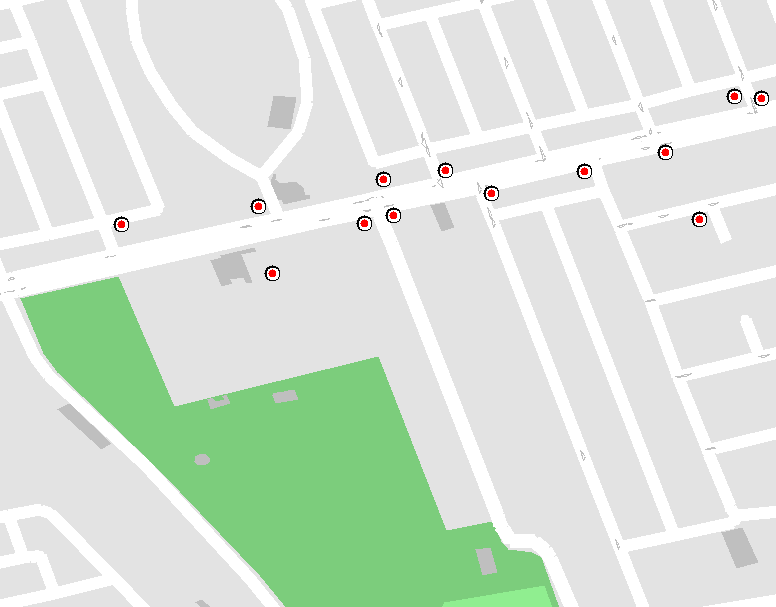


Fig. 2. Points of interest for a particular area of Toronto

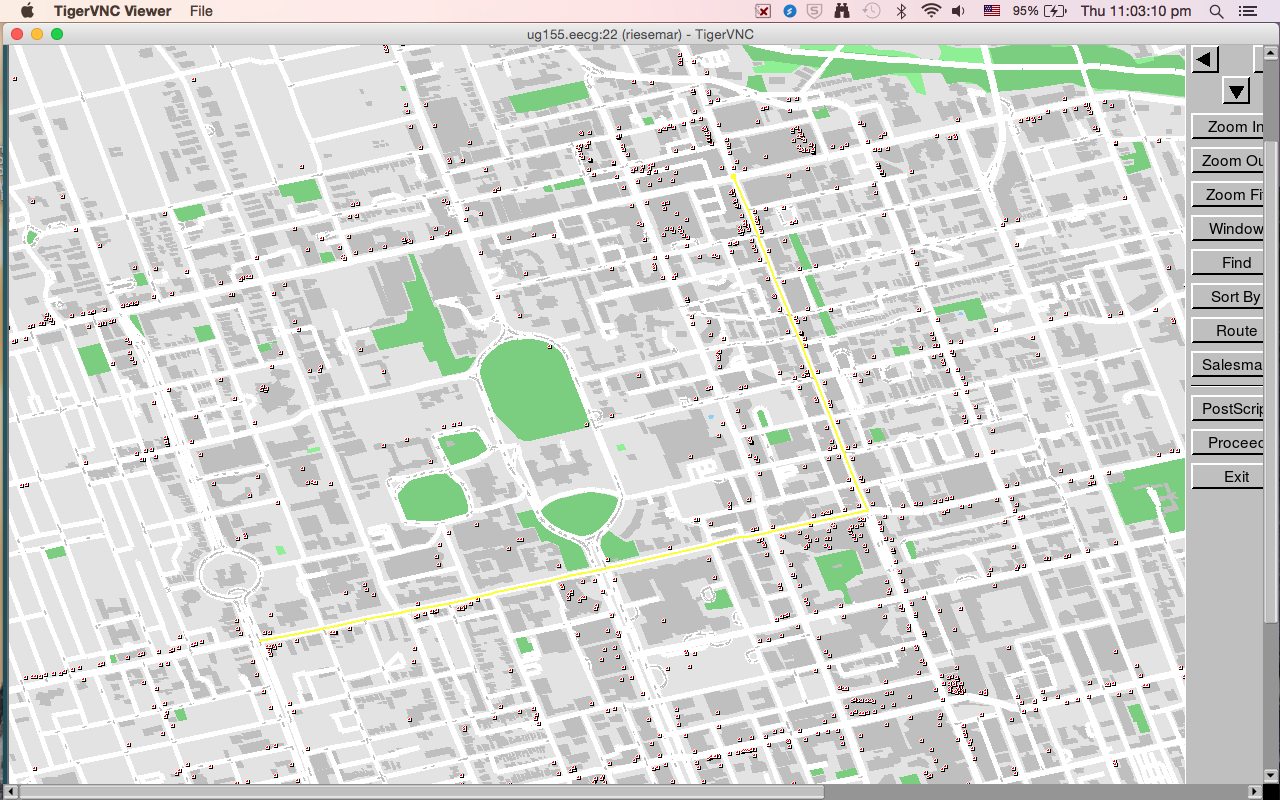


Fig. 3. Original implementation of path highlighting feature

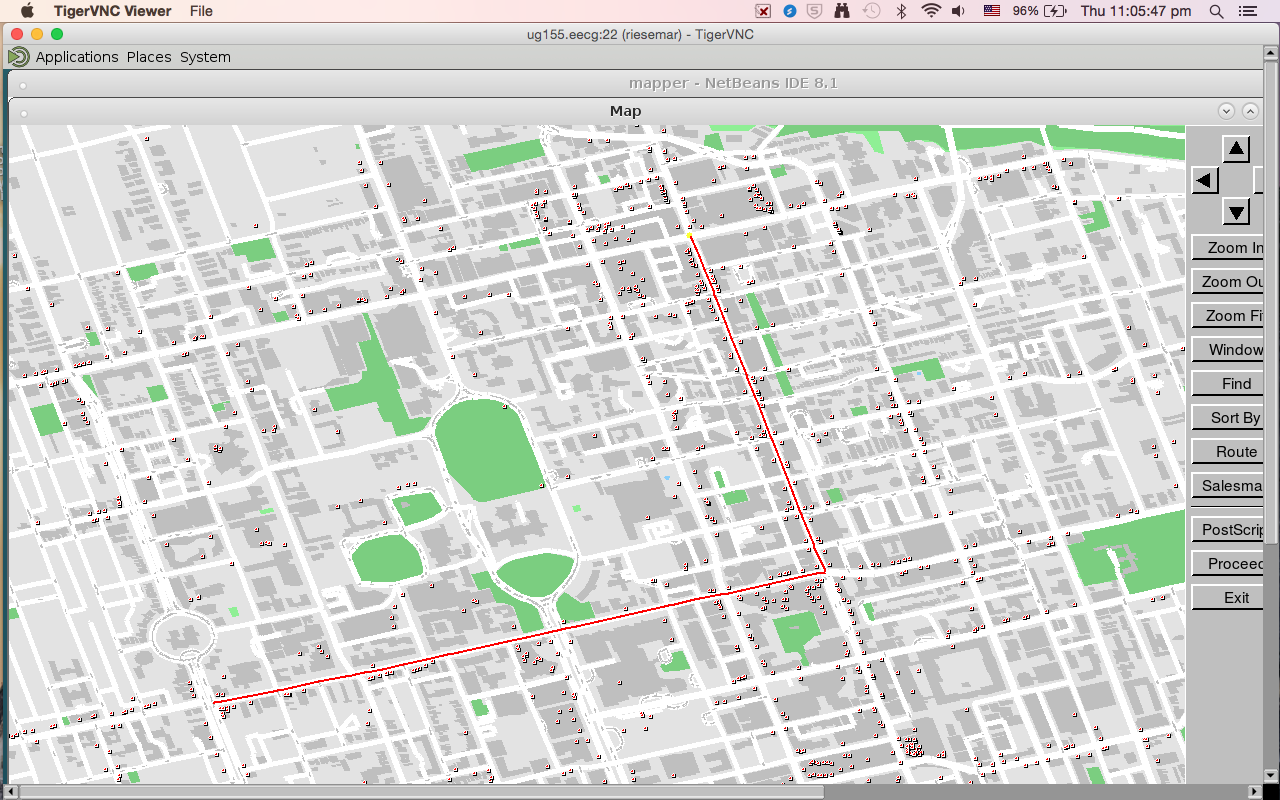


Fig. 4. Final visualization of highlighted path

**2.2 Elements Displayed at Various Zoom Levels**

Adhering to the theme of simplicity, varying amounts of information are made available at different zoom levels to avoid a cluttered look.

Table 1: Details visualized at different zoom levels

|  |  |  |
| --- | --- | --- |
| **Zoom Level** | **Screenshot** | **New Information** |
| 1 |  | Bodies of Water  Green Areas  Buildings  Highways |
| 4 |  | Major Roads |
| 6 |  | Minor Roads |
| 8 |  | POI  Street Names |

**2.3 Methods of User Input and Interactivity**

Fig. 5 shows the buttons that have been implemented to allow the user to interact with the program. Some new functions include: ‘Find’, ‘Route’, and ‘Salesman’. Interactivity is a vital component to the map because of how good of an educational tool it is[5]. Being able to interact with the program is a more concrete way of communicating information[5]. When the user presses the ‘Find’ button they are helpfully directed to type in the name of either a point of interest, or two street names to locate their intersection should one exist. If the user does not know the full name of the street or the point of interest the program will provide suggestions for what they may have meant to be searching for. The intersection or point of interest will then be highlighted to create a metaphorical bridge or link between the information the user wants and the program’s output. When the ‘Route’ button is pressed, a bar at the bottom of the screen will prompt them to click on their origin and destination on the map and the program will find the fastest path between them. The ‘Salesman’ button began as a developers tool to aid the creation of a function that finds an efficient path between pickup and drop off points for a courier company. The function could also be used by the general public for finding efficient ways to make multiple stops on a user’s journey. For example, if a tourist were to use our map they could use it to find efficient routes between multiple tourist spots they would like to visit. We decided to add prompt messages in the bar at the bottom of the screen so that it could enrich the program as yet another way for the user to interact with the program should they so choose.

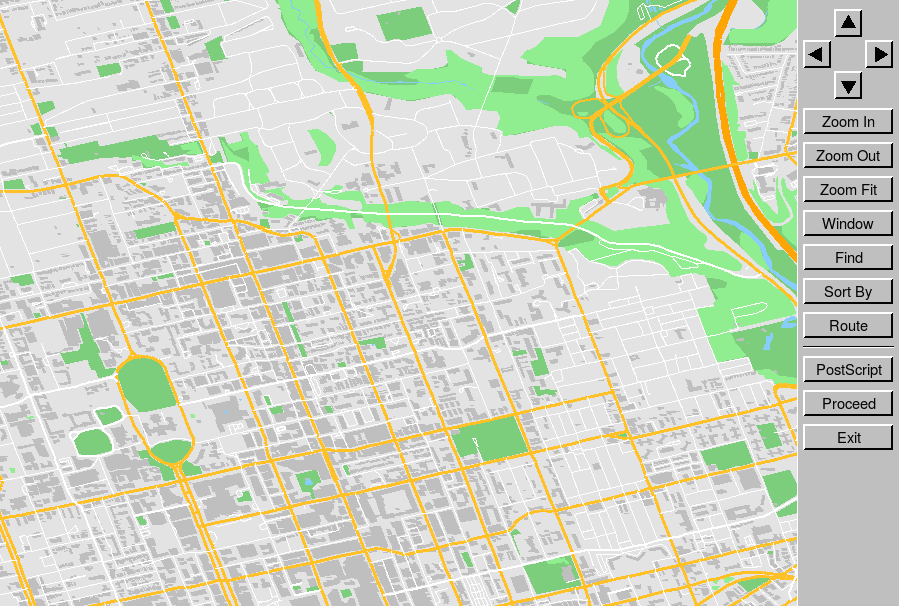


Fig. 5. Buttons implemented for user input and interactivity

**2.4 Special Features**

Fig. 5 also shows a ‘Sort By’ button which sets our program apart from other possible GIS. The team implemented it as a special feature; it allows the user to toggle on and off various types of POI, depending on what they may be looking for, such as restaurants and parking lots, as in Fig. 6. The significance of this function is that it helps communicate to the user the information they are looking for faster, more clearly, and in an interactive and personal way.

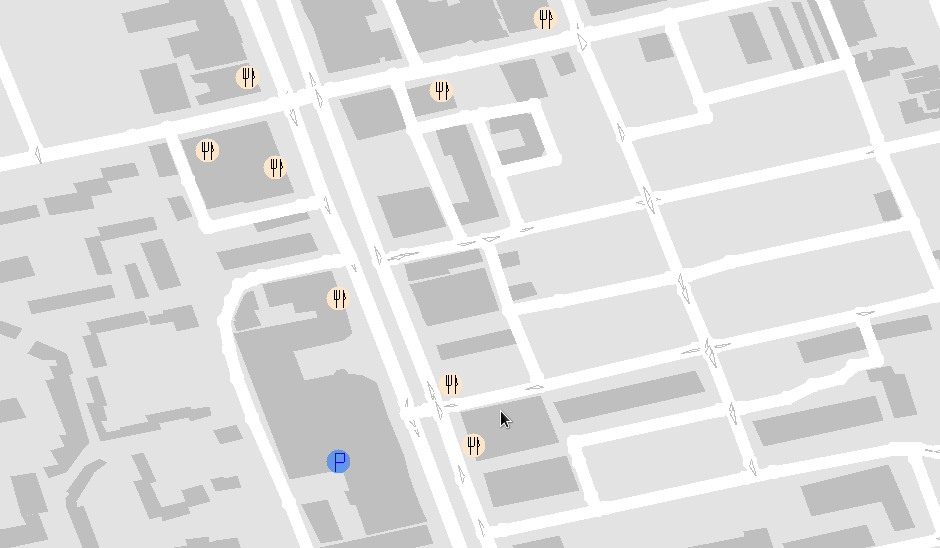


Fig. 6. Restaurants and parking lots for a specific area of Toronto

**2.5 Response Time**

Response time of programs is vital as large numbers of people expect programs to load in two to three seconds or face abandonment[6][7], however, the speed of our map depends on the size of the city. Geographically larger cities take longer to draw than smaller ones. Our program draws Toronto in less than two seconds, while New York, which is considerably larger, can take up to four seconds. The speed with which our program calculates paths between two points also depends upon their geographic distance from each other because the farther apart they are the more possible paths there are between them from which our program must find the most efficient. For example, finding a path from College and Spadina to Bloor and Yonge, as in Fig. 7, takes 0.046 seconds while a journey from one side of Toronto to the other, as in Fig.8, takes 0.058 seconds.The time taken for our program to find the best courier path for deliveries is relative to the amount of deliveries that must be made. The longer the program is allowed to process the information the more efficient the route will be. For example, 25 deliveries in Toronto takes 2.48 seconds whereas couriers with more than 25 deliveries have their paths improved by the program for a maximum of 30 seconds. Refer to Appendix A for confirmation and more detailed response results.

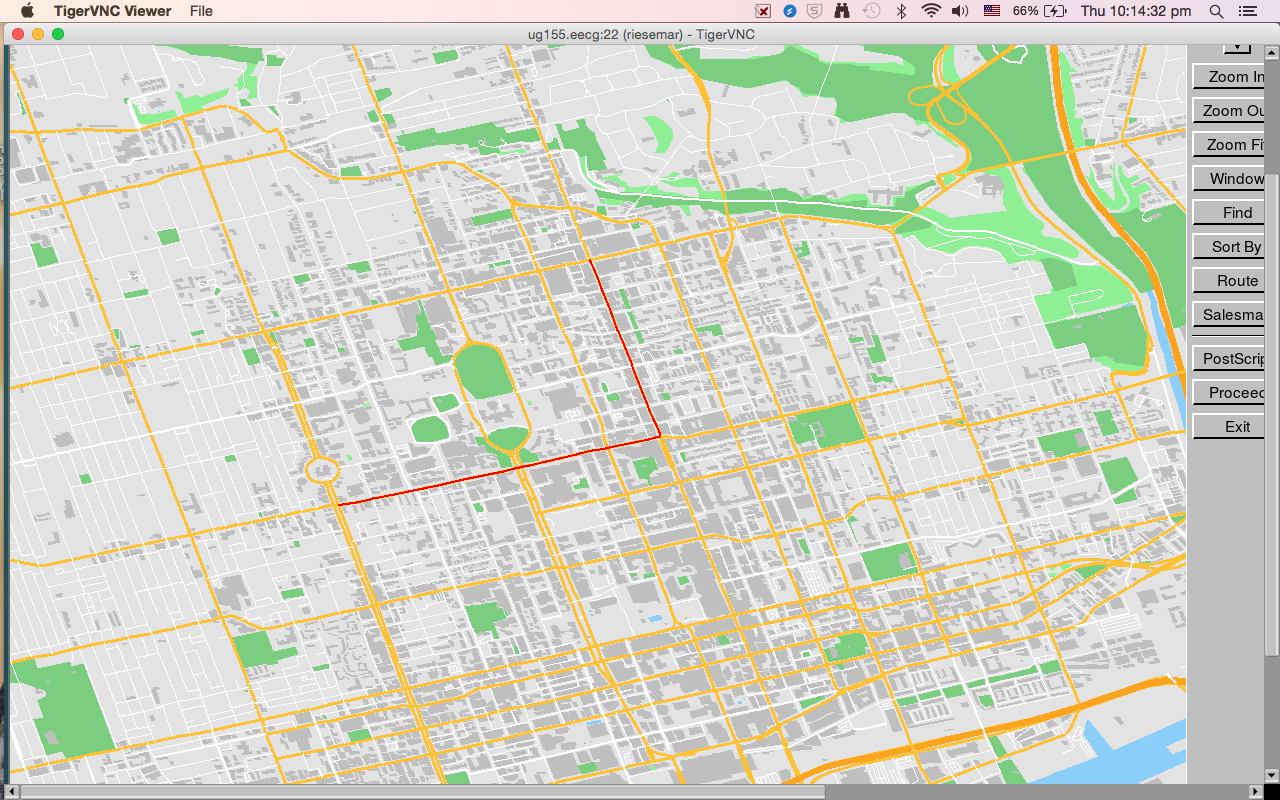


Fig. 7. Path found from College and Spadina to Bloor and Yonge

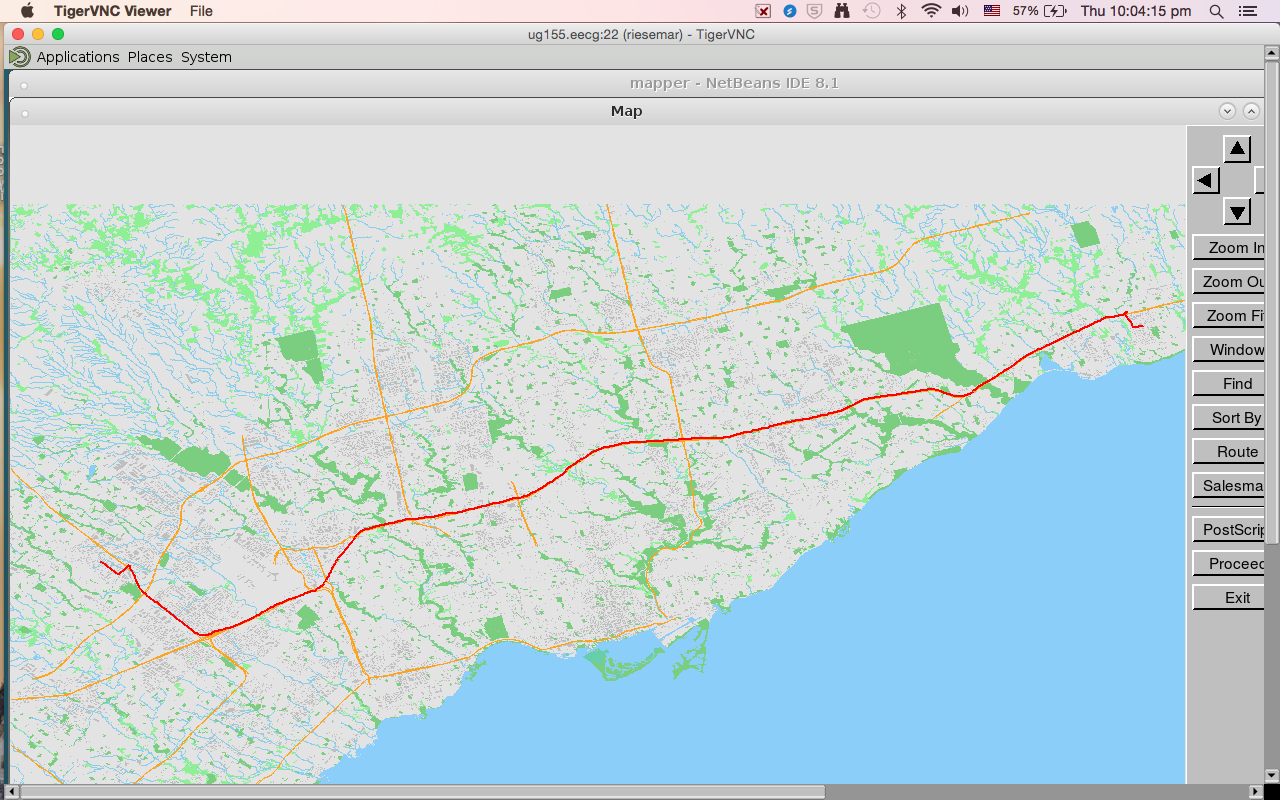


Fig. 8. Path found from one side of Toronto to the other

**3 Technical Overview**

The technical overview will discuss how the team’s code works, the file structure, methods we used to implement complex path finding functions, and major structures that may help a theoretical external coder take over the project with ease.

**3.1 Code Architecture**

The program’s architecture is split into the four source files, and their header files, for each respective milestone. The first (m1) organizes information from the streets databases into data structures for quick access, the second (m2) extracts information from these databases and creates visuals to represent them on the map, the third (m3) finds the quickest path between two points, and the fourth (m4) finds routes for picking up and dropping off multiple deliveries for a courier company.

The main function calls the ‘draw\_map’ function which parses which city the user would like to load and draws it. The ‘m2’ file contains buttons for various functions called either elsewhere in ‘m2’ or, in the case of the route finding and courier functions, in the ‘m3’ and ‘m4’ files respectively.

**3.2 Algorithms**

The team used a breadth first search algorithm with a priority queue for our path finding function and a basic greedy algorithm with an iterative improvement function for our traveling salesman function.

Initially we chose a breadth first search (BFS) algorithm over a depth first search (DFS) because of it potential order (O(2N) at worst for DFS, O(N) at worst for BFS). Originally the wavefront of the BFS was a vector but, when optimizing the function’s speed, we changed it to a priority queue with a new structure for its elements which held their respective cumulative weights or costs, a visited flag, and other information with which to sort them by. Unintentionally, we created something similar to Dijkstra’s Algorithm.

For the ‘traveling salesman’ function a simple greedy algorithm was implemented which always travels to the closest valid destination. An iterative swap function improved the QoR scores. This function iterates through the visited intersections in the path created by the greedy algorithm in a double for loop, swap intersections, check if the new path is valid, check if the new path is shorter than the original, and then replace the original path if both of these conditions are true. We designed this function because it would assess whether small improvements in QoR could be made by rearranging the visiting order similar to how a human would approach the problem. The decrease in average QoR from 1613.43 to 1606.07, as seen in Table 2, solidified the functions importance to the process. The function was repeatedly called in a while loop until a timeout bool was triggered at eleven seconds to allow the function time to push back all the street segments into a vector to return before the time limit was reached.

Originally we assigned our starting depot by choosing the one closest to any given pick up point despite knowing that it may not have yielded the most QoR loss possible. To improve, we used a for loop to apply the greedy algorithm to all of the depots and to only iteratively improve the shortest path among them. Using this function, the team’s QoR score decreased from 1606.07 to 1578.90, as seen in Table 2, therein justifying the function’s relevance to the team.

Table 2: QoR score for different algorithms used

|  |  |
| --- | --- |
| **Algorithms Used** | **Average QoR for Hard and Extreme Cases** |
| Greedy | 1613.43 (Appendix 8D) |
| Greedy + Iterative Improvement | 1606.07 (Appendix B) |
| Greedy + Multistart | 1584.93 (Appendix C) |
| Greedy + Iterative Improvement + Multistart | 1578.90 (Appendix A) |

**3.3 Data Structures**

The team implemented many data structures for various purposes such as creating reference databases between previously unlinked map elements and quickening searches between elements.

Table 3: Relevant Data Structure Information and Purposes

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Structure Names** | **File** | **Contents of Data Structure** | **Purpose of Data Structure** |
| Street\_database  Segment\_id\_database  Street\_database2  segment\_database | m1 | Streets database API information | Quicken search between Streets Database API information elements. Initialized in load\_map. |
| OSMID\_index\_database | m2 | OSMIDs  OSM indexes | Create a quick reference between OSMIDs and corresponding indexes. Initialized in load\_map. |
| Intinfo  Helper  closestIntToPOI | m3 | Intersections wavefront elements (Nodes, waveElems) | Create a quick reference to wave element information (reaching edges, cumulative travel times, visited flags) for particular Intersections. Initialized in load\_map. |
| depotsToPickUps  IntersectionsToIntersections  IntersectionsToDepots  DropOffsToDepots | m4 | Delivery intersections  Depot Intersections | Quicken search between delivery intersections (PickUps, DropOffs) and their nearest delivery intersections. Initialized in initialize function called by traveling\_salesman. |

**4 Lessons Learned and Future Development**

This section outlines the challenges the team encountered while coding the map including building big components of code before required components were functional and dishonesty between team members. Some future implementation possibilities for our map such as improving the ‘Sort By’ function are also outlined.

**4.1 Challenges Faced and Lessons Learned**

Due to the different working habits of each member, the team experienced unexpected conflicts, both technical and communicational, throughout the project. Both coding and communication skills are crucial to deliver each component of a project on time and to the best possible quality. The team’s communication problems resulted in many new challenges.

* ***Technical***

The group had a challenging time with milestone 1 (m1) because of the lack of planning, time management, and the unfamiliarity with the code. In m1, instead of writing the required functions, the team began by setting up structures and databases that we felt were necessary. This was naive because we ended up not using some of the databases we wrote and wasted valuable time. For the next three milestones, the team worked on the algorithms first, making sure the functions met the required functionality before optimizing them. This way, the team saved a lot of time by avoiding unnecessary coding.

Another major setback for m1 was not figuring out how to run the unit tests for a long time. The team wrote all of the functions assuming that they worked without testing them. After running the unit tests, the team found that only some of our functions worked the way we expected. Debugging the code was difficult as some functions were called inside of others. The team realized that the most efficient way to build large projects with many components, is to complete small pieces of code and make sure they works before building on top of them.

* ***Communication and Teamwork***

Through the first two milestones and the Graphic Proposal, the team was faced with many trust and communication issues. We felt that our third member had behaved disrespectfully and dishonestly towards us and that led to the dissolution of the team.

In m1, the team had a miscommunication problem when a member tried to complete functions that were not assigned to him without informing the team. This led to confusion over whether certain functions were completed or not. In response, the team made a rule stating that team members should not change other member’s functions without their permission. However, he broke this rule in milestone 2 (m2) which the team considered very disrespectful.

Another communication problem was his dishonesty towards the other members. He was in charge of displaying street names for m2. Days before the deadline, he told us that the function was complete; however, when we tested the function, it had obvious errors such as street names overlapping and not being oriented correctly. These errors were not fixed by the night before the deadline, and the rest of the members had to stay overnight to fix them.

The team made multiple attempts to improve the group dynamic. For m1 we coded together, but this introduced tension into the team as he disagreed with our methods. We tried working independently, which led to him being dishonest with us about his progress. We organized team meetings where we talked about our frustrations and encouraged him to open up about problems he may have had with our work methods in hopes that it would end to our communication problems. Our attempts failed and our group dynamic only got more toxic. These factors eventually led to the dissolvement of the team.

From this, we learned that some groups are less compatible than others and that one should get to know their potential partners before agreeing to work together. One should attempt to see if team work habits and abilities are compatible. Due to the fact that our team dynamic improved after the previous team was dissolved, if we found ourselves in a similar situation in the future, we would once again try to dissolve the team.

**4.2 Future Enhancements**

Due to poor time management and unexpected communication challenges, there were elements of the map we did not have time to implement that we would like to add to our map as well as some new features.

Enhancing our ‘Sort By’ function, which allows users to toggle on and off various points of interest, would be a priority. Currently, it only displays restaurants and parking lots. The group would add transportation to this function by visualizing public transportation routes. To make the map more user friendly, users should be able to categorize different types of food, making the search more efficient.

The team would print directions for routes as it was not accomplished in milestone 3. Currently, all the information is printed to the terminal, but as more information is printed, the uncleared information in the terminal looks confusing and hard to find information. To address this, information could be printed directly on the map, with buttons to customize what is displayed at any given time.

**5 Conclusion**

Simplicity is a core principle that outlines our city map project. Embracing simplicity, the team adopted a simple color scheme for the visualization and implemented functions to make the user experience effortless. The code was also enhanced to make route finding for single and multiple stops faster. The team faced many challenges throughout the project due time management and communication.

With future improvements, such as displaying transportation routes, the team hopes that the visualization along with the smart algorithms will act like building blocks for the bridge that link the user and the specific geographic informations they are looking for.

**Reference List**

[1] redcuadrada. (2015, June). *Color psychology applied to graphic design* [Online]. Available: <http://www.redcuadrada.com/en/color-psychology-applied-to-graphic-design/>

[2] J. Neidlinger. (2014, July). *Color Psychology In Marketing: The Complete Guide* [Online]. Available: <http://coschedule.com/blog/color-psychology-marketing/>

[3] Help Scout. (n.d.). The Psychology of Color in Marketing and Branding [Online]. Available: [http://www.helpscout.net/blog/psychology-of-col](http://www.helpscout.net/blog/psychology-of-color/)

[4] Business 2 community.(n.d.). *10 colors that increase sales, and why* [Online]. Available: <http://www.business2community.com/marketing/10-colors-that-increase-sales-and-why-0366997>

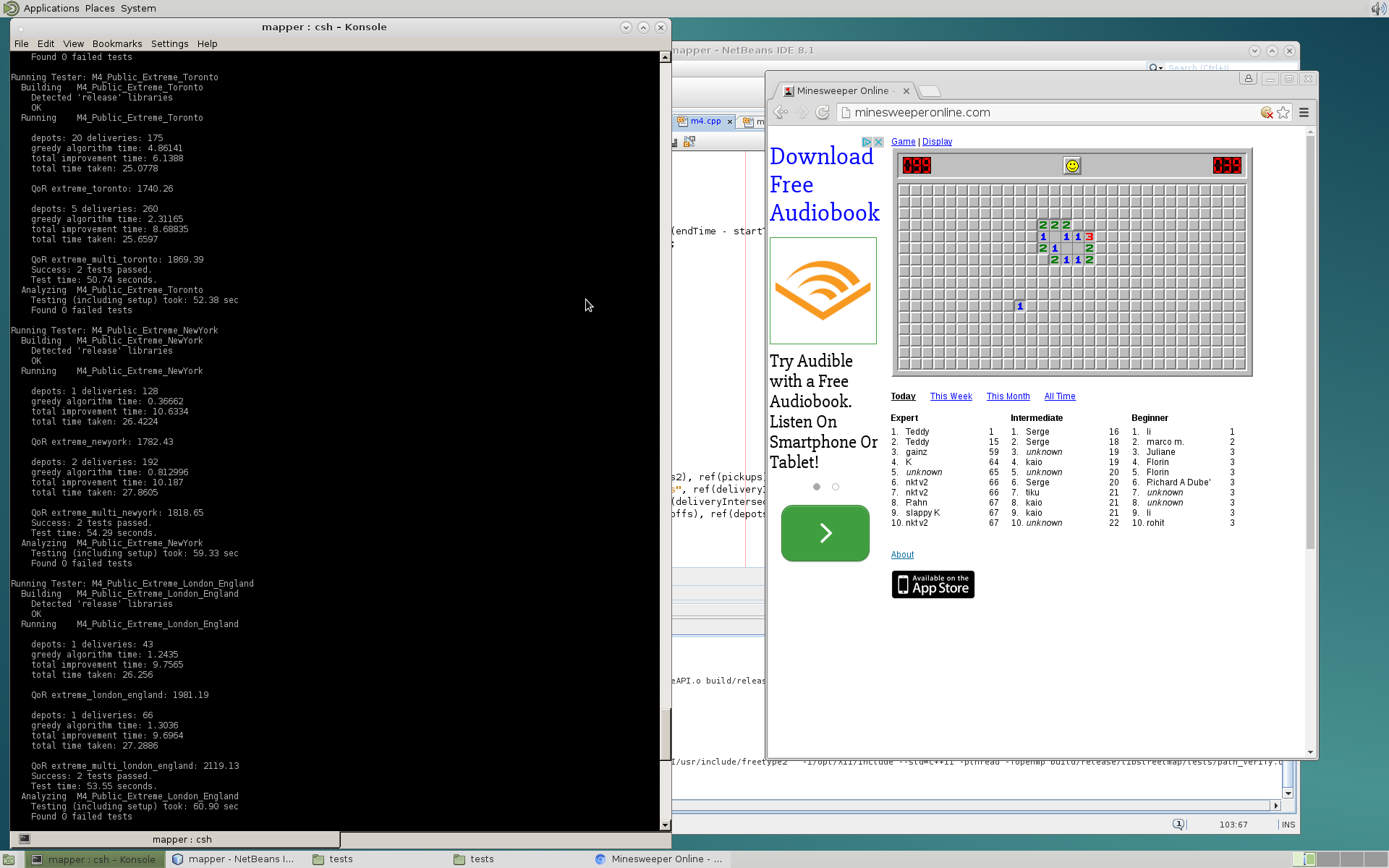
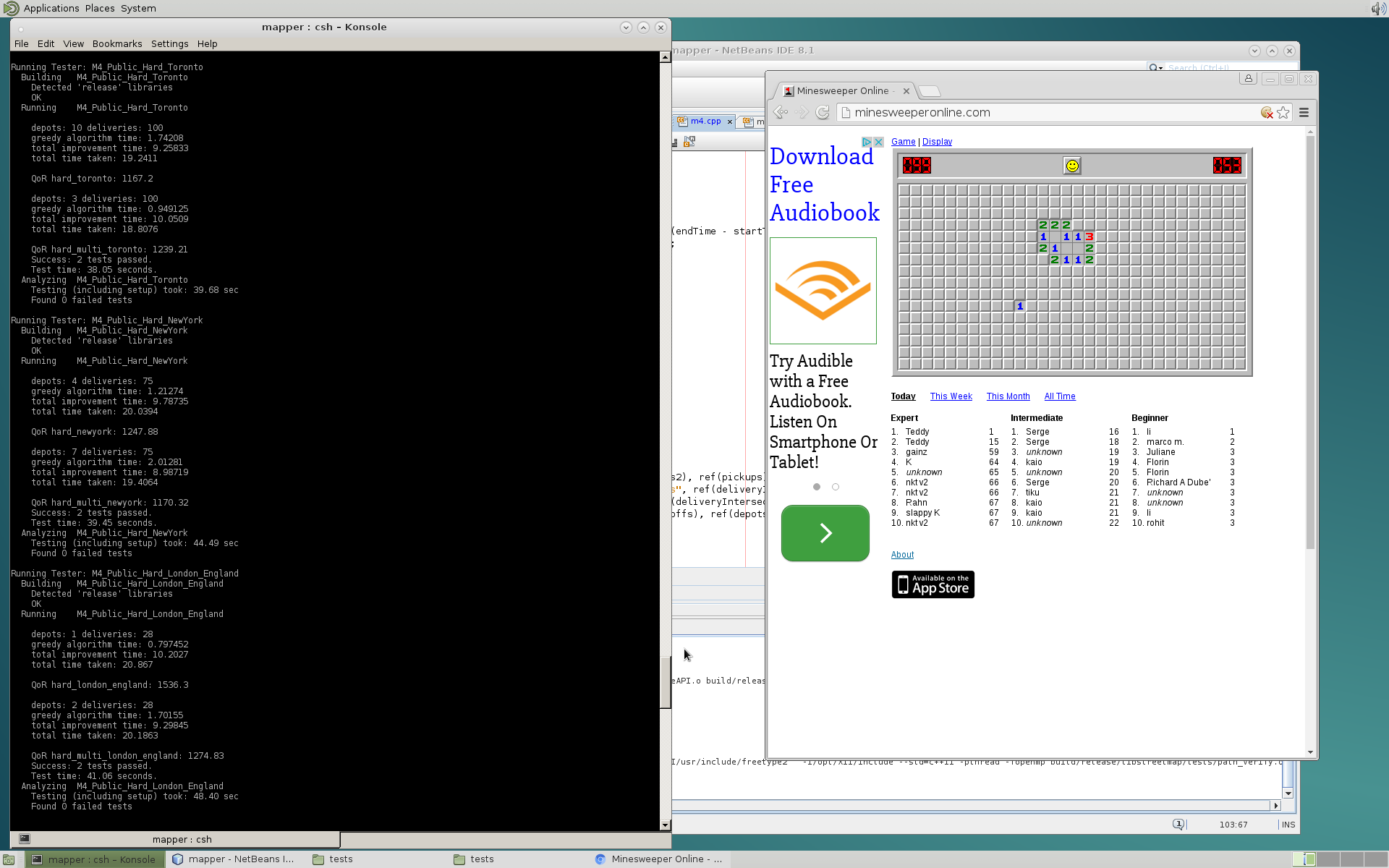
[5] L. Palacios, C. Evans. (2013). *The Effect of Interactivity in e-Learning Systems* [Online]. Available: <https://books.google.ca/books?id=NNQxBwAAQBAJ&pg=PA1&lpg=PA1&dq=interactivity+importance&source=bl&ots=WGH6gUPa83&sig=IHTrKMNyKIsfCNAVi5N_qIixFE0&hl=en&sa=X&ved=0ahUKEwic0_qijv3LAhXFXh4KHeDsDwg4FBDoAQgaMAA#v=onepage&q=interactivity%20importance&f=false>

[6]Kissmetrics Blog. (n.d.). *Speed is a Killer - Why Decreasing Page Load Time Can Drastically Increase Conversions* [Online]. Available: <https://blog.kissmetrics.com/speed-is-a-killer/>

[7]Mos. (n.d.). *Page Speed* [Online]. Available: <https://moz.com/learn/seo/page-speed>

**Appendix**

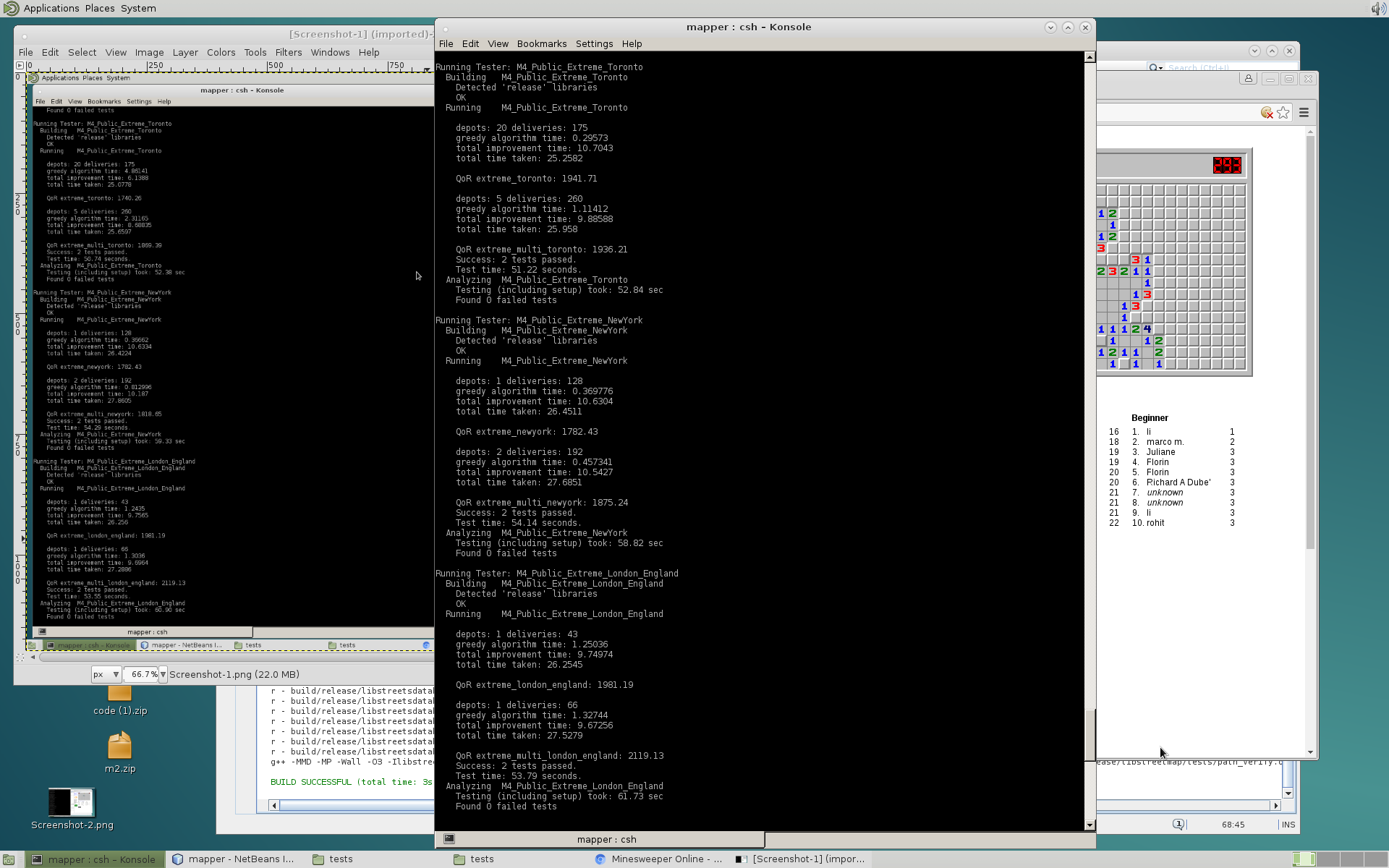
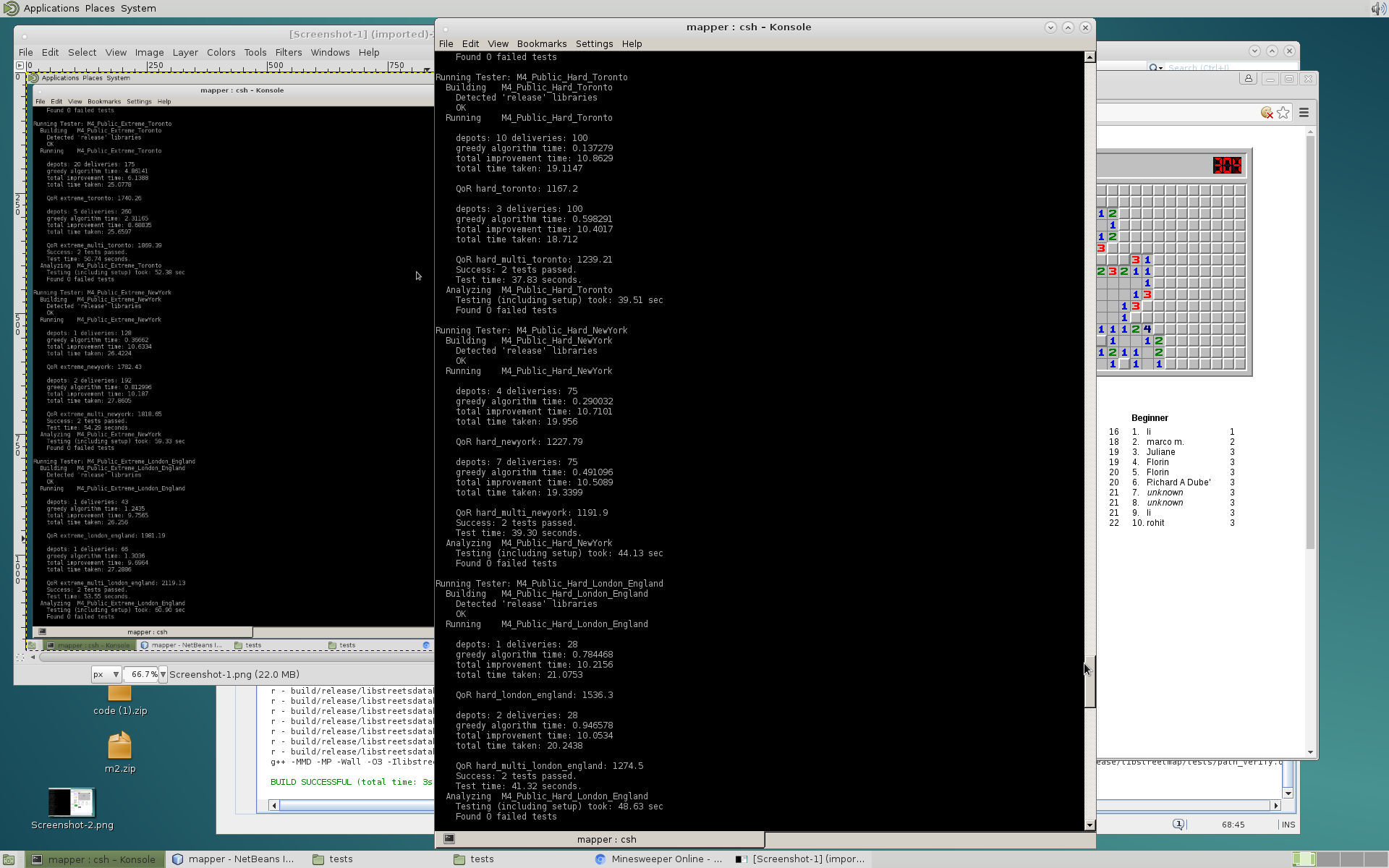
**Appendix A - Screenshots of exercise results with Greedy algorithm, Multistart, and Iterative Improvement for hard and extreme test cases**

****

1. Hard test case results (b) Extreme test case results

Fig. 1. Results for Greedy algorithm, Multistart, and Iterative Improvements for (a) hard test cases and (b) extreme test cases

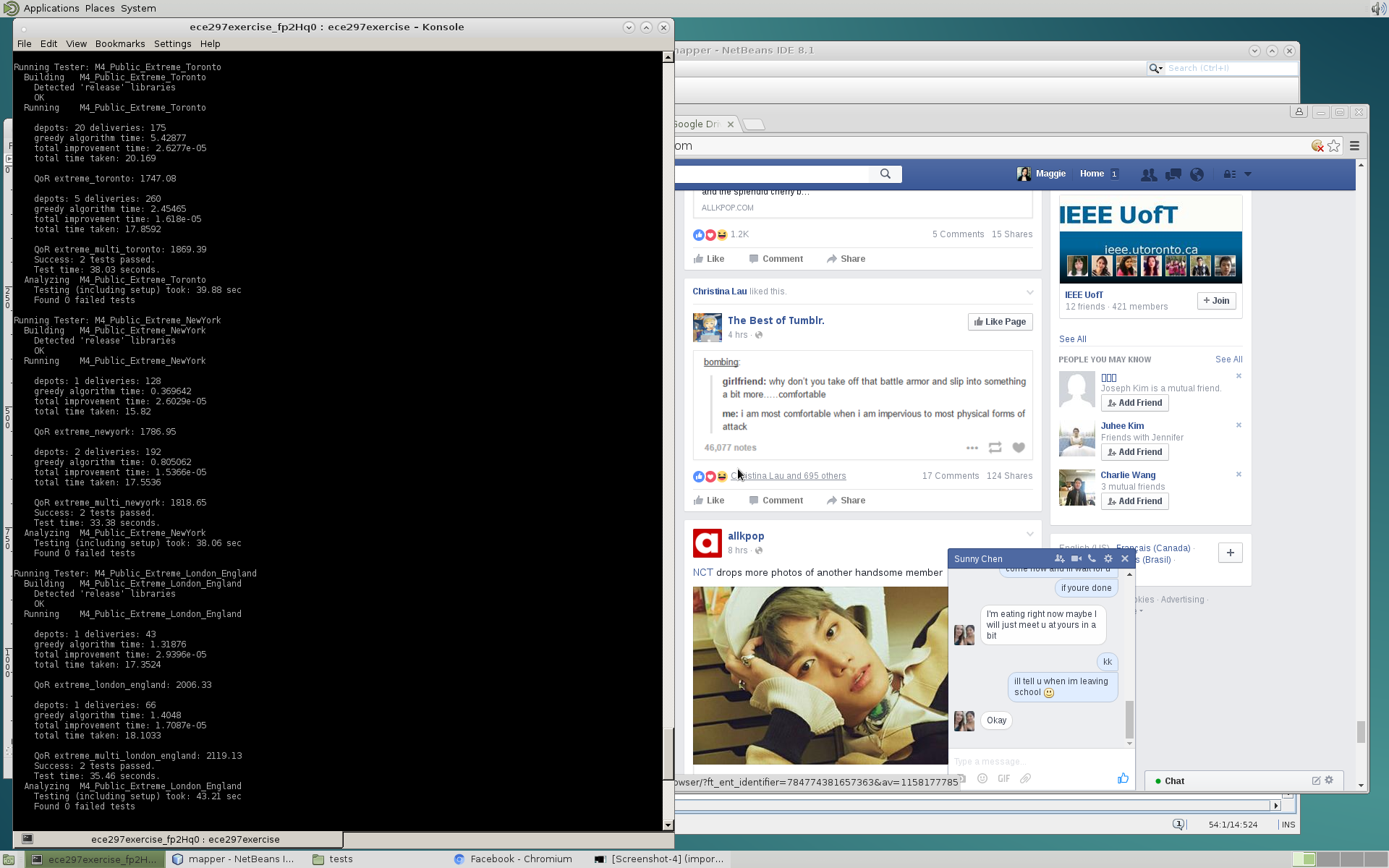
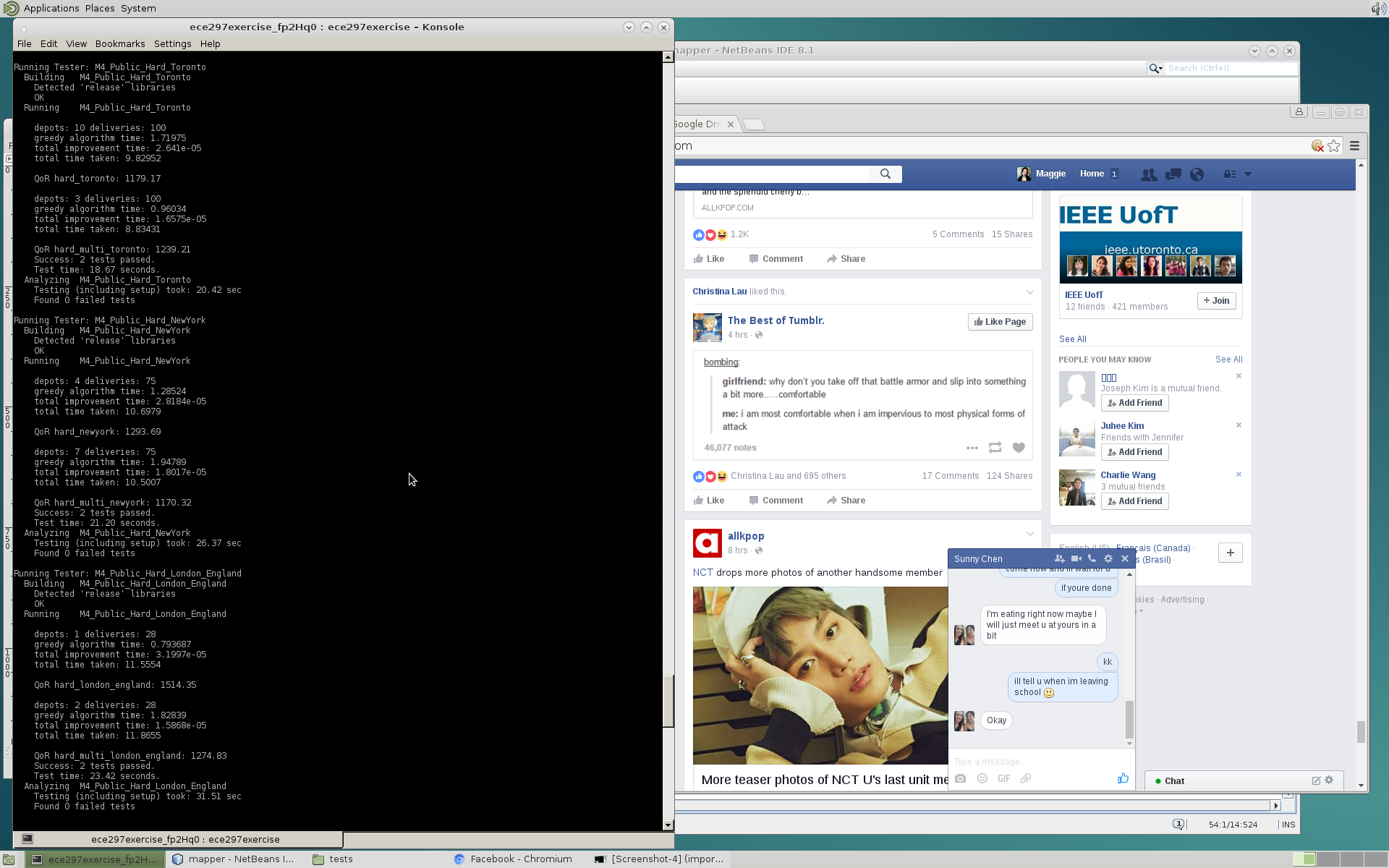
**Appendix B - Screenshots of exercise results with Greedy algorithm and Iterative Improvement for hard and extreme test cases**

****

1. Hard test case results (b) Extreme test cases results

Fig. 2. Results for Greedy algorithm and Iterative Improvements for (a) hard test cases and (b) extreme test cases

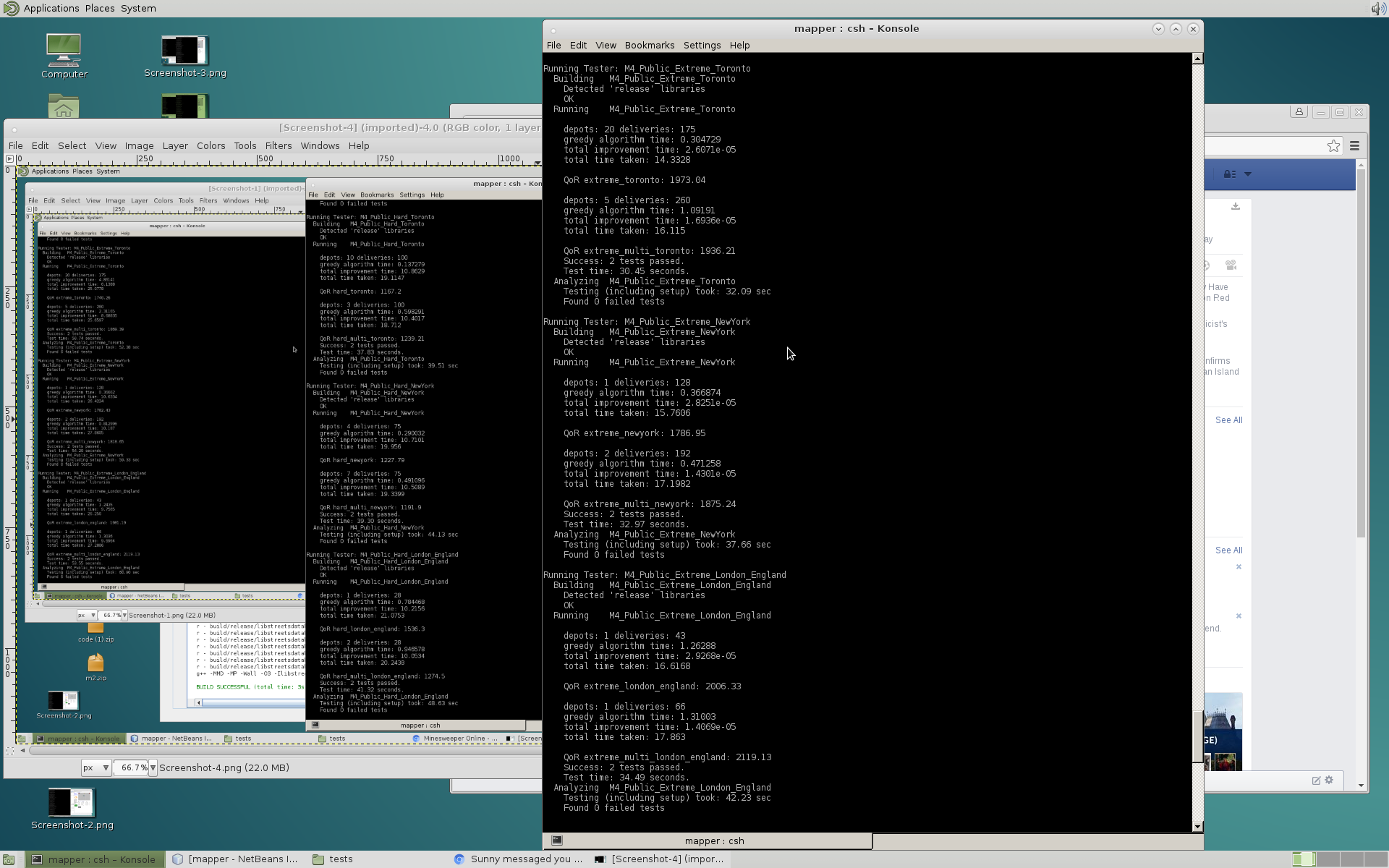
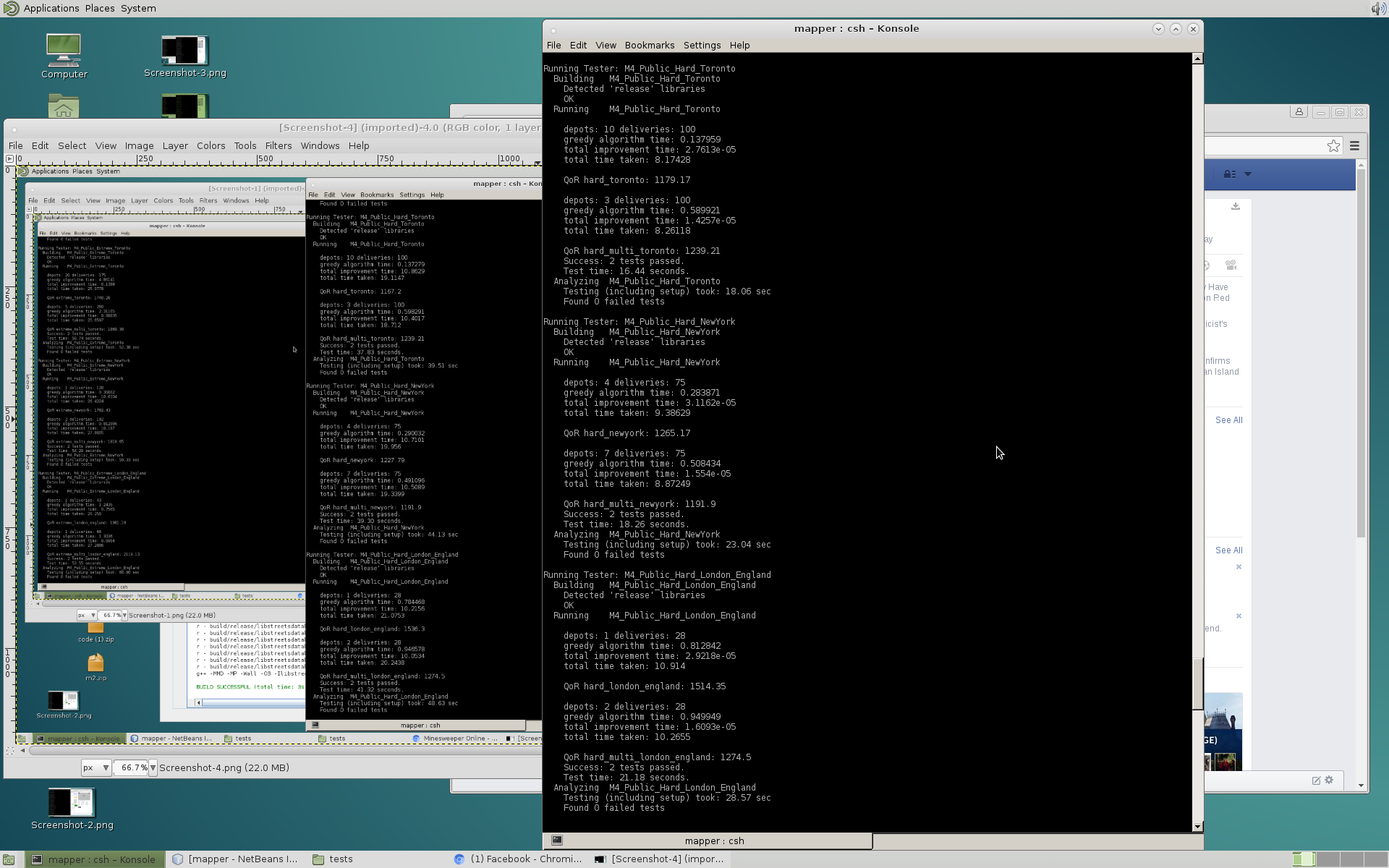
**Appendix C** - **Screenshots of exercise results with Greedy algorithm and Multistart for hard and extreme test cases**

****

1. Hard test case results (b) Extreme test case results

Fig. 3. Results for Greedy algorithm and Multistart for (a) hard test cases and (b) extreme test cases

**Appendix D** - **Screenshots of exercise results with Greedy algorithm with no improvements for hard and extreme test cases**

****

1. Hard test case results (b) Extreme test case results

Fig. 4. Results for Greedy algorithm for (a) hard test cases and (b) extreme test cases

**Attribution Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Research** | **Writing** | **Editing** | **Revision** |
| **1. Introduction** | Shiyao  Margaret | Shiyao  Margaret | Shiyao  Margaret | Shiyao  Margaret |
| **2. Design Overview** | Shiyao | Margaret | Margaret | Shiyao  Margaret |
| **3. Technical Overview** | Shiyao | Margaret | Margaret | Margaret |
| **4. Lessons Learned and Future Development** | Shiyao | Shiyao | Shiyao  Margaret | Shiyao  Margaret |
| **5. Conclusion:** | Shiyao | Shiyao | Shiyao  Margaret | Shiyao  Margaret |
| **Reference List** | N/A | Shiyao | Shiyao | Shiyao |
| **Appendix** | N/A | Shiyao  Margaret | Shiyao  Margaret | Shiyao  Margaret |