Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019

Subdue Web Interface and GUI

Rohan Shah and Lawrence Holder
rohan_y_2000@yahoo.com, holder@cse.uta.edu

Technical Report CSE-2004-9

Subdue Web Interface and GUI

Rohan Shah and Lawrence B. Holder
Department of Computer Science and Engineering
University of Texas at Arlington

November 2004

Abstract

Subdue is a graph-based relational learning system. We are attempting to create two other methods to use Subdue. The Web interface allows users to access programs of Subdue as a service, where they can submit their jobs. When the program completes, results are emailed to the users. The Subdue GUI is an easy-to-use program that allows users to use Subdue programs through a graphical user interface created with Java Swing. This paper describes these systems. This is a basic version, and enhancements can be done as the GUI or Subdue evolves.

1 Introduction

An easy user interface not only improves the customer experience, but also accelerates the learning curve. Different interfaces are suitable for different applications. It also depends on the target audience. If your target audience is experienced in the domain, faster access and simple interfaces are suitable. While if the audience is new to the domain, easy-to-use user interfaces with extensive help are suitable. Moreover, since most of the operating systems come with some kind of windowing systems, use of Graphical User Interfaces (GUI) is becoming more popular.

Subdue [1] is graph based data mining system. Subdue provides a command-line interface. There is separate HTML documentation that comes with the package. Subdue is distributed with source code and uses the MAKE utility for producing binaries. We have created two other methods or user interfaces to access Subdue. Besides the usual command-line mode, it is now possible to access Subdue by a web browser or using a GUI. The whole idea is to make using Subdue easy. Both the web version and the GUI version have extensive help available at the click of the mouse. This paper describes the various features of these two modes. We have only made certain features of Subdue accessible using these interfaces.

2 Subdue Data Mining System

Graph-based data mining techniques are becoming more popular. The graph representation is the most explicit way to express relationships within data. Moreover, graphs can model most of the data. However, they are more widely used in cases where data has explicit or implicit structural components. When data is in the form of graphs, data mining becomes discovery of commonly occurring interesting patterns. However, different graph-based data mining systems use different criteria for determining interesting substructures.

Subdue [1] is relational learning system. The work on Subdue is one of the pioneering works in the field of graph-based data mining. Subdue discovers substructures that appear repetitively in a graph database. Inputs to the Subdue system can be a single graph or set of graphs. The graphs can be labeled or unlabeled. Subdue outputs substructures that best compress the input dataset according to the Minimum Description Length (MDL) principle.

The MDL [2] Principle states that the best description of a dataset is the one that minimizes the description length of the dataset. In this context, the best substructure is the one that minimizes the Description Length (DL) of the dataset compressed with the substructure. The DL of the input dataset can be calculated using the formula I(S) + I(G|S), where S is the substructure used to compress the dataset G. I(S) and I(G|S) represents the number of bits required to encode S and the dataset G, after being compressed by S.

The main discovery algorithm is a computationally-constrained beam search. The algorithm begins with a substructure matching a single vertex in the graph. Each substructure is incremented by edge-expansion. Substructures are then evaluated. The length of the search beam defines the number of substructures retained for the next iteration. This procedure repeats until all substructures are considered. Subdue then returns the best substructures.

There are three different methods for evaluating candidate substructures:

1. Minimum Description Length: The value of substructure S in graph G is value(S, G) = DL(G)/(DL(S)+DL(G|S) where DL is description length in bits and G|S is graph G compressed with substructure S.
2. Size: The value is now size(G)/(size(S)+size(G|S) where size(G) = (# of vertices(G)+# of edges(G). This method is faster that MDL but has less consistent behavior
3. Set Cover: The value of substructure S is the sum of the number of positive graphs containing S and the number of negative graphs not containing S, divided by the total number of examples. This method is used for supervised learning.

During the next iteration, data is simplified by replacing instances of the discovered substructures with a pointer to the definition of the newly discovered substructures. This creates a hierarchical description of structural data in terms of discovered substructures. The graphical representation of a sample input and one of the substructures discovered by Subdue is shown in figure 1.1.

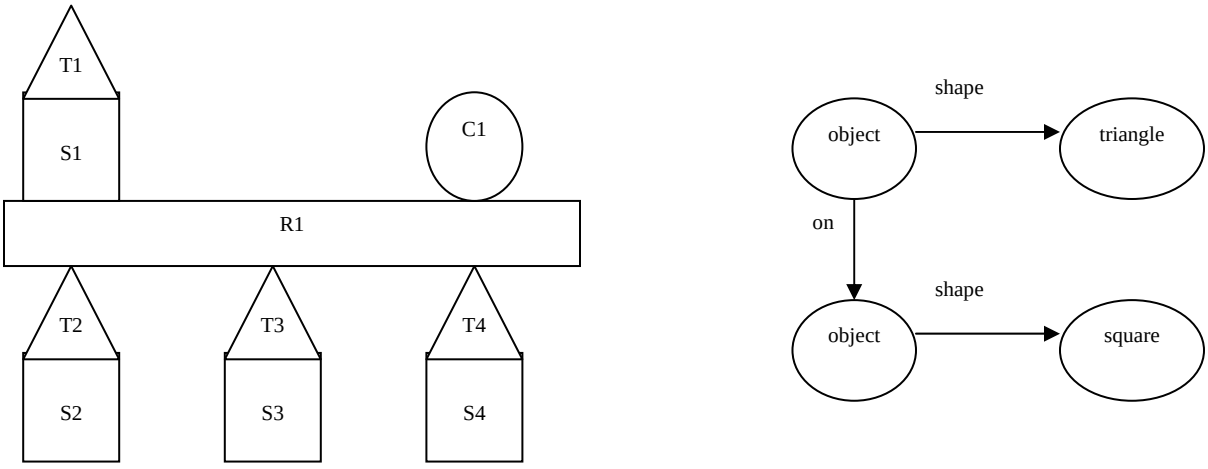

Figure 1.1: Subdue sample input (left) and discovered substructure (right).

3 Subdue Web Interface

3.1 Aim

With the increasing popularity and reach of the internet, many systems are using the internet as an alternate access mode. One example is the use of web services. More and more people are using the internet, and they are comfortable with form-based interfaces for accessing an application.

The Subdue Web Interface provides a similar form-based interface for accessing Subdue. Users can submit their data mining jobs through the internet. Users can select the data they want to send by using a simple "Open File" dialog. The results of their submitted jobs are then sent to the email address specified by the user. We request the user to submit their name and email address. These details are stored in our job log with the specifics of the jobs submitted. However, this is only for research purposes and is not disclosed in an unauthorized manner.

By making Subdue available through the web, we want to increase Subdue's accessibility. We also want to target audiences that would like to try Subdue and become aware of various features of Subdue without actually downloading the system. We have made the following programs of Subdue available through the web:

1. Subdue: The Subdue discovery system finds interesting and repetitive subgraphs in the input data.
2. Inexact Graph Match: This program computes the cost of transforming the larger of the two input graphs into the smaller according to transformation costs.
3. Minimum Description Length: The program calculates the description length of graph 1 (g1), graph 2 (g2) and g2 compressed with g1, along with final MDL compression measures.
4. Subgraph Isomorphism: This program finds all the instances of graph 1 (g1) in graph 2 (g2).

All the options that are available to these programs are also available through the web form. Moreover, there is also help associated with each option.

3.2 Tools and Technologies

Following is the list of different technologies we used for developing the Subdue Web Interface:

1. HTML: Hyper-Text Markup Language, the lingua-franca of the web, was the obvious choice. Since users need to submit the jobs and collect information in the process, we used HTML FORMS. Each program's interface was a form.
2. JavaScript: Users can select the different systems using JavaScript. Moreover, JavaScript was also used for some client-side verification before jobs are actually submitted.
3. PERL-CGI: Since we need to process the data submitted by user, we had to use server-side scripting. We used PERL for this purpose. PERL is easy to learn and has a syntax very similar to shell scripting. Moreover, it has very powerful language processing capabilities. The CGI module that comes with all distributions of PERL was used to process the form data. Some of the advantages of using PERL-CGI over plain PERL are:
    a. In-built functions to process form data. These functions are easier to use and less tedious then actually parsing the environment variables.

b. Function to generate HTML. These functions have unusual syntax but are very powerful. They create HTML code based on arguments you provide. Once mastered, they can be handy to use.
c. Functions to handle different form encodings. Users select the input data to process. These data is first stored at the server and then submitted as a SUBDUE job. Therefore, we had to encode form data in "multipart/form-data" format. PERL-CGI has functions that make it easy to decode and store this data.
d. It is possible to interact with system commands from your program.

4. Apache HTTP Server: All the PERL scripts were hosted on an Apache HTTP Server.

3.3 Installation

The Subdue Web Interface package comes with HTML files that form the interface and CGI-PERL files that do all the work like communication with Subdue, sending results to the user, etc. You need to have a CGI-enabled web-server run the Perl scripts. We will describe the installation process, assuming a directory structure similar to the Apache HTTP Server.

All the HTML files go into the "html" directory or other directory of your choice on your web-server. And all the scripts go into the "cgi-bin" or "bin" or any other directory with execute permission. Directories where you store html pages and scripts should be at the same level, i.e., should have a common parent directory.

You also need to have the Subdue "bin" directory (with all Subdue executables) in the same parent directory. Also, create an empty directory called "proj". This directory is used to maintain the log of all the jobs submitted. The details relevant to the jobs that are submitted but not finished are also kept in this directory.

Finally, we use the "mutt" email program to send an email. Please check if you have this program installed.

3.4 Using the System

This section will walk through the steps of using the Subdue Web Interface system.

1. The Subdue Web Interface provides all the options needed when accessing the regular Subdue. Each parameter of Subdue is now a form member. Each of the members has its default value assigned. For example, Beam Size is set to 4 and Number of Subs is set to 3. "?" in the third column provides information about that option. It is also possible to navigate to other programs using the drop-down menu as depicted in the Figure 3.1.

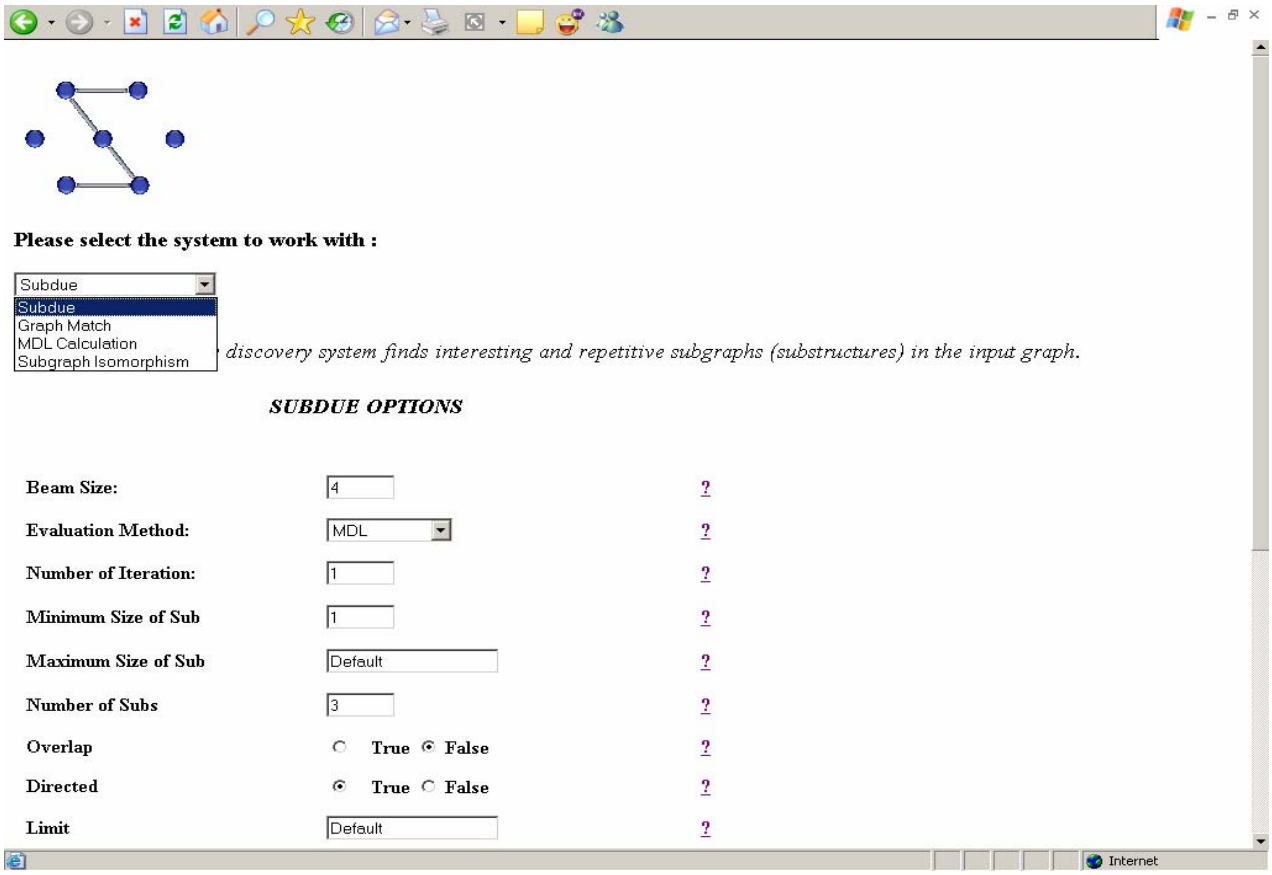

Figure 3.1 Subdue Web Interface

2. The help window gives the user information about that particular option. The Figure 3.2 illustrates the help window for the “Predefined Substructures” option.

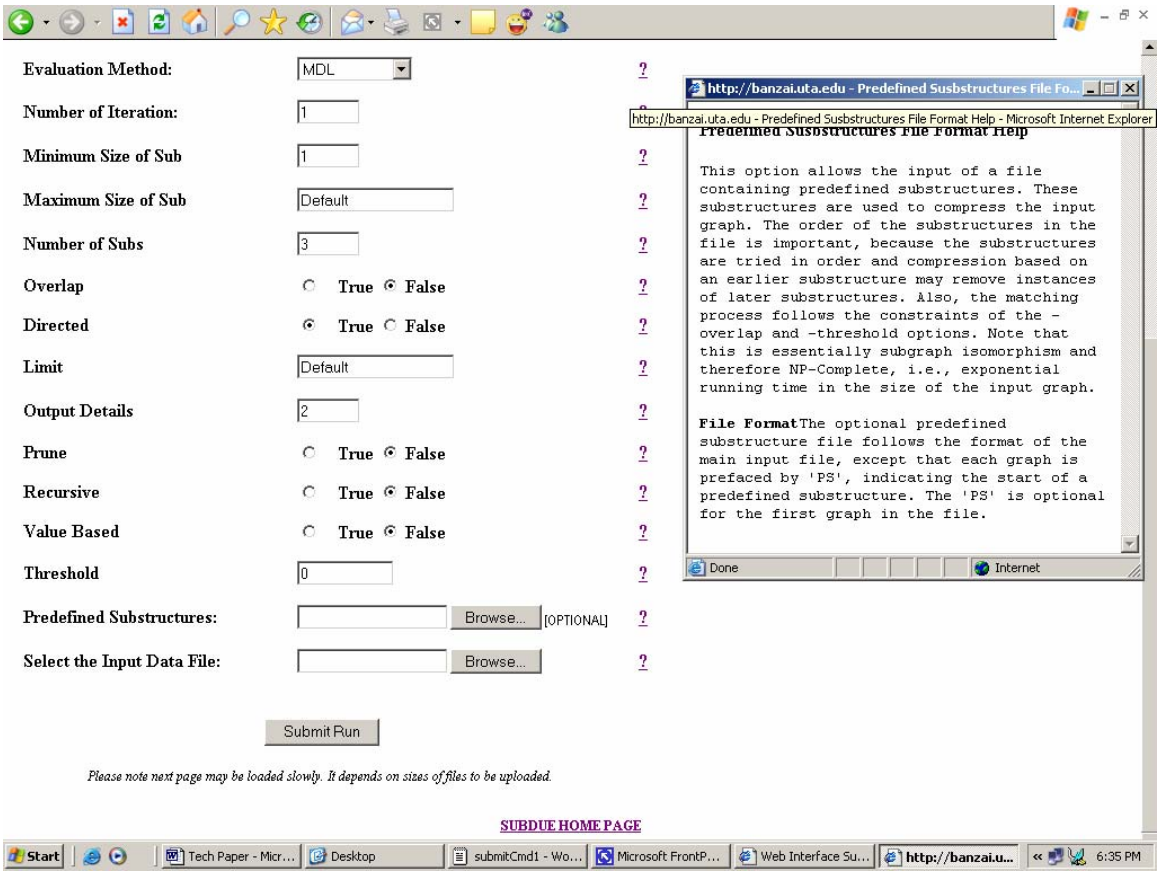

Figure 3.2 Subdue Web Interface Help

3. Users can select the input data file using the “File Open” dialog. The input data file argument is not optional and is checked before the user can submit the actual run. Figure 3.3 shows how users can select the input data file. Figure 3.4 illustrates the error message when the user tries to submit the program without selecting the input data file.

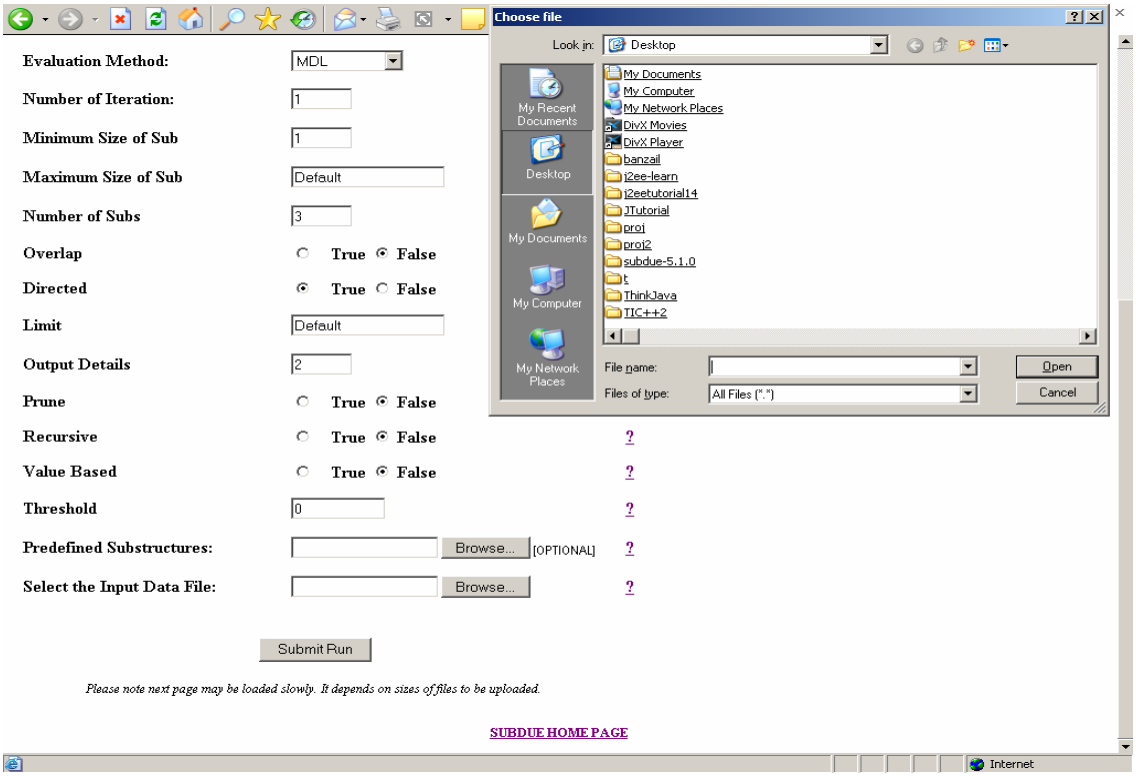

Figure 3.3 Subdue Web Interface File Dialog

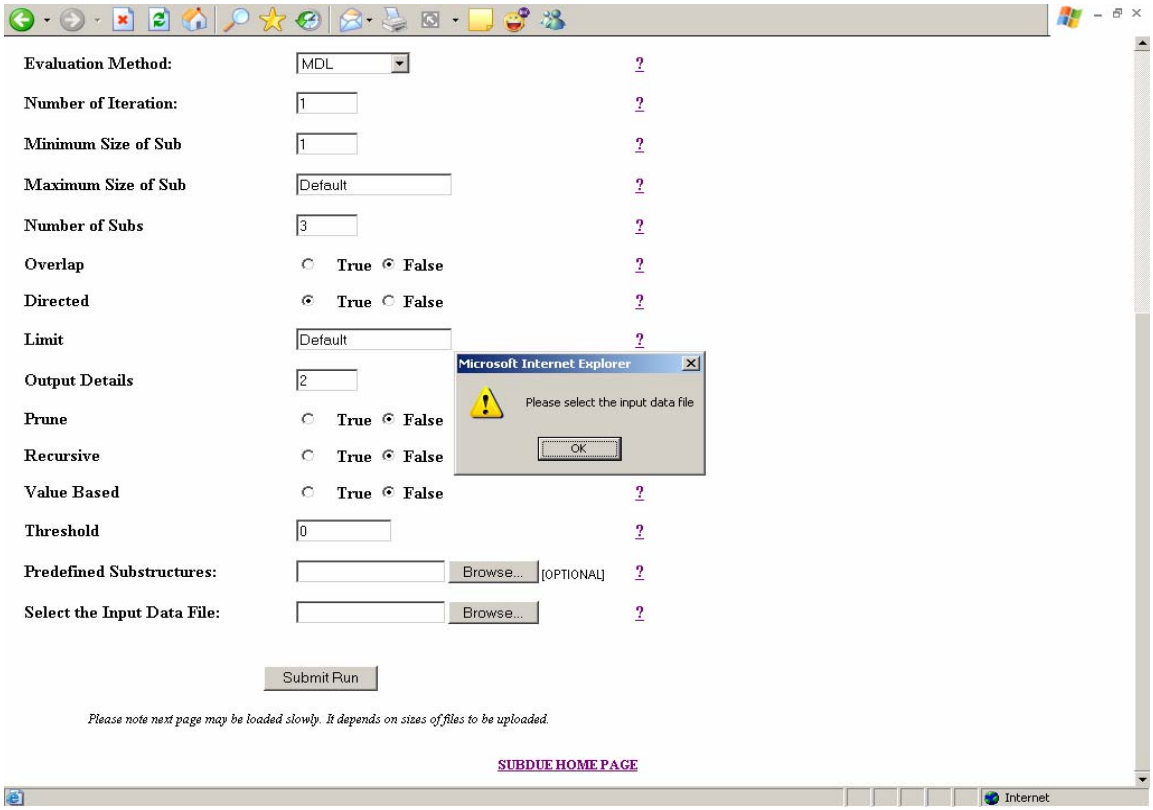

Figure 3.4 Subdue Web Interface Error Message

4. After users have selected the program they want to run, the input data file, and various options, they can submit the job. When the job is successfully submitted, the screen as illustrated in Figure 3.5 is displayed. This screen requests the user's email address for sending the results.

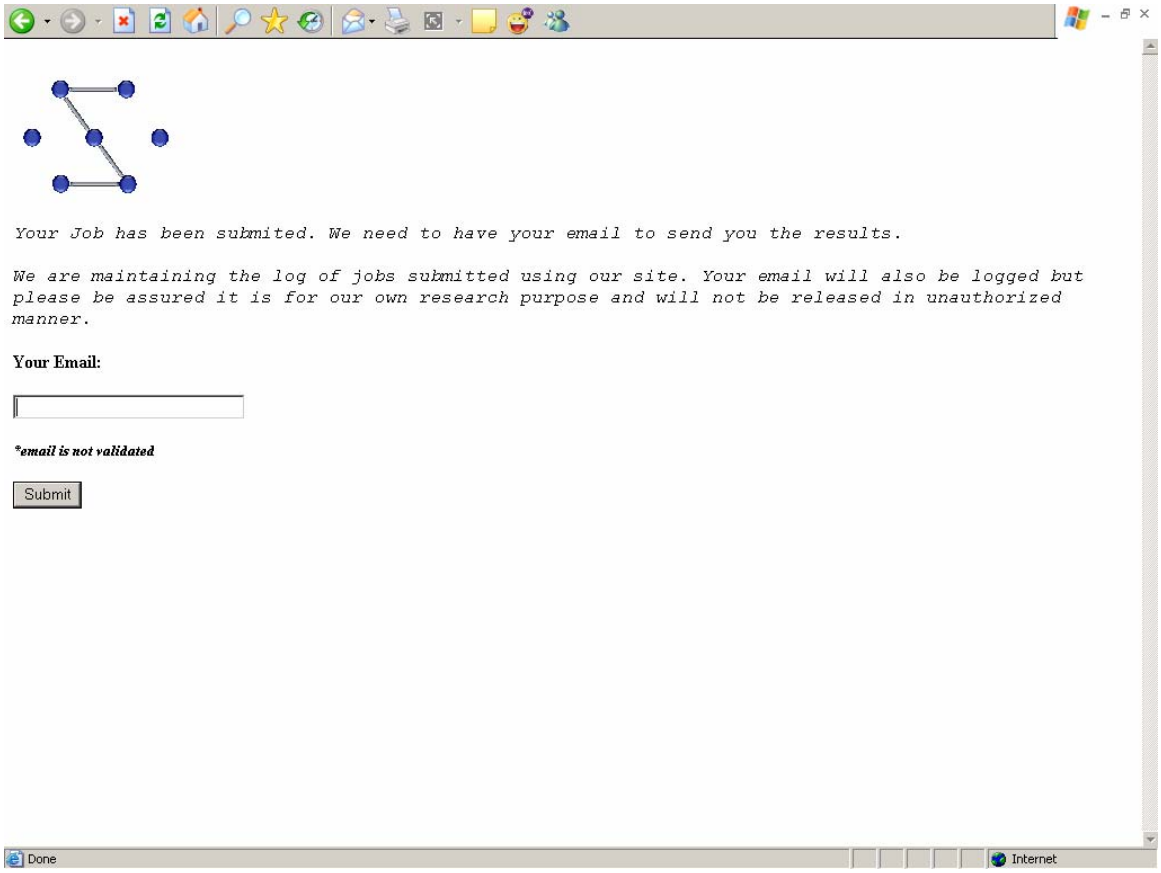

Figure 3.5 Subdue Web Interface Email

5. It should be noted that the above page may take time to load based on the size of file and internet connection speed since the user data file is uploaded to the server. Sometimes, if the size of the user file is more than allowed, users may see the error page as shown in Figure 3.6. The program only allows input data files having size less than 100kb for testing purpose.

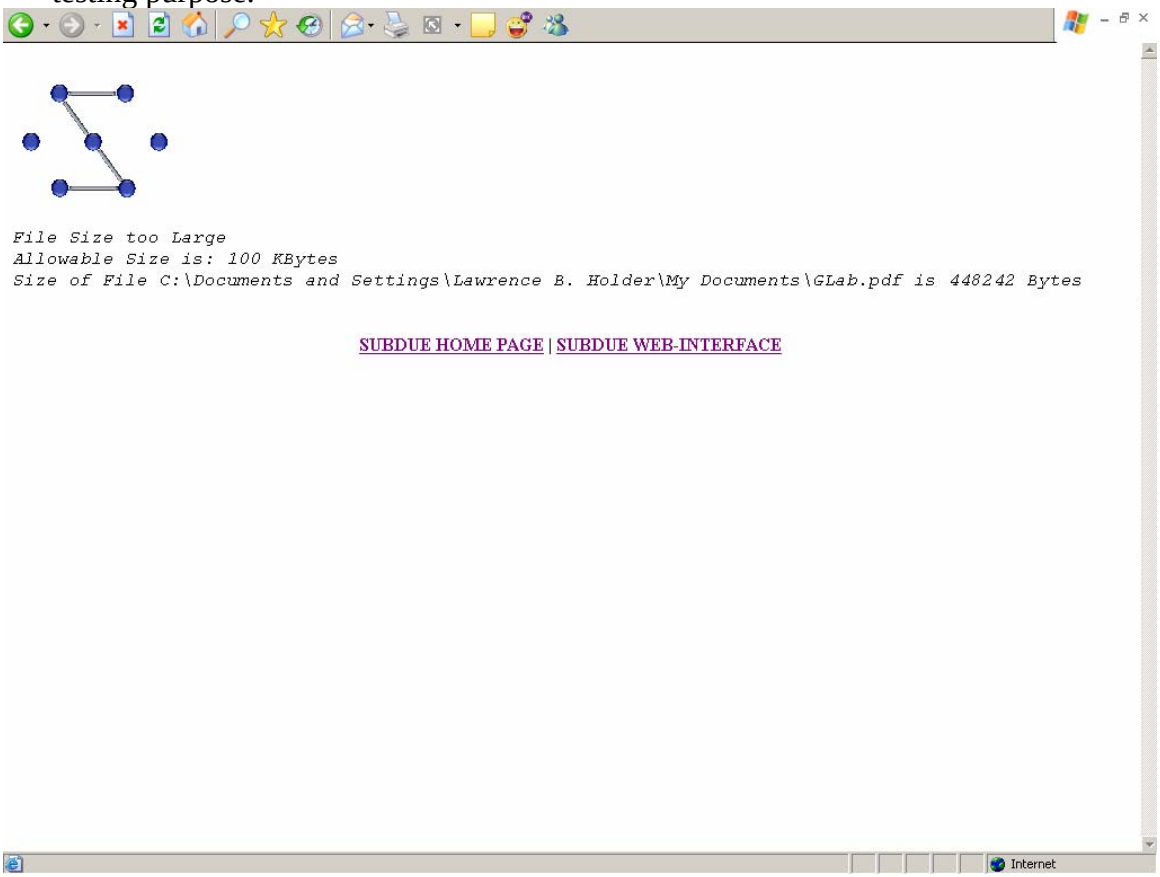

Figure 3.6 Subdue Web Interface File Size Error

6. Once the email address is submitted, a final confirmation screen showing the details about your run is shown. Meanwhile, your job is also started on the server in the background and a log is generated.
7. After the submitted job is complete, an email is sent automatically to the user-specified email address. The results are in the form of an attachment of the email. If there was a problem encountered in executing your job, then it is also mentioned.

3.4 Future Enhancements

This is a very preliminary attempt at creating a Web Interface for Subdue. There can be a number of enhancements and modifications to this version. A few enhancements that can improve the value of this interface are:

1. Encompassing all the programs in the Subdue package
2. Better client-side validation including validation of input data
3. Using features that are not browser dependent (e.g., JavaScript or some form fields like multi-part form data)

4 Subdue Graphical User Interface

4.1 Aim

The increasing popularity of windowing systems has made graphical user interfaces a necessity for all applications. Besides being easy to use, these interfaces improve customer experience and enhance the learning rate. More and more sophisticated tools are now available for creating graphical user interfaces.

The Subdue Graphical User Interface is just a layer above the actual Subdue system. The idea is to use the GUI instead of the normal command-line mode. All the functionalities remain the same except that you invoke Subdue and other programs from the GUI, and the results of these programs are also displayed on the GUI.

This GUI integrates help and is easier to use and learn than the command-line mode. The GUI is a good option for users who are not very familiar with using Subdue. We have integrated the following programs of Subdue in the GUI:

1. Subdue: Subdue discovery system finds interesting and repetitive subgraphs in the input data.
2. Inexact Graph Match: This program computes the cost of transforming the larger of the two input graphs into the smaller according to transformation costs.
3. Minimum Description Length: The program calculates the description length of graph 1 (g1), graph 2 (g2) and g2 compressed with g1 with final MDL compression measures.
4. Subgraph Isomorphism: This program finds all the instances of graph 1 (g1) in graph 2 (g2).
5. Cross-validation Test: This program performs a cross-validation experiment on a set of graphs.
6. Test: This program outputs the statistics for a given substructure to correctly classify the given graph examples.

4.2 Tools and Technologies

There are a number of different technologies available for creating the GUI. But the following issues were critical to us when selecting any one technology:

1. The most important issue was minimum modification to Subdue. We did not want to rewrite Subdue for integrating to the GUI. This is very important since the GUI should work with future releases of Subdue and can be extended to integrate other programs too. Therefore, we decided to use the client-server approach. We made Subdue programs act as server and GUI program act as client. The only modification in Subdue was to add one method which will open sockets. There were also modifications in another method for parsing parameters provided to Subdue for reasons explained below.
2. The same Subdue code works for both GUI and command-line modes. This was achieved by passing the parameter “-gui” secretly from the GUI client. This parameter will tell the Subdue programs whether the program was invoked from the GUI or the command-line. If invoked from the GUI, a special method that opens a connection with the GUI will be called, and all the output will be redirected to sockets.

3. The GUI should be easy to use and should work on different platforms without any modifications

Based on the above criteria, we decided to pursue the following technologies for developing the Subdue Graphical User Interface:

1. Socket programming: Berkeley Sockets are standard on Unix systems, and we used Unix socket libraries for making Subdue programs as the server.
2. Java: Java is an object-oriented language developed by Sun. Java has gained great popularity because of its compile once, run anywhere concept. Moreover, there are large numbers of inbuilt APIs that are available which simplify the process of developing applications in Java. We made extensive use of the following APIs available with Java 2 Standard Edition version 4.1.0:
    a. Networking: Java's network programming APIs are easy to use and make network programming very similar to writing files, a concept similar to Unix.
    b. Java Swing: Swing libraries are used for creating GUIs. This library is different from AWT libraries in Java, since these libraries are light-weight and less resource-consuming. The reason being that all the widgets in this library are written in pure Java and do not use the windowing system of the operating system. This makes the Swing widgets have a look-and-feel independent of the underlying system. Moreover, they are easily extensible since these widgets are pure Java objects. Also, Swing libraries are very comprehensive and include many more widgets than available with the AWT library. The other reason for using Swing is that in a Swing application it is possible to make the GUI to be managed by a separate thread without using Thread libraries. This integrated threading improves the response time of the GUI and improves the user experience.
    c. Thread Library: Even though threading is available with Swing, we use Thread to repaint output windows. We present output in a separate window because of increased flexibility. A separate thread is used to manage the output window to improve response time and avoid the GUI from being frozen when it is waiting for output from Subdue programs.

4.3 Installation

The GUI package consists of all the Java files and Readme. The short description of Java files and their functionalities are as follows:

- MainGui1.java: The class that ensembles all the GUI programs and initiates the window.
- SubGui.java, mdl_sgisoGui.java, gm_test.java: These are the programs that represent the GUI class for corresponding Subdue programs.
- MyFileGui.java: The helper program that creates a widget to select/browse the files.
- YesNoPanlel.java: The helper program that emulates 2-button (Yes/No) radio buttons.
- HelpPanel.java: The helper program that will display help for each Subdue option.
- OutputPanel.java: The program that displays the windows, manages the connection with Subdue and displays the output.

  To start the program, compile MainGui1.java using command "javac MainGui1.java". Then start the MainGui1 program using the command "java MainGui1".

4.4 Using the System

This section will walk through the steps of using Subdue Graphical User Interface.

1. The Subdue Graphical User Interface has a tabbed interface. Six programs are integrated in the GUI and are available on separate tabs. The programs that are available through this interface are: Subdue, MDL, SGISO, GM, Test, CVTest. Figure 4.1 illustrates the basic Subdue GUI. Figure 4.2 shows the state of the GUI after selecting the CVTest program by clicking on that tab.

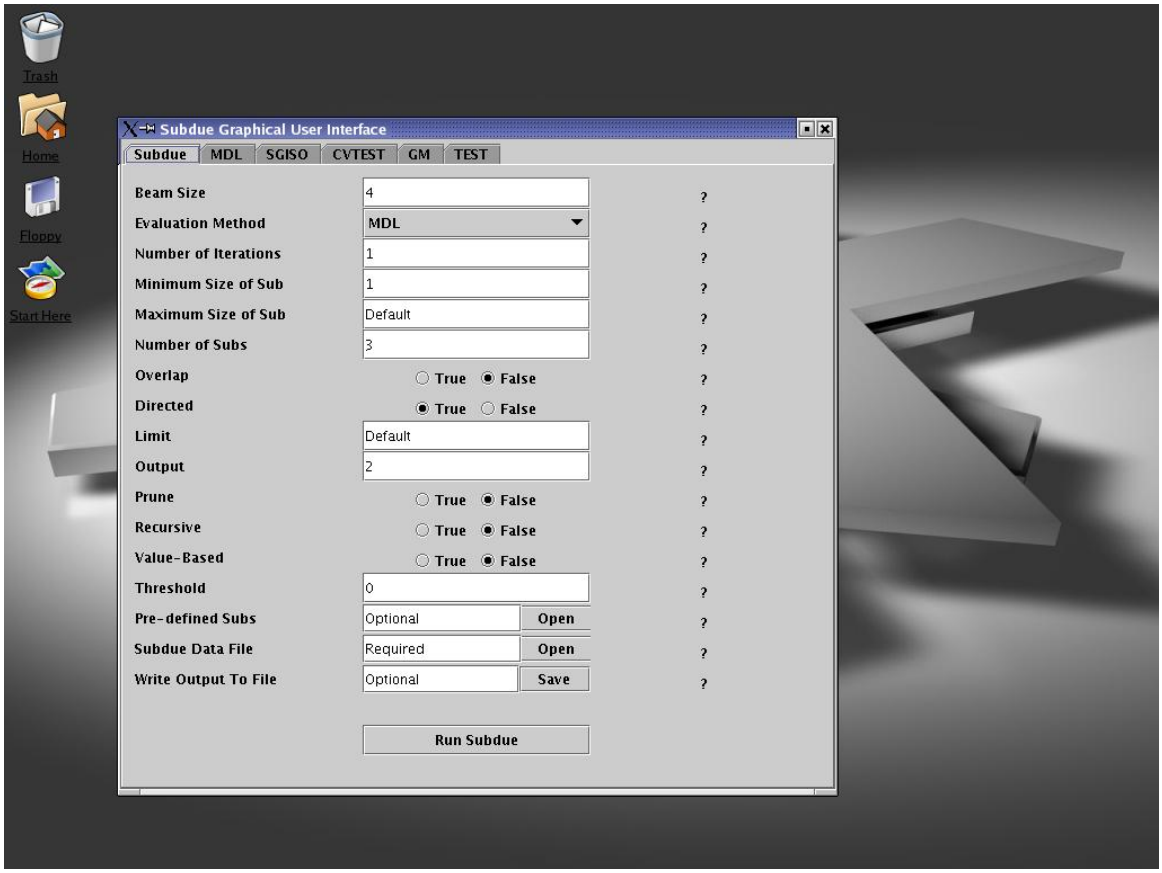

Figure 4.1 Subdue GUI

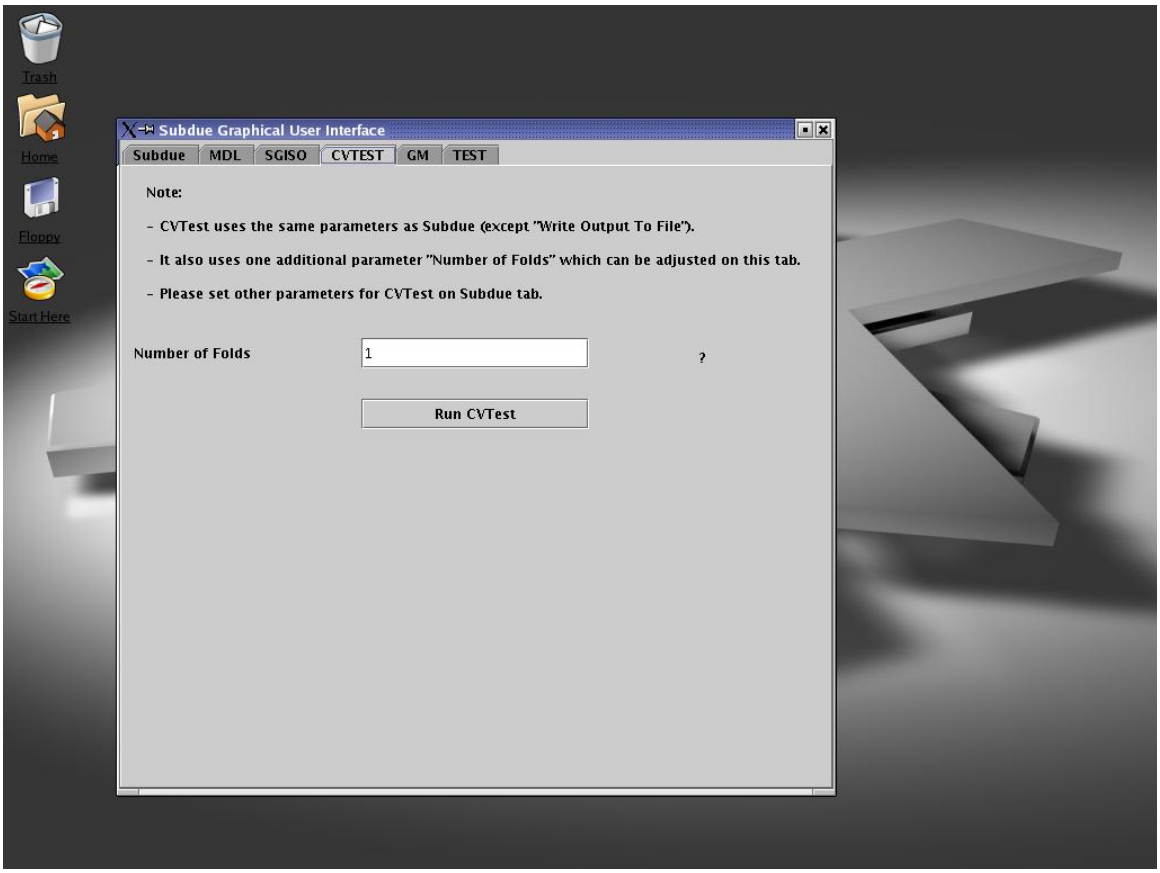

Figure 4.2 Subdue GUI CVTest

2. There is help integrated in GUI. Information is provided using tool-tips and separate window messages. Tool-tips provide basic information to the user. For example, the tool-tip for each tab is a description of the program on that tab. Tool-tips are also used on "?" or help buttons for each option. In this picture, the tool-tip provides information about the "CVTest" program. Figure 4.3 illustrates the tool-tip feature of the program.

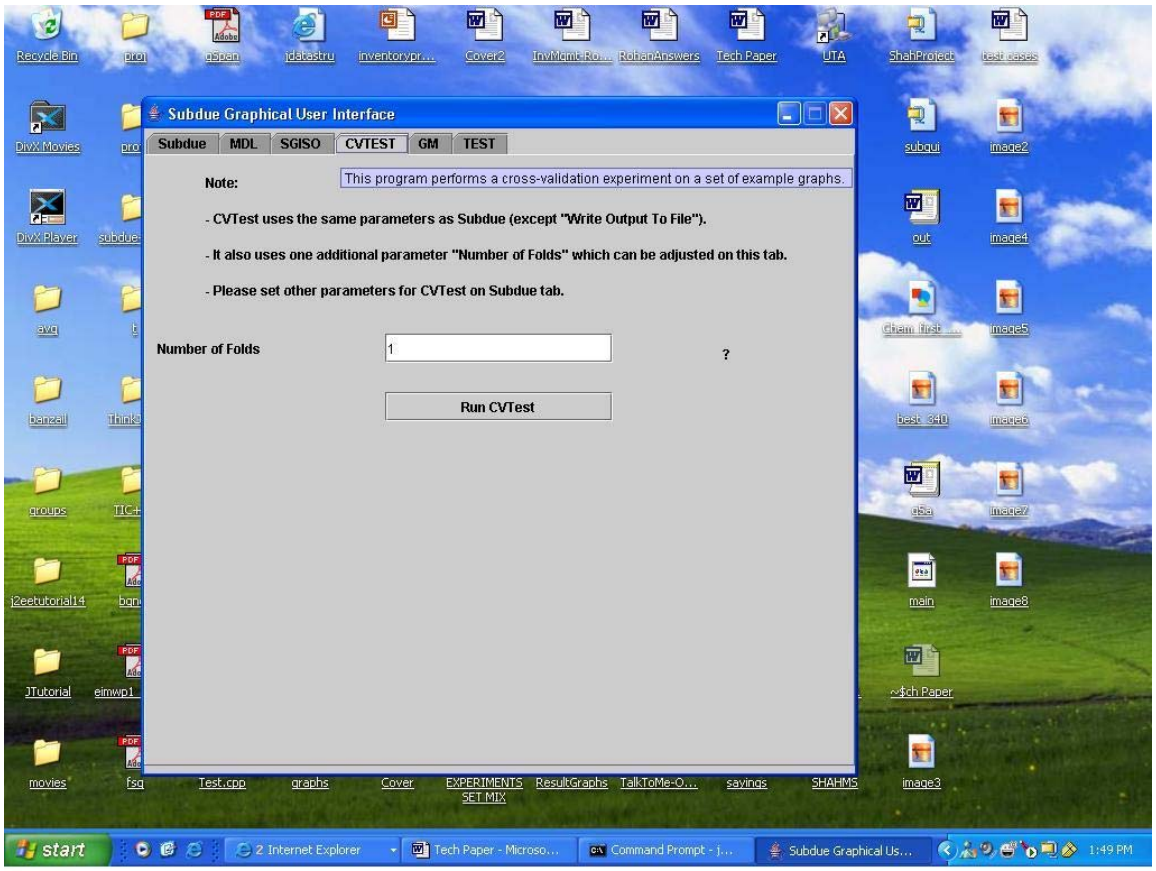

Figure 4.3 Subdue GUI Tool-tip

3. The message windows provide information to the user about the options in the program and error messages. When clicked, the help buttons (“?”) provide information to the user about the corresponding option. Error messages are displayed when the user does not provide the required parameter, e.g., the input file on the Subdue tab. If the user does not specify values for optional parameter, default values are used. Figure 4.4 shows the integrated help. Figure 4.5 illustrates the error message shown when the user does not provide the input data file.

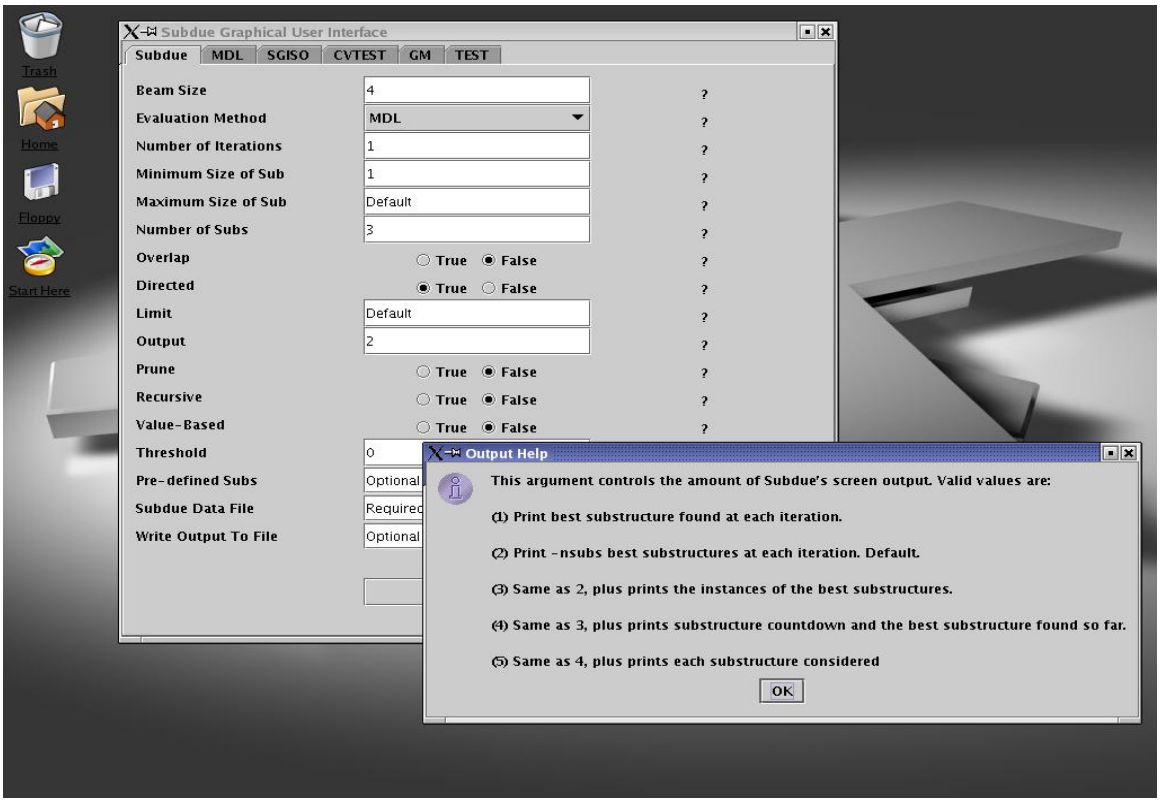

Figure 4.4 Subdue GUI Help

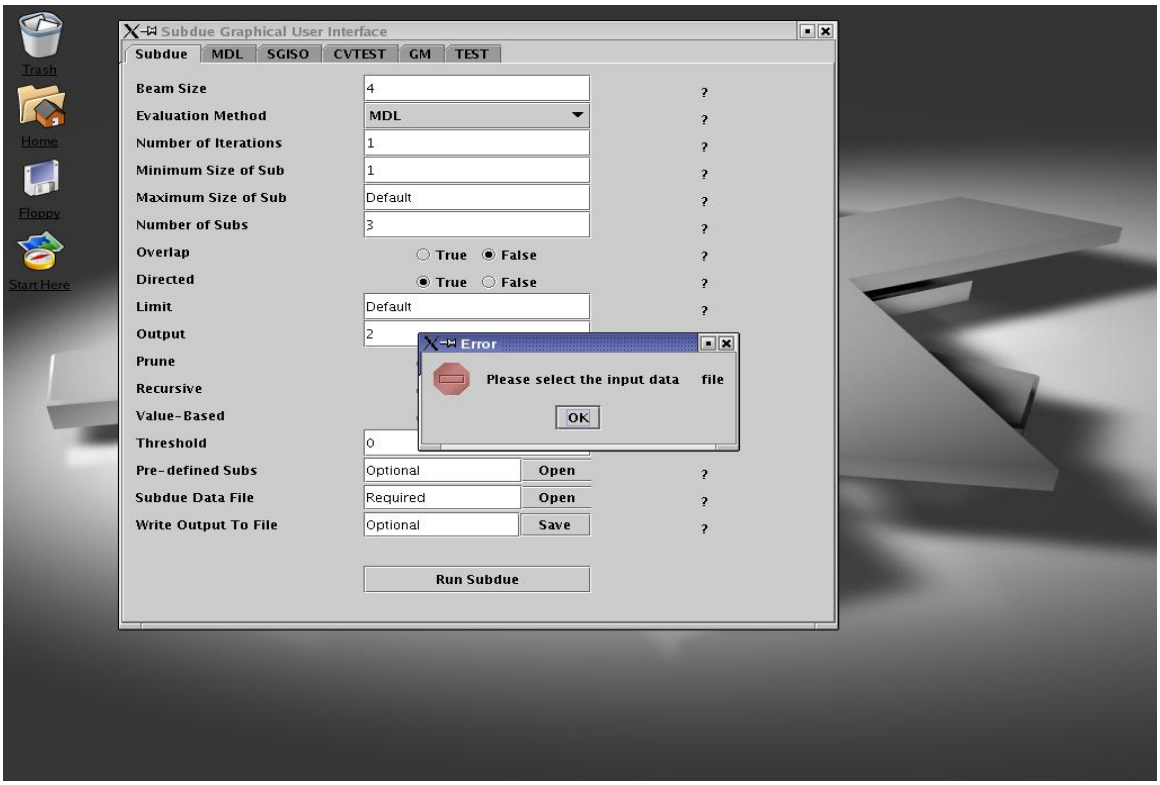

Figure 4.5 Subdue GUI Error

4. Users can select the input data file using the Java “File Open” dialog. There is also a “File Save” dialog in the Subdue program where the user has the option to save the output to a file. If an existing file is selected, the program will display a warning. Figures 4.6 and 4.7 illustrate these features.

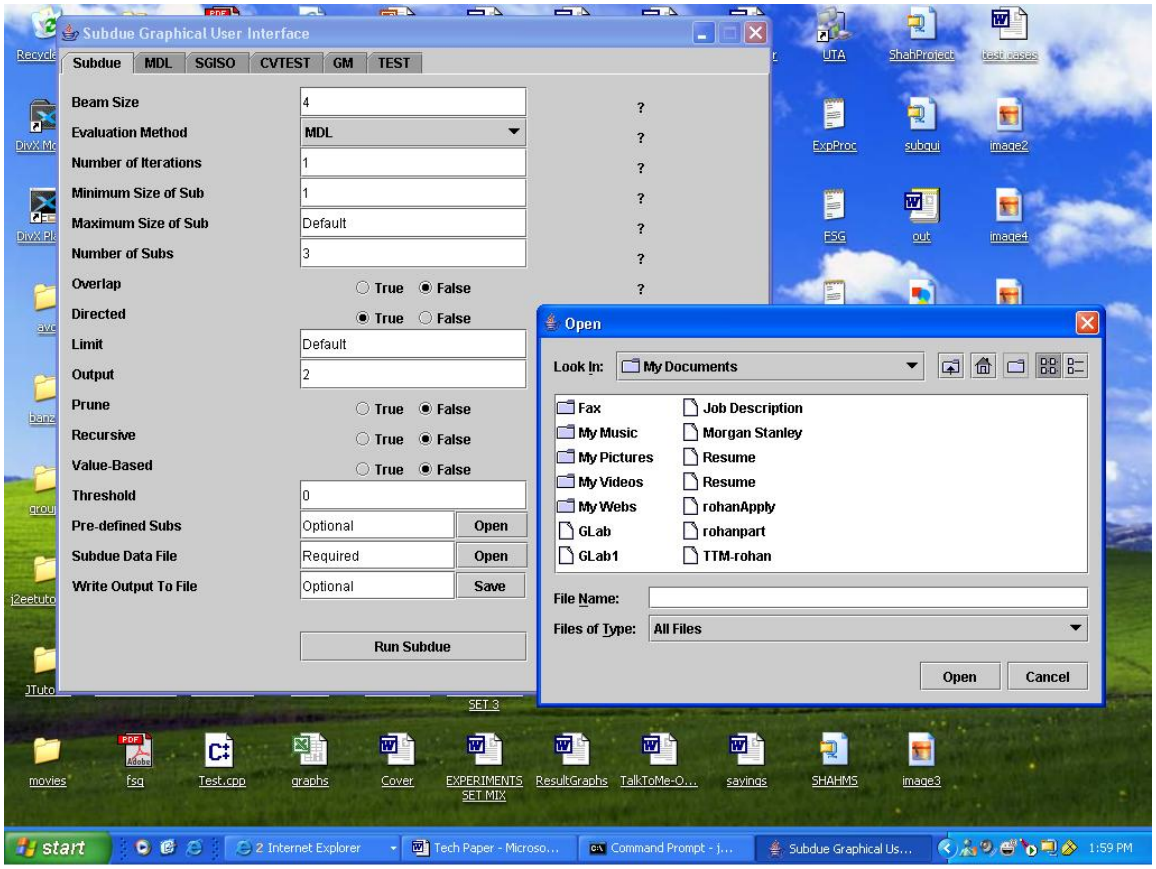

Figure 4.6 Subdue GUI File Open

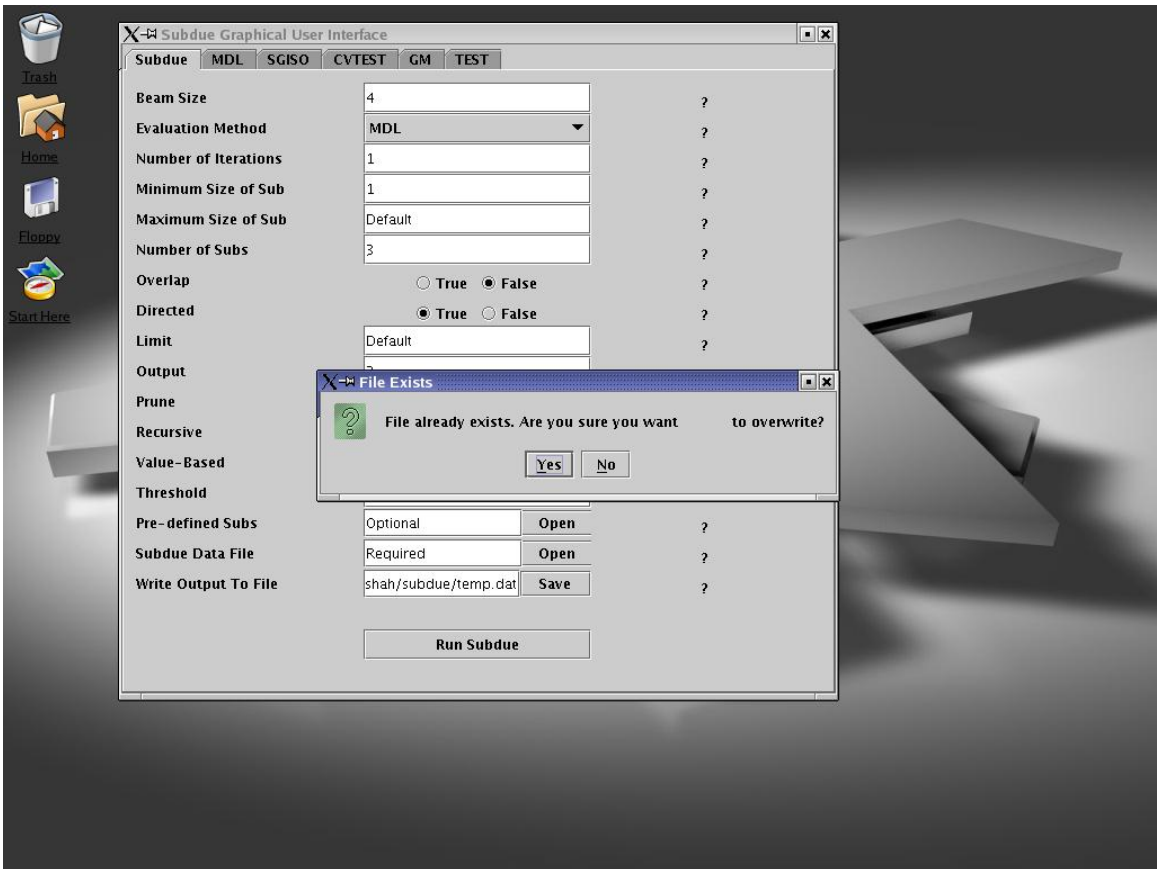

Figure 4.7 Subdue GUI File Save Warning

5. When the user presses the “run” button, the output for the command is displayed in a separate window. The output appears on the window as generated by Subdue (or sent by Subdue through a socket). However, there is a slight delay in displaying the output (compared to running these programs from the command line). This is generally system dependent. Figure 4.8 demonstrates the output of the Subdue program in a separate output window.

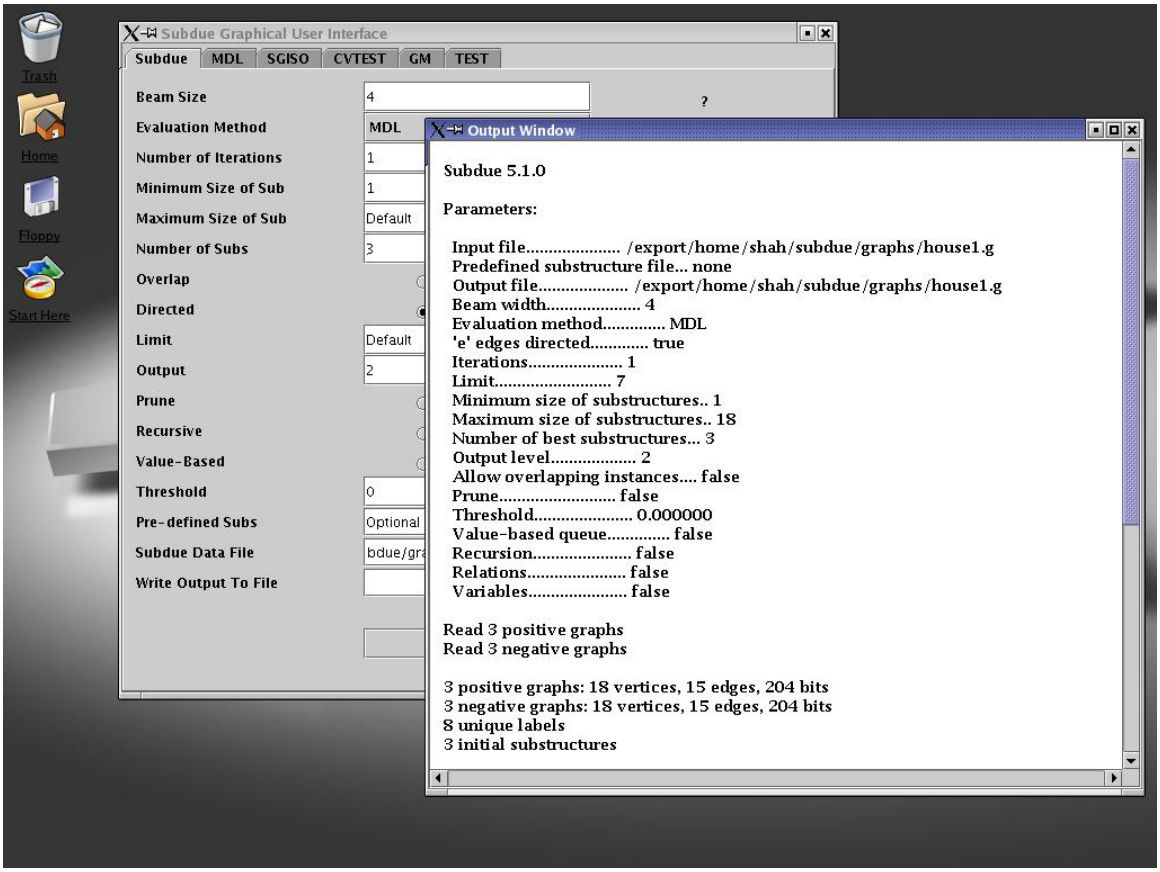

Figure 4.8 Subdue GUI Output

6. Many times that the server socket created by Subdue is not properly disposed. In that case, an error message is displayed in the output window notifying the user about the problem. If Subdue could not successfully complete the command, an error message is displayed in the output window. If you get the error “Connection is momentarily not available. Please try again”, as shown in figure 4.9, just try to run the program again, i.e., press the “Run Subdue” button again if you are trying to run Subdue. This problem occurs because the socket or connection from the program is not released.

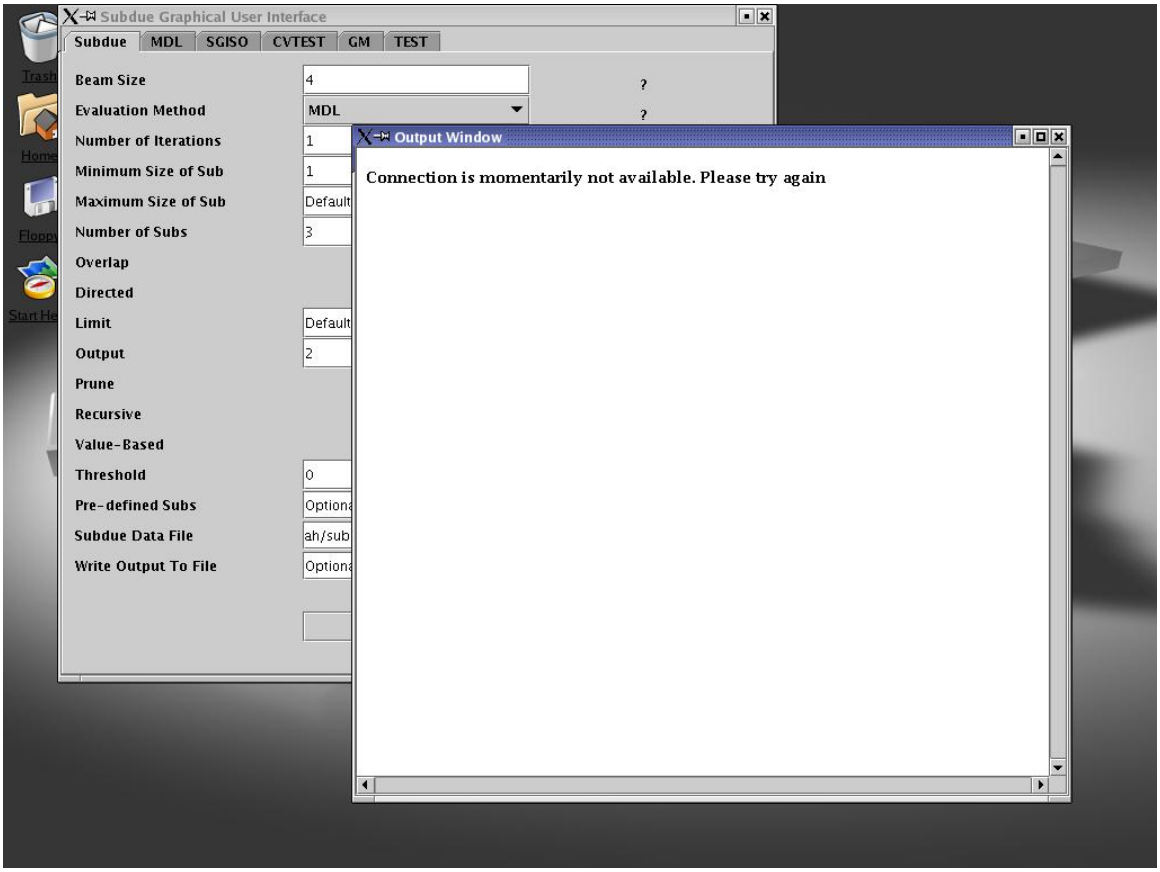

Figure 4.9 Subdue GUI Socket error

4.4 Future Enhancements

There can be a number of enhancements and modifications to this version. A few enhancements that can improve the value of this interface are:

1. Encompassing all the programs in the Subdue package.
2. Integrating a menu in the application.
3. Other methods for integrating Subdue programs with the GUI.
4. Ability to visually display graphs.

5 References

1. J. Gonzalez, L. B. Holder, D. J. Cook, Graph-Based Relational Concept Learning, Proceedings of International Machine Learning Conference, 2002.
2. D. J. Cook and L. B. Holder. Substructure Discovery Using Minimum Description Length and Background Knowledge. In Journal of Artificial Intelligence Research, Volume 1, Pages 231-255, 1994.