Implementation of an Open Source Optimization Software Package for Debian-Based Operating Systems

Emmanouil Emmanouil, Angelo Sifaleras

Department of Technology Management, University of Macedonia, Economic and Social Sciences, Loggou-Tourpali, Naoussa 59200, Greece

E-Mails: tm086@uom.gr, sifalera@uom.gr

Abstract

This paper aims to describe the development of an open source program (jNetPro) for network optimization problems. This application has been implemented in Java programming language and it has been mainly developed for educational purposes rather the solution of large-scale problems. Our motivation was to assist students to learn about some graph algorithms and to promote open source software into the Department of Technology Management at the University of Macedonia. The jNetPro application implements three network optimization algorithms; Dijkstra algorithm for the shortest path problem, and Kruskal and Prim algorithms for the minimum spanning tree problem. Although there exist several open source software packages, emphasis was given at the implementation of a program which can be installed to a Debian-based Linux operating system. The first stage of our methodology is to present some implementation issues for the creation of a .deb executable package, which can be installed to a Debian-based operating system. Next, we describe a short guide for the creation of a repository, which could be used by a student in order to install the software. Furthermore, we present the graphical user interface of jNetPro. Also, the proposed tool has the capability to import and/or export the problem data files. This work has two important goals; firstly to provide the users of Debian-based operating system with the ability to use the jNetPro for better comprehension of such algorithms, and secondly to familiarize students/readers with the implementation of open source applications and repositories.

Key words: Mathematical Programming, Network Optimization, Open Source Software.

1. Introduction

Graph theory is a part of discrete mathematics with applications in computer science, engineering, telecommunications, chemistry, and sociology. Graph theory became a very active research field after the study of Leonhard Euler on the problem of seven bridges of Konigsberg [2][1]. The goal of that problem was to find a path which crosses all the bridges and return at the starting point, passing only one time from each bridge. Several other well-known problems attracted the interest of researchers, such as the four-color problem [14]. The use of computers significantly assisted the developments in graph theory, since algorithmic approaches were not able to solve large graph theoretic problems without efficient computer implementations.

A plethora of software packages nowadays exist, for the teaching of graph theory and network optimization. Dosios et al. developed the NetPro [4] which is a standalone Matlab-based application, for the teaching of Linear Network Optimization. A few years later Karagiannis et al. extended the previous work, by developing WebNetPro [7] which is a web-based educational software based on Apache web server and Matlab web server.

The motivation of our work was to develop an open source optimization software package software, that could be installed in Debian-based operating systems. More specifically, the proposed tool jNetPro is a .deb package that is intended mainly for educational purposes rather than large-scale optimization. jNetPro was based on previous research efforts with NetPro and WebNetPro, but is now implemented using the object-oriented Java programming language. The proposed software package jNetPro provides implementations of three network optimization algorithms for the shortest path problem and the minimum spanning tree problem. The description of these two problems follows in Section 2. In Section 3, a description of the Debian GNU/Linux operating system and a guide about creating .deb packages are presented. A visual presentation of jNetPro follows in Section 4. Section 5 provides a short discussion about the economics of open source software. Finally, conclusions are given in Section 6.

2. Network optimization algorithms

The shortest path problem is one of the most well-studied problems in optimization [10], with several applications in Logistics. The most well-known algorithm for the shortest path problem with nonnegative arcs was proposed by Dijkstra in 1959 [3]. Assume a directed network $G = (N, A)$ with n nodes and m arcs, where N and A are the sets of nodes and arcs, respectively. Each arc $(i,j) \in A$ has a distance c_{ij} that denotes the distance between nodes i and j. The goal of this problem is to find the minimum distances $d(i)$ between each node $i \in N$ and node s. The pseudo-code of the Dijkstra algorithm is as follows:

Table 2.1. Dijkstra Algorithm.

```
1  $d(s) = 0, d(i) = \infty \ \forall i \neq s$
2  Partition the set of nodes into two disjoint sets $S = \{s\}$, and $S' = N \mid S$
3  While $S' \neq \emptyset \wedge \{(i,j) : i \in S, j \in S'\} \neq \emptyset$
4      $d(k) + c_{kq} = \min\{d(i) + c_{ij} : (i,j) \in A, i \in S, j \in S'\}$
5      $S = S \cup \{q\}, S' = S' \mid \{q\}$
```

Furthermore, in the minimum spanning tree problem we are given a weighted undirected graph $G = (V, E)$ with n vertices and m edges, where V and E are the sets of vertices and edges, respectively. The goal of the problem is to find a subset of edges $E' \subset E$, such that it constitutes a tree of minimum total weight. The minimum spanning tree problem has a lot of applications in telecommunications, computer science, etc. Two of the most well-known algorithms for finding a minimum spanning trees, are due to Prim [12] and Kruskal [8]. The pseudo-code of the Prim algorithm is as follows:

Table 2.2. Prim Algorithm.

```
1  Select a starting node $s \in N$
2  While there are isolated nodes do
3      Choose the minimum weight edge which connects the current
       tree solution and the remaining isolated nodes
4      Add this edge and the corresponding incident node into the tree
       solution
5      Update the total weight of the current tree solution as also the
       new neighborhood of the potential isolated nodes.
```

Kruskal algorithm first sorts the edges in ascending order, according to their weight. The pseudo-code of the Kruskal algorithm is as follows:

Table 2.3. Kruskal Algorithm.

```
1  Sort the edges in an ascending order according to their weight
2  $t = 0$
3  While $( t < n - 1)$ do
4      Select the next edge $(u,w)$
5      If $( component(u) \neq component(w) )$
6          Add this edge into the tree solution $T$
7          $component(u) = component(w)$
8          $t = t + 1$
```

The above mentioned pseudo-codes show the main algorithmic steps of each algorithm. However, the selection of data structures for the implementation of these algorithms may affect not only the theoretical complexity but also the efficiency of the corresponding computer implementation.

3. Debian-Based Operating Systems

3.1 The Debian GNU/Linux Operating System

Debian-based operating systems are part of a biggest operating system family [9] [15] and a distribution of free software with the same name, based on GNU/Linux. This distribution consists of a large number of packages. Every package of this distribution has executables, scenarios, documentation, information for the settings and it has a maintainer which is the main responsible for package updating, and watching for errors in order to communicate with the upstream software package creator. The large number of users as well as the error tracking system, guarantee quick solutions to almost any problem. Debian-based operating systems are extremely popular among the professional users of the Linux community due to the stability and technical excellence. Furthermore, Debian is the first Linux distribution which has a package management system for easy installing or uninstalling a software package. It was also the first distribution which it could update itself, even without previously uninstalling the operating system.

Today Debian has a leader position in Linux development. This development process is only one example which proves that the Open Source is a development model which can work very well in practice. This is true, even when complex procedures should be executed such as building and maintaining an open source operating system. Perhaps, the most well-known feature which makes Debian to stand out from other distributions is the package system management. This tool gives full control to the Debian system manager for maintaining the installed packages at his system. Moreover, for security purposes against Trojans and other malicious software, the Debian distribution servers, verify that the packages which are uploaded by programmers are secure. Should a security problem might appear into a package, then the required corrections by programmers are usually very fast. In the following sub-section, a guide on creating .deb packages will be visually presented step-by-step.

3.2 Implementation Issues for Creating .deb Packages

A .deb package can be used into different distributions of Debian-based operating systems, such as Ubuntu which was used in our work. These .deb packages have two archives; the one is the control information which has the information of the package and the other has the program data. One of the most popular programs for package management of .deb archives is the "dpkh", usually using other software programs like for example "apt/aptitude". The .deb packages can also be converted to other archive types from other distributions. Also, they can be created by source code with the "Checkinstall" program or the "Maker" Debian package. A .deb package is actually a zip archive and thus its contents can be easily seen after extracting the .deb archive. Figure 3.2.1 shows us the two folders that exist inside the .deb file; the "DEBIAN" and the "Usr".

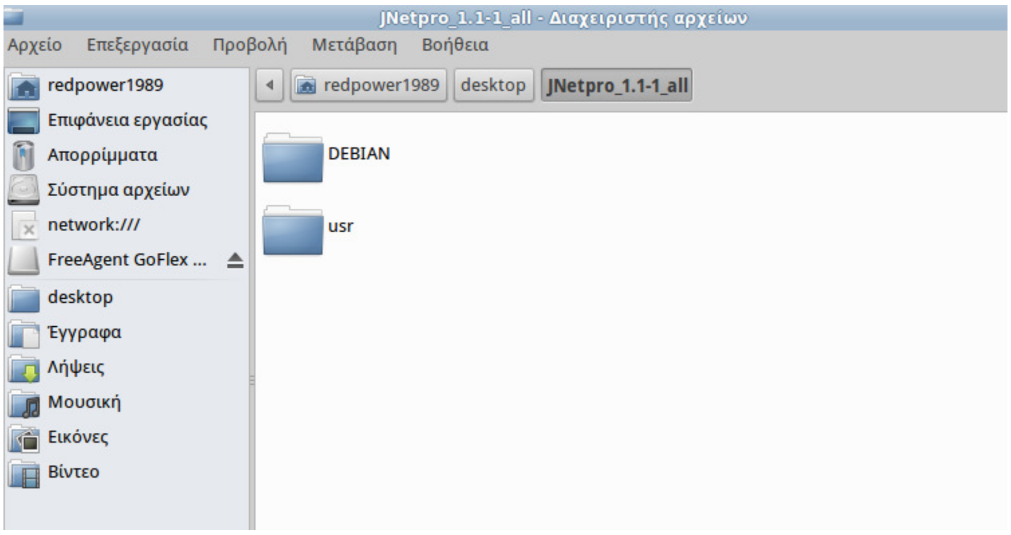
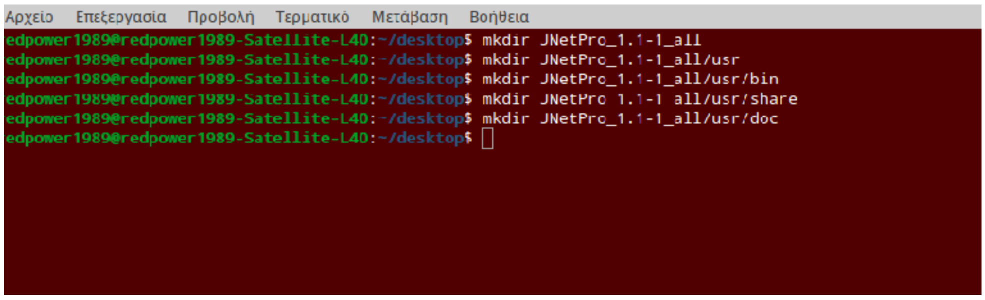


Figure 3.2.1. The Contents of .deb File.

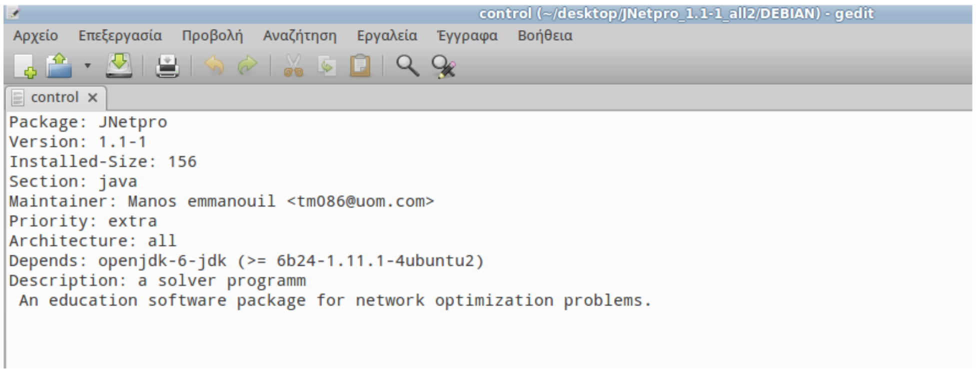
The name of our proposed software package has the form "JNetPro_1.1-1_all", which denotes the following information: (package name)_(Biggest version).(smallest version)-(Computer architecture). The next step is to create the two folders as above and the required subfolders. One of the subfolders is /usr/bin where the executables archives of our program will be put. The other subfolder is /usr/share/doc where the picture of our program will be put and also one archive which allow the user to call the program. Figure 3.2.2 depicts the necessary commands for the above tasks.



```
edpower1989@redpower1989-Satellite-L40:~/desktop$ mkdir JNetPro_1.1-1_all
edpower1989@redpower1989-Satellite-L40:~/desktop$ mkdir JNetPro_1.1-1_all/usr
edpower1989@redpower1989-Satellite-L40:~/desktop$ mkdir JNetPro_1.1-1_all/usr/bin
edpower1989@redpower1989-Satellite-L40:~/desktop$ mkdir JNetPro 1.1-1 all/usr/share
edpower1989@redpower1989-Satellite-L40:~/desktop$ mkdir JNetPro_1.1-1_all/usr/doc
edpower1989@redpower1989-Satellite-L40:~/desktop$
```

Figure 3.2.2. Terminal Commands.

In the "DEBIAN" folder an archive with the name "control" must be created*.* The "control" archive is literally the brain of our program. All the necessary information related to our software, exist in this archive. Without this archive the .deb package does not exist. The contents of this archive are shown in Figure 3.2.3.


control (~/desktop/JNetpro_1.1-1_all2/DEBIAN) - gedit
Αρχείο Επεξεργασία Προβολή Αναζήτηση Εργαλεία Έγγραφα Βοήθεια
control

```
Package: JNetpro
Version: 1.1-1
Installed-Size: 156
Section: java
Maintainer: Manos emmanouil <tm086@uom.com>
Priority: extra
Architecture: all
Depends: openjdk-6-jdk (>= 6b24-1.11.1-4ubuntu2)
Description: a solver programm
 An education software package for network optimization problems.
```

Figure 3.2.3. Inside the Control Archive.

The minimum required parameters which someone must write in the "control" file are those shown in Figure 3.2.3. However, the complete parameters are as follows:

- Package: The name of the package (case sensitive) without blanks between letters and without version or computer architecture.
- Version: The current version of the program, written also to the name of the file we created. By changing this number the system is informed that an update exist
- Priority. The values of this parameter might be: "optional", "high", "low", "extra". If the package is for the needs of the operating system or for security reasons, then this value should be set to "high".
- Architecture: The computer architecture that our package if for. The values are "i386", "amd64", or "all" for 32bit architecture, 64bit architecture, or both architectures, respectively.
- Depends: This is a list of packages which are required to be pre-installed (dependencies).
- Conflicts: This is a list of package which should not be installed into our system, due to conflicts.
- Replaces: This field informs the system that this package replace an older package. Also, if the name of the new version of package is changed and we want to keep the older one, then the new name of the new version should be written here.
- Description: This provides a description of the program. A title and a detailed description should be given.

These are the elements of the "control" archive. In the "Debian" folder two extra archives can be created for achieving a better performance of the program. Those archives are "postinst" and "prerm".

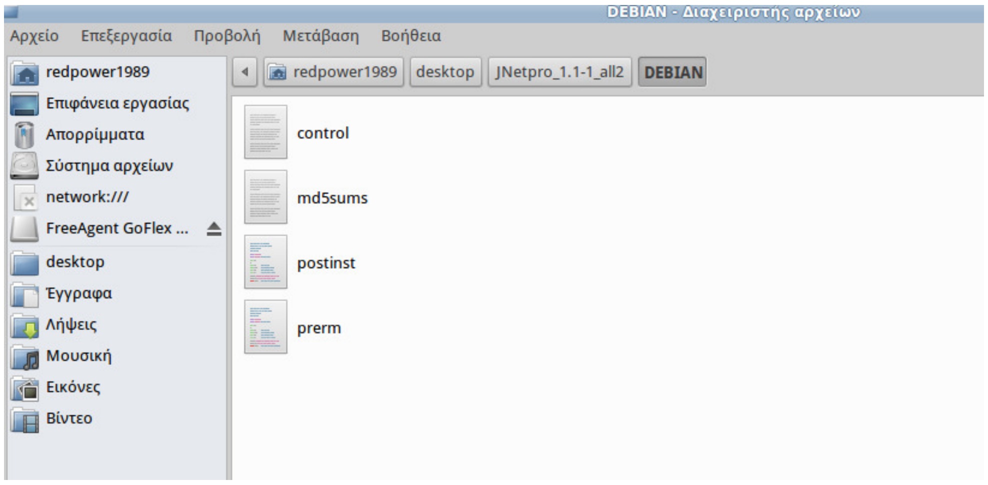
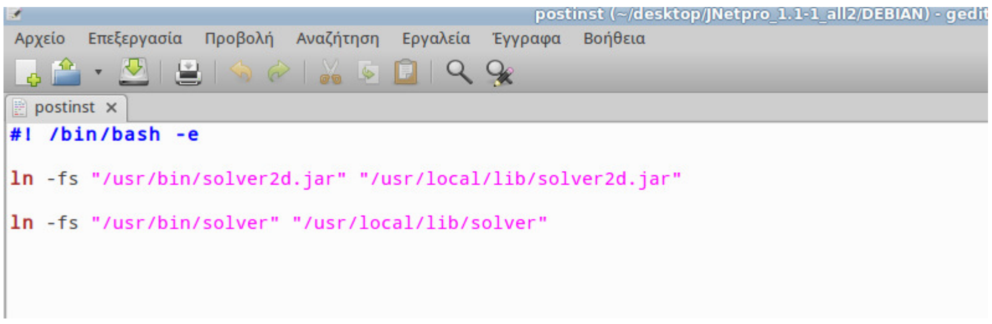


Figure 3.2.4. Inside the DEBIAN Folder.

The archives "postinst" and "prerm" are script executables archives. The "postinst" archive is a script file which executes some commands after the installation of the program. An example of "postinst" is shown in Figure 3.2.5.


postinst (~/desktop/JNetpro_1.1-1_all2/DEBIAN) - gedit
Αρχείο Επεξεργασία Προβολή Αναζήτηση Εργαλεία Έγγραφα Βοήθεια
postinst

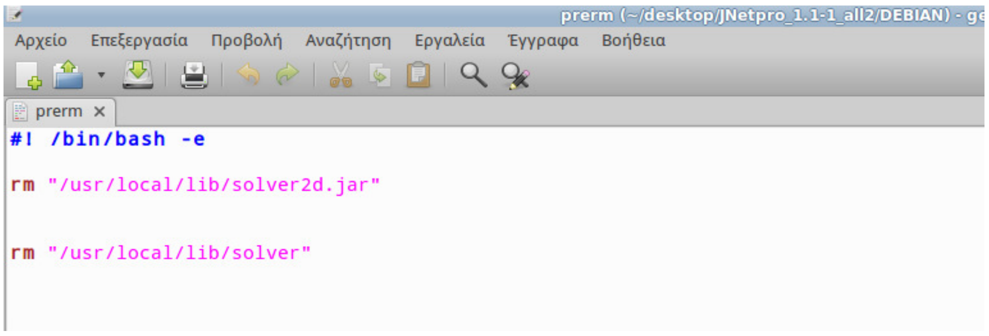
```
#! /bin/bash -e

ln -fs "/usr/bin/solver2d.jar" "/usr/local/lib/solver2d.jar"

ln -fs "/usr/bin/solver" "/usr/local/lib/solver"
```

Figure 3.2.5. Contents of the Postinst Archive.

Therefore, after the installation of the program those commands will be executed. The next archive "prerm" is also a script archive which is executed after uninstalling the program. An example of the "prerm" archive is shown in Figure 3.2.6.


prerm (~/desktop/JNetpro_1.1-1_all2/DEBIAN) - ge
Αρχείο Επεξεργασία Προβολή Αναζήτηση Εργαλεία Έγγραφα Βοήθεια
prerm

```
#! /bin/bash -e

rm "/usr/local/lib/solver2d.jar"


rm "/usr/local/lib/solver"
```

Figure 3.2.6. Contents of the Prerm Archive.

Both of these scripts must have the appropriate permissions for users and also be executables; the permissions must be set between "755" and "775". Those are the basics archives for the "DEBIAN" folder. The next important folder for our package is the "usr". An example of the "usr" folder is shown in Figure 3.2.7.

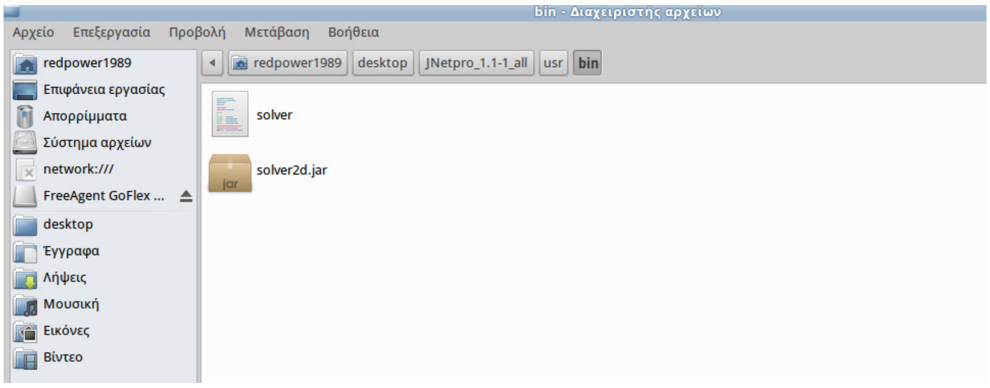
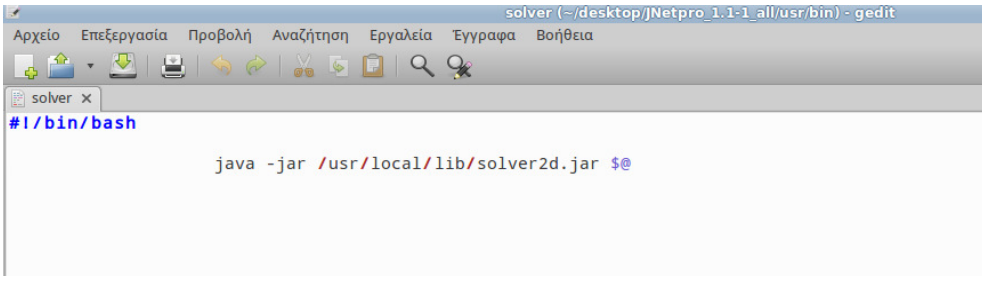


Figure 3.2.7. Contents of the usr/bin Subfolder.

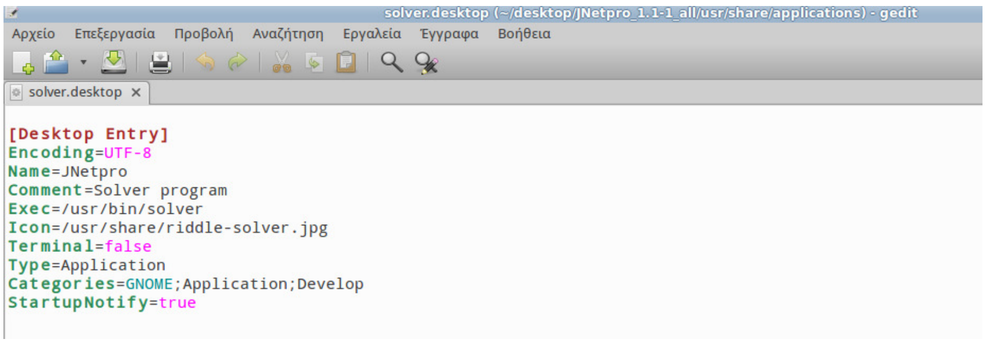
The required archive of this folder is the solver archive. This archive is a script executable which is executed when a user calls this program. In our case, our program executes the solver2c.jar archive. The contents of the solver archive are shown in Figure 3.2.8.



```
#!/bin/bash
                        java -jar /usr/local/lib/solver2d.jar $@
```

Figure 3.2.8. Contents of the Solver Archive.

Similar to the "postinst" and "prerm" archives, the solver archive must have appropriate permissions and should be executable. Needless to say, a user might create different and more complex structure in the "usr" folder. This depends on the complexity of the program which someone creates. In this point the user should be warned that he/she should be cautious if he/she has already installed the program, since the apt-get will give an error. Afterwards, the program should be accessible from the Gnome Application Menu. In order to do this, an archive is required which describes our package. This archive should be put at the /usr/share/application folder. The name of our archive was "jNetPro.desktop", and this is the name that afterwards appears at the menu. The contents of the "jNetPro.desktop" archive are shown in Figure 3.2.9.


solver.desktop (~/desktop/JNetpro_1.1-1_all/usr/share/applications) - gedit
Αρχείο Επεξεργασία Προβολή Αναζήτηση Εργαλεία Έγγραφα Βοήθεια
solver.desktop

```
[Desktop Entry]
Encoding=UTF-8
Name=JNetpro
Comment=Solver program
Exec=/usr/bin/solver
Icon=/usr/share/riddle-solver.jpg
Terminal=false
Type=Application
Categories=GNOME;Application;Develop
StartupNotify=true
```

Figure 3.2.9. Contents of the jNetPro.desktop archive.

The parameters of this archive are as follows:

- Encoding: The coding of the letters.
- Name: The name of the package.
- Comment: Comments about the program.
- Exec: The path of the executable inside our package.
- Icon: The path of the image inside the package.
- Terminal: This value can be "true" or "false". If it is set to "true" the program can be started from the terminal.
- Type: The type of the software.
- Categories: The name of the menu category that the application will be found.
- Startup Notify: This value takes "true" of "false" options. If it set to "true" the system informs the user that the program is going to start.

The next step is to create the copyright of the program. Each user might use a different license according to the principles of open source or free software. In our case we used the GPL license. For this reason, a subfolder entitled "doc" was created under the "share" folder, and a copyright archive with our license was put there, as shown in Figure 3.2.10.

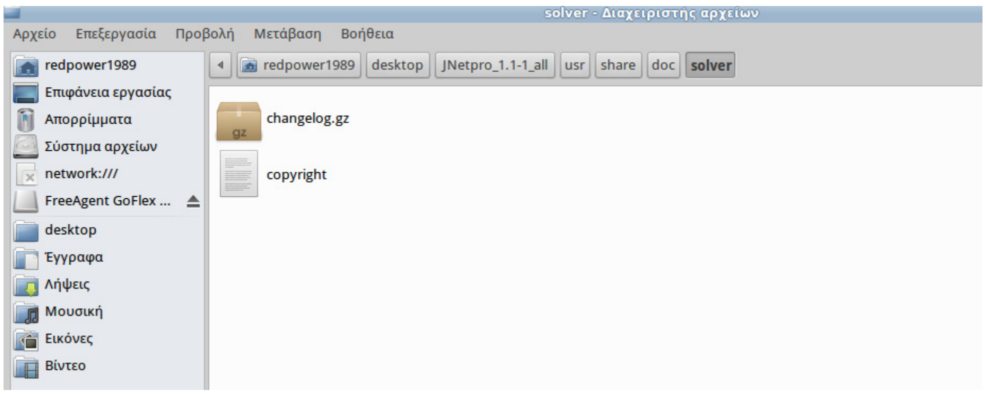
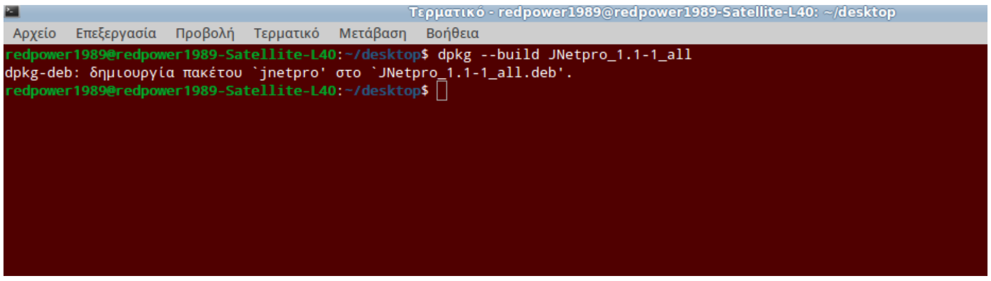


Figure 3.2.10. Contents of the doc Folder.

The next archive inside this folder is the "changelog" file. This archive describes the updates of the program. Once all the required archives are made, the .deb package can be created as it is shown in Figures 3.2.11 and 3.2.12.


Τερματικό - redpower1989@redpower1989-Satellite-L40: ~/desktop
Αρχείο Επεξεργασία Προβολή Τερματικό Μετάβαση Βοήθεια

```
redpower1989@redpower1989-Satellite-L40:~/desktop$ dpkg --build JNetpro_1.1-1_all
dpkg-deb: δημιουργία πακέτου `jnetpro' στο `JNetpro_1.1-1_all.deb'.
redpower1989@redpower1989-Satellite-L40:~/desktop$
```

Figure 3.2.11. The commands for creating a .deb package.

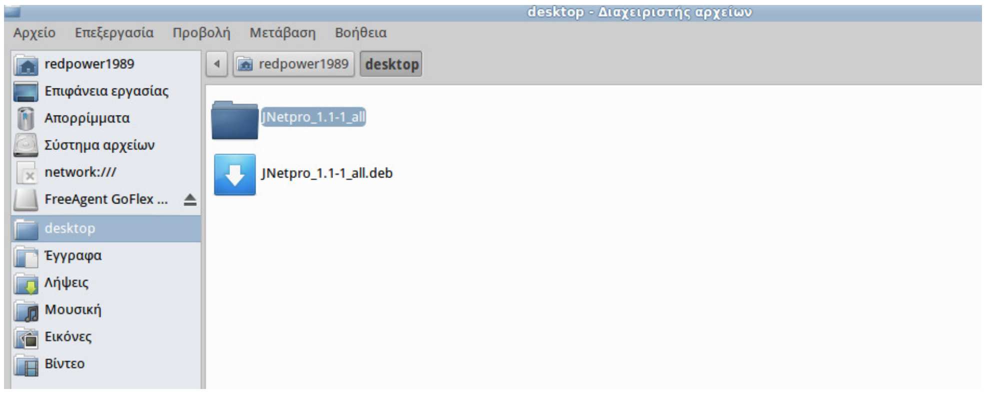


Figure 3.2.13. The .deb package which was created.

There are two ways to install the .deb package. The first is through the software management system of the users' distribution, (e.g., Ubuntu software center). The other way is to execute the following command from a terminal:

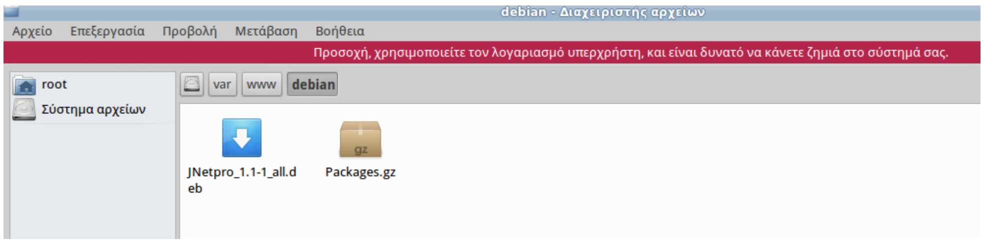
- sudo dpkg-i JNetPro_1.1-1_all.deb

Afterwards, the application can be found to the menu and the user can start it from there.

3.3 A Tutorial for Creating a Local Repository

This sub-section presents the creation of a local repository in order to install the proposed package from there. A motivation of this guide is to create in the future a repository to the web server of the Technology Management Department of the University of Macedonia. This way, all the students/developers can upload their packages in order that other students/user may install them to their systems. The apache web server needs to be previously installed to our system. This application can be easily installed through Ubuntu software center or by terminal. The path of default apache is /var/www/. Here we will create our repository. So, firstly we open our file manager with root privileges. In this work, we used the file manager Thunar; nautilus in mainly used in ubuntu.

Using our file manager we visit the previous path /var/www/ where we created our file inside folder "debian" and we move there our package that was created, as shown in Figure 3.2.14. In order to create our repository, we go back to /var/www folder, open a terminal and execute the command "dpkg-scanpackages debian /dev/null | gzip -9c > debian/Packages.gz".



Προσοχή, χρησιμοποιείτε τον λογαριασμό υπερχρήστη, και είναι δυνατό να κάνετε ζημιά στο σύστημά σας.

Figure 3.2.14. The creation of our Local Repository.

Afterwards, our repository should be inserted to the list of repositories in order that our system scan them and install the package. To do that, we open the list of repositories with an editing tool and add our repository at the end of the list, by writing: "deb http://127.0.0.1 debian/". Obviously, the address 127.0.0.1 corresponds to our local server. In the future, we aim to replace this with the address of the web server of the Technology Management Department. Then, once the list of repositories is updated, we can easily install our package through the terminal.

4. The jNetPro Optimization Software Package

The jNetPro application is mainly for educational purposes, developed to solve small network optimizations problems. It features three algorithms (i.e., Dijkstra, Prim, and Kruskal) from two different problem categories (i.e., shortest path problem, minimum spanning tree problem). Also, it provides the user with the ability to import or export the problem data. The data file that is exported holds information about the data as also which algorithm was used. The menu bar of the program provides with three choices, as shown in Figure 4.1. The user can create a new problem, export/import data or close file, using the "File" choice. Next, there are two problem categories where the user can choose between shortest path and minimum spanning tree problem. From there, the user can select the algorithm which he wants to call.

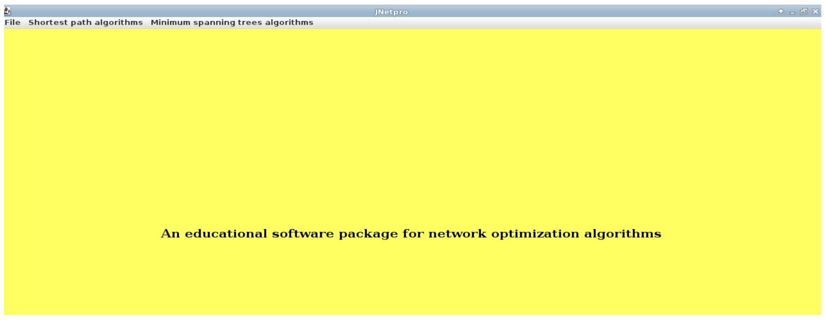
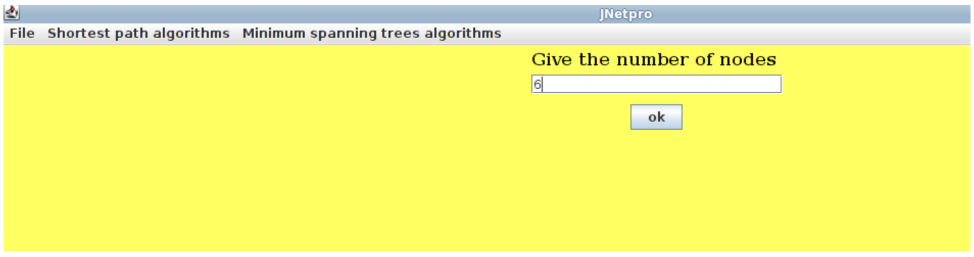

jNetpro
File
Shortest path algorithms
Minimum spanning trees algorithms
An educational software package for network optimization algorithms

Figure 4.1. Graphical User Interface (G.U.I.) of jNetPro.

As an example, assume that the user selects the Dijkstra algorithm. Afterwards, the user must enter the total number of nodes of the problem (e.g., 6 nodes), as shown in Figure 4.2.

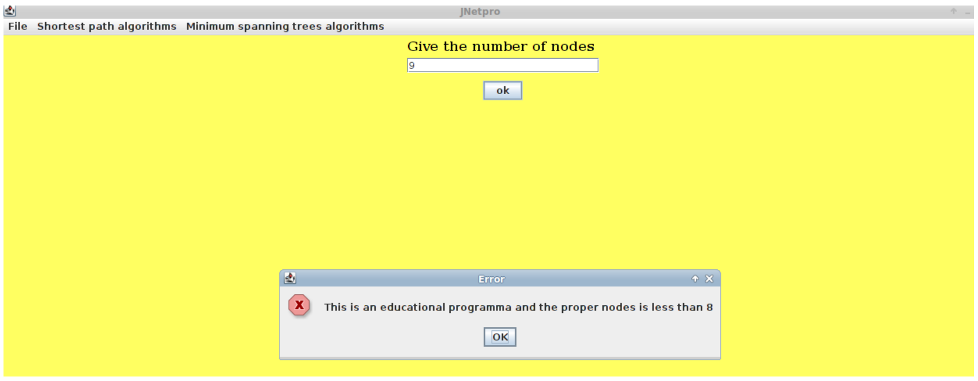


Give the number of nodes

6

Figure 4.2. Entering the Number of Nodes.

In this case, the program checks that the entered number is between two and eight nodes. Since the proposed optimization software package is for educational purposes, the application does not accept problems having more than 8 nodes. This is because the user needs to enter all the details by hand in text boxes. A warning message is shown to the user, if any of the above errors occur, as it is depicted in Figure 4.3.



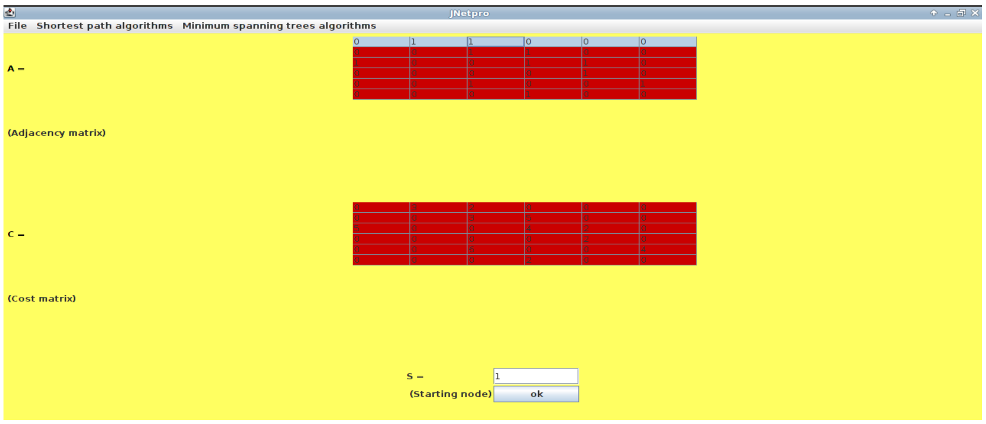
Give the number of nodes

9

This is an educational programma and the proper nodes is less than 8

Figure 4.3. A warning Message.

Next, the program creates two forms for entering the adjacency matrix and the corresponding weights and also a text box in order to state which is the starting node (e.g., node 1), as shown in Figure 4.4.



A =

0	1	1	0	0	0
0	0	1	1	0	0
1	0	0	1	1	0
0	0	0	0	1	0
0	0	1	0	0	1
0	0	0	1	0	0

(Adjacency matrix)

C =

0	3	2	0	0	0
0	0	3	5	0	0
5	0	0	4	2	0
0	0	0	0	2	0
0	0	6	0	0	4
0	0	0	2	0	0

(Cost matrix)

S = 1

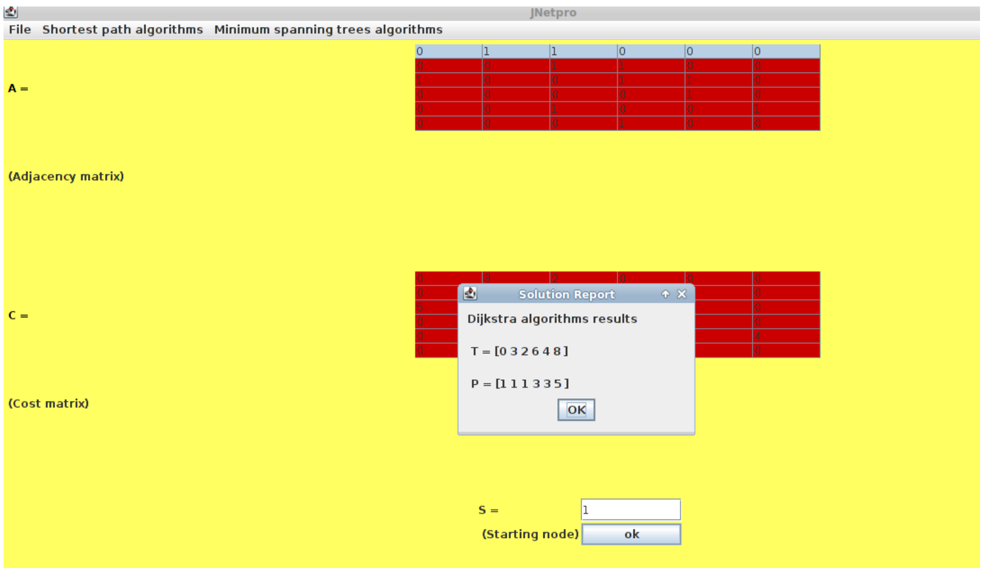
(Starting node)

Figure 4.4. Entering Problem Data.

The application informs the user for any error that he/she makes, with a warning message. The jNetPro does some checks before giving the results of the problem to the user. These checks due to the structure of the adjacency matrix are as follows:

- Checks if the user has entered the number 1 into the diagonal cells, (i.e., arcs of type (i,i)).
- Checks if the user has entered number different than 1 or 0.

The solution is displayed into a new window where the user is informed about the shortest distances from node 1 to all the others nodes, (as also the corresponding parent nodes), as shown in Figure 4.5.



0	1	1	0	0	0
0	0	1	1	0	0
1	0	0	1	1	0
0	0	0	0	1	0
0	0	1	0	0	1
0	0	0	1	0	0

Dijkstra algorithms results

T = [0 3 2 6 4 8]

P = [1 1 1 3 3 5]

S = 1

Figure 4.5. The Result of our Problem.

The jNetPro package provides the user with the ability to save and import or export the problem data. For example, a data file is saved locally with a name "JNetPro10.txt".

The number after the name jNetPro is a random number between 1 and 60. This way, the user may import the saved problem data without entering then from scratch. A saved problem data file has the following form:

- Algorithm used: The name of the algorithm which was used
- Number of nodes: The number of nodes of the problem
- Initial node: The starting node of the problem
- Array A: The adjacency matrix
- Array C: The weight matrix

The user can save a problem data file by selecting "Save problem file" from the

"File" menu. Afterwards, a pop-up window asks user for the path that the file will be saved, as it is depicted in Figure 4.6.

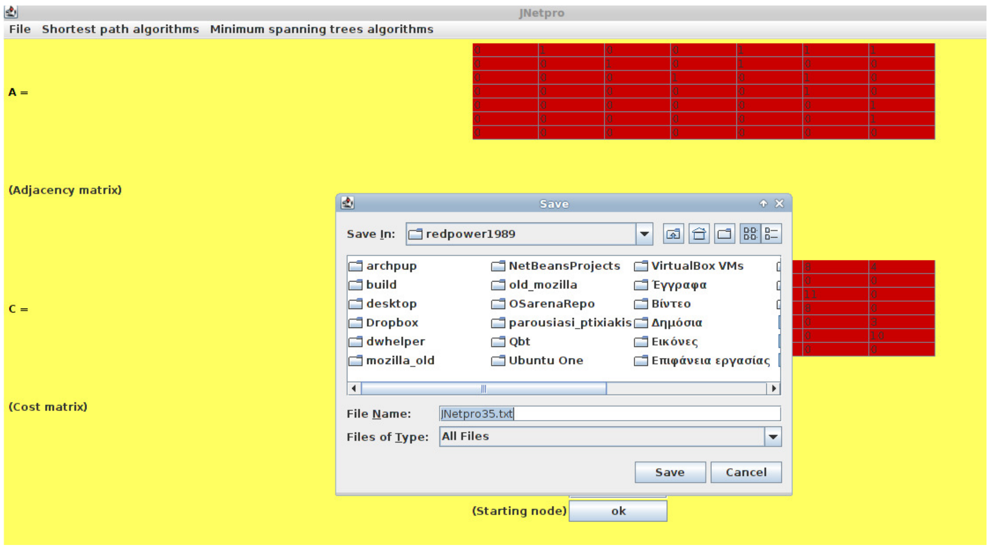
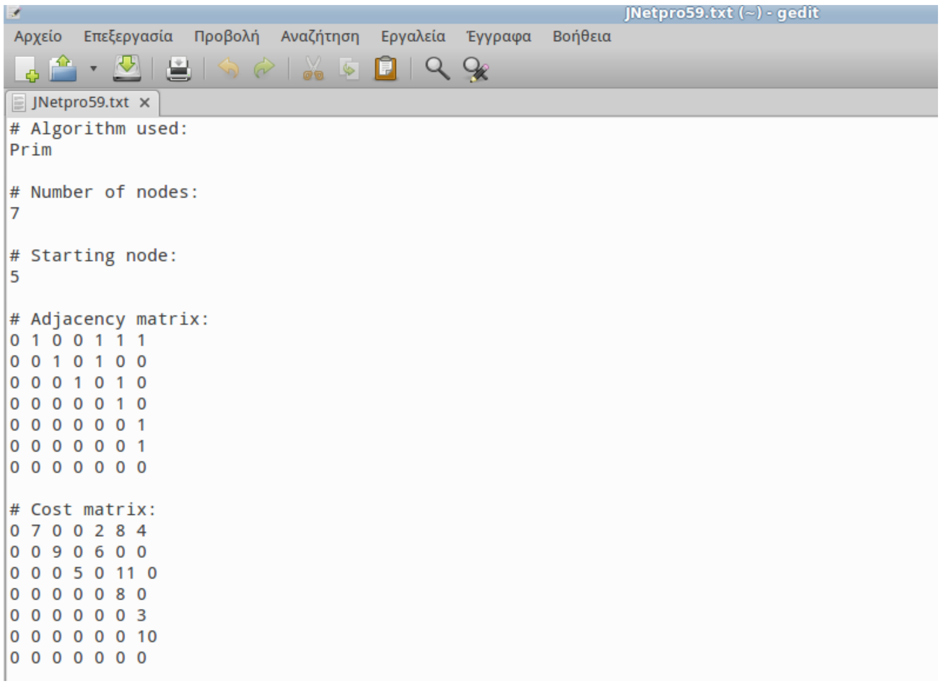


Figure 4.6. Saving a Problem data File.

An example showing the structure of a problem data file is depicted in Figure 4.7.



```
# Algorithm used:
Prim

# Number of nodes:
7

# Starting node:
5

# Adjacency matrix:
0 1 0 0 1 1 1
0 0 1 0 1 0 0
0 0 0 1 0 1 0
0 0 0 0 0 1 0
0 0 0 0 0 0 1
0 0 0 0 0 0 1
0 0 0 0 0 0 0

# Cost matrix:
0 7 0 0 2 8 4
0 0 9 0 6 0 0
0 0 0 5 0 11 0
0 0 0 0 0 8 0
0 0 0 0 0 0 3
0 0 0 0 0 0 10
0 0 0 0 0 0 0
```

Figure 4.7. Structure of a Problem data File.

Similarly, a user may import a problem data file by selecting "Import problem file" from the "File" menu.

5. Open Source Software and Green Economics

During the last years the open source software has significantly been developed. This growth attracts the interest of the business market in open source. Also, the number of open source software products diffuses at the market with the apache web server dominating the corresponding product category. It is estimated that the users of Linux-based open source operating system are between seven to twenty-one million users, with 200% growth rate. Nowadays, Linux is a leading potential challenger to Microsoft Windows in this important market of operating systems. A large number of well-known companies, (e.g., Hewlett Packard, IBM, and Sun) invest to open source software by funding R&D projects for the development of open source software. Also, the collaborative development of open source software is considered to be an important organization innovation.

An interesting point about the vast majority of the open source software developers is that they work without payment. However, those open source software products will be indeed improved, since the collaboration always assists in correcting errors. Most of the open source software developers, consider programming not only a form of intellectual work, but also as an art form. Moreover, several developers describe open source programming as artistic satisfaction associated to solving complex computer problems.

Furthermore, another important issue is that each open source software project has a core team of programmers. This team firstly identifies the needs of certain type of software and start discussing about it. Afterwards, this team of programmers calls the community and programmers all over the world for testing and correcting the program and proposes further improvements and modifications of the product. Once the core team has received adequate feedback from the community, it decides which ideas are best for implementation and then they add the new proposed functionalities to the product. Needless to say, internet plays a crucial role to this process, since the communication of all the developers is cheaper and easier than ever.

There are important economic benefits for business companies that adopt open source software products [1] [6] [13]. Some of those benefits are the low acquisition costs, low total cost of ownership, and great flexibility in choosing hardware and software products. Moreover, open source software products usually have a less number of bugs, since a large number of developers all around the world check them.

Thus, the majority of the bugs in open source software products are not crucial. To continue with, Linux-based operating systems are more secured and not so vulnerable in malicious software and viruses. Other, commercial operating systems have those types of problems and therefore business companies using commercial software products and/or operating systems have significant more expensive costs (e.g., need for commercial antivirus, firewall, etc.).

It is noteworthy that, a number of organizations such as Mozilla which develops the well-known Firefox browser have incomes from donations. The philosophy of the community of the users is that, if one software product is really very successful then, it would be nice to donate a small amount of money in order to support the development of this software. Additionally, other companies such as Canonical, charge their costumers for maintenance services. Although, a software product / operating system can be indeed open and free, charges could be made for services such as configuration and system maintenance. Therefore, business opportunities exist for software companies investing in open source software products.

An increasing number of companies, in public and/or private sector, and governments now prefer to use open source software products in order to reduce total cost. Moreover, the open source philosophy inspires other organizations such as Open Source Ecology[1], which is a network of people who want to build the Global Village Construction Set (GVCS). GVCS is an open technological platform allowing for the easy fabrication of the 50 different i*ndustrial machines* required to build a small civilization with modern comforts.

6. Conclusions and future work

The application jNetPro can be used by the students in order to help them understand these graph algorithms. By using this program, the student/user can easily and quickly solve problems. Also with the .deb package the user can install this application into a Debian-based operating system. A future task is the implementation of more solution algorithms to solve different type of network optimization problems [5] [11].

[1] Open source ecology: http://opensourceecology.org

References

[1] Bonaccorsi, A., and Rossi C. (2003), "Why Open Source software can succeed", *Research Policy*, 32(7), 1243–1258.

[2] Harris, J.M., Hirst, J.L., and Mossinghoff, M.J. (2008), *Combinatorics and Graph Theory*, (2nd edition), Oxford University Press.

[3] Dijkstra, E.W. (1959), "A note on two problems in connection with graphs", *Numeriskche Mathematik*, 1(1), 269–271.

[4] Dosios, K., Paparrizos, K., Samaras, N., and Sifaleras, A. (2003), "NetPro, an Educational Platform for Network Optimization", i*n Proc. of 16th National Conference of Hellenic Operations Research Society* (HELORS), Larissa, Greece, 1, 287–295.

[5] Geranis, G., Paparrizos, K., and Sifaleras, A. (2012), "On a dual network exterior point simplex type algorithm and its computational behavior", *RAIRO - Operations Research*, 46(3), 211–234.

[6] Jürgen B., and Schröder, P. (2006), *The Economics of Open Source Software Development*, Elsevier.

[7] Karagiannis, P., Markelis, I., Paparrizos, K., Samaras, N., and Sifaleras, A. (2006), "E – learning technologies: employing matlab web server to facilitate the education of mathematical programming", *The International Journal of Mathematical Education in Science and Technology*, 37(7), 765–782.

[8] Kruskal, J.B. (1956), "On the shortest spanning subtree of a graph and the travelling salesman problem", *in proc. of American Mathematical Society*, 7, 48–50.

[9] MacKinnon, J.G. (1999). "The Linux Operating System: Debian GNU/Linux", *Journal of Applied Econometrics*, 14(4), 443–452.

[10] Migdalas, A., Sifaleras, A., Georgiadis, C.K., Papathanasiou, J., and Stiakakis, E., (Eds.), (2013), *Optimization Theory, Decision Making, and Operations Research Applications*. In Series: *Springer Proceedings in Mathematics & Statistics*, 31, Springer, New York, USA.

[11] Paparrizos, K., Samaras, N., and Sifaleras, A. (2009), "An exterior Simplex type algorithm for the minimum cost network flow problem", *Computers & Operations Research*, 36(4), 1176–1190.

[12] Prim, R.C. (1957), "Shortest connection networks and some generalizations", *Bell Systems Technical Journal* 36, 1389–1401.

[13] Weber, S. (2005), "*The Success of Open Source"*, Harvard University Press.

[14] Wilson, R. (2004), "*Four colors suffice: How the map problem was solved"*, Princeton University Press.

[15] Zacchiroli, S. (2011), "Debian: 18 years of free software, do-ocracy, and democracy". *In Proc. of the 2011 Workshop on Open Source and Design of Communication* (OSDOC '11), ACM, New York, USA, 87–87.