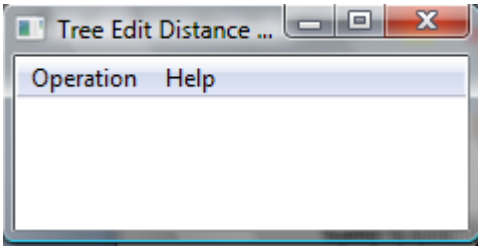
User Guide

By:Boris Pismenny

This user guide is meant to be used only for the Final TED software and for not any other.

Getting Started Menu

After running the software a menu will appear.



Under “Operation” there are 2 options Run & Test. Both of which are going to be covered next. In short Run means to run one of the Tree Edit Distance algorithms(DMRW, Klien, Shasha & Zhang) or in short TED on 2 trees. Using some distance metric which you may render.

Test basically means to run the 3 algorithms mentioned above on an amount of random trees and compare running time and results

(results of the compared algorithms should obviously be equal)

Basic Instruction

Tree Format

The trees should be stored in text files(*.txt) each tree is represented in the following way

(root-label(first-child)(second-child)(third-child)…)

Each child is a recursive structure like this one.

Examle:

(1(2(3)(4))(5(6)(7))) – this is a full binary tree of depth 2. Whose nodes are label according to the preorder traversal of the tree.

Another Example:

(1(2(3(4(5(6)))))) – this is a “spaggeti” tree whose root is one.

Note: the tree should be written in a single line no new lines are allowed. Spaces and tabs are allowed.

Distance Format:

The default distance metric is Delete Cost = 1 Match Cost = 1 if labels are different and 0 otherwise. Your customized distance function should be loaded from a text file(*.txt).

The format of a distance file is the following:

First comes the distance type (a more detailed explanation will be given soon) it should be “NORMAL” (written exactly like this) if you haven’t added your own distance type.

Then a new line, afterwards must come the “Match:” tag what comes afterwards and until the “Delete:” tag is your customized match costs. Each new line is another customized match cost. A customized match cost is of the following format:

Label1 | Label2 | New-Cost

So from now on the cost of matching Label1 with Label2 is New-Cost.

Then comes the “Delete:” tag everything that comes after it is your customized delete costs.

A customized delete cost is of the following format:

Label | New-Cost

From now on the cost of deleting Label is New-Cost.

Example:

```
NORMAL
Match:
a |b |5
Delete :
a| 2
b| 3
```

Here we changed the cost of matching “a” and “b” to 5. Also the cost of deleting “a” is now 2 and the cost of deleting “b” is now 3. All other labels delete and match costs are determined through the NORMAL distance metric meaning their cost is 1.

Note: It is very important that the format is exactly like this otherwise the behavior is unpredictable. Also no empty lines are allowed.

Editing The Distance Type:

To edit the distance type you need to edit the code in a few places.

First you must give your new edit type a name say we call it “CooKoo”.

So now that you have a name lets start changing the code. The first change is in the Distance.h file on the line: (add your type here)

```
enum DistanceType { NORMAL};
```

It should look like this:

```
enum DistanceType { NORMAL , CooKoo};
```

Now we need to change 2 functions and we are done. The functions are:

```
void Distance::Load(string& filename)

int Distance::DeleteNode(Tree* t1)

int Distance::MatchNodes(Tree *t1, Tree *t2)
```

We start with the first function, here the change is quite simple you should change this part:

```
if(line == "NORMAL")
	Dtype = NORMAL;
else
{
	cerr<<"Cannot Recognize Distance Type"<<endl;
	exit(1);
}
```

To be like this:

```
if(line == "NORMAL")
	Dtype = NORMAL;
else
{
	if(line == "CooKoo")
		Dtype = CooKoo;
	else
	{
		cerr<<"Cannot Recognize Distance Type"<<endl;
		exit(1);
	}
}
```

All we added here is a check if the line (the first in the file) is CooKoo meaning the distance type is CooKoo.

The change in the second function is the in this part:

```
if(Dtype == NORMAL)
   return 1;
```

Here you should add

```
if(Dtype == CooKoo)
```

```
{

Your distance calculations…

}
```

Your distance calculations means, for every time you delete a node and the delete cost of this node is not predefined then this code will run it should return the cost of deleting this node.

Note: If you need any more detailed information about other parts of the software you could find them in the Programmer's Manual.

The third and final change is much like the second one the part you need to change is:

```
if(Dtype == NORMAL)
{
      if(t1->getData()->getLabel() == t2->getData()->getLabel())
            return 0;
      else
            return 1;
}
```

Instead you should write:

```
if(Dtype == NORMAL)
{
      if(t1->getData()->getLabel() == t2->getData()->getLabel())
            return 0;
      else
            return 1;
}
if(Dtype == CooKoo)
{
      Your code here..
```

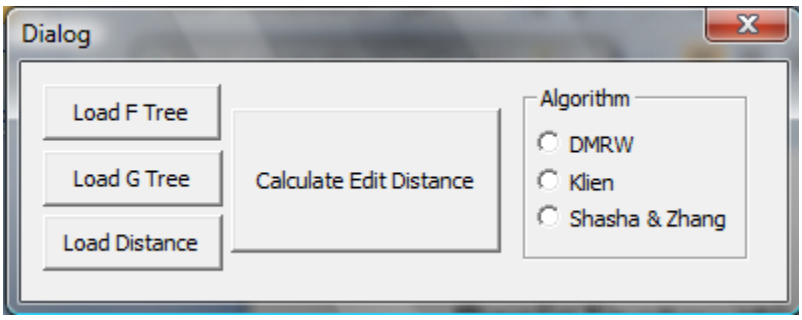
```
}
```

Your code here will run when you match two nodes. It should return the cost of matching the nodes.

Note: If you need any more detailed information about other parts of the software you could find them in the Programmer's Manual.

Run

After pressing the Run operation you will see the following dialog box.

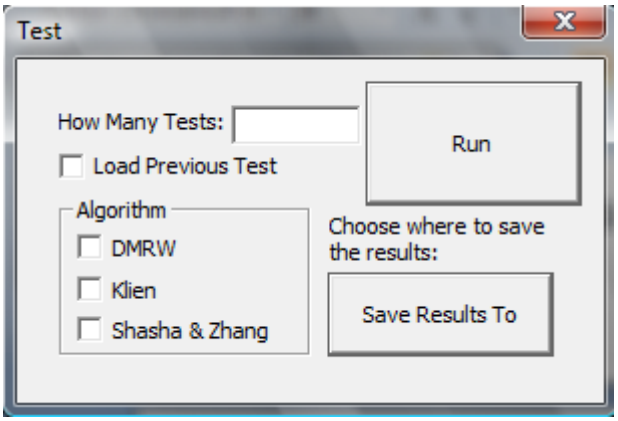


First you should load two trees F and G, of the format mentioned above. Afterwards you may load a distance function customized for your needs. If you wouldn't load one the default distance metric will be used. Also you must choose an algorithm to be used. Finally press

the Calculate Edit Distance button and the program will run until a result will be presented on the screen.

Test

After pressing the Test operation you will see the following dialog box.



Here you may run tests on the following 3 algorithms: DMRW , Klien , Shasha & Zhang. To do this you check the boxes of the algorithms which you will test . Specify the amount of tests to be run then specify where to save the results. Notice that all other data will be saved together with the results including running times in a file named “time.txt”. After you click “Run” the test will be preformed and when this

dialog box is closed then you know that the test is over. You can finish the test at any moment by closing the dialog yourself. To load a test check the “Load Previous Test” CheckBox and choose a place to save the results, this is also where your test files should be. When you create a new test then the amount specifies not only the amount of tests to be created but also the maximal size of a test tree. When running load amount specifies only how much tests to load.

Note: when you load a test the names of the files to be loaded must be “Test0F.txt” “Test0G.txt” “Test1F.txt” “Test1G.txt” etc… also notice that when you create a test the test trees will be saved under the same names and if you would like to investigate a specific test you could find it under its index in the path where you saved your results.