# User Guide
# to the ModuLand program package

(Version 1.3; June 2012)

(An algorithm package for the determination of a hierarchical community structure of networks with adjustable overlaps)

## *Changes in version 1.3*

With version 1.3 several optimizations were introduced, resulting in an increased performance (lower memory footprint and quicker runtime). As a part of this the output data file format was also changed to require less storage space. Due to this change the new output data files can not be opened with the earlier ModuLand program versions. However, all the changes are backward compatible, so the 1.3 ModuLand version handles all output data files created by the earlier versions. The main changes of version 1.3 are listed below:

- sparse matrix structure in the binary cxb and cxp output data files (resulting in smaller files)
- the GradientHill method described on page 37 of the supplement of the original manuscript is implemented
- all the programs are linked statically, so no additional libraries are required to run the ModuLand programs
- parallel running is supported for the PerturLand method
- many of the programs require less memory and/or less runtime.

## *Brief overview of the method family*

This User Guide describes the use of the algorithms of the ModuLand method family for uncovering overlapping modules. The ModuLand method family includes the following main steps (for more details see the related Manuscript and its Supplementary Material downloadable from **HERE**):

0. Converting the network data to the input format of the current implementation
1. Calculation of influence functions
2. Construction of the community landscape
3. Identifying modules and assigning elements and links of the network to modules.

For each step, a choice of methods is available to use. Though the algorithms we describe in this User Guide are both generally applicable and effective, we would like to note that each step of the process may be optimized further to the unique properties of the network. We offer appropriate clues for this optimization process in the Supplementary Material downloadable from **HERE**.

In our current implementation, networks are stored in **cxg** format, which is a custom binary format for the effective storage of network elements, links and their various attributes. We provide the **pajek_conv** tool to convert Pajek **net** files to **cxg** files.

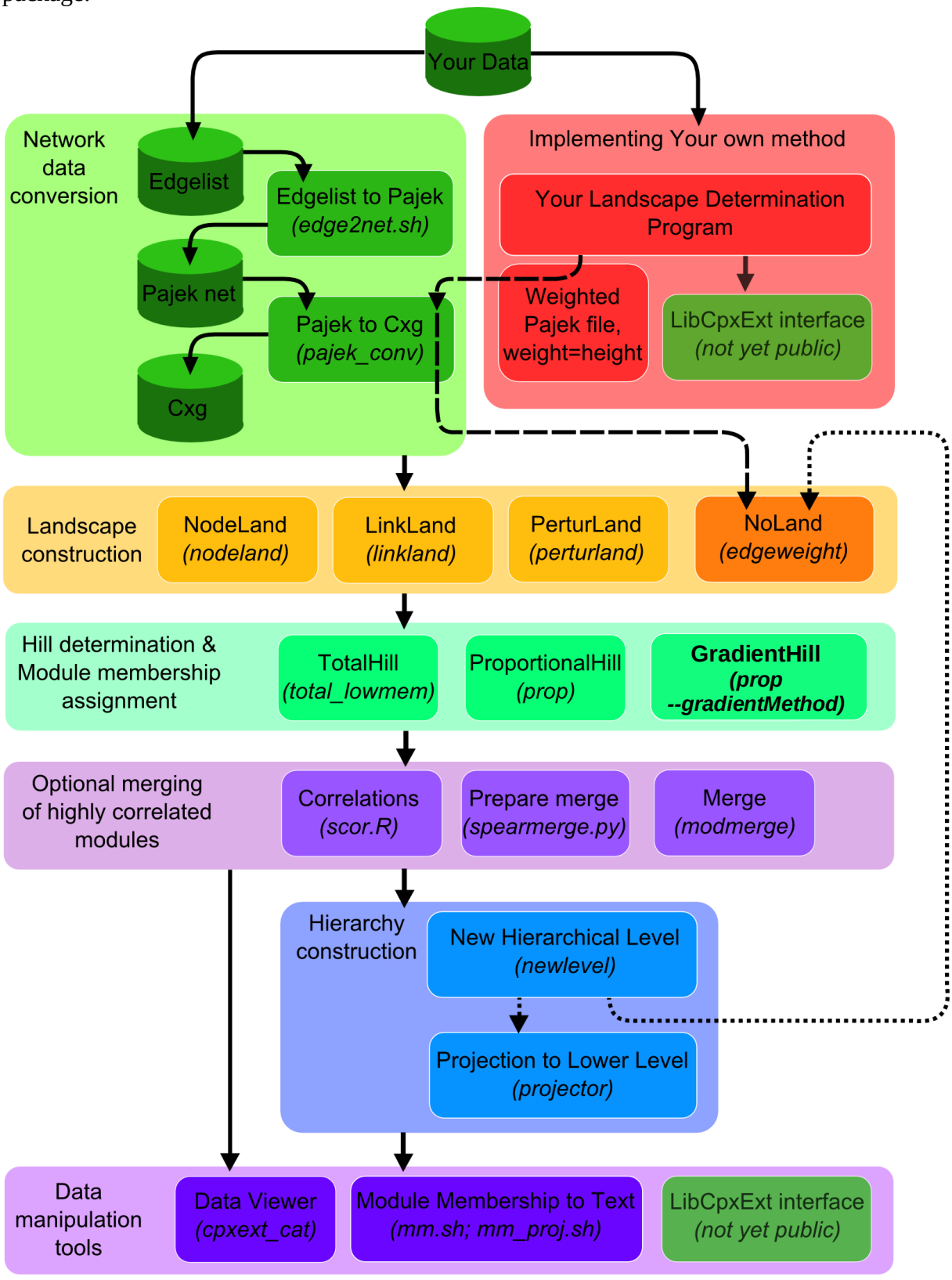
In the applications we offer the **NodeLand**, **LinkLand**, **PerturLand** and **EdgeWeight** (NoLand) influence function calculation methods, where steps 1 and 2 of the ModuLand method family have been merged. These influence function calculation methods take the network input data in **cxg** format, and directly output the community landscape data in a **cxl** format.

Step 3 may be performed using any of the **ProportionalHill, GradientHill,** or **TotalHill** module assignment methods. These methods take both the original network topology data in the **cxg** format and the community landscape data in the **cxl** file as inputs, and output the module membership assignment matrix of network elements in a **cxb** format. Since version 1.3, only the "ordered" versions of the hill determination programs are supported. The "ordered" versions differ from the original versions in that they output modules in the same order, that is: module *i* of the ordered ProportionalHill output will correspond to module *i* of the ordered TotalHill and ordered GradientHill output, while this was not the case for the non-ordered versions. This additional ordering property is useful for merging strongly correlated modules.

Optionally, the next level of the hierarchical representation of network modules – where the original modules became the elements – can be constructed using the **NewLevel** method, which requires the **cxg**, **cxl** and **cxb** input data, and outputs the network topology of the new level as a **cxg** file.

**Please note that a Linux/x86 compatible operating system is required for running the programs provided in the ModuLand program package. If you do not have access to such a system, you may use a prebuilt VirtualBox image of a Linux system with all necessary programs (please see the Appendix of this manual for instructions to use the VirtualBox image).**

The following flowchart summarizes the different phases and possible options of the ModuLand algorithm package. Cylinders represent data storage options, while boxes represent different operations. Box captions refer to the name of the operation, while the name in parentheses refers to the executable program name as found in the ModuLand program package.



## *Network data*

### Networks provided in the program package

We provide the network data we have analyzed in the related manuscript downloadable from **HERE**: in both **cxg** and Pajek **net** format. The following networks can be found in the *networks/* directory of the program package:

1. Word association network
2. School-friendship network
3. Electrical power-grid of the USA
4. Yeast protein-protein interaction network
5. Network science collaboration network
6. Zachary karate club social network

For the detailed description of these networks see Section I. of the Supplementary Information downloadable from **HERE**.

### Converting Pajek net network data to the cxg format with pajek_conv

After installing the programs of the enclosed program package downloadable from **HERE**, you can use the **pajek_conv** program to convert the network topological parameters in the Pajek **net** format to the **cxg** files used by the ModuLand method family by issuing the following command:

```
pajek_conv mynet.net mynet.cxg
```

where **mynet** is the name of the network file you have selected. You can inspect and verify the output by looking at the content of the **cxg** file using the program of **cpxext_cat** by issuing the command:

```
cpxext_cat mynet.cxg | head
```

### Converting network data in edgelist format to cxg format

After installing the programs of the enclosed program package downloadable from **HERE**, you can use the **edge2net.sh** program to convert an undirected edgelist to Pajek net format, and then use the **pajek_conv** program to convert the network topological parameters in the Pajek **net** format to the **cxg** files used by the ModuLand method family by issuing the following commands:

```
edge2net.sh mynet.edgelist mynet.net
pajek_conv mynet.net mynet.cxg
```

where **mynet** is the name of the network file you have selected. You can inspect and verify the output by looking at the content of the **cxg** file using the program of **cpxext_cat** by issuing the command:

```
cpxext_cat mynet.cxg | head
```

The edgelist file is assumed to have two integers per line separated with a whitespace character, where the integers refer the node numbers, starting from either zero or one.

## *Influence function calculation and community landscape construction methods*

You may use the provided algorithms describing the implementation of the NodeLand, LinkLand or PerturLand methods to determine the community landscape of your network. Alternatively, you may use the EdgeWeight method, if you simply want the community landscape heights to be the link weights of the original network. The EdgeWeight method is also useful for the inclusion of any of the community landscape determination methods you wish to try in the ModuLand method family framework.

### The NodeLand and LinkLand methods

After installing the programs of the enclosed program package downloadable from **HERE**, you can execute the NodeLand or LinkLand methods by issuing the following commands:

```
nodeland -0 mynet.cxg mynet.cxl
```

or

```
linkland -0 mynet.cxg mynet.cxl
```

where **mynet** is the name of the network file you have selected. The **nodeland** or **linkland** programs take the **cxg** format network data as input, and output the community landscape data to the **cxl** file. The **-0** (minus zero) switch prevents the programs to write detailed influence function data to the output **cxl** file.

### The PerturLand method

After installing the programs of the enclosed program package downloadable from **HERE**, you can execute the PerturLand method by issuing the following command:

```
perturland mynet.cxg <X> <higher level: 0/1> mynet.cxl <heapbased method: 0/1>
```

where **mynet** is the name of the network file you have selected. The program takes the **cxg** format network data as input, and outputs the community landscape data to the **cxl** file. The **<X>** parameter describes the attenuation of the information perturbation during the propagation process. **<X>** can be in the range of [0;1]. The higher the value of *X*, the smaller the attenuation of the perturbation is spreading in the network (for details see Section IV. 2. of the Supplementary Information downloadable from **HERE**). The **<higher level: 0/1>** parameter can be set either 0 or 1, telling the program, if the supplied network is an original network [**<higher level: 0/1>** = 0] or any higher level of the network hierarchy constructed by the ModuLand method [**<higher level: 0/1>** = 1], respectively. In the latter case, the *X* parameter has no effect (for explanation of this see Section IV. 2. of the Supplementary Information downloadable from **HERE**). The <**heapbased method: 0/1**> parameter must be set to 0 in all cases.

From the ModuLand version 1.3, you can run the perturland program parallel on different machines. In each machine you can specify a node region where the program will calculate the influences:

```
perturland mynet.cxg <X> <higher level: 0/1> mynet.cxl <heapbased method: 0/1> [<start nodeId> <end nodeId>]
```

After all the sub-landscapes are calculated, you can combine them into the final landscape by using the landscape_union command. In the following example we will create the landscape of the word association network contains 10617 nodes on three different machines.

<copy the network file (assoc.cxg) to the three different machine>

run on machine 1: **perturland assoc.cxg 0.05 0 assoc-part1.cxl 0 0 3000**

run on machine 2: **perturland assoc.cxg 0.05 0 assoc-part2.cxl 0 3001 6000**

run on machine 3: **perturland assoc.cxg 0.05 0 assoc-part3.cxl 0 6001 10616**

<copy the second and the third landscape parts (assoc-part2.cxl, assoc-part3.cxl) to the first machine>

run on machine 1: **landscape_union assoc.cxg assoc.cxl assoc-part1.cxl assoc-part2.cxl assoc-part3.cxl**

< the combined landscape can be found in the assoc.cxl file>

### The EdgeWeight method

After installing the programs of the enclosed program package downloadable from **HERE**, you can execute the EdgeWeight method by issuing the following command:

```
edgeweight --in mynet.cxg --out mynet.cxl
```

where **mynet** is the name of the network file you have selected. The program takes the **cxg** format network data as input, and outputs the community landscape data to the **cxl** file. In the **cxl** data the link weights of the original network become the community landscape heights of the links.

For networks of higher hierarchical levels the EdgeWeigth method should be used as influence function construction method (for explanation of this see Section VII. 2. of the Supplementary Information downloadable from **HERE**).

You can also supply your own community landscape data by writing the community landscape heights to a Pajek **net** file as link weights, using **pajek_conv** and finally the EdgeWeight program to assemble the community landscape data in **cxl** format.

## *Module identification and module membership assignment methods*

### The ProportionalHill, GradientHill and TotalHill methods

The ProportionalHill, GradientHill and TotalHill module membership assignment programs take the **cxg** format network data and the **cxl** format community landscape data as inputs, and output the module membership assignment matrix of the network elements in a **cxb** format.

After installing the programs of the enclosed program package downloadable from **HERE**, you can execute the ProportionalHill, GradientHill or TotalHill method by issuing the following commands:

```
prop mynet.cxg mynet.cxl mynet.cxb H  --noEdgeBelong
```

or

```
prop mynet.cxg mynet.cxl mynet.cxb H --noEdgeBelong --gradientMethod
```

or

```
total_lowmem --no-edge-belong -m 900 mynet.cxg mynet.cxl mynet.cxb
```

where **mynet** is the name of the network file you have selected. The **H** parameter of the ProportionalHill program ensures that the module membership assignment vectors of the network links will be normalized to the community landscape height of the links. Note: One could substitute '1' or 'W' instead of 'H' in order to normalize the module membership assignment vector of any given link either to the sum of one or to the sum of the original weight of the given link ('W'), respectively. The **--noEdgeBelong** parameter of the **prop** program prevents writing the module membership assignment vectors of the links to the output **cxb** file – it is advised to use this option, as the assignment vectors of links are not required in later steps and their inclusion would needlessly enlarge the output file size.

The GradientHill method can be chosen by giving the extra **--gradientMethod** parameter to the program **prop**.

The **--no-edge-belong** parameter of the TotalHill program prevents writing the module membership assignment vectors of the links to the output **cxb** file – it is advised to use this option, as the assignment vectors of links are not required in later steps and their inclusion would needlessly enlarge the output file size. The **-m 900** parameter instructs the program not to use more than 900 Megabytes of memory. Select this number according to the available memory of your computer, where half of the total free memory may be a safe choice.

### Extracting module membership data

While the function library for directly manipulating the **cx*** files of the ModuLand program package (such as reading or writing their content) is not available yet, you can use the **mm.sh** script in combination with the **cpxext_cat** tool provided in the enclosed program package downloadable from **HERE** to extract the module membership assignment matrix from a **cxb** file by executing the following command:

```
cpxext_cat mynet.cxb | mm.sh > module_node_mat.txt
```

where **mynet** is the name of the network file you have selected. The **mm.sh** script utilizes the standard GNU Awk program available in standard Linux distributions. The **module_node_mat.txt** output file will contain as many rows as the number of network modules, with as many comma-separated float values in each row as the number of network elements. The float value in row *i* and column *j* of the **module_node_mat.txt** output file is the membership strength of element *j* in module *i*.

## *Merging strongly correlated modules*

### Determining the module-module correlation matrix

When applying the PeakHill based ProportionalHill, GradientHill or TotalHill module membership assignment methods on a 'noisy' community landscape, each local maxima will result a new separate (and possibly highly overlapping) module. Therefore it can be useful to apply a simple yet effective post-processing step for merging the groups of extremely overlapping modules. To this end the first step is determining the similarity, or correlation between any two modules.

To determine the module-module correlation matrix based on the **.cxb** output of a given module membership assignment method, first you have to convert that output into a textual matrix form using the **mm.sh** script as seen in the previous section. Assuming that the name of the textual matrix file is **module_node_mat.txt**, perform the following steps:

```
ln -s module_node_mat.txt mat.tmp
scor.sh
rm mat.tmp
```

where the **ln -s** Linux command creates an alias of the textual matrix file, calling the alias "**mat.tmp**". This is required because the **scor.sh** script expects its input file to be named "**mat.tmp**". Then the **scor.sh** script calculates the module-module correlation matrix and writes it into an output file named **scor.mat**. Internally the **scor.sh** script utilizes the R statistical program to execute the calculations. After this, the **rm** command erases (removes) the alias because it is no longer needed.

Now optionally a histogram can be created to picture the relative frequency of module-module correlation values using the following command:

```
dohist.sh
```

which takes the **scor.mat** file as input and creates the **hist.pdf** file as output, which can be viewed with a standard pdf viewer.

### Creating a list of module groups to be merged

Once the **scor.mat** file described in the previous section is available, execute the **spearmerge.py** script using:

```
spearmerge.py scor.mat 0.9 full > merge_list.txt
```

where **0.9** is the correlation threshold above which groups of modules will be merged and the parameter "**full**" denotes that the output should list not only module groups to be merged, but also individual modules which are left intact. The name of the output file is **merge_list.txt** in this example, and contains a group of modules to be merged per line, each module denoted by its number, starting from zero.

### Merging the correlated groups of modules

Having the **merge_list.txt** file prepared as the end of the process described in the previous section, the groups of modules can actually be merged by the command:

```
modmerge mynet.cxb merge_list.txt mynet_merged.cxb
```

where **mynet.cxb** is a module membership assignment data file for the given network and **mynet_merged.cxb** is the output module membership assignment data file with the groups of modules listen in **merge_list.txt** merged.

Note that it is possible to merge module membership assignment data produced by the ordered TotalHill method based on the module-module correlation data resulting from processing the ordered ProportionalHill module membership assignment data. The reason why one would want to do this is that the excessive amount of module-module overlap produced by the TotalHill method results in excessive module-module correlations, which makes it hard to determine a feasible correlation threshold based on this data alone.

## *Higher level networks*

### Creating a higher level network

It is possible to create a higher level network based on the overlapping module information of the original network: Modules of the original network will be the elements of the new network, and the links between the elements of the new network will represent the overlaps between the modules of the original network.

The NewLevel program determining the immediate next level of the hierarchical network structure defined by the above definition takes the **cxg** format network topology data, the **cxl** format community landscape data and the **cxb** format module membership assignment matrix as inputs, and outputs the higher level network topology data to a **cxg** format data file.

After installing the programs of the enclosed program package downloadable from **HERE**, you can execute the NewLevel method by issuing one of the following commands:

```
newlevel mynet.cxg mynet.cxl mynet.cxb mynet-1.cxg
```

or

```
newlevel --threshold Y mynet.cxg mynet.cxl mynet.cxb mynet-1.cxg
```

where **mynet** is the name of the network file you have selected. The second way of execution differs in supplying the additional --**threshold Y** parameter, where Y is a percentage-like number between 0.0 and 100.0. By default, this **Y** threshold parameter is set to zero, meaning that even the smallest overlaps between the modules of the original network are taken into account, which may result a network at the immediate next level of the hierarchy with as many as $M^2$ links, where M is the number of modules of the original network, if the module overlaps have been very extensive. The inclusion of a non-zero **Y** threshold value (in percentage) will cause that the value of all components of those module membership assignment vectors will be set to zero, which do not reach the **Y** threshold percentage of the sum of all components. In case of a non-zero **Y** threshold parameter the calculation of modular overlaps takes place only after the pruning step described here. It is especially advised to use at least a small value of the **Y** parameter, e.g. 0.1%, or 1%, if the module membership assignment matrix has been determined using the TotalHill method, which results in extensive and detailed overlaps of the network modules. After determining the modules of both the original and any hierarchy levels of the higher level networks the module membership assignment data will be stored in the **mynet.cxb**, **mynet-1.cxb**, ..., **mynet-k.cxb** files, respectively, where **mynet** is the name of the network file you have selected, and **k** refers to the number of the highest level of the hierarchy you wanted to determine (obviously this should be at least one level below the highest level of the hierarchy, where all modules of the original network are represented by a single element).

### Projecting module assignment of higher levels to the original network

After determining the modules of both the original and any hierarchy levels of the higher level networks and storing the module membership assignment data in the **mynet.cxb**, **mynet-1.cxb**, ..., **mynet-k.cxb** files, respectively, you can project the module membership assignment of any intermediate level *i* ($0 \leq i \leq k$) or even back to the original 0 level by executing the following command (after installing the programs of the enclosed program package downloadable from **HERE**):

```
projector mynet.cxb mynet-1.cxb [...] mynet-i.cxb mynet-0-i.cxp
```

where **mynet** is the name of the network file you have selected. The output file, **mynet-0-i.cxp** contains the module membership assignment data of the elements of the original network assigned to the modules of the network hierarchy level *i*.

### Extracting projection data

Similarly to the extraction of the module membership assignment data from **cxb** format files described earlier, you can use the **mm_proj.sh** script provided in the program package downloadable from **HERE** in combination with the **cpxext_cat** tool to extract the module membership assignment matrix from a **cxp** file containing the module membership assignment data of the higher hierarchical level projected to the original network elements by executing the following command:

```
cpxext_cat mynet-0-i.cxp | mm_proj.sh > module_node_mat.txt
```

where **mynet** is the name of the network file you have selected. The **mm_proj.sh** script utilizes the standard GNU Awk program available in standard Linux distributions. The **module_node_mat.txt** file will contain as many rows as the number of network modules on network level *i*, with as many comma-separated float values in each row as the number of network elements in the original network. The float value in row *j* and column *k* of the **module_node_mat.txt** output file is the membership strength of element *k* in module *j* at network level *i*.

## *Examples*

### A complete example

In this example, we will convert an original network data in a Pajek **net** format to a **cxg** format, determine the modules of the original network, create the immediate next layer of the higher level networks, determine the modules of this immediate next layer of the higher level networks, and finally project the modules of the immediate next higher level network to the elements of the original network.

```
pajek_conv mynet.net mynet.cxg
nodeland -0 mynet.cxg mynet.cxl
total_lowmem --no-edge-belong -m 900 mynet.cxg mynet.cxl mynet.cxb
cpxext_cat mynet.cxb | mm.sh > module_node_mat-0.txt

newlevel --threshold 0.01 mynet.cxg mynet.cxl mynet.cxb mynet-1.cxg
edgeweight --in mynet-1.cxg --out mynet-1.cxl
total_lowmem --no-edge-belong -m 900 mynet-1.cxg mynet-1.cxl mynet-1.cxb

projector mynet.cxb mynet-1.cxb mynet-0-1.cxp
cpxext_cat mynet-0-1.cxp | mm_proj.sh > module_node_mat-0-1.txt
```

After the execution of all the above commands, the results of the modular analysis will be in the following files:

- *mynet.cxg*: network topology in the cxg format
- *mynet.cxl*: community landscape data in cxl format
- *mynet.cxb*: module membership data in cxb format
- *module_node_mat-0.txt*: module membership data in standard text format
- *mynet-1.cxg*: hierarchical level 1 network topology in the cxg format
- *mynet-1.cxl*: hierarchical level 1 network community landscape data in cxl format
- *mynet-1.cxb*: hierarchical level 1 network module membership data in cxb format
- *mynet-0-1.cxp*: module membership data as projected from the modules of the level 1 network to nodes of the original network, in cxp format
- *module_node_mat-0-1.txt*: projected module membership data in standard text format.

### An example for merging strongly correlated groups of modules

This example takes a Pajek **.net** network dp2.net as input, converts it to **.cxg** format, constructs a LinkLand-based community landscape, determines both the ProportionalHill and the TotalHill module membership assignment data, and merges the modules of the TotalHill module membership assignment data based on the strongly correlated modules of the ProportionalHill module membership assignment data. Finally, it prints the number of modules before and after the merger.

```
pajek_conv dp2.net dp2.cxg
linkland -0 dp2.cxg dp2.cxl
prop dp2.cxg dp2.cxl dp2_fps.cxb H noedge
total_lowmem --no-edge-belong -m 128 dp2.cxg dp2.cxl dp2_ts.cxb

cpxext_cat dp2_fps.cxb | mm.sh > mat_fps.txt

ln -s mat_fps.txt mat.tmp
scor.sh
rm mat.tmp

dohist.sh

spearmerge.py scor.mat 0.9 full > merge_list_fps_09.txt
modmerge dp2_ts.cxb merge_list_fps_09.txt dp2_ts_merged.cxb
cpxext_cat dp2_ts.cxb | head -n6
cpxext_cat dp2_ts_merged.cxb | head -n6
```

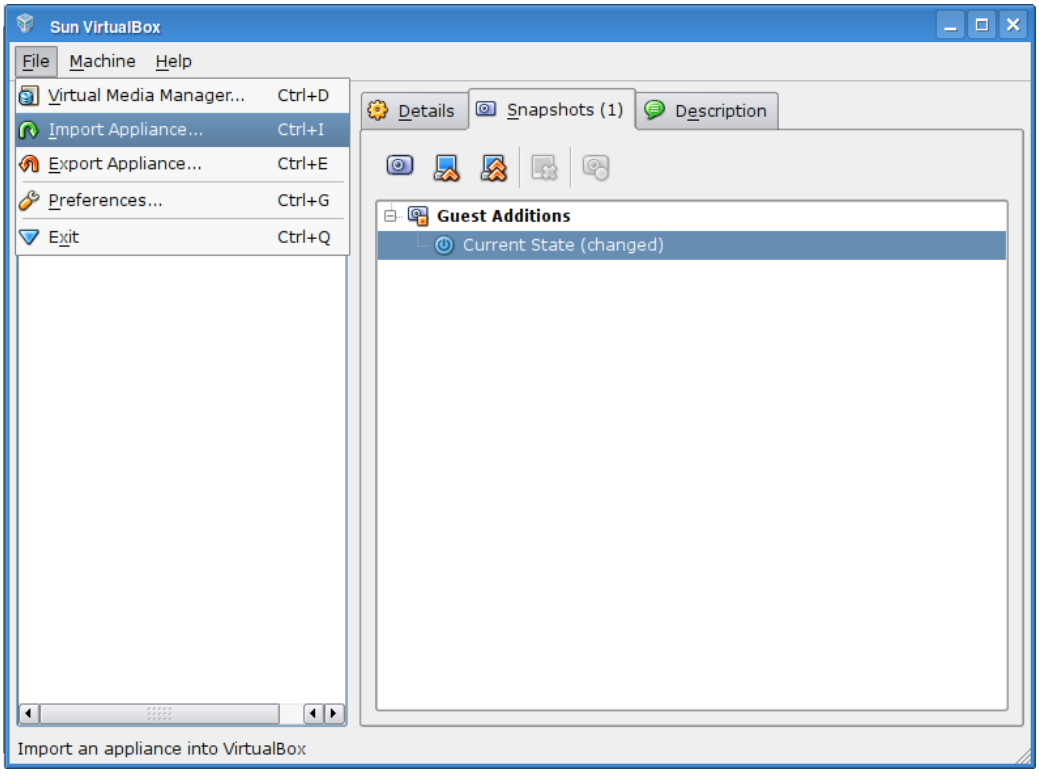
## *Appendix: VirtualBox image for running the ModuLand programs*

### Setting up VirtualBox and the ModuLand Image

VirtualBox is a free virtualization technology for running a virtual computer inside your real computer. First, please download and install the latest VirtualBox from http://www.virtualbox.org/wiki/Downloads. Please also see if your computer meets the requirements for running VirtualBox (a recent computer should have no problems coping with this task).

After downloading and installing the VirtualBox program as shown above, at the registration for the ModuLand program package **HERE**, you will receive a confirmation email also containing a link to the zip-compressed ModuLand VirtualBox image which makes the interface between the VirtualBox program and the ModuLand programs. The zip file contains a ModuLand.ovf[1] and a vmdk file. Extract these files to a suitable location.

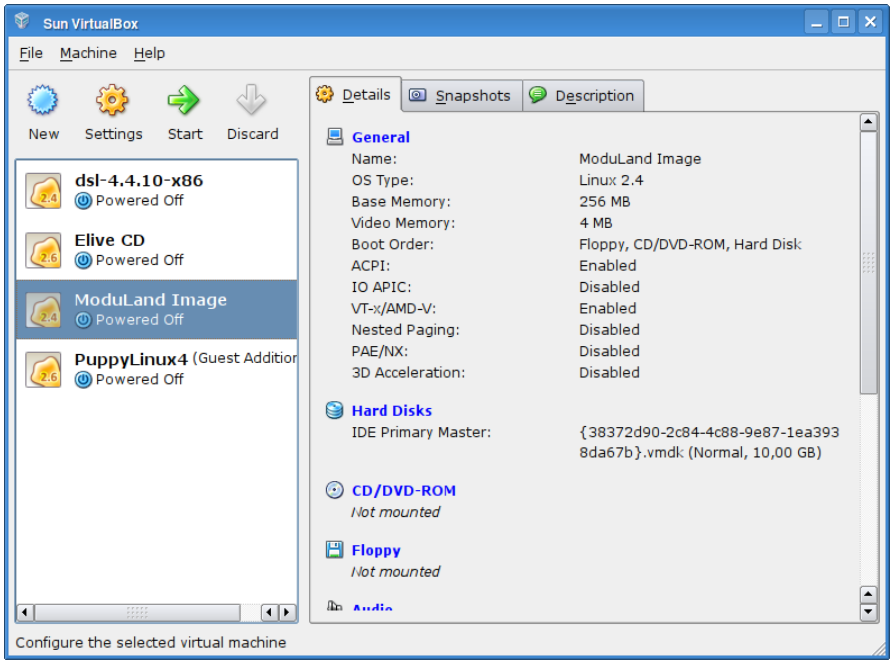
Start the VirtualBox program (which has been previously downloaded from http://www.virtualbox.org/wiki/Downloads), and choose *Import Appliance* option from the *File* menu as shown below.




Choose the extracted ovf file and follow the process of importing, accepting the default values. Importing may take some minutes to finish. After the import process is done, the

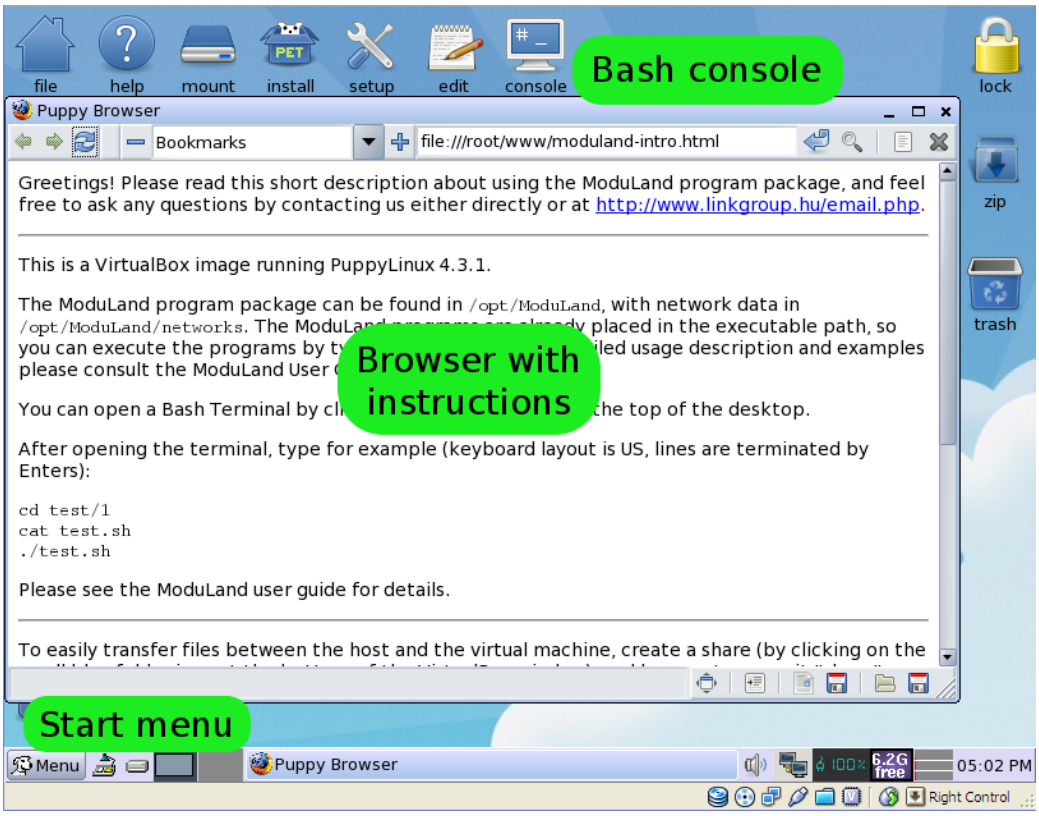
[1]ovf stands for Open Virtualization Format. If you are already familiar with a virtualization software capable of using ovf images (such as VMWare), you may also use that software for importing and running the image. However, here we restrict ourselves to only give instructions for the free VirtualBox software. Please note that the ovf image provided here has certain convenience features (via the VirtualBox Guest Additions) installed which are only available when using VirtualBox.

"ModuLand Image" is listed under your images, and you can start the virtual machine, as shown below.



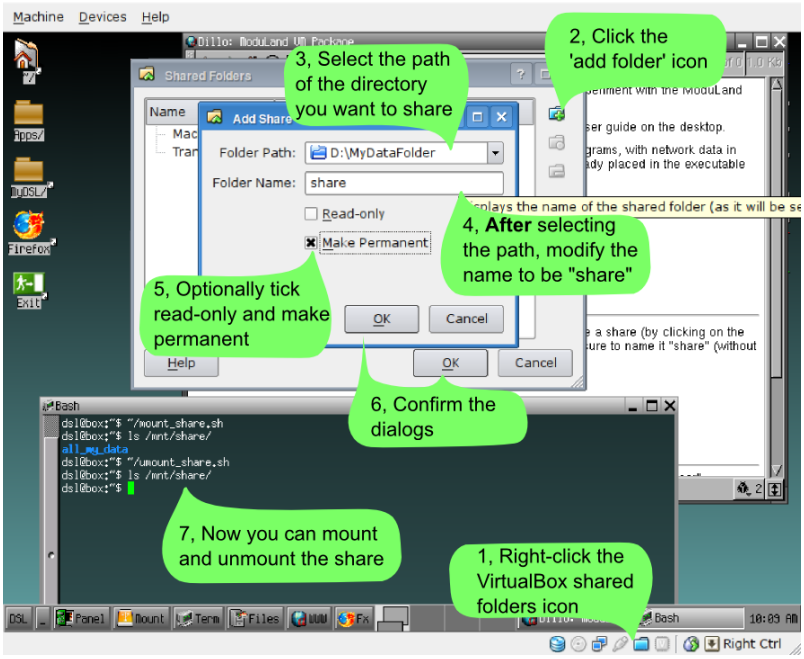

## Getting started with the ModuLand Image

After the Virtual Machine has booted, a small browser window will open with usage instructions. Important elements of the user interface are shown in the picture below.



## Sharing a folder between your computer and the virtual computer

You can share a folder of your computer with the virtual computer (either with just reading or with read and write support) for easing the transfer of your data from and to the virtual computer. The process is shown in the picture below. Please also read the instructions presented in the virtual computer after startup.




## Stopping the Virtual Computer

After you have finished you work, please shut down the virtual computer using the *Menu* > *Shutdown* > *Power-off computer* option as shown in the screen below.