1 General description of the projects

The term projects are aimed at providing a test ground for the skills and knowledge you picked up during the course and optionally give you a chance to join ongoing network research. You are expected to give a 5-8 minute long presentation of your project with a few slides on the last lesson of the course.

There are two type of projects you can choose from: simple projects and research projects. The simple projects in most cases correspond to numerical studies of one of the problems discussed during the course. To fulfill a simple project you have to go through basically the following steps:

- understand the problem posed, (i.e., dig up the slides from the course related to the problem, look up in Wikipedia, etc.),
- write/download the program(s) needed to carry out the project,
- calibrate and run your programs: in some cases parameters have to be tuned, in other cases the same routine has to be re-run several times with different parameters, etc.
- prepare figures showing the results,
- prepare your slides for the presentation. In your presentation you should
  - explain the problem to the others,
  - outline how you solved it,
  - show the results.

The research projects are more complex, need much more work (along with frequent consultations with the project leader), and are not expected to be finished by the end of the course. (However, a significant progress should be still achieved). A student choosing a research project could continue working on the problem in contact with the project leader even after finishing the AIT course, and in the ideal situation the outcome of such a collaboration is a research publication in a peer reviewed scientific journal. The presentation of a research project in the AIT course should explain the problem posed, detail the progress made, and describe the future plans related to the project.

2 The problems

2.1 Simple projects

1. Write a program that can randomize a network,

   - by preserving only the number of nodes and links,
   - or by preserving the degree sequence of the nodes,
   - or by preserving the degree sequence and turning the network to assortative or dissasortative depending on the choice of the user.

   Demonstrate how the program works on the network examples used in the practical sessions: show a few figures of the networks before and after the randomization, and show a figures of the changing of some network parameters during the randomization.

   (Project leader: Gergely Palla)

2. Write a program that can randomize a network by preserving the degree sequence and enhancing or decreasing the frequency of a chosen motif consisting of 3 nodes. (The user should be able to change the motif and to decide whether its frequency should be increased or decreased).

   Demonstrate how the program works on the network examples used in the practically sessions: show a few figures of the networks before and after the randomization, and show a figures of the changing of some network parameters during the randomization.

   (Project leader: Gergely Palla)

3. Compare random graphs generated with the Holme-Kim model and with the configuration model. Set the number of nodes and links to be the same, and also the degree distribution to be the same. Is there any difference between the behavior of the clustering coefficient, the behavior of the average nearest neighbors degree, and the closeness?

   In the presentation show pictures of the networks generated by the two different algorithm, and also present figures showing the examined statistics (e.g., the clustering coefficient as a function of degree, the k_{nn} as a function of degree, etc.).

   (Project leader: Gergely Palla)

4. Write a program that can extract the triad motif significance profile of an input network. Compare the motif profile of the networks used in the practical session.

   In the presentation (beside the motif significance profiles) show how the frequency of a given motif is changing when randomizing the network, and also snapshots of the networks before and after the randomization.

   (Project leader: Gergely Palla)

5. Study the percolation transition of the E-R graph numerically. How does the size of the largest connected component scale with the system size at the critical point? How does the size of the largest connected component grow with p at the critical point?

   In the presentation show pictures of the generated graphs both below and above the critical point, and also figures revealing the scaling of the largest connected component at p_c.

   (Project leader: Gergely Palla)

6. Study the average shortest path length in the W-S model numerically. If p is kept fixed and N is varied, how does the critical N_c (where the small world effect takes place) scale with p? Reproduce the data collapse for $\langle l \rangle$ discussed in the course.

   Include pictures of the generated graphs both below and above the critical point in the presentation, and show figures depicting the scaling and the data collapse.

   (Project leader: Gergely Palla)

7. Calculate the distribution of the node visiting frequency of a random walker on a graph! Use some real network examples from the course and use some randomly generated networks! Try to compare the results with the topological properties of the graph (degree, clustering, betweenness, motif and communities)!

   (Project leader: Péter Pollner)

8. Compare the networks used in the practical sessions according to the resilience against random breakdown and attack! Analyze the problem of resilience on the corresponding random graphs as well. Use an ensemble of ER graphs and the randomly rewired graphs.

   (Project leader: Péter Pollner)

9. Implement the Girvan-Newman algorithm and test it on the "classical" computer generated test bed and on the networks used in the practical sessions.

   Details:

   - The computer generated test bed is defined as follows: consider 4 groups of nodes with 32 members each, these correspond to the communities the algorithms should find. The links are drawn between the nodes at random, however the average number of links going from one node to other nodes in its own community is given by $z_{\rm in}$, whereas the average number of links going to the rest of the network is given by $z_{\rm out}$, and $z_{\rm in}$ and $z_{\rm out}$ always add up to $z_{\rm in} + z_{\rm out} = \langle k \rangle = 16$. When $z_{\rm out}$ is small and $z_{\rm in}$ is close to 16, the algorithms have an easy job, the communities can be spotted even by eye, since they the links are very dense inside and occur very rarely in between the communities. However, as $z_{\rm out}$ is increased the communities become less and less well defined, and around $z_{\rm out} = 8$ the graph becomes homogeneous.
   - The evaluation of the community finding results is based on a matching between the predefined 4 groups and the found communities. Choose the matching which gives the highest number of correctly classified nodes. (E.g., index the original groups by A, B, C, D, then index the found communities also with A, B, C, D, and search for the permutation of these indices maximizing the number of nodes on which the two community index is the same).
   - In the presentation plot the ratio of correctly classified nodes over the total number of nodes as a function of $z_{\rm out}$. Also show figures about the found communities in the test graph and in the real networks.

   (Project leader: Gergely Palla)

10. Implement a greedy optimization of the modularity Q and test it on the "classical" computer generated test bed and on the networks used in the practical sessions.

    The outline of the greedy optimization is the following:

    - In the initial state each node is considered to be in a separate community.

- At each step merge a pair of communities into a single community. Select the pair for which the increase in Q resulting from the merging is maximal.
- Carry on until the network contains only a single community. Choose the partition with highest Q value in the process.

The further details are the same as in the previous problem.

(Project leader: Gergely Palla)

11. Treat directed, weighted networks as pipelines, where fluid can flow from one node to the other. The fluid moves on the links only into the direction of the link, and the amount of the flow on a link is proportional to the link-weight.

    Demonstrate this dynamics on some small example networks! Start from some initial density distribution of the amount of the fluid on the nodes, and show how this distribution is changing over time. Find the characteristic time scales of the dynamics and examine how the distribution approaches its infinite time limit!

    Note: use only strongly connected networks for the demonstration, and discuss shortly what do you expect if there are weakly connected parts as well.

    For the dynamics you assume the following:

    - The dynamics is discrete in time.
    - At each time step:
      - The fluid at each nodes is in an "outgoing container".
      - The full amount of the fluid leaves the "outgoing container" and arrives in the empty "incoming containers" of the neighbors.
      - Then the fluid flows from the incoming container to the empty outgoing container.

    (Project leader: Péter Pollner)

2.2 Research projects

2.2.1 Project options for AIT students in the LINK-Group (www.linkgroup.hu)

The LINK-Group offers the following projects for AIT students:

- comparative analysis of the overlapping network modules of the real world networks using our in-house developed modularization method (www.linkgroup.hu/modules.php)
- analysis of the cooperation of complex, real world networks using the model of spatial games (where social dilemma games, such as the Prisoners Dilemma game are played by agents, who are neighbors in a real world network) and the NetworGame program recently developed by the LINK-Group (http://www.linkgroup.hu/NetworGame.php)
- analysis of signal propagation of complex, real world networks using the Turbine perturbation analysis program recently developed by the LINK-Group (http://www.linkgroup.hu/Turbine.php)

The real world networks examined are the following:

- analysis of the recently described onion-type networks (http://www.pnas.org/content/108/10/3838.full.pdf)
- analysis of crisis events on large-scale social networks (http://www.plosone.org/article/info:doi%2F10.1371%2Fjournal.pone.0017680)
- comparative analysis of the non-stressed and stressed yeast protein-protein interaction network (see original paper here: http://www.linkgroup.hu/docs/11PLoS-Comput-Biol.pdf), or the protein-protein interaction network from young and old organisms
- comparative analysis of free and substrate-bound (drug-bound) protein structure networks, and allosteric drug effects on combined protein structure networks of protein complexes (see the concept here: http://www.linkgroup.hu/docs/11TIPS.pdf).

The student working on the project should

- read the most important papers related to the project obtained from Prof. Csermely
- get acquainted with the network data and computer programs serving the purpose

- be able to modify the data and programs if necessary
- analyze the network by the programs, understand the results and propose the next step of investigation
- make a report on her/his work and findings.

Interested students should approach Prof. Peter Csermely at the email: `csermely@eok.sote.hu`. If the student stays interested after the completion of the project report, the project may continue to a distance-collaboration and may lead to a scientific publication. (We published several papers with colleagues from China and Singapore we have never met ever personally.)

2.2.2 Project options for AIT students at the Eötvös University

1. The PUBMED (`www.pubmed.org`) is a free archive for several bio-related publications. Download a set of articles (please take into account the download policy of PUBMED, and do not overload their server!), and create a co-authorship network. Find communities in the co-authorship network with several community finding algorithms! Try to identify the most representative author/publication in each community by calculating centrality measures! Verify your findings by general scientometric measures, e.g. using the Google scholar archive! By using keywords from the publications, try to identify the topics of the communities!

2. There are a huge amount of free, open source computer programs. Some of them are really badly written, and some of them are really excellent examples for "good programming practices". Choose one "bad" and one "good" software, and analyze the call graph of the functions, routines etc, that build up the software! Compare the graph structure of the call graphs of the chosen programs by comparing usual network measures! Collect some programming practice guide lines, available at the Internet. Compare the network measures of softwares, that do follow these guidelines with such softwares, that do not follow them!

3. The Flickr (`www.flickr.com`) is publicly available photo-archive site, where users can upload photos. Each photo can be labeled by the owners and by any other user as well. Download a set of public photos and their labels. Create a topic map from the co-occurrence of the labels! Try to find "well tagged" and "not well tagged" photos by

measuring similarity between the co-occurring labels. The similarity can be measured e.g. by the length of the shortest path between the two labeling words on an word-encyclopedic network (e.g. Wordnet, wordnet.princeton.edu)

For more details contact Péter Pollner or Gergely Palla.