Third Year Projects 2011/2012
Rules and Guidelines

Third Year Projects Committee

September 12, 2011

1 Introduction

This booklet concerns third year projects. It contains all the information that you will need from choosing a project title and a supervisor, right up to writing up, submitting and being assessed.

Due to pressure on staff you may perhaps not be able to get one of the supervisors/projects that you have chosen.

1.1 Types of project

A third year project is a piece of individual work done under the guidance of an individual supervisor. It should be about 200 hours of work (100 hours for a half unit project), which will include the preparation of a report which, together with any programs that have been written, will be what is assessed by the examiners.

Not all third year projects are the same. Different projects may stress theoretical aspects of a problem, or practical (programming/implementation) aspects, or may be essentially of a survey type.

It is important that you discuss the requirements of your chosen project with your supervisor.

1.2 Half unit projects

The Department **requires** all single honours Computer Science students to do a full unit third year project. This is necessary to obtain British Computer Society (BCS) accreditation for their degree.

Students on joint honours courses may also do a full unit project, but it is recognised that the desire to study other (taught) modules may mean that a half unit project is more appropriate. A half unit project is identical in structure to a full unit project, except that the choice of topics will be limited to those that can be successfully completed with only 100 hours of work.

1.3 Why do a project?

Because you are a single honours Computer Science student and the Department requires you to do a project.

Probably the strongest single reason for doing a third year project is that it is your opportunity to demonstrate your ability to work individually. Success in your third year project will provide evidence of your skills to any future employer and students often use their project supervisors as referees after completing their degree.

Your third year project should be the most enjoyable part of your undergraduate studies. You get to choose a topic of personal interest and to study it in depth.

2 Choosing a project and finding a supervisor

Please read these instructions carefully.

There are two kinds of final year project you may choose to do:

- ◇ Full unit projects **required for single honours Computer Science.**
- ◇ Half unit projects.

If you wish to do any project, then you must run the ballot program which will be available from **15 September, 2011**. Only run this program when you have decided on your project choices.

The Office has NO BALLOT FORMS. You will run a program to register your preferences.

2.1 Finding a supervisor

In order to do a final year project (Full or half unit), you must complete (electronically) a project ballot form.

This year's list of project titles will be available on the departmental website. Choose from the list those projects you may be interested in doing. Every project in the list is associated with a supervisor; you may only choose (at most) one project from each supervisor on your project ballot form. **You can return the completed ballot form any time after 15 September, 2011.**

Each supervisor can only supervise a limited number of project students. Some supervisors may not allow more than one student to do the same project. Therefore it is important to choose a variety of projects on your ballot form.

You MUST return your completed ballot form by 22 September, 2011, otherwise the Department may choose a project for you without consideration of your wishes!

2.2 Project prerequisites

Many projects have *prerequisites* attached. Typically, this means that you must register for certain final year courses, or must pass certain second year courses, in order to do the project. Unless otherwise stated, passing CS2520 is a prerequisite for every project.

Note that students will normally be required to pursue a project chosen by their allocated supervisor if their grade in CS2520 is less than 50% and if their overall grade is strictly less than 50% in their second year.

Note also that if you do not pass CS2520, or if you do not satisfy any specific prerequisites for your chosen project, you will not be allowed to do the project.

2.3 Project topics

You do not have the right to do a third year project on *any* subject of your choosing. You must choose a selection of projects from the list and then the Department will make the final allocation of students to projects and supervisors.

You can make up your own project title as one of the projects that you would like to do — but this will have to be accepted by the Department as a valid project. However, you will still be entered in the normal ballot, and you may be allocated one of the projects from the list.

This is to ensure that all projects have a sufficient academic content, and that they are not too ambitious.

2.4 The project list

On the departmental web-site, there will be a current list of projects. As each supervisor has a limited capacity for project students, and some supervisors will only allow one student to do a particular project, it is imperative that you choose as wide a range of projects as you can on your ballot form. Don't just pick Java programming!

2.5 The allocation process

Any problems with submitted ballot forms will be dealt with by **23 September, 2011**. If you are a single honours Computer Science student and we have not had a ballot paper from you by **23 September, 2011** we will make an allocation on your behalf.

All single honours Computer Science students, and those joint students who have submitted a ballot form by **22 September, 2011** will be allocated a project and a supervisor by **26, September 2011**.

Note that repeating students are exempt from the allocation process.

3 Important dates

Project list 12 September, 2011

Project ballot form program available 15 September, 2011

Deadline for completing the ballot form (online) 22 September, 2011

Final allocation 26, September 2011

Review meeting during the week November 28, 2011 – December 2, 2011

Draft report ready 11 March, 2012

Submission deadline 21 March, 2012, at 2pm

During the last week of the second term we will have a demo day. You will demonstrate the software developed for your project to staff, postgraduates, and fellow students.

4 Organisation of the project

This section describes the process of doing a third year project. The basic structure is that students meet with their supervisor each week for 20 minutes. There is also a formal first term review.

4.1 Student's responsibilities

As soon as possible after the start of the first term the student should arrange a regular weekly meeting time with their supervisor. Each week you will meet with your supervisor for 20 minutes to discuss your project work. You should plan these meetings ahead of time. It is vitally important that you attend these meetings as they are the best way for both student and supervisor to monitor progress on the project.

4.2 Project milestones

You should agree with your supervisor by 22 Oct, 2011 of the first term the important milestones in your project and the dates by which they should be completed. It is clear that the milestones will be different for the different types of project. Note that each milestone is associated with a **deliverable** that you can discuss with your supervisor. The following is a general example:

- ◇ Detailed Specification Written
- ◇ Review of Current Theory/State of the Art Completed
- ◇ First Term Review/Presentation
- ◇ High-Level Design
- ◇ First Prototype Program
- ◇ Specification of Testing Procedures
- ◇ Outline of Report
- ◇ Final draft of Report

4.3 Project diary

It is also important that you keep a **diary** or workbook log of your project work. This will be invaluable when you come to write the report as it will help you to remember problems that you found and dead ends that you investigated.

4.4 Supervisor's responsibilities

The supervisor is there to monitor and advise. They are not there to teach. There will not be time in 20 minutes per week for supervisors to teach all required new material. Supervisors should supply references to new material and be willing to discuss any problems that the student has with this material after the student has studied it.

It is the supervisor's responsibility to attend each of the weekly meetings, or to re-organise the meetings if this is not possible.

The supervisor must keep an attendance register of all student meetings. The supervisor will be present in the first term review to formally monitor the progress of the project.

4.5 Paperwork

During the course of the project there is a minimum level of paperwork that must be maintained.

- ◇ Formally there is a requirement for the **supervisor** to maintain a register and a milestone sheet. The register is important as it may form part of the assessment if it can be shown that a student's attendance has been unsatisfactory. The milestone record is important as it clearly shows how progress was made during the project. These records may be maintained electronically.

- ◇ It is in the **student's** interest to maintain a diary or log book of all the work that they do. It is very useful when it comes to writing up as it is very hard to remember all that was done early in a project! Also, it can be used as a basis for discussions with the supervisor about problems and progress.

  Whilst you have only 20 minutes per week of meeting time it is reasonable to ask your supervisor to read and review your work before the weekly meeting. You should get your supervisor to look at your diary periodically to make sure that you are keeping a log of all that you are doing.

5 Documenting the project

5.1 Writing the report

Remember that your project report will be the most important deliverable that you will produce. It will serve as a guide to the other deliverables and to the structure of your project.

You should aim to agree on the outline of your report with your supervisor as early as possible. This will allow you to **write up your work as you go along**. An example of a typical list of headings for a project report is attached. As you work on your project keep together all the work you do, including early, incorrect ideas and program fragments. These will all be essential in explaining the development of your work.

You should read the section on project marking in order to better understand the criteria used in assessing a project.

Your supervisor must see a complete draft of your report before the final version is prepared.

Your supervisor may however wish to see other deliverables in support of your report. You should discuss with your supervisor exactly what is required in addition to the report. For example:

- ◇ Long or complicated test output (referred to in the report).
- ◇ Examples showing the use of the project.
- ◇ A working program with instructions on executing it.
- ◇ Copies of papers, reference material used for the project.

5.2 What might your report contain?

1. Some kind of **Introduction**. This should be broken into an **abstract** of the project, a section of **motivation**, and a list of **project goals**.

2. Some kind of **theory section**. This might include a **literature survey**, sections on **specific theory**, or even an **interesting discussion** on what you have achieved in a more global context.

3. If your project is mostly based on a **software product** then you will need sections describing the **software engineering method**. This may even be most of your report. You should look back at CS2520 to decide what sort of sections you might need. Remember that programs might need a **user manual**, and a short **technical manual**.

4. If your project is mostly **theoretical** then you should include some **theory development**. This might include **small programs** to investigate certain things, **explanations of algorithms**, or even descriptions of the **particularly hard bits of theory**. A theoretical project will probably have a section of **results** and probably some sections of **analysis of results**.

5. If your project is hardware oriented then you will need to describe the hardware development process. The details of what sections should be in your report should be discussed with your supervisor.

6. Every report needs some kind of **assessment** section. This might include a **realisation** section describing your experience of the project, a

summary of what you achieved, and a section on **problems and enhancements**. Of course you should have a short **conclusions** section summing up the whole process.

7. Most projects will also need a **Bibliography** of works referred to in the text, or that have been read in order to understand the project.

8. Lastly there are some added extras you might want to include. Perhaps a **program listing** (use `psify` and `up`, see the manual). Perhaps some **sample output**, or **layout diagrams**. Certainly you should have a section on **How to Use My Project** that tells the markers about examples they might look at, or demonstrations to run, or simply where to find the sources for material referred to in the report!

Remember to agree the structure of your project, including the report, as early as possible in the process.

6 Plagiarism and acknowledgement of sources

It is in the nature of a third year project that much of the material will not be original. You will have researched around your subject and discovered many sources of information. It is vital that any quote made from any source (including the web) should be properly acknowledged, both where it is used *within the report text*, and at the end of the report in the bibliography. Under no circumstances should copyright material be included in a project report without the proper permissions having been obtained, and any such inclusion should be agreed with your supervisor.

Your project and report must be your own work. Plagiarism (the unacknowledged use of other people's work) is a very serious offence and will be severely penalised. In 1998, three students in this department submitted final year projects consisting of programs which they had down-loaded from the Internet and then presented as their own work. This plagiarism was discovered, with the result that the students left the College without graduating.

All sources of information which you use during your project must be listed in your bibliography. This includes books, articles, research papers, course notes and Internet sites. Direct quotations must be acknowledged, for example:

Henry Smith [1] states that "The problem of wild animals on campus can only be solved by the introduction of even wilder animals which will eat them." but the results of this project seem to contradict him.

In this case, the citation [1] would refer to a bibliography entry such as

1. Smith, H. *Modern University Life.* Wombat Press, 1997.

If you express someone else's idea in your own words, then you must also acknowledge their original expression of the idea. For example:

Smith [1] believes that an infestation of wild animals in a university can only be cured by introducing suitable predators. However, the obviously recursive nature of his proposal led me to consider more feasible alternatives.

If you use examples from a lecture course or a book to illustrate your background theory, then you must acknowledge the original source. Similarly if you follow a book or lecture notes when presenting background theory, for example:

The following sequence of definitions is based on [1], with simplifications due to the fact that we are only considering finite *widgets.*

If your program includes any pieces of code which you did not write yourself, then you must identify them and say where they came from. For example, if you copy an implementation of a particular algorithm from a book, you must make it clear that you did not write those lines of code.

If you are in any doubt about what needs to be referenced and acknowledged, ask your supervisor for advice.

7 Submitting the project

This section tells you how to produce and submit your project report. It also defines the circumstances in which the submission deadline can be extended.

7.1 Submitting the report

In order to produce and submit your project report, you need to do the following.

◇ Prepare the text of the report with a word processor or a document preparation system e.g. LaTeX, using a font size and line spacing which will make it easily readable. You need to leave a margin of at least 2cm on the left hand edge of the paper, to allow for binding.

◇ Make sure that the front cover of the report contains the following information:

  – your name
  – the project title
  – the year
  – either "Full unit project" or "Half unit project", as appropriate

  Your name and the year must also appear at the very top of the cover (this makes searching through the projects easier for us).

◇ Print two copies of the report on a good quality printer—again, the final report must be clearly readable. You can print it either single or double sided, as you prefer.

◇ Bind each copy of the report with a plastic spine (available from the department office) and a transparent cover (also available from the office). If the report is too long to be bound in one volume, then you should bind it in as many separate volumes as necessary. The front cover of each volume must contain the information listed above, and must also state "Volume *m* of *n*" as appropriate. Note that the production of very long project reports is discouraged and is rarely necessary.

- ◇ Hand in both copies of the report to the department office by 21 March, 2012, at 2pm. Remember that printers, binding machines and the office itself will be in great demand at the time of the deadline; try not to leave preparation and submission of your report to the last minute. You must also hand in the student feedback form and then you will be given a receipt for your project.
- ◇ Hand in an electronic copy of the report into your own submission directory (see the section on *Submitting the program*).

7.2 Submitting the program

You must make sure that it is possible for your project markers to run any programs which you have produced as part of your project. It is also essential that you not modify your program after submission. To this end you **must copy all relevant files to the directory:**

```
/CS/projects/submissions/YOUR_LOGIN_NAME
```

This includes:

- ◇ Source files for programs,
- ◇ Any Makefiles,
- ◇ Results files,
- ◇ A machine readable version of your final report,
- ◇ Other files specific to your project.

In this directory you must include instructions for building any programs included from the source code. If your program was not built on the departmental computer system then you must say how it was built on whatever system you used.

Remember that your project report must include full instructions on running any programs that you submit.

- ◇ If you have used the departmental computer system, then your report must say where your program is located (relative to your submission directory).
- ◇ If you have used your own computer, for example a PC, then you must say where relevant files are located relative to your submission directory, and how to install the files needed to run your project.

7.3 Student Feedback Form

Students should fill in the project feedback form and hand this in along with their projects reports. You will only receive a receipt for your submitted project if you hand in this feedback form (available from the project web-site).

7.4 Extensions

Extensions for projects will normally only be given for medical reasons, and are difficult to obtain. An extension can only be given by your academic advisor, with the agreement of your project supervisor and the projects committee. An email confirming your extension **must be e-mailed to the departmental administrator**. If a student is not granted an extension and hands in the project late a marking penalty will be applied (see formal procedures).

8 Guidelines for marking

This section is concerned with the overall assessment of a project.Ultimately, the final mark of the project will be determined by the external examiners to ensure that the project marks are commensurate with individual projects across the University. Thus, the following criteria are guidelines for choosing the final classification of a project and have been constructed in consultation with our external examiners.

8.1 The criteria

Examiners should mark from 0% — 100%.

Because every project is different, it is impossible to give a universal marking scheme. However, these guidelines specify the features which a project must have in order to be awarded a mark in each class range.

Half unit projects must be marked against the same criteria as full unit projects. The examiners should consider, when marking projects, that a full unit project represents 200 hours of work, whilst a half unit project represents only 100 hours of work.

1. A **first class project** should be outstanding in most respects. It should show a mature, accurate grasp of the issues raised by the topic and should conduct a sustained, coherent argument in a style that is fluent. It should demonstrate an excellent knowledge of appropriate reference material, techniques and relevant theoretical perspectives. It should also demonstrate excellent technical/practical skills where these are needed for software and/or hardware. So that first class marks may make their proper contribution to the final average, it is important that they should not be limited to the low 70s. Examiners should use the full band analogously to others and be prepared to give marks above 80% for outstanding work.

   A mark between 70% and 79% should clearly be a first class project.

   A mark of 80% or more should only be given for projects with excellent reports and where the student has achieved one of the following:

   - ◇ The achievement of the student would be a very good basis for continuing research in the area.
   - ◇ The practical work is of such a high professional standard that it could be distributed without significant extra effort.

  - ◇ The student has worked independently, exceeded expectations, and clearly demonstrated a very deep understanding of their results.

   Exceptional projects may occasionally be awarded a mark in excess of 90%.

2. A clear **upper second class project** should have many of the features distinguishing a first class project. Creativity and originality or breadth and depth of response could compensate for some weakness in style or some incoherence of argument. Alternatively, a well written thoughtful report could compensate for a narrowness in the range of issues addressed. Projects in this class should contain little or no irrelevant material and should be generally well constructed.

3. A **borderline 2-1/2-2 project** would show that the candidate has grasped the basic issues raised by the topic, but has not achieved the qualities listed for a first or clear upper class second grade. There should be some theoretical analysis and/or experimentation and implementation. This grade might alternatively be awarded where the quality of argument is mitigated by clear weaknesses, eg. a detailed report with considerable insights, marred by a lack of background material, poor organisation of material or incomplete work or errors.

4. A **lower second class project** gives some evidence that the main part of the topic is understood. However, recognition of issues is incomplete, and analysis and discussion is restricted. Evidence may be unbalanced or of dubious relevance with a tendency to excessive generalisation or unsupported assertions of a subjective nature. This class is also appropriate where much of the content seems worthy of a higher grade but where poor presentation, buggy implementation, or lack of a good introduction or conclusion etc obscures the report.

5. A **third class project** makes an attempt to approach the topic but without fully understanding its implications. It may be characterised by insufficient or low quality implementation and theory.

6. A mark just above the pass/fail boundary represents a failure to approach the topic. The project may show some evidence of study but a failure to mobilise it as an implementation or with background theory, with a tendency towards incoherence.

7. A mark **from 0 — 39%** should be used where what is written is characterised by complete lack of relevance to the topic and where there is little or no evidence of the candidate having studied, and where the work contains serious errors and is poorly constructed.

8. Where only a few lines of work have been submitted, the candidate is not deemed to have made a serious attempt at the project and a mark of 0 - **Incomplete** - will be recorded.

8.2 What the markers mark

The above list is enough to classify each project submitted. Within these (wide) classification bands markers will use many different criteria for assessing a project.

1. Difficulty. Each project title has the scope for an ambitious attempt, or a straightforward progression. You might award marks for making an intelligent try at a more difficult goal.
2. Presentation. Presentation of the material and arguments in a way that makes their development clear to a computer scientist who is not a specialist in the project topic.
3. The Journey. Any project that required many new topics to be assimilated during the process should get credit for this.
4. Content. Has a minimum of ground been covered? Most projects should contain an exposition of an area of theory, as well as some practise.
5. Scale. Is the project is equivalent to two third year half units?
6. Results. Obviously we are interested in results. However, the students have been told that **a good first can be obtained for a project which is incomplete**. This includes non-working programs, theories which are wrong, hardware which melts, etc., The work can still be judged against all of the other criteria above.
7. Review. The critical review or assessment section will be of most importance to the **second marker** second marker. It helps you to gauge the student's appreciation of the difficulty and relevance of the work involved in the project.
8. Theory. Evidence of a good understanding of the background theory or current state of the art.
9. Objectives. Clarity of statement of objectives.
10. The **first marker** may also award (or deduct) marks because of the conduct of the student. Did they attend meetings, did they meet deadlines?
11. The **first marker** may award (or deduct) marks for students who worked completely independently, or who plagiarised, or who had excessive support.
12. The markers may take into account the students' performance in their presentations at the first term review.

9 Formal procedures

The rest of this report covers the departmental regulations associated with third year projects.

9.1 Finding a supervisor

1. Any student doing single honours Computer Science must pursue a project. A joint degree with Computer Science is eligible to do a third year project.
2. Supervisors may refuse to accept any project title not proposed by themselves.
3. Supervisors may refuse to allow two students to register for the same project title.
4. The Department will make all reasonable attempts to satisfy the requests made by students on their ballot forms.
5. Students will normally not be offered any choice over their final project allocation.

9.2 Choosing a project title

1. All projects from the published list are acceptable project titles.
2. Students will normally be required to pursue a project title chosen by the supervisor if their grade in CS2520 is strictly less than 50% and their overall grade is strictly less then 50% in the second year.
3. Any student wishing to attempt a project not from the published list must produce a project description, typically one side of A4, which must be signed as acceptable by a member of the projects committee.
4. A student pursing a Computer Science with AI or Computer Science with Communications is normally required to pursue a project with a significant AI or Communications content.
5. Students repeating projects are normally required to pursue the same project title as previously undertaken and will be supervised by the same supervisor.

9.3 Complaints

1. Students may complain about their supervisor only on grounds of insufficient monitoring, or inappropriate demands.
2. In the first instance a student should bring complaints to their supervisor.
3. If, after bringing a complaint to the supervisor, the student still has a grievance then they should bring their complaint to a member of the projects committee.
4. If the committee feels that the complaint is justified, and the student wishes, the projects committee will make representations to the supervisor.

5. If the committee feels that a complaint is justified then they will minute this and the grievance will be considered when assessing the project.

9.4 Marking

1. All projects will be marked by two independent markers, one of whom will be the project supervisor.
2. **The markers may take the student's presentation performance at the end of the first term review into account (see progression procedures)**. The performance will be recorded on a mark form.
3. In the event of a small discrepancy (less than 10%) the marks will be averaged with a two to one weighting towards the supervisor's mark.
4. In the event of a larger discrpancy, a third independent marker will mark the project, and the projects committee shall seek agreement amongst all markers as the final outcome.
5. Markers will mark according to the marking criteria supplied and will justify marks with a short paragraph on the marking form.
6. A recommendation of **Incomplete** will be made to the external examiners for a student failing to submit their project.
7. No marking penalty will apply to a project submitted by a student on time or within the limits of an agreed extension. Extensions for projects will normally only be given for medical reasons. An extension can only be given by the academic advisor, with the agreement of the project supervisor and the projects committee.
8. The marking penalty for late submission is 10% of the agreed mark for a project submitted within 24 hours of project deadline. After 24 hours no project will be accepted and a mark of incomplete will be given.

9.5 Monitoring

1. Monitoring will include a meetings register and a milestone sheet.
2. Students may be required to complete a half-unit project and take more taught courses in the second semester (instead of a full-unit) if their attendance is unsatisfactory, if they do not produce timely milestone deliverables, or if their performance is deemed unsatisfactory in the first term review.
3. An student feedback form will be filled out by the students at the end of the project and handed in with their project reports. It will remain confidential until the completion of the examination process in July. It will be used to inform procedures and practices of the projects committee.

9.6 Progression - First Term Review

1. In the penultimate week of term, (during the week November 28, 2011 – December 2, 2011), a review will be conducted to determine the student's progress on the project to date.
2. A review panel will consist of three people, the supervisor, one independent marker and one member of the projects committee.
3. The review will comprise of up to a ten minute presentation by the student about their project.
4. The review panel will grade the presentation using the criteria
   - ◇ **A - progressing well**. The student is making good progress on his/her clearly identified milestones.
   - ◇ **B - satisfactory**. The student is making acceptable progress on his/her clearly identified milestones.
   - ◇ **C - maybe required to change**. The student is not making satisfactory progress on the project and/or the project goals are not clearly identified. Nonetheless, the student has demonstrated in the presentation a grasp of the ideas of the project objectives.
   - ◇ **D - required to change**. The student is not making satisfactory progress on the project and the project goals are not clearly identified.
5. The grade will be justified on the mark form with a short paragraph.
6. The grade given for a student's presentation may be used when determining the final mark of the project.
7. The students will be informed of their grades at the end of the review meeting.
8. Students receiving an A grade can continue with their project in their second term.
9. Students receiving a B grade will be required to clarify their project goals with their supervisor and agree to a certain set of deliverables which are to be completed by the beginning of the second term. In the first supervisory meeting of the second term these deliverables must be discussed. If they are considered to be of unsatisfactory quality by the supervisor the student will normally be required to complete a half-unit project and take more taught courses in the second term.
10. Students receiving a C grade will be required to change their registration from a full-unit project to a half-unit project and take more taught courses. They will be advised of this by the projects committee and should then meet with their advisor.
11. Students who are registered for a half-unit may choose to de-register from their project and take more taught courses in the second term.

10 The project list

In this section you will find a list of all projects available this year. A project will normally only be supervised by the supervisor named in the project description. Projects marked with a * are suitable for Computer Science with AI students.

10.1 Creation of a web-based Gene Ontology Semantic Similarity Tool

Supervisor A. Paccanaro

Value Full unit

Abstract

* This is a small research project. The final aim of this project is the creation of a software tool of high quality that will be published in a bioinformatics journal and will be freely distributed to the scientific community *

Knowledge of a gene's function is held in scientific natural language which is not machine friendly. The Gene ontology project [1] provides a controlled and machine read-able vocabulary for describing gene products in various organisms. The terms in the Gene ontology (GO) project are arranged in a directed acyclic graph (DAG). Each of the terms in the GO project have definitions and have a parent-child relationship.

Although the GO project addresses the need for consistent descriptions of gene products, measuring the functional similarities between genes based on their functional annotation remains a challenge. Typically, researchers collect gene annotation data from databases using web based tools such as AmiGO [2] and TAIR [3], and visually examine the biological functions. This process is very time consuming and can be accelerated by semantic similarity measures [4, 5] which can identify genes with similar annotations. Although there have been a few web based semantic similarity tools for examining the similarity in gene functional annotation, their usage has been limited due to limited functionality of the tools available.

The project involves building a web based GO semantic similarity tool. The tool would be able to take a list of gene identifiers. It would then retrieve the annotation data and plot the gene identifiers on the GO DAG. The semantic similarity between the annotations of the genes will be calculated using various methods such as [4, 5]. The student is expected to gain understanding of ontologies, constructing and visualizing large graphs and semantic similarity measures.

Prerequisites Interest in biological problems and good programming skills

References

1. The Gene Ontology Consortium (2000), Gene Ontology: tool for the unification of biology. Nature Genet. 25: 25-29
2. AmiGO amigo.geneontology.org

3. The Arabidopsis Information Resource http://www.arabidopsis.org/
4. Jiang J, Conrath D. (1997), Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. Tenth International Conference on Research on Computational Linguistics (ROCLING X): Taiwan
5. Resnik P. (1999), Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language. J. Artificial Intelligence Res. 11:95-130

10.2 Implementation of GO Similarity Algorithms

Supervisor A Paccanaro

Value Full unit

Abstract

* This is a small research project. The final aim of this project is the creation of a software tool of high quality that will be published in a bioinformatics journal and will be freely distributed to the scientific community *

The Gene Ontology (GO) project is a collaborative effort to address the need for consistent descriptions of gene products in different databases. It aims at delivering a shared, structured, and controlled vocabulary that can be applied to any organism. The terms in the vocabulary are organized by three directed acyclic graphs called GO trees. In order to establish how two GO terms relate to one another, we can exploit the concept of semantic similarity, or alternatively, its inverse concept, semantic distance. The aim of this project is to provide an efficient implementation of some well known semantic similarity measures including those proposed by Resnik et al. [1], Jiang et al. [2], Lin et al. [3] and Yang et al [4]. The software will be implemented in Java. The implementation should be efficient, and easy-to-use. Given a list of genes or a list of Gene Ontology terms and a choice of semantic similarity, the program will output the semantic similarity matrix between the given genes/terms.

Prerequisites Interest in biological problems and good Java programming skills

References

1. P. Resnik, Using Information Content to Evaluate Semantic Similarity in a Taxonomy, Proc. 14th Int Joint Conf. Artificial Intelligence, pp. 448-453, 1995.
2. J.J. Jiang and D.W. Conrath, Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy, Proc. Int Conf. Research in Computational Linguistics, ROCLING X, 1997.
3. D. Lin, An Information-Theoretic Definition of Similarity, Proc. 15th Int Conf. Machine Learning, pp. 296-304, 1998.
4. H. Yang, T. Nepusz and A. Paccanaro. Improving semantic similarity measures using downward random walks. In preparation.

10.3 A software tool for the automatic selection of transcriptomics experiments for functional genomics

Supervisor A Paccanaro

Value Full unit

Abstract

* This is a small research project. The final aim of this project is the creation of a software tool of high quality that will be published in a bioinformatics journal and will be freely distributed to the scientific community *

Microarray is perhaps the most widely used gene expression measurement technology today. Gene expression profiles obtained from microarray experiments provide an insight into gene function and it is considered a powerful exploratory technique. However, understanding gene function from the expression profiles is a challenge. To this end, scientists often apply the Guilt by Association (GBA) principle where, genes are considered to be involved in similar functions if the gene expression profiles are correlated. GBA principle is used in a variety of functional analyses techniques ranging from simple co-expression analysis, clustering techniques to network based approaches. GBA driven strategies generally use large collections of microarrays for calculating correlation. However, our research has shown that instead of a large collection, using only those experiments which are relevant to the function under investigation could lead to significantly improved results [1]. However, selecting relevant experiments manually is a non-trivial task. Therefore, we have recently developed a novel algorithm which is able to identify the relevant experiments given a large collection microarray experiments. The prototype of the algorithm which we have developed has given excellent results and the algorithm promises to be extremely useful in practice for gene functional annotation. Therefore the aim of this project will be to provide an efficient user friendly implementation of our algorithm to be freely distributed to the scientific community. More details about this research and the algorithm can be found at www.paccanarolab.org/papers/CorrGene The software tool implementing the algorithm would be a stand-alone desktop application running on the users computer. The system would allow the user to build a microarray collection locally. This can be done by simply downloading datasets from databases such as NCBIs GEO or EBIs ArrayExpress into a directory. Users should also be able to add their own datasets into the collection. When building the microarray collection the software will also extract information regarding the experiments (such as experiment name, experimental conditions and author names) and it will perform quality checks on the data (such as dealing with missing values and mapping gene names to standard gene identifiers). Having built a microarray collection the user will provide as input a list of gene identifiers or simply specify a Gene Ontology identifier. The system will then run the algorithm for the experiment selection. The final list of selected

experiments is output on the screen or as a text file. The user should also be able to export the microarray data for the selected experiments as a text file.

Prerequisites Interest in biological problems and good C++ programming skills

References 1. Bhat P., Yang H., Bgre L., Devoto A., Paccanaro A. A Computational selection of transcriptomics experiments improves Guilt-by-Association analyses (under review, preprints available from A. Paccanaro) .

10.4 BioSurf a tool for the visualization and analysis of large biological networks on the Microsoft Surface

Aim The aim of this project is to begin the development of BioSurf an application for the MS Surface that will allow scientists to visualize and analyze different type of networks either independently or relating to each other.

Supervisor A Paccanaro

Value Full unit

Abstract

* This is a small research project. The final aim of this project is the creation of a software tool of high quality that will be published in a bioinformatics journal and will be freely distributed to the scientific community *

An important idea that has emerged recently in biology is that a cell can be viewed as a complex network of inter-relating proteins, nucleic acids and other bio-molecules. A bio-molecular network can be viewed as a collection of nodes, representing the bio-molecules, connected by links, representing relations between the bio-molecules. Examples of bio-networks are, for example, regulatory networks, metabolic networks and signaling networks. At the same time, data generated by large-scale experiments often have a natural representation as networks such as protein-protein interaction networks, genetic interaction networks, co-expression networks. From a computational point of view, a central objective for systems biology is therefore to develop methods the analysis of large-scale biological networks (recently this has been the focus of the research in my lab check out www.paccanarolab.org for an overview of the lab research).

A big challenge for this analysis lies in the heterogeneity and sheer size of the biological networks at play. In order to reason about these networks, often researchers find it useful to visualize them. Several tools have been proposed recently which are able to display these networks (mainly in 2D), but they are often difficult to use especially when thousands of nodes and connections have to be analyzed and visualized. Moreover, these tools are severely limited in the way in which these networks can be controlled and manipulated by the user.

The MS Surface is the ideal tool the analysis of large biological networks: the researcher could use its gesture and object recognition capabilities to manipulate the networks, while its large display would allow the visualization of large sections of the networks (or multiple networks at the same time).

The meaning of the nodes and edges will depend on the biological networks and the problem at hand. Nodes could constitute genes, proteins, metabolites (to make just the few obvious examples) and edges within these graphs could represent (and be labelled with) the correlation between gene expression profiles or metabolite abundance, or the probabilities of two proteins to interact, etc

Biological networks from different organisms could be visualized at the same time and the connection to internet resources could allow easy retrieval of information about the various biomolecules being visualized (gene sequences, metabolite structures, etc.)

A problem to solve will be define a proper technique to map the nodes in 2 or 3 dimensions, as in general the labels weighting the edges will not constitute Euclidean distances (here the performance of already existing techniques, such as multi-dimensional scaling or t-SNE would have to be tested).

The tool would allow scientists to contrast and compare the same type of network across different organisms, or different networks in the same organism. One could even imagine extending this tool to include networks whose nodes are diseases, or linking genes to diseases, or drugs, or attribute of chemical compounds.

Prerequisites Interest in biological problems and excellent programming skills. Knowledge of Windows Presentation Foundation and/or XNA would be very helpful.

10.5 A general Sorting and Searching Testbed.

Aim To provide a tool for demonstrating sorting and searching algorithms.

Abstract This package will display the continuing calculations and the results of certain sorting and searching algorithms. It will provide a graphical user interface and statistics in order for user's to compare various algorithms. It will be written in C, C++ or in Java.

Supervisor A. Yeo

Value Full unit

10.6 A File Compressor.

Aim To provide a tool for compressing a selection of files.

Abstract This package will offer the user a file tree for them to select files from. It will compress each file with an appropriate algorithm and create an archive in which to store the resulting compressed files. The package will also selectively decompress files from such an archive. The student will have to determine file type dynamically in order to select the compression technique from Lempel-Ziv, Huffman, and JPEG. The compression algorithms for JPEG will not be coded by the student.

Supervisor A. Yeo

Value Full unit

10.7 A Chess Program

Aim To produce a program that can play chess against a user.

Abstract The program shall provide a graphical user interface that allows the user to play chess against the computer. The program should incorporate a number of game-theoretical principals, that will allow the program to play at a basic level.

Supervisor A. Yeo

Value Full unit

10.8 The Travelling Salesman Problem.

Aim To implement and analyse several heuristics for the TSP.

Abstract The student should implement several heuristics for the TSP (e.g. Nearest Neighbour, Repeated Nearest Neighbour and the Greedy algorithm). The student should then give some theory behind the heuristics, as well as performing some experiments with the implemented algorithms. Finally the student may either consider generalisations of the TSP or implement a nice user interface.

Supervisor A. Yeo

Value Full unit

10.9 A Black Jack simulator

Aim To produce a nice Black Jack program, that is capable of evaluating given strategies by simulation.

Abstract The program should provide a nice graphical user interface which allows the user to play Black Jack against the computer. One should furthermore be able to enter a given strategy, and ask the computer to evaluate it, using simulation.

Supervisor A. Yeo

Value Full unit

10.10 A Chinese checkers program

Aim To produce a program that can play Chinese checkers against a user.

Abstract The program shall provide a graphical user interface that allows the user to play Chinese checkers against the computer. The program should incorporate a number of game-theoretical principals, that will allow the program to play at a decent level.

Supervisor A. Yeo

Value Full unit

10.11 Fractal Patterns Using Java

Supervisor C. Watkins

Value Full unit

Abstract There are simple techniques for producing beautiful fractal patterns based on random walks of fractional dimension, and on iterated function systems. The aim would be to produce a usable Java applet in which a user could specify a fractal pattern, see the result, and then re-specify the pattern.

A second and important aim is to come understand the nature of some classes of fractal patterns, and the principles by which they can be produced.

Prerequisites Strong programming skills.

10.12 Drawing Plants using L-systems

Supervisor C. Watkins

Value Full unit

Abstract Many plant-like forms can be elegantly described in terms of simple growth-rules, which may be represented in a formalism called an L-system. The project would be to implement software for “growing” plants using L-systems, and then rendering the resulting forms using a graphics package.

There would be considerable room for creativity in designing the plants and rendering them.

The aim of the project would be to produce L-systems that described the form of certain plants as realistically as possible, to write software to generate and render plant-forms from L-system specifications, and to produce some nice pictures.

Prerequisites Strong programming skills.

10.13 Reinforcement learning (perhaps for an idealised acrobat) *

Supervisor C. Watkins

Value Full unit

Abstract Reinforcement learning is a way of learning what to do, with feedback consisting only of delayed "rewards" and "punishments". The project would be to implement a demonstration of reinforcement learning for a simple problem, such as parking a car, or, more interestingly, for an idealised acrobat swinging from a horizontal bar to swing back and forth and finally balance upside down on the bar.

There is a considerable amount of theory to master.

Prerequisites Programming Skills

10.14 Learning and Combinatorial Optimisation *

Supervisor C. Watkins

Value Full unit

Abstract Combinatorial optimisation is a generic term for a broad class of problems where the optimal configuration for some discrete system has to be found. A common method of solving such problems is to use heuristic search, in which various trial configurations are repeatedly modified in search of improvements.

The effectiveness of heuristic search depends critically on how good the heuristic measure of configuration quality is. There have been a small number of papers that report encouraging results on improving the effectiveness of heuristic search by learning the heuristic, instead of defining it a priori.

The aim of the project would be to implement heuristic search algorithms for several combinatorial optimisation problems, to implement or use learning algorithms to learn heuristics, and to test the effectiveness of the learned heuristics by performing computational experiments.

Prerequisites Strong Programming Skills, a willingness to tackle a challenging and open-ended project that would be based on (fairly) recently reported research.

10.15 Digital Fountain Codes

Supervisor C. Watkins

Value Full unit

Abstract Digital fountain codes are a beautiful and (relatively) simple idea, invented in 1998. A digital fountain encoder takes a large file (such as a movie) and produces an endless, randomised sequence of packets, or "drops" of information. To recover the file, you need to collect enough "drops" (usually a total of about 1.4 times the file length), and then decode. The beauty is that any random selection of drops will do. Digital fountain codes, therefore, are suitable for broadcasting over channels over which packets are lost (erasure channels): if some packets are lost in transit, there is no point in the receiver asking for them to be resent — the receiver just waits until it has accumulated a sufficient number of "drops" to commence decoding. Digital fountain codes are also used for distributed file storage and downloading — collections of "drops" can be stored on different machines, and then a sufficient number retrieved to reconstruct the original file when it is needed.

The aim of the project would be to implement an efficient digital fountain encoder and decoder, and to perform experiments to optimise some of the encoding parameters.

10.16 Gallagher Codes

Supervisor C. Watkins

Value Full unit

Abstract (For the more mathematically minded. . .) Gallagher codes are state-of-the-art error correcting codes, which nearly achieve the Shannon bounds on channel capacity. The aim of the project would be to understand some information theory and channel coding theory, and to implement an encoder and decoder for Gallagher codes, and to perform experiments to determine the capacity and reliability of these codes. This is a challenging but elegant project, which will involve studying some beautiful and classical results in information and coding theory.

10.17 A Sudoku solver using Constraint satisfaction (and other techniques)

Aim To understand simple techniques from Constraint Satisfaction and to apply these techniques to solve a complete information puzzle.

Abstract Constraint Satisfaction is one of the leading technologies in Artificial Intelligence for solving search problems. A constraint satisfaction problem instance can be seen as a set of related questions that need to be answered. We cannot answer any particular question without knowing the answers to those questions which affect it.

This model is applicable in machine vision, fleet scheduling, staff rostering, machine shop design, game playing, frequency assignment, vehicle routing etc.,

It is ideal for solving Sudoku. Constraint solvers are relatively easy to implement in Java and can solve even hard Sudoku instances in very little time.

Supervisor D. Cohen

Value Full or half unit

10.18 A general arbitrary precision arithmetic program

Aim To produce a package that will calculate series expansions or mathematical formulae to arbitrary, but determined precision.

Abstract This package will take as input an expression for a term in a series and will calculate the successive values of the series to arbitrary precision. The term in the series will only include the standard mathematical operators, addition, subtraction, multiplication and division. The final project must be able to calculate pi to 1,000 decimal places in less than an hour.

Supervisor D. Cohen

Value Full unit

10.19 Mancala - The world's oldest Game.

Aim To write a program to play Mancala.

Abstract The program should either have a nice graphical user interface, or should learn to play the game using a well-known algorithm. This game is simple. The programming will be straightforward.

Value Full unit

Supervisor D. Cohen

10.20 A Traffic Simulator.

Aim To design and implement a road traffic model to determine the effectiveness of road layouts.

Abstract The program must have a full object-oriented design. It will involve the use of threads and a graphical user interface. Ideally the program should be written in C++ or Java. It will model traffic behaviour based around individual driver preferences, vehicle types, and local road conditions.

Supervisor D. Cohen

Value Full unit

10.21 Frequency Assignment

Aim To investigate the allocation of frequencies to transmitters in broadcast networks.

Supervisor D. Cohen

Value Full unit

Abstract In a broadcast network based on frequencies (channels) it is necessary to allocate channels to transmitters in such a way that a receiver anywhere in the region of interest can receive a decent signal. This project will investigate this problem and produce tools to solve simple examples. It will involve the use of graphics for presenting results. The program will be written in Java.

10.22 Structural alignment of proteins

Aim To implement a piece of software that will align two protein structures.

Abstract Sequence alignment is well understood method for identifying proteins that are close to each other on an evolutionary basis (as either orthologues or paralogues). The Needleman-Wunsch algorithm is a well understood method for identifying such relationships. Deeper evolutionary relationships can be detected by generalising the Needleman-Wunsch algorithm to detect structural rather than sequence similarities.

The project will read in two arbitrary protein structures and perform the structural alignment method as outlined by Orengo and Taylor for their implmentation of SSAP. The program can also be run in a batch mode so from a list of protein structures all possible similarities can be computed.

The program be written in Java or C# .

Value Full unit

Supervisor Hugh Shanahan

References: 1) Taylor WR, Flores TP, Orengo CA. Protein Sci. 1994 Oct;3(10):1858-70. 2) Needleman, Saul B.; and Wunsch, Christian D. (1970). ”A general method applicable to the search for similarities in the amino acid sequence of two proteins”. Journal of Molecular Biology 48 (3): 443?53. doi:10.1016/0022-2836(70)90057-4. PMID 5420325

10.23 An implementation of string kernels using SVM's

Aim To develop an SVM that uses string kernels to classify sequences.

Abstract The greatest success of Bioinformatics has been in the fields of sequence alignment, which is key in identifying homologous families of proteins. However, alignment methods are based on a largely ad-hoc sets of weights to determine the likelihood of alignment between a particular pair of residues or amino acids. A class of non-alignment based methods have come to the fore in the recent past which allow the opportunity to identify classes of families without recourse to alignment. The student will write a piece of software that will read in two sets of sequences and determine frequency vectors for k-mers of a variable length. Using these as feature vectors an SVM will be generated which can then be used to classify another set of sequences.

The software may be written in either Java or C#.

Supervisor Hugh Shanahan

Value Full unit

References: 1) Christina Leslie et al. Bioinformatics 20 (4), 2004.

10.24 A framework for deploying Bioinformatics applications on Azure

Aim To develop a piece of software that minimises the difficulty of deploying Bioinformatics applications on the cloud computing platform Azure.

Abstract The use of cloud services (where computational resources are based at a unspecified non-local location which makes use of novel platforms and extensive use of virtualisation) has come to the fore in Bioinformatics in the last three years. The focus of most of the research in this area has been in using Unix-based virtualisation models such as that fouind in the Amazon EC2 service. Microsoft Azure provides an alternative to this approach. However deployment is largely hampered by the requirement of extensive programs to be written in C# to allow the deployment of any piece of software. In this project, the student must develop a piece of software that will locate an executable and a set of necessary files, bundle them together and generate C# code so that the resulting package cn be deployed onto Azure with the results of the executable available as a web service based on Azure. The service will be tested on a simple Blast run.

The code must be developed on Java or C# and must compile on a 64 bit Windows 7 machine.

Supervisor Hugh Shanahan

Value Full unit

References : 1) Alex Bateman and Matt Wood Bioinformatics 25 (12), 1475 (2009) 2) Ling Qian et al. Lecture Notes in Computer Science 5931, 626-31 (2009) 3) Windows Azure tutorial `http://www.microsoft.com/windowsazure/learn/tutori`

10.25 String Factorization Techniques

Aim To implement and apply string factorization algorithms.

Abstract Strings are fundamental in Computer Science, and the theory of strings includes formal grammars, regular expressions and automata. Practical applications of string manipulation techniques arise in sorting problems, bioinformatics, compression, cryptography and musicology. For example, some periodic musical structures in African rhythms can be described in terms of Lyndon words. A string is a Lyndon word if it is the least in lexicographic (dictionary) order amongst the set of its rotations. For example, "data" is not a Lyndon word since it is not lexicographically least in the set data, atad, tada, adat, whereas "adat" is a Lyndon word. Similarly, "computer" is a Lyndon word while "science" is not. An important technique for handling string computations is to split an input string into substrings, each with a common pattern. One such method is the Lyndon factorization, where each substring is a Lyndon word. In this project you will study the mathematics of strings and Lyndon theory. You will then implement Lyndon factorization algorithms. Finally you will apply your programs to a practical application. For instance, you may choose to investigate the Burrows-Wheeler transform, which applies the Lyndon factorization, and is useful as a preprocessing step in compression methods like run-length encoding.

Supervisor Jackie Daykin

Value Full unit

Prerequisites Mathematical aptitude and competent programming skills.

References
1. M. Lothaire, Combinatorics on Words, Addison-Wesley, Reading, MA, 1983; 2nd Edition, Cambridge University Press, Cambridge, 1997.
2. J.P. Duval, Factorizing words over an ordered alphabet, J. Algorithms 4 (1983) 363381.
3. http://en.wikipedia.org/wiki/Burrows-Wheeler_transform
4. Search for "Lyndon words" and Lyndon algorithms" on the web.

10.26 Topics in Computational Geometry

Aim To study, implement and apply geometric computations.

Abstract In this project, either a range of more elementary geometric topics can be investigated or one topic can be studied more thoroughly. In all cases, data structures, computational complexity and robustness issues will be explored, and benchmarking may be appropriate. Geometric problems provide a good opportunity for developing an interesting GUI. Applications of the theoretical problems and algorithms are also relevant. Elementary topics include: polygon traversal; calculating polygon area from vector cross product; line segment intersection; simple triangulation; 2D convex hull; the point-in-polygon problem. More detailed studies include: Computing the convex hull. Intuitively, the convex hull of a set of points in the plane is the shape taken by a rubber band stretched around those points. Commencing with representational issues, convex hull algorithms will be implemented in 2D and then 3D. You will further demonstrate your algorithms in practice. A wide range of applications of efficient convex hull algorithms include robotics, shape analysis and smallest box problems. Triangulation algorithms. Commencing with a simple triangulation algorithm, the Delaunay triangulation will be implemented followed by the dual Voronoi diagram. Applications arise in computer graphics, network communications and geographical information systems, eg terrain mapping and GPS systems. Computing polygon intersection. The computation of polygon intersection arises in VLSI, graphics and robotics. Commencing with the primitive operation of line segment intersection, the design and analysis of algorithms for determining the intersection of convex and arbitrary polygons will be studied. Efficient algorithms here may involve advanced data structures.

Supervisor Jackie Daykin

Value Full or half unit

Prerequisites Mathematical aptitude, competant programming skills, and elementary geometry.

References

1. J. O'Rourke. Computational Geometry in C, Cambridge 1994.
2. M. de Berg, M. van Kreveld, M. Overmarsand O. Schwarzkopf. Computational Geometry: Algorithms and Applications, 2000, 2nd Edition, Springer-Verlag.
3. Preparata and Shamos. Computational Geometry an Introduction, Springer-Verlag, 1985.
4. Search the web - there are some cool demos of geometric algorithms!

10.27 Implementing the Event Calculus in mySQL

Aim To implement the Event Calculus as a mySQL database and perform performance evaluation of the resulting implementation.

Abstract The event calculus is a logical language for representing and reasoning about actions and their effects first. The basic components of the event calculus, as with other similar languages for reasoning about actions and change are fluents and actions. In the event calculus, one can specify the value of fluents at some given time points, a narrative of the actions that took place at given time points, and the effects of actions. The calculus can then deduce, using a logic-based axiomatization, which properties hold as a result of these descriptions.

The project should implement the event calculus in mySQL. The reason why such an implementation will be important is that it will utilize the efficient indexing and query processing mechanisms of relational databases to support the way conclusions are drawn. The approach will need to be tested in existing application data with large narratives of actions and evaluate their performance.

Supervisor K Stathis

Value Full unit

Prerequisites Excellent programming ability with SQL and Java. Good knowledge of Prolog is desirable but not strictly necessary.

References 1. M. Shannahan (1999) "The Event Calculus Explained," Springer Verlag, LNAI (1600): 409-30.

10.28 Virtual Agents control Lego Robots Avatars

Aim To implement an application in which the computationally limited LEGO robots present cognitive and cooperative behaviours and perform evaluation of the resulting implementation.

Abstract Enable low-cost commercial robots (LEGO) with limited resources to present more complicated reasoning, planning and communication capabilities. A way to achieve this is by enabling virtual agents in a Multi-Agent-System (MAS) platform to control the LEGO robots as avatars. Taking advantage of the planning, reasoning and connectivity capabilities of the agents, the resulting agent-LEGO robot entities should be able to explore their environment, create an ontology and communicate about previously unknown objects in order to perform cooperative tasks. All above tasks are not feasible using just the custom LEGO hardware.

The project should implement a LEGO robot remote control API that would enable virtual agents to control the robots as their avatars. Also, the virtual agents should be programmed to control the robots and implement simple cognitive and cooperative multi-robot scenarios. The agent framework will be provided. A lot of ongoing research deals with enabling simple robotic hardware to present complex behaviours and this implementation provides an opportunity to study an approach using virtual agents.

Supervisor K Stathis

Value Full unit

Prerequisites Excellent programming ability with Java. Good knowledge of Prolog is desirable but not strictly necessary.

References 1. Nikolaos Dipsis, Kostas Stathis. "EVATAR - A Prototyping Middleware Embodying Virtual Agents to Autonomous Robots.", ISAmI 2010: 167-175.

10.29 Cooperative Strategies in Multi-agent Systems

Aim To implement a virtual tournament of a set of programmable players engaging in a round robin for the Iterated Prisoner's Dillema game, test different strategies for these players to follow, identify winning strategies and evaluate the final results.

Abstract Imagine the following social interaction presented here as a kind of game that requires intelligent reasoning: You and your accomplish are both held prisoners, having been captured by the police and charged with a serious crime. The prosecutor interrogates you separately and offers each of you a deal. If one of you, the defector, incriminates the other, while the partner remains silent, then as the defector you will be convicted of a lesser crime and your sentence will be cut to one year for providing enough information to jail your partner. Meanwhile, your silent confederate will be convicted of a more serious crime and burdened with a four year sentence. If you both remain silent, and thus cooperate with each other, there will be insufficient evidence to convict either of you of the more serious crime, and you will each receive a sentence of two years for a lesser offense. If one the other hand, you both defect by incriminating each other, you will both be convicted of the more serious crime, but given reduced reduced sentences of three years for at lest being willing to provide information.

If you follow a purely selfish strategy as a player, the best outcome in terms of years in jail is that (a) you defect and the other person cooperates, (b) you both cooperate, and (c) you both defect. If both you and your accomplish are selfish, then you will both try to defect, hence this means 3 years in jail, which is the worst outcome. So if you could only trust each other, by following a strategy that is cooperative, you would both be better off than if you had acted selfishly. A more cooperative strategy will also give you benefits in the future, especially if you had to meet your accomplish after the sentence, engage in more crime, and as it likely need to play the game again, but now with knowledge of how your accomplish behaved in the previous game. This is called the Iterated Prisoner's dilemma game.

The project should design and implement a generic virtual tournament framework for running the Iterated Prisoner's dilemma game and test different types of player agents. A type of agent will be characterised simply by the kind of strategy the agent is using in the game (e.g. always defect). Your work should show how to implement these kinds of agents and evaluate their performance in terms of their cooperation characteristics. A successful project will need to be balanced both in terms of the intelligent behaviour of individual players (strategies) and the generality of the software engineering techniques used to implement the tournament.

Supervisor K Stathis

Value Full unit

Prerequisites Excellent programming ability with Java. Good knowledge of Prolog/SQL is desirable but not strictly necessary.

References
1. M. Nowak, R. May and K. Sigmund. *The Arithmetics of Mutual Help*, Scientific American, June 1995.
2. K. Stathis *A game-based architecture for developing interactive components in computational logic*, Journal of Functional and Logic Programming, 20(1), Jan 2000, MIT Press.

10.30 Proof search in propositional logic

Aim To produce a program that will take a set of premises and a conclusion, written as formulas in propositional logic, and decide whether the conclusion is a logical consequence of the premises.

Abstract Several decision procedures exist for this problem, including the construction of truth tables and resolution-based proof. You may choose to implement several decision procedures and compare their efficiency; or you may choose to extend one decision procedure to a more advanced logic, such as predicate logic or modal logic.

Supervisor R. Adams

Value Full unit

Prerequisites An interest in logic and good programming skills.

10.31 Proof assistant for propositional logic

Abstract The aim of this project is to produce a program that will help the user to build step-by-step a proof, written in the language of propositional logic, in such a way that the finished proof is guaranteed to be correct.

Supervisor R. Adams

Value Full unit

Prerequisites An interest in logic and good programming skills.

10.32 Exact Algorithms for NP-hard Problems

Abstract NP-hard problems cannot be solved in polynomial time (unless P=NP), but that doesn't mean we have to just give up. There are two main approaches to solving an NP-hard problem: finding algorithms that give approximate answers in polynomial time (approximation algorithms), and finding algorithms that give exact answers in exponential time (exact algorithms).

This project shall involve investigating some exact algorithms for NP-hard problems, including the Travelling Salesman Problem and the Minimum Fill-In Problem. These algorithms use dynamic programming, a technique for breaking one large decision problem into several smaller ones.

The student shall implement several exact algorithms and investigate their behaviour in practice, and their theoretical properties.

This project shall therefore involve both theoretical work and implementation. The student may choose to make it a mostly theoretical project, or a mostly practical project, as time proceeds.

Supervisor R. Adams

Value Full unit

Prerequisites Mathematical ability, an interest in the theory of algorithms, good programming skills.

10.33 Analysis of genetic texts to predict splice sites*

Aim To apply pattern recognition techniques to analyze genomic data for splice sites identification

Abstract Finding protein-coding genes in sequenced genomes is one of major practical and theoretical tasks of computational biology. Splice sites are functional elements that define positions of exons that encode protein sequences. The project involves an application of a few pattern recognition techniques (such as weight matrices, discriminant analysis, neural networks or SVM) to splice sites identification. The study should analyze the effects of different features selection and comparative characteristics of different methods, simplicity of their implementation, execution speed and accuracy of splice site prediction in genomic sequences.

Supervisor V. Solovyev

Value Full unit

Prerequisites Machine Learning or Neural Network course would be helpful. Programming skills.

References 1. Richard O. Duda, Peter E. Hart, David G. Stork, Pattern classification, New York; Chichester : Wiley, 2001.

2. Burset M., Seledtsov I., Solovyev V. (2000) Analysis of canonical and non-canonical splice sites in mammalian genomes. Nucl. Asids Res., 28, 21, 1-12.

10.34 Search for repeats in genomic sequences

Aim To apply fast text searching techniques to analyze genomic data for imperfect repeats identification

Abstract Different types of repeat elements occupy often more then 40% of genome DNA. Identifying and masking repeats is important initial step in genome annotation. Project involves devising strategies for finding different classes of repeats: low complexity regions, tandem repeats, dispersed repeats. It is important to apply different hashing techniques and design effective computer programs to analyze long genetic texts (chromosome sequences comprising 100 000 000 letters).

Supervisor V. Solovyev

Value Full unit

Prerequisites Programming skills. Bioinformatics course would be helpful.

References

1. Donald E. Knuth, Donald Ervin Knuth. The Art of Computer Programming, Volumes 1-3. Addison-Wesley Pub Co; 2nd edition, 1998.
2. Initial sequencing and analysis of the human genome. Nature. 2001 Feb 15; 409 (6822), 860-921.

10.35 Application of Hidden Markov Models to recognizing various states of protein sequences*

Aim To apply HMM techniques to analyze protein sequence for their partition to ordered and disordered segments or to predict protein secondary structures.

Abstract It is increasingly evident that intrinsically unstructured protein regions play a key roles in cell-signaling, regulation and cancer, which makes finding them extremely useful for discovery of anticancer drugs. This two classes partition task can be extended to predict three classes of protein sequence regions such as a-helix, b-strand and coil using variants of powerful Hidden Markov Models technique.

Supervisor V. Solovyev

Value Full unit

Prerequisites Bioinformatics course would be helpful. Programming skills.

References 1. R. Durbin et al. Biological sequence analysis. Cambridge University Press, 1998

2. Dunker et al., Biochemistry (2002) 41 (21), 6573 -6582.

10.36 Android game development

Aim To develop a simple game for Android platform using Android SDK (Software Development Kit).

Abstract Game technologies and game user interfaces are now being more commonly used in serious applications, and the market for serious games is growing. This makes it important for students to learn how to develop games even the students do not target to work in the game industry. You will study how to build well-designed software architectures that copes with variations in hardware configurations, functional modifications, and network real-time constraints.

Supervisor V. Solovyev

Value Full unit

Prerequisites Good Java programming skills.

References 1. Mario Zechner. Beginning Android Games. 2011 - Springer.

10.37 Android game development

Aim To develop a simple game for Android platform using Android SDK (Software Development Kit).

Abstract Game technologies and game user interfaces are now being more commonly used in serious applications, and the market for serious games is growing. This makes it important for students to learn how to develop games even the students do not target to work in the game industry. You will study how to build well-designed software architectures that copes with variations in hardware configurations, functional modifications, and network real-time constraints.

Supervisor V. Solovyev

Value Full unit

Prerequisites Good Java programming skills.

References 1. Mario Zechner. Beginning Android Games. 2011 - Springer.

10.38 Comparison of machine learning algorithms*

Aim To implement and compare on benchmark data sets various machine learning algorithms.

Abstract This project involves implementing a range of machine learning algorithms, from the simplest to very sophisticated, such as: nearest neighbours [1], decision trees [1], support vector machines [2-3], and nearest neighbours using kernels. Good sources of data are the UCI data repository [4] and the Delve repository [5]. The project can concentrate on applications to some specific area, such as medical diagnosis or option pricing.

Supervisor V. Vovk

Prerequisites Java or C++ programming skills

Value Full unit

References

1. Tom M. Mitchell. *Machine Learning.* McGrow-Hill, New York, 1997.
2. Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995. Second edition: 2000.
3. Vladimir N. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.
4. Seth Hettich and Steven D. Bay. *The UCI KDD Archive.* University of California, Department of Information and Computer Science, Irvine, CA, 1999, `http://kdd.ics.uci.edu`.
5. Delve archive. `http://www.cs.toronto.edu/~delve`.

10.39 Conformal predictors for classification and regression*

Aim Implement and test machine learning algorithms that complement their predictions with measures of confidence.

Abstract A disadvantage of standard machine learning algorithms is that they only produce "bare predictions", i.e., predictions without any indications of their reliability. The technique of conformal prediction [1-3] makes it possible to build measures of confidence on top of almost any prediction algorithm. Possible specific areas of application of conformal predictions include medical diagnosis and option pricing (data sets will be available from the supervisor).

Supervisor V. Vovk

Prerequisites Java or C++ programming skills

Value Full unit

References 1. Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9:371–421, 2008.

2. Alex Gammerman and Vladimir Vovk. Hedging predictions in machine learning (with discussion). *The Computer Journal*, 50:151–177, 2007. Also published as Technical Report `arXiv:cs/0611011` [cs.LG], `arXiv.org` e-Print archive, November 2006.
3. Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, New York, 2005.

10.40 Prediction with expert advice*

Aim To implement algorithms for prediction with expert advice and analyze their performance.

Abstract The technique of prediction with expert advice [1] is applicable in the situation where the prediction algorithm has access to predictions made by a range of experts; it guarantees that the loss suffered by the prediction algorithm is not much worse than the loss suffered by the best expert. In this project, various algorithms for prediction with expert advice will be applied to the predictions made by bookmakers or prediction markets for the results of various sporting and political events (such as football matches, as in [2], or elections). The next step is try and perform better than the best expert (even negative results will count!).

Supervisor V. Vovk

Prerequisites Java or C++ programming skills

Value Full unit

References 1. Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games.* Cambridge University Press, Cambridge, England, 2006.

2. Vladimir Vovk and Fedor Zhdanov. Prediction with expert advice for the Brier game. *Journal of Machine Learning Research* 10:2413–2439, 2009.

10.41 Markov chain Monte Carlo algorithms*

Aim Explore Markov chain Monte Carlo methods.

Abstract Markov chain Monte Carlo methods have become extremely popular in machine learning and science in general. An interesting application is to deciphering codes ([1], Section 1).

Supervisor V. Vovk

Prerequisites Java or C++ programming skills

Value Full unit

References 1. Persi Diaconis. The Markov chain Monte Carlo revolution. *Bulletin (New Series) of the American Mathematical Society*, 46:179–205, 2008.

10.42 Solving games using prediction with expert advice*

Aim Apply prediction with expert advice to finding solutions of two-person zero-sum games.

Abstract A famous result in game theory is von Neumann's statement that in two-person zero-sum games the two opponents always have optimal randomized strategies. The technique of prediction with experts advice will be applied to finding the optimal strategies (see [1], Chapter 7 and the references therein). The next step would be to apply similar techniques to more general games.

Supervisor V. Vovk

Prerequisites Java or C++ programming skills

Value Full unit

References 1. Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games.* Cambridge University Press, Cambridge, England, 2006.

10.43 Value at Risk

Aim Write a program for computing Value at Risk. Check its performance using backtesting.

Abstract It is a very difficult and important problem to estimate the riskiness of a portfolio of securities. Value at Risk (VaR) is a tool for measuring financial risk. The goal of this project is to implement one of the methodologies for VaR computation (either under normal conditions or using "crash metrics") and test it on the real financial data. Other monetary measures of risk may also be studied.

Supervisor V. Vovk

Prerequisites Java or C++ programming skills

Value Full unit

References 1. John C. Hull. *Options, futures and other derivatives*, 7th ed., Pearson, Upper Saddle River, NJ, 2009.

2. Hans Folmer and Alexander Schied. *Stochastic Finance: An Introduction in Discrete Time*, 3rd ed., de Gruyter, Berlin, 2011.
3. http://www.gloriamundi.org

10.44 File Compression Utilities

Aim To implement file compression tools using different compression algorithms and to compare their performance on different kinds of data

Abstract File compression utilities compress files and thus allow us to save disk space. What stands behind utilities like zip or arj is very interesting and highly nontrivial algorithms for processing information. Different kinds of algorithms are good for compressing data of different kind. The area of mathematics concerned with this kind of problems is called information theory. This project consists of implementing different compression methods, including Shannon-Fano, Huffman, and Lempel-Ziv codes, testing them on different types of files (e.g., text files, picture, or executables) and comparing them against standard compression utilities.

References:

1. D. Salomon, Data compression: the complete reference. New York, Springer, c1998

2. T. Lynch, Data compression: techniques and applications, Belmont, Calif., c1985

Supervisor Yuri Kalnishkan

Value Full unit

Prerequisites C++ (preferably) or Java programming skills

10.45 Prime Numbers and Cryptosystems

Aim To implement utilities dealing with prime numbers and cryptographic applications based on them.

Abstract Modern cryptography allows us to perform many types of information exchange over insecure channels. One of these tasks is to agree on a secret key over a channel where messages can be overheard. This is achieved by Diffie-Hellman protocol. Other tasks include public key and digital signature schemes; RSA key exchange can be used for them. These protocols are of great importance for bank networks.

Most such algorithms are based upon number theory, namely, the intractability of certain problems involving prime numbers. The project involves implementing basic routines for dealing with prime numbers and then building cryptographic applications using them.

The project may be developed into several directions: modern algorithms for dealing with prime numbers, integration with symmetric techniques such as DES or AES, or cryptographic attacks.

References:

1. T.H.Cormen, C.E.Leiserson, and R.L.Rivest, Introduction to algorithms, Cambridge, Mass.; London: MIT , c1990

2. J.Pieprzyk, T.Hardjono, J.Seberry, Fundamentals of Computer Security, Springer, 2003

3. N.Ferguson and B.Schneier, Practical cryptography, New York: Chichester: Wiley, 2003

Supervisor Yuri Kalnishkan

Value Full unit

Prerequisites 1. Taking CS3760 (Information Security) in the 3rd year is recommended.

2. C++ or Java programming skills

3. Interest in mathematics.

10.46 Regression Methods *

Aim To implement the Ridge Regression algorithm for machine learning and to study its performance on various datasets under various circumstances

Abstract Ridge Regression (RR) is a popular machine learning kernel algorithm. It is applied to problems similar to the following. The Boston housing database contains records describing houses in Boston. For each house the values of attributes like the number of rooms, the distance from the city centre etc are known and the price, as calculated by an expert estate agent, is given. The goal of the learner is to determine the dependency between attributes and prices. The program is first shown a number of training examples and then is tested on test examples. This project may have several variations targeting different problems. One is the problem of overfitting. It is known that in many situations more sophisticated techniques achieving better results in training later behave worse than coarse methods; this is called overfitting. The task is to investigate where and when overfitting occurs. Another task is to compare the performance of Ridge Regression against the performance of the recently developed Kernel Aggregating Algorithm Regression algorithm and its iterative version. A possible direction is applying Ridge Regression and its variations such as recently developed KAARCh to financial data, e.g., applied volatility of options. The project may lead to new results of scientific interest.

References:

1. B.Schoelkopf and A.J.Smola, Learning with Kernels, MIT Press, 2002

2. N.Cristianini and J.Shaw-Taylor, An Introduction to Support

Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2000

3. A.Gammerman, Y.Kalnishkan, and V.Vovk, Online Prediction with Kernels and the Complexity Approximation Principle. In Uncertainty

in Artificial Intelligence, Proceedings of the Twentieth Conference, pages 170-176. AUAI Press, 2004.

4. S.Busuttil and Y.Kalnishkan, Online Regression Competitive with Changing Predictors. In Proceedings of The 18th International Conference on Algorithmic Learning Theory, ALT 2007.

Supervisor Yuri Kalnishkan

Value Full unit

Prerequisites 1. Taking CS3920 (Computer Learning) in the 3rd year is recommended

2. Good command of mathematics (particularly, Linear Algebra) would be an advantage because the project is mathematically intensive

3. Good command of C++ and/or Java

10.47 A game-playing environment

Aim Design and develop a game-playing program.

Abstract A game-playing program should be developed in a language such as Java. Besides providing appropriate error reports when illegal moves are made, the program should have a nice graphical interface and/or provide more advanced features such as allowing multiple games to be played at the same time.

Supervisor Zhaohui Luo

Value Full unit

Prerequisites Good programming skills

10.48 Searching for reuse in formal analysis

Aim To investigate searching methods in formal analysis.

Abstract This project is to investigate searching techniques for libraries of proof development systems, and develop a supporting tool to help the user to find reusable artifacts in formal analysis.

Supervisor Zhaohui Luo

Value Full unit

10.49 Program transformation and refactoring

Aim To investigate tool support in program transformation or refactoring.

Abstract Programs can be transformed into better programs by applying provably correct transformation rules. For example, by changing the structure of a program without changing its semantics, one may obtain a more efficient or more understandable program.

You should study and obtain a good understanding of one of the following techniques: (1) Refactoring of programs; (2) Transformation in program development. An experimental transformation tool for program refactoring or development by transformation is to be implemented.

Supervisor Zhaohui Luo

Value Full unit

10.50 Optimal winner determination in combinatorial auctions *

Aim To implement and experiment a search algorithm for optimal winner determination in combinatorial auctions

Abstract Auctions are popular ways of allocating items (goods, resources, services, etc) among agents. They are extensively used among human bidders in a variety of task and resource allocation problems. In combinatorial auction setting, there is one seller and multiple bidders. Each bidder may place bids on combinations (i.e. bundles) of indivisible items and any number of his/her bids can be accepted. The auctioneer determines the winners – that is, decides which bids are winning and which are losing – so as to maximise the seller's revenue. Unfortunately, winner determination in combinatorial auctions is hard. The project will require implementing an approximate algorithm for searching optimal winner determination and analysing its performance on example cases.

Supervisor Z. Luo

Prerequisites C++ or Java programming skills

References 1. Tuomas Sandholm "Algorithm for optimal winner determination in combinatorial auctions", Artificial Intelligence 135 (2002), pages 1-54.

Value Full unit

10.51 A simple TCP simulator

Aim To model a simple TCP protocol engine and demonstrate its flow control behaviour

Abstract Transmission control protocol (TCP) is one of the key communication protocols used on Internet. TCP is responsible for verifying the current delivery of data from client to server. Data can be lost in the network due to congestion. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received. This project will implement a TCP simulator in Java and demonstrate flow control behaviour of TCP under congestion conditions. The simulator will have graphical user interface for configuring TCP model and displaying data being sent or received.

Supervisor Z. Luo

Value Full unit

Prerequisites Good Java programming skills and basic knowledge on data communications

References
1. http://www.ietf.org/rfc.html
2. RFC 793, J. Postel "Transmission control protocol", Sept. 1981.
3. RFC 2581, M. Allman, V. Paxson and W. Stevens "TCP congestion control", April 1999. http://www.faqs.org/faqs/internet/tcp-ip/resource-list/

10.52 Discovering Bayesian belief networks from data *

Aim Write a programme to construct Bayesian belief networks from data

Abstract Bayesian belief networks are powerful knowledge representation and reasoning tools under uncertainty. A Bayesian belief network is a directed acyclic graph with a conditional probability distribution for each node. Typically, nodes represent domain variables and arcs between nodes indicate probabilistic dependencies. The project will implement an efficient algorithm for constructing Bayesian belief networks from data. The construction process is based on the computation of mutual information of node pairs. The implemented algorithm will then be evaluated on example data sets.

Supervisor Z. Luo

Value Full unit

Prerequisites C++ or Java programming skills

References
1. J. Pearl "Probabilistic Reasoning in Intelligent Systems: networks of plausible inference", Morgan Kaufmann, 1988.
2. C.K. Chow and C.N. Liu "Approximating discrete probability distributions with dependence trees", IEEE Trans. on Information Theory, 14, pages 462-467, 1968.
3. J. Cheng, D.A. Bell and W. Liu "Learning belief networks from data: an information theory based approach", In Proceedings of ACM CIKM'97, 1997.

10.53 Microarray gene expression data analysis *

Aim To apply machine learning techniques to analyse microarray gene expression data for cancer classification

Abstract A reliable and precise classification of tumours is essential for successful treatment of cancer. DNA microarrays are part of a new class of biotechnologies which allow the monitoring of expression levels for thousands of genes simultaneously. DNA microarray gene expression data can be used to characterise the molecular variations among tumours by monitoring gene expression profiles on a genomic scale. However, an unusual feature of the microarray gene expression data is the very large number of attributes (genes) relative to the small number of observations (tumour samples).

This project involves an application of machine learning techniques to an example of Leukemia data set and experiments of different feature selection methods for cancer classification.

Supervisor Z. Luo

Prerequisites Strong Java or MATLAB programming skills, good knowledge of machine learning

References

1. Eng-Juh Yeoh, Mary E. Ross, Sheila A. Shurtleff, W. Kent Williams, Divyen Patel, Rami Mahfouz, Fred G. Behm, Susana C. Raimondi, Mary V. Relling, Anami Patel, Cheng Cheng, Dario Campana, Dawn Wilkins, Xiaodong Zhou, Jinyan Li, Huiqing Liu, Ching-Hon Pui, William E. Evans, Clayton Naeve, Limsoon Wong, and James R. Downing (2002) "Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling" Cancer Cell: March 2002 Vol.1. (http://www.stjuderesearch.org/data/ALL1/index.html)
2. Richard O. Duda, Peter E. Hart, David G. Stork, Pattern classification, New York ; Chichester : Wiley, c2001.

Value Full unit

10.54 Routing protocol and algorithm simulation

Aim To use simulation technique to study the behaviour and performance of a routing protocol

Abstract Data communication networks are interconnected via routers. Internet communications are controlled by routers which switch data from one network to another at the crossing points. The main function of the routers are to run routing protocols and algorithms in order to determine optimal routing paths. Performance of the routing protocol and algorithm affects the network stability and load-balancing.

This project will develop a computer programme to study the behaviour and performance of a routing protocol using simulation techniques. Examples of such protocols are RIP and OSPF.

Supervisor Z. Luo

Prerequisites Good Java programming skills, knowledge of communications and networks

References

1. Kurose, James F. and Ross, Keith W., Computer networking :a top-down approach featuring the Internet, Reading, Mass. : Addison-Wesley, 2001.
2. System Simulation: The Shortest Route to Applications (http://www.ubmail.ubalt.edu/ harsham/simulation/sim.htm)

Value Full unit

10.55 Kerberos attacks and improvements (IY3821 only)

Aim To investigate practical weaknesses in Kerberos.

Supervisor Chris Mitchell

Value Full unit

Abstract Kerberos is a widely implemented authentication system, which depends on a secret shared by a user and an Authentication Server (AS). It is well-established (and documented widely on the Internet) that Kerberos v5 has a major vulnerability if this shared secret is based on a weak password, such as a word in the English language.

This project involves implementing an attack on Kerberos exploiting this vulnerability, by using an intercepted message sent from the client machine to the AS as the basis of an exhaustive password search, e.g. using a dictionary of likely passwords. If time permits, possible modifications to Kerberos should be investigated which would prevent such an attack, using secure password-based key establishment techniques (e.g. such as SPEKE).

Prerequisites Programming skills.

References There is an excellent introductory article on the Kerberos protocol on Wikipedia. An outline of the operation of the protocol was given in IY2760.

More specific references include the following:

[1] B. Clifford Neuman and Theodore Ts'o, 'Kerberos: An Authentication Service for Computer Networks', *IEEE Communications*, **32 (9)** (September 1994) pp.33–38.

[2] John T. Kohl, B. Clifford Neuman, and Theodore Y. T'so , 'The Evolution of the Kerberos Authentication System', in: D. Johansen and

F. M. T. Brazier (eds.), *Distributed open systems*, IEEE Computer Society Press (1994) pp.78–94.

10.56 Browser root key management tool (IY3821 only)

Aim To produce a tool to manage the root public keys held by a web browser.

Supervisor Chris Mitchell

Value Full unit

Abstract Just about every web browser incorporates an implementation of SSL/TLS, used to set up secure channels to particular web sites. Establishing an SSL channel requires the server to send the client a copy of its public key certificate, which is verified using a root public key held by the browser. The security of the SSL channel thus depends on the browser only having 'good' root public keys. However, it has been shown that it is not difficult to add corrupt public keys to the root public key store of at least one widely used browser. Hence it would be helpful to have a means of better managing these root public keys.

Browsers typically provide the means to view, add and remove root public keys, but the functionality of this management interface is usually very limited. Stand alone scanners have also been produced to verify the contents of a root public key store for specific browsers. This project has the goal of producing an integrated management tool for one or more browsers, which allows for scanning and other management functions for the root public key store, e.g. highlighting root public keys which have questionable security properties.

Prerequisites Programming skills.

References There is nice introductory article on SSL available on Wikipedia. It is also briefly covered in IY2760.

[1] A. Alsaid and C. J. Mitchell, 'Installing fake root keys in a PC', in: D. W. Chadwick and G. Zhao (eds.), *Public Key Infrastructure, Second European PKI Workshop: Research and Applications, EuroPKI 2005, Canterbury, UK, June 30 – July 1, 2005. Revised Selected Papers*, Springer-Verlag LNCS 3545, Berlin (2005), pp.227–239.

[2] A. Alsaid and C. J. Mitchell, 'A scanning tool for PC root public key stores', in: C. Wolf, S. Lucks, and P.-W. Yau (eds.), *WEWoRC 2005 — Western European Workshop on Research in Cryptology, Leuven, Belgium, July 2005*, LNI P-74, Gesellschaft fuer Informatik (2005), pp.45–52.

10.57 Fuzz-testing (IY3821 only)

Aim To write and deploy a fuzz-testing program.

Supervisor Chris Mitchell

Value Full unit

Abstract Fuzz testing or fuzzing is a software testing technique that provides invalid, unexpected, or random data to the inputs of a program. If the program fails (for example, by crashing or failing built-in code assertions), then a defect in input handling has been detected (and can then be rectified). File formats and network protocols are the most common targets of fuzz testing, but any type of program input can be fuzzed. Interesting inputs include environment variables, keyboard and mouse events, and sequences of API calls. Even items not normally considered 'input' can be fuzzed, such as the contents of databases, shared memory, or the precise interleaving of threads.

The objective of this project is to choose a type of input, to build a fuzz-tester for this type of input, and to test it on one or more programs. A good target might be programs taken from the large suites of open source programmes available — indeed, if faults are found, it would be helpful to try to determine the cause of the problem, and the use of an open source program should make this more straightforward.

Prerequisites Programming skills.

References There is nice introductory article on fuzzing available on Wikipedia, on which the first paragraph of the above description was based. There is also an extensive literature on this type of testing.

10.58 SSL/TLS investigation (IY3821 only)

Aim To develop an SSL/TLS version and cryptography suite tester, and to use it to survey the security of web sites.

Supervisor Chris Mitchell

Value Full unit

Abstract SSL/TLS is a widely used security protocol, which provides a secure channel between two communicating parties. It is built into just about every web browser, and is used by web sites wishing to establish secure channels with clients, e.g. prior to performing e-commerce transactions. The establishment of the channel is typically transparent to the end user of the browser (except for the presence of a padlock symbol, or something similar).

The protocol starts with a negotiation phase, in which the client proposes the use of a particular version of the protocol (3.0 is the latest version of SSL, and 3.1 indicates TLS), and also proposes a possible set of cryptographic algorithms. The server then decides whether it will support the version of SSL/TLS proposed by the client, and selects the set of algorithms it would like to use from amongst those offered by the client.

The purpose of the project is to write a tool designed to discover which versions of SSL and which cryptographic algorithms are supported by a web site, and then to use the tool to test a selection of widely used web sites. This is of interest as it has been alleged in the past that some sites will accept old and potentially insecure versions of SSL (e.g. version 2.0) and also may allow the use of algorithms of dubious security (e.g. MD5).

Prerequisites Programming skills.

References As always, there is a very helpful introduction to SSL/TLS on Wikipedia. There are also several books devoted to this topic, and a wide range of academic literature.

10.59 ISO/IEC 27002 security management tool (IY3821 only)

Aim Develop a tool to assist in security management audits using ISO/IEC 27002.

Supervisor Chris Mitchell

Value Full unit

Abstract The international standard ISO/IEC 27002, formerly known as ISO/IEC 17799, was developed from BS 7799, a British Standard. This standard provides a 'Code of practice for information security management'. More specifically, it provides detailed advice on best security practice for the management of IT systems.

The document provides a long list of recommendations (a sort of checklist). The goal of this project is to provide a tool which can be used to aid the implementation of ISO/IEC 27002 in an organisation, by enabling structured information relevant to each item in the standard to be stored, and for individual items on the list to be 'checked off'. This would then enable reports to be automatically generated indicating to what degree the organisation is compliant with the standard.

Completing the project will require gaining a detailed understanding of the standard, and the development of a tool which meets envisaged management needs.

Prerequisites Programming skills.

References There is a helpful (albeit brief) introductory article on ISO/IEC 27002 on Wikipedia. There is a growing number of books on the topic.

10.60 EMV (chip and PIN) card capability tester (IY3821 only)

Aim Develop an EMV (chip and PIN) card capability tester, and use it to survey existing card issue practices.

Supervisor Chris Mitchell

Value Full unit

Abstract In the UK and much of Europe, all newly issued credit and debit cards are chip-enabled, and can communicate with terminals conforming to the EMV standards. The EMV standards specify three different Card Authentication Methods (CAMs), i.e. three different methods to be used by a merchant terminal to verify that a card is genuine. The least secure method, called SDA, is believed to be widely used.

The purpose of this project is to develop the software necessary to interrogate debit and credit cards and to discover (a) which CAM they support, and (b) the key lengths used by the card and any public key certificates it contains (every card contains at least one certificate). This tool can then be used to try to understand the policies of banks with regard to CAM use, and also their policies with regard to key lengths and frequency of card issuer key changes. The project will entail programming a smart card reader, and understanding the protocols used to communicate with smart cards (as well as analysing the data recovered from cards). Data recovered from cards (which could be potentially sensitive) will need to be handled with care, and in accordance with a policy agreed at the beginning of the project. [This project will only proceed if a generally acceptable policy for handling sensitive card data can be agreed].

Prerequisites Programming skills.

References The EMV standard and its use in chip and PIN cards was described in IY2760. The EMV standards are available from www.emvco.com There is a brief but not very helpful historical article on EMV on Wikipedia.

10.61 User authentication token (IY3821 only)

Aim Develop software to enable a PDA or a mobile phone (or some other mobile device) to be used as a generator of one-time passwords.

Supervisor Chris Mitchell

Value Full unit

Abstract For many years a range of products have been available designed to provide mobile users with 'one time passwords'. These products typically involve equipping every user with a special token, which shares a unique secret key with a server database. The secret key is then used to generate one-time passwords, e.g. as a function of a challenge generated by the server, or as a function of a clock in the token.

The objective of this project is to develop a low-cost system of this type, using an existing user device in place of a special-purpose token. The project will involve designing an appropriate protocol, and writing software to run on the mobile device. Software for the server will also need to be produced.

Prerequisites Programming skills.

References The course notes for IY2760 contain an extensive discussion of user authentication techniques.

10.62 Test suites for cryptographic algorithms (IY3821 only)

Aim To develop a test suite for a set of standardised cryptographic techniques.

Supervisor Chris Mitchell

Value Full unit

Abstract There are now a wide range of standardised cryptographic techniques, covering encryption (both symmetric and asymmetric), MACs, signatures, hash functions, authentication protocols, and key establishment techniques. Most of the standards contain a small number of 'example' outputs of the algorithm, designed to enable implementors to test that their implementations work correctly. However, more comprehensive testing facilities are not always available.

The objective of this project is to choose a class of algorithms, and to develop a set of tools for testing implementations of these algorithms. This will include testing for correctness and for performance.

Prerequisites Programming skills.

References Copies of standards relevant to the subject area can be supplied.