COMP114 Experimental Methods in Computing

Introduction

Module Information

- Coordinator/Lecturer: Paul Dunne
- Office: Room 2.04, Ashton Building
- e-mail: P.E.Dunne@liverpool.ac.uk

Module Aims

1. Make clear the importance of experimental methods in CS.
2. Distinguish differences between testing *correctness* and evaluating *performance*.
3. Introduce aspects relating to choice of experimental data.
4. Present standards methods for generating data.
5. Consider methods for critical analysis of experimental results.
6. Give practical experience of experiment.

Intended Learning Outcomes

1. Awareness of different areas where experiments are used in CS.
2. Ability to distinguish between methods for testing from methods for evaluating.
3. Understand factors involved in designing experiments.
4. Ability to choose and apply suitable experimental methods in a range of cases.
5. Ability to assess, analyse, and present experimental results in an objective style.

Module Assessment

- 100% Continuously Assessed.
- 4 Practical Exercises.
    - Exercise 1: 10%
    - Exercise 2: 20%
    - Exercise 3: 30%
    - Exercise 4: 40%
- Details provided later in the module.

Why are experiments needed?

- The use of experimental methods is important in “traditional” sciences:

  Physics, Chemistry, Biology
- For example: clinical trials of new drugs (Medicine); testing hypotheses about atomic structure (Physics); testing properties of materials (Chemistry), etc.

Why are experiments needed?

- In traditional sciences, experiment is often the *only* method by which *evidence* in support of a *model* or *theory* or *hypothesis* can be produced.
- The validity of a model often cannot be shown by a *rigorous* "*mathematical proof*" – instead suitability is judged by testing how accurately the behaviour *predicted* by the model matches the results of observations obtained by *experimental studies*.

Why experiments in Computing?

- Programs, computing systems, etc. are *artefacts* constructed through human effort.
- If X has produced a program, P, to carry out some task T, how would experiment convince others that P did deliver T?
- Why would experiments even be needed at all – X could simply "*mathematically prove*" that P was a correct program.

Why experiments in Computing?

- This, however, is not always a realistic option: even in cases where, in principle, formal proof could be used. Reasons?

a. A complete formal analysis (correctness, run-time performance, memory usage) may be very difficult to carry out.

b. Users may be unconvinced by such formal analysis if they do not understand how it was produced.

Why experiments in Computing?

- A number of applications of Computing – both traditional and very recent topics – raise significant problems when *evaluating* solutions.
- In such cases, precise, formal analytic proofs that "*method P out-performs method Q*" are not possible.
- Convincing evidence that one approach "*is better than*" another can only be derived by suitable experimental study.

Why experiments in Computing?

- For assessing how fast an approach is, users are often more concerned with "typical" (i.e. average) speed than "worst-case" aspects.
- Thus users wish to know "does this method work well *most of the time*?" rather than "what's the *worst possible time* this *could ever take* with certain data?"
- Analytic methods have tended to focus on the latter – formal analysis of "typical" performance is, often, extremely hard or infeasible.

Traditional Computing Examples

- Software testing –

  Attempt to find bugs in a program by running experiments with suitable data.
- System performance tuning –

  Varying system parameters, - e.g. page sizes in virtual memory, priority given to system processes, etc, to identify settings that give "best" performance.
- Queuing Strategies

  Some resources can only handle one job at a time – e.g. printers – experimental models used to assess "fairness" of queuing methods such as "first-in first-out", "longest job runs last" and so on.

More recent areas of interest

- Game-playing strategies – both standard two player games such as Chess, and models from so-called "Mathematical Game Theory" (used in economics and other fields).
- Bidding agents in e-auctions.
- Coalition formation and "profit" sharing techniques in multi-agent systems settings.
- Simulation of new computing paradigms – e.g. Genetic algorithms, Neural networks.
- Studies of "algorithm engineering".
- Studies of heuristics for "hard" optimisation problems, e.g. scheduling, timetabling.

Aspects in Common

- These applications have one important feature that motivates using experiment rather than analysis –

  A formal, mathematical analysis, when possible at all, is extremely challenging except in very naïve settings.

Issues in using Experimental Methods

- Rationale
  What is the study intended to demonstrate?
- Reproducibility
  Can the study be repeated and similar results obtained?
- Is the choice of data reasonable?
- Are results given and analysed objectively?
- Would “similar” behaviour occur with much larger examples?

Summary

- The questions raised on the previous slide are the main focus of this module.
- We will consider aspects of experiment design, conduct, and reporting.
- We will also look in more detail at some example experimental studies in Computing.
- The aim of the subsequent assessments is to give some experience of this important aspect of Computing in a practical setting.