An Open and Reproducible Paper on Openness and Reproducibility of Papers in Computational Science

Sylwester Arabas[*], Michael R. Bareford[†], Ian P. Gent[†], Benjamin M. Gorman [‡], Masih Hajiarabderkani[†], Tristan Henderson[†], Luke Hutton[†], Alexander Konovalov[†], Lars Kotthoff[§], Ciaran McCreesh[¶], Ruma R. Paul[‖], Karen E. J. Petrie[‡], Daniël Reijsbergen[**]

August 8, 2014

Abstract

This paper was written by participants of the First Summer School on Experimental Methodology in Computational Science Research. We report several case studies performed during the school and systematize our experiences. This paper is *open* and *reproducible*: the whole process of writing this paper is captured in the GitHub repository hosting both the source of the paper, supplementary codes and data; we are providing setup for several experiments on which we were working; finally, we are trying to describe what we have achieved during the week of the school in a way that others may repeat (and hopefully improve) our experiments.

This paper is a work-in-progress. It reports the state of our work at the end of a busy week of a Summer School, but may be incomplete in various ways.

1 Introduction

This paper has been written during the week of the Summer School on Experimental Methodology in Computational Science Research [1]. The school was comprising of lectures, tutorials and a hackathon, all united by the topic of reproducible research in computational sciences.

One goal of the Summer School was to write a paper about reproducible research in Computational Sciences *by the end of the week*, and this is what you

[*]University of Warsaw, Warsaw, Poland
[†]University of St Andrews, St Andrews, Fife, UK
[‡]University of Dundee, Dundee, UK
[§]University College Cork, Ireland
[¶]University of Glasgow, Glasgow, Scotland
[‖]Université catholique de Louvain, Louvain-la-Neuve, Belgium
[**]University of Edinburgh, Edinburgh, Scotland
[1]St Andrews, Scotland, August 4-8, 2014, https://blogs.cs.st-andrews.ac.uk/emcsr2014/

are reading now. The hackathon aspect of the Summer School therefore became focussed on working on this paper and then writing it.

For the hackathon (actually split into several sessions), attendees were asked to 'bring a paper', i.e. to bring the suggestion of a computational experiment to be reproduced as part of the summer school. The experiment should be either from a published paper or, if unpublished, have the agreement of all authors concerned for this activity including the publication of a replication.

Following participants' presentations of their selected papers, we decided that they fell naturally into three topics, and three groups were performed to work on these topics as case studies. The resulting case studies will be presented below in Sections 6 – 8.

2 The state of the art

In this section we are going to describe the current state of reproducible computational science, and in particular the Recomputation.org project, which is aimed at implementing the infrastructure for recomputable experiments following the statements of the Recomputation Manifesto [5]. We will give an overview of other activities in this direction, such as, for example, the Mozilla Science Lab, GitHub and Figshare joint project to fix code citations by assigning a DOI to the particular revision, and training initiatives such as Software Carpentry to increase the awareness of researchers about tools that they can use to make their research more reproducible – in other words, promote "Recomputaliteracy" (a word coined by Ian Gent at one of the group discussions at the Collaborations Workshop 2014).

This section is in a preliminary state

3 The EMCSR summer school

The summer school comprised two activities: a set of speakers describing ongoing developments and challenges in reproducible research, and a set of projects chosen by participants to address some of these outstanding challenges.

Ian Gent from St Andrews and Lars Kotthoff from Cork gave presentations outlining the current state of the recomputation project, covering such issues as what it means to "recompute" a computational experiment, and the requirements for storing and reproducing an experiment using virtual machines.

Kenji Takeda from Microsoft Research delved further into the use of virtualisation, by describing the impact of cloud computing on computational research[2]. As more and more research projects involve the use of virtual machines running on cloud infrastructure, it might perhaps be possible to leverage this for creating reproducible experiments by sharing these virtual machines. After that, Alexander Konovalov from St Andrews gave a tutorial on creating virtual machines in Microsoft Azure (6-month passes to access the latter were provided freely to all participants thanks to Microsoft Research's generosity).

Two members of the Software Sustainability Institute[3], Neil Chue Hong and Steve Crouch, discussed the role of sustainable research software in reproducible

[2] e.g., the various projects at http://azure4research.com/

[3] http://www.software.ac.uk/

computational experiments. While Neil focused on the need for reproducibility, Steve discussed how to make research software sustainable, by using best practices for programming, and indeed thinking about what "good" code means.

Darren Kidney from the University of St Andrews introduced the use of standard statistical tools such as R and SWeave to create recomputable papers. Indeed this paper is written using SWeave.

In addition to sharing code and the systems used to run an experiment, much reproducible research requires the sharing of research data, which in itself involves many challenges. Tristan Henderson from St Andrews discussed his experiences in running the CRAWDAD wireless network data archive[4], and the difficulties in documenting data, convincing researchers to dedicate the time to share their data, citing data correctly and the privacy issues surrounding the sharing of sensitive data.

Sharing code, data or systems may also involve many legal issues: code may have particular licenses that make sharing hard or easy, data may be subject to data protection legislation, or indeed the law might suggest best practices that could make recomputable research easier to achieve. Burkhard Schafer from the Edinburgh School of Law gave a set of presentations covering these and many other aspects of the law.

Not all recomputable computational experiments solely involve computations. Miguel Nacenta from the St Andrews Human-Computer Interaction (HCI) research group discussed the issues around reproducibility in HCI experiments that involve human participants, and led a session in which summer school attendees attempted to reproduce the analysis from an HCI research paper.

4 Obstacles to Reproducibility in Computational Sciences

There are various obstacles to reproducibility, and many of these were discussed by speakers and participants during the week. Although not intended to be an exhaustive list, amongst the obstacles and challenges that we identified are: technical (long-term persistence of experiments; amount of efforts required from researchers to create reproducible experiments, which varies very much for different areas), legal and ethical (licensing issues; data protection issues, e.g. with researcher's personal data drifting into the virtual machine while it is in use), community building (why to make your research reproducible and how to to motivate your colleagues/collaborators/supervisors/etc to think the same).

Other problems that came out from the discussions include: performance measurement; possibly non-trivial amount of knowledge that is still needed to rerun an experiment; ease of reproduction vs independent reproducibility; automation tradeoff (recomputation tends to need automated systems); ability to get ethical consent (did they ask to release the data in a form we can use?); not knowing what data was decided not to publish in the paper.

[4] http://crawdad.org/

5 Open, Reproducible, Executable

This paper is intended to be Open, Reproducible, and Executable. We discuss here what we mean by these words, how we tried to achieve them, and to what extent we achieved this.

By "Open", we mean that the paper was openly developed and written. The authors collaborated via GitHub[5]. Although authors were usually in the same room, with so many of us other forms of collaboration would have been very difficult. More importantly, for the current discussion, this means that the development of the paper can be tracked throughout through the commit history on github. Anybody can download not only the final paper, but its source, and code and other materials collected during its development. An interesting aside is that this means that anybody can see what each author committed on the paper or supporting materials at any time, although it must be borne in mind that a commit by one person may represent the work of several authors working together offline.

By "Reproducible", we mean that it is our intention that other scientists (or ourselves at later dates) will be able to reproduce our work to assess if statements we make are correct and if conclusions are valid. To enable this we have attempted to collate materials necessary for each study, and make them available to future researchers. In most cases this has also been done in git, with materials such as ethics forms and experimental results put into the repository. However, in some cases we have constructed virtual machines (VM) to recompute experiments. Even a small VM might be half a gigabyte, and such large files can be problematic for git. This is one reason these are not included directly in the repository.

Finally, by "Executable", we mean a paper that can be reconstructed from source materials, and that data can be reanalysed as it changes and new versions of the paper produced, and possibly executable code rerun. This has numerous advantages because as we add data, the paper does not need to be rewritten. [6] To make our paper executable, we used Sweave, a package that integrates the statistics system R and LaTeX. To ease the workflow and to make execution of the paper easy, we wrote a Makefile for the generation of the paper pdf, although there are still some issues which need manual intervention such as installation of the necessary R packages.

To what extent have we succeeded in making this paper open, reproducible, and executable? At this stage, *remembering that this draft is a work in progress*, we would say our success is mixed. We cannot be completely open because some of the data and/or programs used in various parts of our paper do not allow us to share them. Also our paper is not fully executable in the sense that many computations involved in constructing the data must be run by hand. In terms of reproducibility, it is interesting to speculate: for example, if we run a second summer school in 2015, we might ask participants to reproduce this paper and see how we did this year.

[5] specifically `https://github.com/larskotthoff/recomputation-ss-paper/`

[6] The name "reproducible paper" is sometimes used for this, but can lead to confusion because a paper can be executable in the sense of being able to produce new figures with changed data, but not reproducible if that data can not be reconstructed ab initio.

6 Case study #1: Ethical requirements for recomputation

6.1 Introduction

The goal of recomputation[7] is to follow the recomputation manifesto[8] and make computational experiments recomputable by providing tools and a repository to store experiments in. The mission is stated as:

> "If we can compute your experiment now, anyone can recompute it 20 years from now"

One of the considerations perhaps under considered in regards to recomputation to date is the ethical implications.

6.2 Background

Due to the nature of an HCI experiment, in that it involves gathering data from human participants, it is necessary in order to successfully replicate an HCI experiment you must gather a new set of participants of which must meet the original experiments inclusion and exclusion criteria. In order for this to occur it is also necessary to replicate the ethical approval which the original experimenter sought, at first this may seem trivial however through examination of ethical approval submissions from ten institutions across the UK, EU, and US we found great differences between each and that a very low amount of overlap currently exists.

6.3 Comparison of Ethical Requirements across Universities

6.3.1 Gathering academic forms

To understand the state of ethics procedures between institutions, we collected the ethics applications forms for ten public universities located in the UK, EU, and USA. A wide range of locations were chosen to represent procedures across research communities, stature of the institution, and engagement with the HCI community. We aim to discover a set of ethical concerns common to all.

In order to compare each ethical approval form from each institution, two researchers independently transcribed each field and noted unique occurences, mapping minor variations in wording to each field. Where a field potentially encompasses additional information, unique requests for additional information were counted as additional fields. For example, requesting information about grants were considered one field, while providing explicit grant codes and whether grant funding was already agreed were considered two further fields. Our analysis uncovered 145 unique fields, encompassing generic details, such as contact details of co-investigators, methodological details, and institution-specific requirements, often for insurance and liability purposes. Agreement was sought on all terms between researchers before producing the final set. Of these fields, only two were common to all ten ethics forms - the name of the

[7] http://www.recomputation.org/

[8] http://www.recomputation.org/blog/2013/04/12/the-recomputation-manifesto/

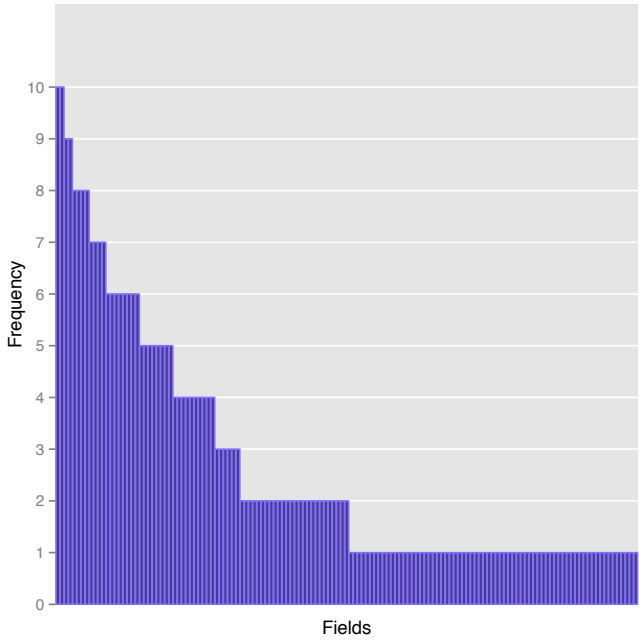

Figure 1: Histogram showing frequency distribution of fields in ethics applications

principal investigator, and whether informed consent was sought. While this is clearly an insufficient minimum specification for capturing ethics requirements, it reveals the crux of the ethics approval process is to ensure participant agency is preserved through the experimental process by mandating informed consent.

6.3.2 Comparison Analysis

6.4 Comparison of Ethical Requirements across Publishers

6.5 Framework for Ethics to accompany data and experiments

7 Case study #2: recomputing parallel and distributed experiments

We wish to know whether using a virtualised environment can affect the quality and reproducibility of parallel and distributed experiments. We looked at a multi-core parallel experiment, and a peer-to-peer system.

7.1 Experiment 1: A Parallel System

We looked at an existing implementation of a parallel algorithm for the maximum clique problem [7]: given a graph, a clique is a subset of pairwise-adjacent vertices, and the maximum clique problem (which is NP-hard) is to find the largest such subset. On real hardware, for any given graph instance, if we run the program multiple times we get very similar runtimes. The aim of this experiment is to see whether runtimes are similarly consistent when run on a virtual machine (VM) on a cloud. For real runtimes (RRT) we used 4 cores of a node with 16 Intel cores (dual Intel Xeon E5-2640 v2 2GHz). The node has 64 Gb RAM and has *Scientific Linux* 6 as the operating system. We have selected 10 different "medium sized" problem instances (i.e. graphs) from the second DIMACS implementation challenge to have a valid and reasonable comparative analysis. For each instance, we run the algorithm 50 times on real harware to collect the RRTs, and then 50 times on a VM with 4 cores on Microsoft Azure [1] to collect virtual runtimes (VRTs). We measure runtimes of the algorithm and ignore the input and output times. We compare the relative standard deviation (standard deviation as a proportion of the mean, since processing models varies between real and virtual hardware) of RRTs and VRTs of each problem instance.

We present the result in Table 1. Each row of the table is a problem instance. We show the relative standard deviation of RRTs and VRTs, expressed as a percentage.

	real	vm
brock400_1	0.54	0.28
brock400_2	0.71	0.20
brock400_3	0.78	0.14
brock400_4	0.95	0.33
MANN_a45	0.50	0.24
p_hat500.3	0.46	0.25
DSJC1000_5	0.42	0.23
p_hat1000.2	0.46	0.26
sanr400_0.7	0.52	0.27
p_hat700.3	0.43	0.21

Table 1: Relative Standard Deviations of runtimes (as a percentage)

In both real and virtual hardware the relative standard deviation is very small (below 1% in every case). We did not encounter any abnormalities when running on a virtual machine; this is contrary to the experiences of Kotthoff [6], who did not always see reliable *sequential* runtimes on virtualised hardware.

The virtual machine image used for the experiments is available via VMDepot[9].

7.2 Experiment 2: A Peer-to-Peer System

Experiments that are performed across multiple machines are challenging to reproduce. This is due to the cost of needed resources, and the complexities in-

[9] http://vmdepot.msopentech.com/Vhd/Show?vhdId=44545

volved in configuring the machines and the relationship between the machines. Cloud services offer increasingly affordable computation. Such services use virtualisation to decrease the cost of system reconfiguration.

In order to study the challenges of making distributed experiments reproducible using cloud services, we deployed a Chord [9] Distributed Hash Table (DHT) of 10 nodes on 10 dedicated machines, and tried to reproduce the same experiment on Microsoft Azure.

Reproducing the same experiment on Microsoft Azure proved to be time consuming. Due to the lack of time we were unable to reproduce the experiment. The main reason is the steep learning curve of using Microsoft Azure's API. The API itself lacks documentation and is complex in design. Since the Azure's API is not as popular as the rivals, it is harder to find example usage of the API.

It is possible to make experiments across multiple machines reproducible by writing a vendor-specific script that starts and configures any needed VMs before running the experiment. This approach raises a number of issues: 1) it relies on external services in order to run the experiment, 2) it is time-consuming to produce such an script, and 3) the script cannot be re-used on other cloud services.

Further studies are needed to identify best practices to make distributed experiments reproducible using virtual machines.

8 Case study #3: recomputing non-CS experiments

The focus of this section is on the issues relevant to research reproducibility that are potentially unique to non-CS computational research. The discussion is based on experience gained while packaging three computational experiments into self-contained virtual-machines. The papers on which the discussion is based (i.e., [8, 3, 2]) deal with urban planning, solar physics, and atmospheric physics respectively, although the discussion is likely relevant to other non-CS domains. Our aim was to offer readers (or reviewers) the opportunity to reproduce the figures presented in the paper, to inspect the code, and to possibly test behaviour of the programs with other parameters, all within a ready-to-use environment, namely a single virtual box constructed using the Vagrant tool and which runs the Debian operating system.

We encountered legal obstacles in all case studies. All of the software had been developed at universities, which typically resulted in the copyright being held by these institutions. Similarly, the decision on whether to allow open-source distribution of the code and the choice of licensing terms are to be taken by a university representative (e.g., a PhD advisor). While this is in no way unique to non-CS domains, it is likely that pre-existing IP procedures are less likely to cover software-dissemination aspects in institutions not dealing with computer science. Even if the legal status of the code developed for a given experiment is settled and matches reproducibility requirements, the environment needed to run it might prevent unconstrained recomputation. This in fact was also the case in one of the programs in question, as the code relied on proprietary software library.

Furthermore, we noted a lack of domain-specific workflows for recomputa-

tion in non-CS journals. A counterexample is the journal Geoscientific Model Development (GMD), which encourages reviewers to check the code.[10] The level of computer proficiency in non-CS domains[11] is also likely to influence the ability of researchers to use the Virtual Machines — it is arguably less likely that a physics or urban-planning journal reviewer will know right away how to deal with a VM, in contrast to CS-related journals.

We give a more detailed description of the experiments in the remainder of this section.

Experiment 1: Urban Planning

For the first case study, we tried to recompute parts of Table 6 of [8], in which the punctuality of a bus service in Edinburgh is evaluated using statistical methods. Table 6 contains confidence intervals that are constructed using a piece of software written in Java. It is possible to compile the source code after installing a Java Software Development Kit on the virtual machine. The software uses the SSJ library for Stochastic Simulation,[12] which had to be downloaded from the Internet. After that, the code was able to run. The resulting virtual machine is about 550MB - the Java SDK and its dependencies account for over 200MB by themselves, although parts of the development kit could be de-installed after compilation.

We encountered legal obstacles in the following three areas.

- Code-related: the software was developed as part of the EU project QUANTICOL, which is funded by the European Commission as part of its 7th Framework programme. In the General Conditions part of the project's grant agreement it is stated that the IP rights are awarded to the beneficiary, which in this case would be the University of Edinburgh. In turn, the University of Edinburgh has issued a position statement on intellectual property rights[13] in which it stated that it "*is the policy of the University of Edinburgh to develop University research capabilities and to assess, develop and promote the transfer of Edinburgh's technology and ideas for society's use and benefit.*" Still, publication of the code would need to be checked with a supervisor.

- Data-related: the code uses a dataset based on bus location measurements provided to us by Lothian Buses. We would need their permission to put this dataset in the public domain.

The use of the SSJ package is not an obstacle because it is released under the GPL licence from GNU.

The resulting virtual machine contains the Java source files, allowing researchers with knowledge of Java to analyse the correctness of the programme. However, the code is not documented and may be hard to read. Specifically, parameters are hard-coded and no interpretation is given of the resulting numbers and LaTeX code.

[10] DOI: 10.5194/gmd-6-1233-2013

[11] For example, see: "*Computational science: ...Error. Why scientific programming does not compute*". DOI:10.1038/467775a.

[12] http://simul.iro.umontreal.ca/ssj-2/indexe.html

[13] http://www.research-innovation.ed.ac.uk/Portals/0/Documents/University-of-Edinburgh-Position-Statement-on-Intellectual-Property_December2011.pdf

Experiment 2: Solar Physics

At the start of the recomputation exercise it was judged that the results presented in [3] would be the easiest (of the three publications outside of computer science) to reproduce. The underlying code is serial in nature and not tied to a specific platform or hardware.

The work discussed in [3] concerns the energy released from an idealised cylindrical magnetic field whenever it becomes unstable and subsequently relaxes to a simpler state. Initially, a field starts in a stable configuration and is then taken on a random walk through a known two dimensional region of stability, see Fig 3 of the aforementioned publication. When the field crosses the boundary of this region (i.e., the threshold for instability), an energy release is determined and the field is moved to simpler stable configuration.

When this process is repeated many times, the results can take the form of energy distributions. The top right plot of Figure 14 in [3] shows the energy releases for 10^5 relaxations involving 100 different stable field configurations. We decided to focus on reproducing only this figure from [3].

This plot was taken from the results produced by a C++ code called Taylor Relaxation of Loop Ensembles, or TRoLE for short, that automates the process described above.

The TRoLE code was written when the first author was a postgraduate at the University of Manchester. It was necessary therefore to check the IP policy for this institution.[14] Section 3.2 of this policy states that the "*...ownership of IP created by a Student, who is not an employee of the University, is with the Student.*", which we took to mean that the first author was free to place the code in a publicly available repository, which will be done at a later date.

The next issue concerned a technical matter: the code was written with the use of a proprietary numerical algorithms library (NAG),[15] the licence for which had expired. Fortunately, there is an open source alternative, the GNU Scientific Library (GSL).[16] The TRoLE code was updated such that all NAG calls were replaced with the GSL alternative.

After the code was recompiled and linked with GSL libraries, it was ready to run the simulation of 10^5 relaxation events. Alas, the code could only simulate a fraction of this total (146 events), and instead the code reported that it had reached a point on the instability threshold for which a stable (relaxed) configuration could not be calculated. The relaxed state is determined by ensuring that certain properties are conserved (i.e., the values match those calculated for the unstable state). This is essentially a root finding exercise, and at the time of writing the cause of this problem was not known. Updates to the TRoLE code subsequent to [3], that were used to provide the figures for a later publication (i.e., [4]), one that featured a more complex magnetic fields, may be behind the immediate failure in recomputation. Given time however (e.g., 2-3 days), this issue is likely to be resolved.

[14] http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487
[15] http://www.nag.co.uk/numeric/CL/CLdescription.asp
[16] http://www.gnu.org/software/gsl/

Experiment 3: Atmospheric Physics

We also attempted to use the virtual machine to run code that uses libcloudph++ [2], a library of algorithms for representing cloud microphysics using numerical models. The library allows one to simulate formation of clouds and precipitation, for instance within a fluid-dynamics simulation of an atmospheric flow. We succeeded in running several pieces of software that were used to create figures used in [2]. The software is already under a GPL and in a Github repository. The libraries needed to run the software are a C++ compiler, CMake, and the Boost and Thrust libraries.

Two legal obstacles were encountered. For one, the libraries optionally use the GPU using non-free-software (CUDA) to make it run faster. Additionally, the employed particle-based algorithm is inspired by a technique covered by several patents including a European one (EP1847939A2). As for potential technical obstacles, the only complication was getting the right software versions on the VM, otherwise all was feasible.

9 Discussion

After presenting the case studies, in this section we are going to summarise obstacles that we have met in which case studies. In the final version of the paper we are going to describe lessons learned, suggest recommendations that could be made from it, and outline possible further directions of addressing the issues that we've identified.

Acknowledgements

The authors of this paper include organisers and participants in the summer school. We are very grateful to all of the people who helped make the school a success.

First we thank the organisers: John McDermott, Angela Miguel, and Lakshitha de Silva for organisational and technical help.

We thank the various sponsors of the Summer School for financial and other forms of assistance. These were: SICSA (the Scottish Informatics and Computer Science Alliance); Microsoft Azure; the Software Sustainability Institute; the School of Computer Science at the University of St Andrews; and the EPSRC Impact Acceleration Award "Recomputation.org: Making Computational Experiments Recomputable" at the University of St Andrews.

We are also grateful to speakers at the Summer School who were not also authors of this paper: Neil Chue Hong, Stephen Crouch, Darren Kidney, Miguel Nacenta, Burkhard Schafer, and Kenji Takeda.

References

[1] Microsoft Azure. https://azure.microsoft.com/en-us/.

[2] S. Arabas, A. Jaruga, H. Pawlowska, and W. W. Grabowski. libcloudph++ 0.1: single-moment bulk, double-moment bulk, and particle-based warm-rain microphysics library in C++. *arXiv preprint arXiv:1310.1905*, 2013.

[3] M. Bareford, P. Browning, and R. Van der Linden. A nanoflare distribution generated by repeated relaxations triggered by kink instability. *Astronomy and astrophysics*, 521, 2010.

[4] M. Bareford, P. Browning, and R. Van der Linden. The flare-energy distributions generated by kink-unstable ensembles of zero-net-current coronal loops. *Solar Physics*, 273(1):93–115, 2011.

[5] I. P. Gent. The recomputation manifesto, 12 Apr. 2013. Online at http://arxiv.org/abs/1304.3674.

[6] L. Kotthoff. Reliability of computational experiments on virtualised hardware. *Journal of Experimental & Theoretical Artificial Intelligence*, 26(1):33–49, 2014. doi:10.1080/0952813X.2013.784812.

[7] C. McCreesh and P. Prosser. The shape of the search tree for the maximum clique problem, and the implications for parallel branch and bound. 2014. Online at http://arxiv.org/abs/1401.5921.

[8] D. Reijsbergen and S. Gilmore. Formal punctuality analysis of frequent bus services using headway data. In *Proceedings of the Eleventh European Workshop on Performance Engineering (EPEW 2014)*, 2014.

[9] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 149–160, San Diego, CA, USA, 2001. doi:10.1145/383059.383071.