Teaching Dossier

Marc Moreno Maza

December 15, 2007

Contents

1 Current Position and Contact information

1.1 Current position

Assistant Professor
(Associate Professor with tenure, effective July 1, 2008)
Departments of Applied Mathematics and Computer Science
University of Western Ontario (UWO)

1.2 Contact information

Office:	Room 383, Middlesex College
	University of Western Ontario
	London N6A 5B7, Ontario (Canada)
Web site:	`http://www.csd.uwo.ca/~moreno`
Telephone:	1 (519) 636 62 72

2 Teaching Responsibilities

2.1 Courses taught at the University of Western Ontario

The table on the next page contains data for the undergraduate and graduate courses I have taught at the University of Western Ontario during the last 5 academic years. For all these courses, I was the only instructor, except for CS210. Two sections were opened for CS210 during Fall 2003. Prof. Roberto Solis-Oba and myself together were in charge of them. Except for CS210, I am the author of each of the course web sites listed below.

2.1.1 Undergraduate courses (cross-listed courses included)

- *CS210, Data-structures and algorithms.*
- *CS211, Software Tools and Systems Programming.*
  `http://www.csd.uwo.ca/~moreno/cs211_moreno/index.html`
- *CS424/CS556, Foundations of Computer Algebra.*
  `http://www.csd.uwo.ca/~moreno/MainPages/CS424-2005.html/index.html`
- *CS447/CS545, Compiler Theory.*
  `http://www.csd.uwo.ca/~moreno/MainPages/CS447-2004.html/index.html`.

2.1.2 Graduate courses

- *CS855, Parallel Scientific Computing: Models, Algorithms and Implementation.*
  `http://www.csd.uwo.ca/~http://www.csd.uwo.ca/courses/CS855b/`
- *CS867, Algorithmic Properties of Polynomial Rings.*
  `http://www.csd.uwo.ca/~moreno/MainPages/CS867-2005.html/index.html`.

- *AM583, Advanced Computer Algebra: Modular Computations.*
  `http://www.csd.uwo.ca/~moreno/MainPages/AM583-2003.html/index.html`.
- *CS652, Algorithms and Software for Symbolic Solvers for Polynomial Systems.*
  `http://www.csd.uwo.ca/~moreno/MainPages/CS652-2004.html/index.html`.
- *CS874, Advanced Computer Algebra: Asymptotically Fast Methods for Exact Computations.*
  `http://www.csd.uwo.ca/~moreno/MainPages/CS874-2003.html/index.html`

Year 2002-2003				
Department & Course number	Comp. Sc. CS447b + CS545b	Comp. Sc. CS874b		
Total lecture hours/year	39	26		
Total student enrolment	11 + 7	5		
Student evaluation (overall effectiveness)	CS447b: 4.7 / 7 CS545b: 4.17 / 5	4.8 / 5		
Year 2003-2004				
Department & Course number	Comp. Sc. CS447b + CS545b	Comp. Sc. CS652b	Comp. Sc. CS210a	Appl. Math. AM583a
Total lecture hours/year	39	39	39	39
Total student enrolment	9 + 4	7	104	6
Student evaluation (overall effectiveness)	CS447b: 5.4 / 7 CS545b: 4.2 / 5	5 / 5	4.2 / 7	6.83 / 7
Year 2004-2005				
Department & Course number	Comp. Sc. CS447a + CS545a	Comp. Sc. CS424b + CS556b	Comp. Sc. CS867b	
Total lecture hours/year	39	39	39	
Total student enrolment	12 + 3	3 + 5	9	
Student evaluation (overall effectiveness)	CS447b: 5.3 / 7 CS545b: N/A	CS424: N/A CS556: 5 / 5	5 / 5	
Year 2005-2006				
Department & Course number	Comp. Sc. CS211a	Comp. Sc. CS424b + CS556b	Comp. Sc. CS855b	
Total lecture hours/year	39	39	39	
Total student enrolment	55	3 + 5	3	
Student evaluation (overall effectiveness)	4.9 / 7	CS424: N/A CS556: 4.33 / 5	4.67 / 5	
Year 2006-2007				
Department & Course number	Comp. Sc. CS211b	Comp. Sc. CS424b + CS556b	Comp. Sc. CS855b	
Total lecture hours/year	39	39	39	
Total student enrolment	13	2 + 5	5	
Student evaluation (overall effectiveness)	5.2 / 7	CS424: N/A CS556: 5 / 5	4.8 / 5	

2.2 Course outlines for all courses taught in the past 2 years

In each of the academic year 2005-2006 and 2006-2007, I taught CS211, CS424 / CS556 and CS855. The entire course outline of CS211 is included in Section 8.1. For the course outline of CS424 / CS556, Section 8.2 does not contain the parts related to Computing Facilities, Email Contact and Ethical Conduct since they are identical to those of CS211. For the graduate course CS855, the

main web page of the course serves as outline and is given in Section 8.3.

2.3 Summary of student rating evaluations

Please, refer to the table in Section 2.1 on page 3.

2.4 List of all students supervised (lifetime summary)

I have supervised the theses listed in Section 2.4.1, the post-doctoral projects listed in Section 2.4.2 and the internships and undergraduate projects listed in Section 2.4.3.

	Number successful completed	**Number in progress**
Doctoral Thesis	1	4
Master's Thesis	4	0
Post-Doctoral Fellows	2	1
Internships	4	0

One post-doctoral fellow, Oleg Golubitsky, and two PhD students, Yuzhen Xie and Xin Li, are co-supervised with Stephen M. Watt.

2.4.1 Completed graduate students

1. Yuzhen Xie. *Fast algorithms, modular methods, parallel approaches and software engineering for solving polynomial systems symbolically.* PhD thesis (co-supervised by S. M. Watt). UWO, started in January 2003, defended on September 4, 2007.

2. Raqeeb Rasheed. *Modular Methods for Solving Nonlinear Polynomial Systems.* Master's thesis UWO, started in September 2006, defended on August 23, 2007.

3. Akpodigha Filatei. *Implementation of Fast Polynomial Arithmetic in Aldor.* Master's thesis UWO, started in May 2005, defended on March 28, 2006.

4. Xin Li. *Efficient management of symbolic computations in compiled and interpreted environments.* Master's thesis UWO, started in May 2004, defended on August 15, 2005.

5. Jinlong Cai. *Unified functional closures extending the Aldor development environment and supporting its interactive debugger*, Master's thesis UWO, started in May 2003, defended in August 2004.

2.4.2 Completed post-doctoral fellows

1. Sylvain Neut during the Fall 2003 for the project *Application of Cartan's equivalence method to the Painlevé equations* at UWO.

2. François Lemaire in Summer 2003 for the project RegularChains library in Maple at UWO.

2.4.3 Completed internships

1. Undergraduate research assistant: Afsaneh Bakhtiari (Applied Mathematics Department). *Implementation of the Half-GCD Algorithms in Aldor.* UWO, May 2005 to May 2006.

2. Undergraduate research assistant: Steve Wilson (Mathematics Department). *Implementation of Multivariate Hensel lifting in Aldor.* UWO, Summer 2004.

3. Internship in *licence MIAGE-2* of Karim Zgoulli. *Automatisation du suivi de la production de l'entreprise OSYS.* Université de Lille 1, France, 2001.

4. Internship in *licence MIAGE-2* of Édith Deman. *Gestion informatique de la bibliothèque interne de l'entreprise UNIS.* Université de Lille 1, France, 2001.

5. Undergraduate project (CS 490 Y) of Wei Hu (co-supervised with Serge Mister). *Security Testing of Protocol Implementations.* UWO, September 2004 to March 2005.

2.4.4 Titles of theses in progress

1. Supervision of the PhD thesis of Raqeeb Rasheed. *Nearly-optimal symbolic solvers for polynomial systems with finitely many solutions.* UWO, started in September 2007.

2. Supervision of the PhD thesis of Wei Pan. *Towards practically efficient symbolic solvers for systems of differential equations.* UWO, started in September 2006.

3. Supervision of the PhD thesis of Changbo Chen *High-performance computer algebra support for dynamical systems.* UWO, started September 2006.

4. Supervision of the PhD thesis of Xin Li (co-supervised by S. M. Watt). *Implementation Techniques for Asymptotically Fast Polynomial Arithmetic in a High-level Programming Environment.* UWO, started in September 2005.

2.4.5 Supervision of post-doctoral fellows in progress

1. Supervision of the post-doctoral fellowship of Oleg Golubitsky, jointly with Stephen M. Watt. *Optimization of Aldor.* UWO, started September 2006.

2.5 Academic advising duties in the past 5 years

Between September 2002 and September 2007, at UWO, I served as an adviser or as an examiner for 17 theses listed in Section 2.5.1. In the Computer Science Department, I am serving as a Graduate Executive Committee Member since September 2004; details are reported in Section 2.5.2.

2.5.1 Theses for which I served as adviser or examiner

1. Examiner for the PhD thesis of Xiaofang Xie *On the Recognition of Handwritten Mathematical Symbols* UWO, December 14, 2007.

2. Examiner for the Master's thesis of Matthew Malefant *A Comparison of Two Families of Algorithms for Symbolic Polynomials* UWO, December 13, 2007.

3. Examiner for the PhD thesis of Azar Shakoori. *Polynomial Algebra by Values for Solving Bivariate Systems* UWO, December 11, 2007.

4. Examiner for the PhD thesis of Laurentiu Dragan. *On Measuring and Optimizing the Performance of Parametric Polymorphism.* UWO, September 4, 2007.

5. Examiner for the PhD thesis of Qing Zhao. *SC-Expressions in Object-Oriented Languages.* UWO, August 9, 2007.

6. Adviser (but not supervisor) for the PhD thesis of Wenyuan Wu. *Geometric Symbolic-numeric methods for differential and algebraic systems.* UWO, July 23, 2007.

7. Examiner for the PhD thesis of Wenqin Zhou. *Symbolic Computation Techniques for Solving Large Expression Problems from Mathematics and Engineering.* UWO, April 19, 2007.

8. Examiner for the Master's thesis of Heba Anbeer. *Complexity measures for biological strings.* UWO, December 8, 2006.

9. Examiner for the Master's thesis of Songxin Liang. *Component-free vector algebra in Aldor* UWO, April 17, 2006.

10. Examiner for the PhD thesis of Elena Losseva. *Optimal Methods of Encoding Information for DNA Computing.* UWO, December 8, 2005.

11. Examiner for the Master's thesis of Nargol Rezvani. *Approximate Polynomials in Different Bases.* UWO, December 2, 2005.

12. Examiner for the PhD thesis of Cosmin Oancea. *Parametric Polymorphism for Software Component Architectures.* UWO, November 1, 2005.

13. Examiner for the Master's thesis of Andrew Skryzhynskyy. *Methods for Improving the Relevance of Search Results from a Search Engine.* UWO, July 7, 2005.

14. Examiner for the PhD thesis of Juntao Ye. *Computational Aspects of the Dynamics of Cloth.* UWO, May 3, 2005.

15. Examiner for the Master's thesis of Ben Huang. *Network Performance Studies in High Performance Computing Environments.* UWO, March 3, 2005.

16. Examiner for the Master's thesis of Kevin Durdle. *Supporting Mathematical Handwriting Recognition through an Extended Digital Ink Framework.* UWO, December 13, 2004.

17. Examiner for the Master's thesis of Yong Lei. *Test Case Minimization and Fault Localization with Random Unit Testing and Log File Analysis.* UWO, September 8, 2004.

2.5.2 Service to the Graduate Executive Committee

Since September 2004, I am a member of the Graduate Executive Committee of the Computer Science Department. Moreover, for the academic year 2006-2007, I was the Chair of the committee for graduate scholarships, and for the academic year 2007-2008, I am chairing the committee for graduate awards.

3 Candidate's Statements

3.1 Teaching Philosophy

Two experiences in my life had a major impact on my teaching philosophy. First, when I was an undergraduate student, I was giving several hours per week of private classes in Mathematics for middle school and high school students. This guided reflexions on learning: what is the proper way to explain this concept? how to make this exercise attractive? Secondly, when I was a graduate student, I was the singer of a rock'n'roll band. This contributed to my reflexions on teaching: how to catch people's interest? how to gain people's participation?

My recent participation to the seminars of the Teaching Support Centre has revitalized these thoughts and I retrieved the feeling of getting on stage during my lectures of the past winter.

As a lecturer, I have to lead my class and give the students the desire to be led. At the same time, I need to keep them engaged, as a team. It is essential, indeed, to obtain frequent feedback from them in order to realize when the course concepts are not properly understood or assimilated.

Bringing students to successfully reach the objectives of the course is my motivation. As described in Section 4.3.1, for my compiler theory course, my mission was to lead students to realize their own program compiler. It was a great satisfaction to me that all students who invested enough effort succeeded in this project. In addition, for the Fall 2004 edition of this course, I was delighted that all enrolled students passed and that all students, but one, put the necessary effort in the course project.

Two students attending this compiler course became my graduate students. Leading students to discovery is, honestly, my greatest professional achievement and happiness. When a student has the desire to do research, I really enjoy helping her or him to succeed. Supervision, this other side of the teacher, has its own challenges. The topic has to be appropriate to the student while challenging enough in the research area. This latter point implies that the supervisor might not know the answer yet. However, the student has to trust the path initiated by the supervisor. That's the magic part, intuition, that cannot be taught, but hopefully acquired.

3.2 Professional Development

During the academic year 2006-2007, I participated in the program *Teaching at the University level* offered by the Teaching Support Centre (UWO). I completed the Modules 1, 2, 3, 4, 8, 12 and 13. I also completed the microteaching sessions (#1 - Nov. 15/06 and #2 and #3 - Feb. 28/07). I need to complete 3 more modules during the academic year 2007-2008 in order to obtain the certificate.

I would like to thank my colleagues of the Teaching Support Center for organizing this series of courses and seminars. This is the first university where I have been working, which invests such an effort in helping professors and lecturers in keeping improving their teaching. I think it is also very important that this teaching support activity is well advertised and recognized locally. Clearly, this is the case at Western. Attending these Wednesday morning meetings was also for me a good oxygen bubble in the middle of a busy week.

Teaching is probably a discipline that most of us have never learnt formally in the way that one learns Mathematics or English grammar. However, this is a subject that we, students and professors, are demonstrated every day. So, is our experience on teaching comparable to what of a Mathematics course consisting only of examples could be? Probably not. Mathematics involve more abstraction, although they aim at understanding the real world. On the other hand, all these

mathematical examples would, hopefully, be correct, whereas in the case of course experiences, we may have been demonstrated many not so correct examples without realizing it.

It was, therefore, very important to me to meet with experts in teaching that could give feedback on my way of lecturing. These micro-teaches were challenging in many aspects (time constraints, audience, camera recording) leading to careful design of the presentations together with a very original and rich experience.

The variety of topics covered during the meetings I attended did also impress me. We were lucky to have all these expert talks. Thinking of the "Interactive Teaching Strategies" meeting, several of the techniques demonstrated are quite natural and could be part of a game. While, putting them into practice, in class, requires to break some barriers or habits. So, being taught these techniques gives confidence in using them. Of course, some adjustments might be needed from discipline to discipline. For the 2nd year computer science course that I was teaching this Winter term (CS 211), I took advantage of the fact that there were enough laptops among the students during the lectures: from time to time, I asked the students to work out some little examples that they could grab from the course web site.

To summarize, participating to this series of courses and seminars was really enjoyable to me. It also had a clear impact on my thinking and acting regarding teaching at the University level.

4 Teaching Innovations

4.1 Development of new courses

I have designed all the 5 graduate courses CS874, AM583, CS652, CS867, CS855 listed in Section 2.1.2 on page 3 and one cross-listed (graduate and undergraduate) course CS424/CS556 listed in Section 2.1.1 on page 3. A short description of each of these 6 courses is given in the Sections 4.1.1 4.1.2, 4.1.3, 4.1.4, 4.1.5 and 4.1.6. The teaching materials that I have developed are presented in Section 4.1.7.

4.1.1 CS424 / CS556: Foundations of computational algebra

Symbolic computation manipulates numbers by using their mathematical definitions rather than using floating point approximations. Consequently, results are exact, complete and can be made canonical. However, they can be huge! Moreover, intermediate expressions may be much bigger than the input and output. One of the main successes of the Computer Algebra community in the last 30 years is the discovery of algorithms, called modular methods, that allow to keep the swell of the intermediate expressions under control. Without these methods, many applications of Computer Algebra would not be possible and the impact of Computer Algebra in the scientific community would be severely reduced. This course introduces the student to these algorithms and their implementation techniques.

4.1.2 CS 874: Advanced computer algebra: asymptotically fast methods for exact computations.

Asymptotically fast methods for exact computations have been known for a quarter of a century. Unfortunately their impact on computer algebra systems has been reduced since it was believed that they were irrelevant in practice. In this course we show that careful programing and accurate

experimentation can lead to a successful implementation of these methods permitting to overcome the size of magnitude of effective computations.

4.1.3 AM583: Advanced computer algebra: modular computations.

The motivation and areas are similar to those of CS424 / CS556. However, the topics covered are more advanced. In particular, symbolic Newton iteration is discussed in deep, including lifting techniques for solving multivariate polynomial systems. Comparing to CS874, this course focuses much more on the theoretical and algorithmic aspects whereas the main concerns of CS874 are complexity results and practical efficiency.

4.1.4 CS652: Algorithms and software for symbolic solvers for polynomial systems.

Systems of equations have been studied for centuries. However, the development of symbolic (or exact) methods for solving them is quite recent. No efficient software existed 10 years ago. Many theoretical and practical questions are still open and many problems in mathematical and engineering sciences are solved today by symbolic solvers. Digital signal processing, theoretical physics, cryptography, auto-correcting codes are some of the areas where exact solutions of systems of equations are used. In this course we will describe the key ideas of some of the most popular algorithms for solving systems of equations symbolically (Gröbner bases, triangular decompositions). These algorithms are quite elegant but their implementation leads to several difficult challenges. We will explain how emerging techniques (modular algorithms, distributed computations) address these difficulties. Finally, a detailed application will be presented.

4.1.5 CS867: Algorithmic properties of polynomial rings

It is a standard problem in mathematical sciences to study which properties of a given structure are preserved in derived structures. A trivial example of such preservation is the fact that every subgraph of a planar graph is itself planar. A more involved example is the fact that, if R is a unique factorization domain, then so is the ring $R[x]$ of univariate polynomials over R. Many algorithms rely on properties that are preserved under certain transformations. For instance, if R is again a unique factorization domain, then the so-called subresultant algorithm reduces polynomial gcd computations in $R[x]$ to number gcd computations in R. In this course, we discuss the algorithmic properties of R that can be lifted to $R[x]$, with an emphasis on those which help solving systems of equations. In particular, we discuss the lifting from R to $R[x]$ of prime and primary decomposition of ideals.

4.1.6 CS855: Parallel scientific computing: models, algorithms and implementation.

The motivation of this course is the parallelization of algorithms for solving systems of equations, both linear and non-linear. The topics covered are the computer science techniques needed for achieving this goal. The first third of this course will be devoted to the complexity analysis of parallel algorithms. In the rest, we will discuss parallel algorithms for scientific computations, with a view toward symbolic computations. Then, we will focus on parallel solvers and their applications.

4.1.7 Teaching materials developed

The course notes of AM583, CS874, and CS424 form a document of **244 pages.** All the chapters are available from the web site of CS424 in two formats: postscript for the students to read and HTML for the instructor to lecture. This document has been designed to serve these two purposes. Several other materials are available at the web site of CS424:

- worksheets and programs from the computer algebra system MAPLE and AXIOM,
- quizzes (offered with elements of correction) and assignments,
- articles (to be used for the course projects).

The course notes of CS867 and CS652 are part of a book which I am currently writing and which contains today **250 pages**; it is available to the students taking these courses. Although CS867 and CS652 are more advanced courses than AM583, CS874, and CS424, quizzes were used in order to help students having an active learning.

For CS855, the approach is quite different. First, the topics are less abstract (or technical) and do not require to have notes formally written as in a book. Therefore, CS855 has about 200 slides instead. Secondly, the motivation is to cover a large spectrum of fundamentals in parallel scientific computing. That is why quizzes were not used. In fact, students were asked to read a set of articles on a subject, say *parallel linear algebra*, and to provide a detailed report at the end of the course. Note that the web site of CS855 lists a large collection of articles and links in the area of parallel scientific computing.

4.2 Revision of existing courses and novel teaching methods

4.2.1 CS447 / CS545: Compiler theory

This course covers both the theory and the practice of compiler development. Few people are likely to build or even maintain a compiler, or a language translator. However, the ideas and techniques of compiler theory apply to many problems in software engineering and in other areas. Therefore, it is important that students taking this course have the opportunity to implement a compiler, as a course project. This is, actually, a challenging objective in a 13-week course.

For this compiler project to have some pedagogical value, the language to translate must be rich enough. However, for the compiler to be completed in this relatively short period of time, the language must be easy to translate. Technically speaking, the compiler should generate the output program in *one pass*. Therefore, I designed two **new programing languages** especially for the purpose of this course:

***MOOL*,** a Minimal Object Oriented Language (Winter 2003) and,

***Allcot*,** A Language for Learning COmpiler Theory (Winter 2004 and Fall 2004).

Both languages compile in one pass to a very low-level C language. The first one is an *Object Oriented* (OO) programing language whereas the second one is a subset of the ALDOR language, which is a *functional language* with an OO flavor. The specifications of *MOOL* and *Allcot*, together with the course notes (**215 pages with 62 figures**) and quizzes, are available from the course web site

`http://www.csd.uwo.ca/~moreno/MainPages/CS447-2004.html/index.html`.

See the comments of Mark Giesbrecht in Section 6.1.

For conducting these compiler projects, I have experimented two strategies:

- First, assign the *compiler project* in the second half of the course, after the students have acquired knowledge on some of the basic tools in compiler development, such as the `lex` and `yacc` software tools.
- Second, split the *compiler project* into 3 dependent assignments, given successively from the beginning to the end of the course.

The first strategy is probably easier to organize for the instructor but creates a big work load for the students in the last weeks of the course. The second strategy is more comfortable for the students working regularly. However, it becomes difficult to manage when a student performs poorly on Assignment n. Indeed, the outcome of Assignment n serves as a basis to realize Assignment $n+1$.

I experimented with the first strategy during Winter 2003 and with the second one during Winter 2004 and Fall 2004. The first strategy succeeded in the sense that all students who invested enough in the compiler project managed to produce a compiler. This required from me to revise and simplify the original goals of the project. However, 2 students out of 18 just ignored this compiler project. See the data in Section 4.3.1. The second strategy succeeded better, especially in the Fall of 2004, where all registered students, but one, produced a compiler. More details are reported in Section 4.3.1.

4.2.2 CS211: Software tools and systems programming

The slides of CS211 were developed by the previous instructors. They include a lot of small computer programs (UNIX shell scripts and C language programs) in order to illustrate the course concepts. I decided to post on the course web site all these computer programs independently of the slides, in a format that made them very easy to run for the students. This happened to be very useful:

- first for myself in order to demonstrate these programs in class,
- secondly for the students, since they could easily experiment with the course concepts.

Actually, both could take place in class since most students had a laptop! Another benefit is to ensure that the example programs on slides can effectively run with the latest version of such or such software tool that the students use.

In addition to these small computer programs, I posted on the course web site many solutions and hints to exercises, together with software documentation and articles on the course concepts. See the comments of Doug Vancise in the Section 6.1.

4.3 Evidence of impact or effectiveness of above innovations

4.3.1 CS447 / CS545: Compiler theory

As mentioned in Section 4.2.1, the challenge of this course, for the instructor, is, in a relatively short period of time, to lead the students to implement a program compiler. For the student,

the challenge is to realize a highly complex software, based on several concepts (lexical analysis, parsing, type checking, code generation, code optimization) freshly acquired during the course.

As mentioned in Section 4.2.1 also, I followed two different approaches. During the Winter 2004 and the Fall 2004, the compiler project was split into three successive and dependent projects, whereas during the Winter 2003, it was a large project preceded by two small and independent projects.

Term	**Initial population**	**Drops**	**Failed**	**Passed**	**Compilers attempted**	**Compilers completed**
Winter 2003	18	0	2	16	14	14
Winter 2004	14	2	0	12	10	10
Fall 2004	15	0	0	15	14	14

The above table shows that all students who attempted to produce a compiler managed to do so. This suggests that the *MOOL* and *Allcot* programing languages (See Section 4.2.1 on page 12) were successively designed. However, in Winter 2003, 2 students gave up at an early stage and failed; moreover, 2 other students passed the course without entering the compiler project, thanks to their high scores with all the other forms of evaluations (quizzes, exams, 2 other projects). In Winter 2004, 2 students dropped the course and 2 succeeded without attempting the third and final part of the compiler. Finally, in Fall 2004, all passed and only one succeeded without attempting the final part of the compiler. This suggests that the approach taken during the Winter 2004 and the Fall 2004 was more successful than that of Winter 2003. The success increased in Fall 2004 is probably due to the following facts. First, the specifications of the compiler (and the language to compile) were introduced progressively, allowing interaction with the students on the design of the language. Secondly, students were also allowed to choose among different possible enhancements of the language, leading to more creativity.

4.3.2 CS652: algorithms and software for symbolic solvers for polynomial systems

Seven students have taken this course for credit and two other were auditing. Student evaluations consisted of quizzes and research projects. Quizzes were aimed to help students studying the course materials on a regular basis. Research projects were based on the concepts presented in class and could be related on the thesis topic of the students. Four out of the seven student projects (by **Wenyuan Wu, Robin Scott, Yuzhen Xie, Wenqin Zhou**) have led to four published papers; the other three projects (by Hui Ding, Xin Jin, Songxin Liang) should also lead to publication in the near future. Projects that have led to published refereed articles:

[1] M. Moreno Maza, G. Reid, **R. Scott** and **W. Wu**. On approximate linearized triangular decompositions. In *Advances on Symbolic-Numeric Computation*, edited by D. M. Wang and L. Zhi, pages 268-287, Springer, 2007.

[2] M. Moreno Maza, G. Reid, **R. Scott** and **W. Wu**. On approximate triangular decompositions in dimension zero. *Journal of Symbolic Computation*, 42(7):693-716, 2007.

[3] X. Dahan, M. Moreno Maza, É. Schost, and **Y. Xie**. On the complexity of the D5 Principle. In proc. of *Transgressive Computing 2006*, J.-G. Dumas et al., editors, pages 149-168, Universidad de Granada, Spain, 2006.

[4] M. Moreno Maza, É. Schost, and **W. Zhou**. Primary decomposition of zero-dimensional ideals: Putting Monico's algorithm into practice. In proc. of *Transgressive Computing 2006*, J.-G. Dumas et al., editors, pages 419-428, Universidad de Granada, Spain, 2006.

5 Classroom Teaching

5.1 Summary of student rating evaluations

Please, see Figures 1, 2 and 3 on pages 37, 38 and 39 respectively.

5.2 Letters

Here's a list of former students who can be contacted for reference.

Name	**Email**	**Relation with me**
Jinlong Cai	erikcai@gmail.com	Former graduate student
Haitong Xu	hxu@csd.uwo.ca	CS545 Winter 2003
Kyle Charbonneau	kcharbo@uwo.ca	CS211 Winter 2007
Erin V. Ireland	evirelan@uwo.ca	CS447 Winter 2003
Jichao Zhao	jichao.z@gmail.com	AM583 Fall 2003
Songxin Liang	sliang22@uwo.ca	CS652 Winter 2004
Luam Pham	lpham2@uwo.ca	CS211 Fall 2005
Peng Jia	pjia2@uwo.ca	CS211 Fall 2005
Liu Yang	lyang47@uwo.ca	CS424 Winter 2005
Renee Lam	rlam8@uwo.ca	CS424 Winter 2005
Jennifer E. Hubbarde	jhubbar@uwo.c	CS556 Winter 2006
David Liu	liuyl.david@gmail.com	CS424 Winter 2007
Jeremy Lundy	jeremy.lundy@gmail.com	CS424 Winter 2007
Noman Y. Shihadeh	nshihade@uwo.ca	CS855 Winter 2007

5.3 Colleague evaluations based on direct classroom teaching

These colleagues work in the Computer Science and Applied Mathematics Departments at the University of Western Ontario, London, Canada.

5.3.1 Comments from Professor Rob Corless

See the letter page 30.

5.3.2 Comments from Professor Éric Schost

See the letter page 31.

5.4 Objective indicators of amount learned by students

5.4.1 CS447 / CS545: Compiler theory

For this course, the report of Section 4.3.1 illustrates the fact that most students learned how to realize a compiler, which is the main objective of the course.

5.4.2 CS211: Software tools and systems programming

For this course, the table below summarizes the final marks obtained by the students. The comparison between the two editions, Winter 2006 and Winter 2007 is probably not easy since the sizes of the population are quite different.

Term	Population	F	D	C	B	A	A+	Mean	Std Dev
Winter 2006	55	1	5	13	17	13	6	76	12
Winter 2007	12	2	0	2	3	1	4	66	31

5.5 Evidence of student success attributable in part to my teaching

I recruited 4 of my graduate students in the courses I taught: Yuzhen Xie (CS874), Jinlong Cai (CS545), Xin Li (CS545) and Akpodigha Filatei (CS424). The latter three defended their Master's theses in one year after starting with me. Xin Li is preparing today a PhD thesis under my supervision, whereas Jinlong Cai and Akpodigha Filatei are working in industry. Yuzhen Xie has defended her PhD thesis recently and she has been awarded an NSERC Post-Doctoral Fellowship. Xin Li was awarded an NSERC Post-Graduate Scholarship in 2006. See Section 7.2 for the awards of my graduate students.

6 Course Content and Course Management

6.1 Colleague evaluations based on course documents

These colleagues work in the Computer Science Department at the University of Western Ontario, London, Canada.

6.1.1 Comments from Professor Mike Bauer

See the letter on page 32.

6.1.2 Comments from Professor Mark Giesbrecht

See the letter on page 33

6.1.3 Comments from Professor Roberto Solis Oba

See the letter on page 35

6.1.4 Comments from Doug Vancise, Undergraduate Chair

See the letter on page 36

6.2 Formal student ratings of course (as opposed to instructor)

It is my understanding that this category refers to the question about the course as a learning experience. Please, see Figures 1, 2 and 3 on pages 37, 38 and 39 respectively.

7 Student Supervision

7.1 Evidence of student success attributable in part to my teaching

The table below summarizes the publications of my past and current graduate students. Please, refer to Sections 2.4.1 and 2.4.4 for the titles of their theses. All these publications are in peer-reviewed international conferences. The numbers below are those used for numbering the publications in my CV.

Student Name	**Publication numbers**
Yuzhen Xie	**[8]**, **[14]**,**[17]**, **[19]**, **24]**, **26]**, **[30]**, **[32]**, **[34]**, **[35]**, **[40]**, **[42]**, **[44]**
Jinlong Cai	**[38]**, **[39]**
Xin Li	**[15]**, **[18]**, **[20]**, **[33]**, **[36]**
Akpodigha Filatei	**[15]**
Changbo Chen	**[8]**, **[22]**, **24]**, **[32]**, **[44]**
Wei Pan	**[8]**, **[22]**, **24]**, **[32]**, **[44]**

The table below summarizes the presentations of my past and current graduate students at workshops and conferences. All these meetings are peer-reviewed international conferences, except

- UWORCS, the UWO Research Conference in Computer Science, organized yearly by the Computer Science Graduate School.
- ACA, which is an international workshop where all speakers are invited (without financial support).

All conference acronymes are defined in my CV.

Student Name	**Conference talk**
Jinlong Cai	CATLAN-04, EACA-04
Yuzhen Xie	ICPSS, ISSAC-05, TC-2006, UWORCS-06, PASCO-07 (2)
Xin Li	ACA-05 (2), ISSAC-06, CASA-07, UWORCS-06, UWORCS-07, PASCO-07, ISSAC-07
Akpodigha Filatei	UWORCS-06
Changbo Chen	UWORCS-07
Wei Pan	CASA-07, UWORCS-07

7.2 Student awards and scholarships

The following is a list of the competitive awards and scholarships received by my students under my supervision.

Year	**Student**	**Scholarship or Award**
2004	Yuzhen Xie	Ontario Graduate Scholarship (OGS)
2005	Yuzhen Xie	Ontario Graduate Scholarship (OGS)
2005	Yuzhen Xie	Distinguished Student Author Award (ISSAC 2005)
2005	Yuzhen Xie	Best Poster Award (ISSAC 2005)
2006	Xin Li	Ontario Graduate Scholarship in Science and Technology (OGSST)
2006	Yuzhen Xie	CS Publications Incentive Award (Computer Science Department, UWO)
2006	Yuzhen Xie	Ontario Graduate Scholarship (OGS)
2006	Yuzhen Xie	UWO Thesis Award
2007	Xin Li	Best presentation at the UWO Research Conference in Computer Science
2007	Xin Li	NSERC Post-Graduate Doctoral Scholarship (for 2 years)
2007	Yuzhen Xie	NSERC Post-Doctoral Fellowship (for 2 years)

8 Annexes

8.1 Course outline of CS 211 for Winter 2007

The University of Western Ontario
London, Canada

Department of Computer Science

Computer Science 211b
Software Tools and Systems Programming
Course Outline - January 2007

Course Description
This course provides an introduction to software tools and systems level programming. Topics include: understanding how programs run (compilation, linking, and loading), an introduction to a complex operating system (UNIX), scripting languages, and the C programming language. As time permits, other topics will be chosen from: system calls, memory management, libraries, multi-component program organization and builds, version control, debuggers and profilers.

Prerequisite: Computer Science 027a/b or 037a/b with a grade of at least 60%

Antirequisites: Software Engineering 250a/b and the former Software Engineering 201a/b

Lecture Hours: Tuesdays 9:30-11:30am and Thursdays 10:30-11:30am in Natural Science Building 7

Instructor: Marc MORENO MAZA
Office: Middlesex College 383
Office Hours: Tuesdays 1:30pm-3:30pm; Wednesdays 1:30-2:30pm
Email: moreno <at> csd.uwo.ca
Phone: 661-2111 x 86891

Required Texts

- Sobell, *UNIX System V: A Practical Guide*. Benjamin-Cummings: 3rd edition.
- King, *C Programming: A Modern Approach*. Norton.

Course Topics
The course will address as many of the following topics as time will allow:

- **UNIX Fundamentals:** UNIX vs. Windows; logging on; files and directories; pathnames, and directory and file structure; editors; shells; I/O redirection; UNIX concurrency (processes); utilities; file permissions and security; regular expressions; shell programming.
- **C programming:** compiling, linking and loading; data types and operators; control structures; formatted I/O; file I/O; connections between I/O and the underlying operating system; function calls; structs; enumerations; arrays; pointers (pointer arithmetics, pointers and arrays, arrays of pointers, pointers to functions); memory management; linked lists and other dynamically allocated data structures; strings; calling C from UNIX; general libraries; standard libraries and headers; the C preprocessor; C program organization.

- **UNIX Tools:** building and managing multi-component programs; the make utility; version control and configuration management; debuggers; code performance and profiling.

Lecture Notes

Course lecture notes will be made available in PowerPoint and PDF on the course website. They are provided as a courtesy by the course instructor. Possessing (and even reading) these notes is not a suitable substitute for attending lectures.

Course Website

The CS211b website is at http://www.csd.uwo.ca/courses/CS211b. Lecture notes, assignments, and class information will be posted on this website. You are responsible for reading this information on a frequent and regular basis.

TA Consulting Schedule: to be arranged

Computing Facilities

Each student will be given an account on the Computer Science Department senior undergraduate computing facility, GAUL. In accepting the GAUL account, a student agrees to abide by the department's *Rules of Ethical Conduct*.

Note: After-hours access to certain Computer Science lab rooms is by student card. If a student card is lost, a replacement card will no longer open these lab rooms, and the student must bring the new card to a member of the Systems Group in Middlesex College Room 346, or to the I/O counter (MC 352).

Email Contact

We will occasionally need to send email messages to the whole class, or to students individually. Email will be sent to your UWO email address. You must make sure that you read your email on a frequent and regular basis, or have it forwarded to an alternative email address if you prefer to read it there.

Note that UWO and most other email providers establish quotas or limits on the amount of space available to you. If you let your email accumulate, your mailbox may fill up and you may lose important email from your instructors. Losing email is not an acceptable excuse for not knowing about the information that was sent.

Student Evaluation

Grades will be based on six assignments worth a total of 40%, a midterm exam worth 20%, and a final exam worth 40%.

If for any reason the assignment schedule given below cannot be adhered to, the assignment marks will be prorated. (The assignments are worth 40% of the overall mark for the course. If an assignment has to be canceled for any reason, the remaining assignment weights will be prorated (scaled) to add up to 40%.)

To be eligible to receive a passing grade in the course, your mark on the final exam must be at least 40%, and your weighted average on the assignments must be at least 40%. To be eligible to receive a grade of C or higher, your mark on the final exam must be at least 50%, and your weighted average on the assignments must be at least 50%.

Assignment and Test Feedback

Every effort will be made to have assignments marked and handed back within 2 weeks of the hand-in date. Midterm exam marks will be posted within 2 weeks of the exam. If we are unable to comply with our intended return dates, revised dates will be posted on the course website.

Test and Exams

Midterm: 1 hr 40 mins, Tuesday March 6th, during class time (location to be announced)
Final: 3 hours during the April exam period; exact time to be announced

There will be no makeup midterm exam. Students who do not write the midterm will have the weight shifted to the final exam, which will then be worth 60 %. Students who do better on the final than the midterm will also have the midterm weight shifted to the final.

Assignments

Due Dates (revised)

Asn 1 - 1% (light) - due between Jan 9 and 19
Asn 2 - 6% (medium) - assigned Jan 19, due Feb 6
Asn 3 - 6%(medium) - assigned Feb 6, due Feb 22
Asn 4 - 7% (medium) - assigned Feb 27, due Mar 15
Asn 5 - 20% (heavy) - assigned Mar 20, due Apr 12
About the Assignments

- Assignment descriptions will be posted on the course website by the dates listed above.
- Any changes, updates, and clarifications to assignments will also be posted on the website. It is your responsibility to monitor these pages closely.
- Assignments may involve programming in C, the use of UNIX operating system utilities, programming using shell scripts, and concept questions (non-programming) related to the lecture material.
- To be eligible for full marks, individual C assignments must run under UNIX on the departmental computing equipment. You may develop assignments on your home computer, but you must allow for the amount of time it will take to get the final product working on Computer Science's machines.

Submission of Assignments

- Your assignment solutions are expected to be your own individual work, not the products of group effort. On occasion, you may be allowed to make use of code from an outside source, such as your textbook. Check with your instructor if you are uncertain about the places in which you can use code written by another person.

- With each assignment, you are required to hand in an *Assignment Submission Form*, on which you certify that the material you've handed in is exclusively your own work. This form must be submitted to the assignment locker in a 9-by-12-inch envelope, bearing your name, which will be used to return your assignment. The precise location of the assignment lockers will be announced on the course website.

- Programming assignments will be submitted electronically. Details will be given on the course website and/or in the assignment descriptions. We reserve the right to use similarity detection software to detect possible cheating cases.

- Some non-programming portions of assignments may have to be handed in on paper. These items must be included in your submission to the assignment locker.

- Assignments are due by 11:59 pm on the due date. In the case of programming assignments, the time stamp on the electronic submission will be used to determine any late penalty. For assignments requiring paper submissions, the times at which they are removed from the assignment locker will be used.

Late Assignment Policy

- Late assignments will be accepted for up to four days after the due date, with weekends (Saturday and Sunday) counting as a single day; the late penalty is 5% of the available marks per day. Lateness is based on the times the assignment is removed from the locker and/or received electronically, not on the time it was printed or last modified.
- **Late Coupons**
  - Each student is given 5 *late coupons*. These are *virtual* (as opposed to physical) coupons maintained by the instructor. Late coupons may be applied toward any of assignments 2 through 6.
  - Each coupon can be thought of as a potential one-day assignment extension. It's entirely up to you to decide when to use your coupons; more than one coupon may be used per assignment. If you wish to use any coupons with an assignment, simply note that fact on your *Assignment Submission Form*.
  - The intent of late coupons is to give you some free days in case of minor illness, work overload, etc. No other extensions will be given in such cases.
  - Coupons are not transferable to another student.
  - Once a coupon has been used, you cannot ``take it back'' to use for another assignment instead.
  - Unused coupons are not redeemable for extra marks.
  - Using a late coupon does not change the final date on which an assignment will be accepted. Whether or not coupons are used, the assignment must be handed in within 4 days of the original due date.

- Extensions will be granted only on serious medical or compassionate grounds. You should take supporting documentation to the office of the Dean of your faculty, who will contact the instructor.

Assignment Marking

- Assignments are marked by the Teaching Assistant(s), who follow marking schemes provided by instructors.
- When assignment marking has been completed, you will be informed via the course website and/or email.
- You will be able to pick up your graded assignments in the first class following the announcement of their availability. Thereafter, they will be placed at the I/O counter on the third floor of Middlesex College. You must present your student card when retrieving assignments from the I/O counter.
- A request for adjustment in an assignment mark must be made within 2 weeks of the date on which it was first available for pickup. (Beyond that date, regrading will not be considered, regardless of whether you retrieved your assignment.) Such a request must be submitted in writing, and must include specific reasons why you believe you deserve more marks. The request must be accompanied by all materials that were originally handed in, as well as the original marker's

grade summary sheet. Regrading requests must be handed in to the lecturer.
- Assignment marks may be mailed out periodically throughout the term. It is your responsibility to check that your marks have been recorded correctly.

Assignment Backups
It is your responsibility to keep up-to-date backups of assignment disk files in case of system crashes or inadvertently erased files. Retain disk copies of all material handed in, as well as the actual graded assignment, to guard against the possibility of lost assignments or errors in recording marks. It is not safe to discard these materials until you are satisfied that your final mark for the course has been computed properly.

Tutoring

The role of tutoring is to help students understand course material. Tutors should not write assignments or tests for the students who hire them. Submitting an assignment that contains material written by a tutor is an academic offense. Having employed the same tutor as another student is not a legitimate defense against an accusation of collusion, should two students hand in assignments judged similar beyond the possibility of coincidence.

Ethical Conduct

All assignments are individual assignments. You may discuss approaches to problems among yourselves; however, the actual details of the work (assignment coding, answers to concept questions, etc.) must be your individual effort. Assignments that are judged to be the result of academic dishonesty will, for the student's first offense, be given a mark of zero with an additional penalty equal to the weight of the assignment also being applied. You are responsible for reading and respecting the Computer Science Department's policy on Scholastic Offenses and Rules of Ethical Conduct.

8.2 Course outline of CS 424 for Winter 2006

The University of Western Ontario
London, Canada

Department of Computer Science

CS 424b -- Foundations of Computational Algebra

Course Outline -- Winter 2006

Course Description

Symbolic computations manipulate numbers by using their mathematical definitions rather than using floating point approximations. Consequently, their results are exact, complete and can be made canonical. However, they can be huge! Moreover, intermediate expressions may be much bigger than the input and output. One of the main successes of the Computer Algebra community in the last 30 years is the discovery of algorithms, called modular methods, that allow to keep the swell of the intermediate expressions under control. Even better: these methods fit almost each of the intermediate values in a machine word. Without these methods, many applications of Computer Algebra would not be possible and the impact of Computer Algebra in the scientific community would be severely reduced. Today, modular computations are well-developed, especially for univariate and bivariate polynomial arithmetic and for linear algebra. They form the foundation for all modern algorithms in Computer Algebra. This will be the main topic of this course. In particular, we will discuss

- Fast multiplication algorithms (FFT, Karatsuba, Strassen)
- Chinese remaindering algorithm
- Newton's iteration and Hensel lifting
- Fast Linear Algebra and the LLL algorithm
- Polynomial gcds and resultants
- Factorization of Univariate Polynomials

Prerequisites

- Math223 (Discrete Structures II)
- OR permission from the instructor

Familiarity with a variety of programming languages is a bonus, although not required.

Instructor

Name: Marc Moreno Maza
Office: MC383
Office Hours: Thursday 3:00-5:00 pm
Email: morenoATcsd.uwo.ca
Phone: 661-2111 x6891

Lecture Notes and Textbook

Notes of each lecture will be available on the course website, approximatively one week after the oral presentation. The format is Postscript and HTML.

There is only one (highly) recommended text for the course. The other references are given for the curious reader.

- **Modern Computer Algebra** J. von zur Gathen and J. Gerhard *(RECOMMENDED)*. Cambridge University Press, 1999.
- **Algorithms for Computer Algebra**. K.O. Geddes and S.R. Czapor and G. Labahn *(SUPPLEMENTARY)*. Kluver Acadenic Publishers, 1992.

Course Website

The course web site is accessible from: http://www.csd.uwo.ca/~moreno/

Please check the site often for updates on lecture notes and errata. Also be aware that the course website is not a substitute for actual classroom attendance!

Lecture Topics

The list of topics will be something on the order of:

1. Introduction to computer algebra and computer algebra systems
2. Fast multiplication algorithms (FFT, Karatsuba, Strassen)
3. Euclidean methods (Chinese remaindering algorithm, rational reconstruction, ...)
4. Newton's iteration and Hensel lifting
5. Fast linear algebra and the LLL algorithm
6. Polynomial gcds and resultants
7. Factorization of Univariate Polynomials

Class Schedule

Lectures: 3 hours (Monday from 9:30 to 11:30am and Wednesday from 9:30 to 10:30am in MC320). In general, a lecture consists of a talk by the instructor followed by an exercise session.

Each student is expected to attend the lectures. In particular, quizzes (short written tests) may take place without notice.

Student Evaluation

- The course is very oriented toward assignments and projects.
- Assignments and projects constitute 30% and 40% of the course mark, respectively.
- Assignments consist of theoretical exercises. Projects can be theoretical work (research article presentations, research work), practical (programming, experimentation and analysis) or combination of both.
- There is no midterm examination and no final examination.
- However, there will be at least four quizzes given, but only your best three will count toward your mark.
- Quizzes constitute 30% of the course mark.

- A quiz is a series of short and simple exercises similar to the examples of the course notes, whereas assignments consist of more advanced exercises.
- In order to successfully complete the course, a student must achieve at least 50% on assignments, 50% on projects and 50% on quizzes. This is to prevent a serious lack of effort in either area. Thus, a student cannot get 100% on assignments and projects and barely pass the quizzes.

Assignment/Project/Quiz Schedule

All dates are **tentative** and currently subject to change, although it is doubtful by any significant amount.

Evaluation Technique	**Weight**	**Posted Date** (*tentative!*)	**Due Date** (*tentative!*)	**Workload**
Assignment One	10%	Th, Jan. 19	Th, Feb. 2	light
Assignment Two	10%	Th, Feb. 2	Th, Feb. 16	light
Assignment Three	20%	Th, Feb. 16	Th, Mar. 9	regular
Project Two	30%	Th, Mar. 9	Mo, Apr. 10	regular
Quizzes	10% each	N/A	various	N/A

If for any reason the schedule given above cannot be adhered to, the assignment, project and quiz marks will be pro-rated. For instance, if an assignment has to be canceled for any reason, the remaining assignment weight will be prorated to add up to 30%.)

Every effort will be made to have assignments, projects and quizzes marked and handed back within 3 weeks of the hand-in date, preferably sooner.

Quizzes

Quizzes may be held without being announced in advance.

Quizzes will be closed book.

Assignments

Assignments will be due on the (tentative) dates listed above. The assignment can be sent by email to the instructor or given to the instructor during the class or office hours.

Extensions will be granted only by the course instructor. If you have serious medical or compassionate grounds for an extension, you should take supporting documentation to the office of the Dean of your faculty, who will contact the instructor.

Projects

Projects will be presented by the students during the class on the

8.3 Course outline/website of CS855 for Winter 2007

Next Up Previous
Next: About this document ...

CS 855 - Parallel Scientific Computing: Models, Algorithms and Implementation

University of Western Ontario
Computer Science Department

Date: September 3, 2007

The motivation of this course is the parallelization of algorithms for solving systems of equations, both linear and non-linear. The topics covered are the computer science techniques needed for achieving this goal.

The first third of this course will be devoted to the complexity analysis of parallel algorithms. In the second third, we will discuss parallel algorithms for scientific computations, with a view toward symbolic computations. Then, we will focuss on parallel solvers and their applications.

Prerequisites.
A good familiarity with linear algebra and complexity analysis of algorithms is necessary.
Lectures (tentative schedule).

Week Jan. 8-14	Performance and scalability of parallel algorithms
Week Jan. 15-21	PRAM, BSP models
Week Jan. 22-28	Parallel Complexity Theory
Week Jan. 29-4	No lecture.
Week Feb. 5-11	Parallel Complexity Theory
Week Feb. 12-18	Parallel Sorting
Week Feb. 19-25	Parallel Sorting
Week Feb. 26-4	No lecture.
Week Mar. 5-11	Parallel Matrix Operations
Week Mar. 12-18	Solving systems of linear equations in parallel
Week Mar. 19-25	Task scheduling and load balancing
Week Mar. 26-1	Parallel algorithms for GCD computations
Week Apr. 2-8	Programming environments for parallel solvers
Week Apr. 9-15	Challenges for parallel solvers
Week Apr. 16-22	Project presentations

Student Evaluation.

No quizzes, exams or assignments. Each student chooses a subject based on the articles posted below in the area of Parallel Symbolic Computing. More papers will be posted soon. Then the student writes a report and presents it during one of the two last weeks of classes.

Software and Courses.

These are links to some books and software related to the course.

- ***MPICH-A Portable Implementation of MPI***
- ***Development Tools for MPICH***
- ***CS 402/CS 535: Parallel and Distributed Computing at UWO***
- ***CS 424b - CS 556b Foundations of Computational Algebra at UWO***
- ***Parallel Algorithms (WISM 459, 2005/2006) by Rob Bisseling***
- ***Topics in Parallel Computing (CS 838, Univ. of Wisconsin-Madison, Spring 1999) by Pavel Tvrdik***
- ***Parallel Algorithms (CS 662, San Diego Univ., Spring 1996)by Roger Whitney***
- ***U.C. Berkeley CS267/EngC233 Home Page by James Demmel***
- ***MIT 6.895 Fall 2003 by Charles E. Leiserson***

Conferences.

- ***PARALLEL PROCESSING AND APPLIED MATHEMATICS***
- ***SIAM Conference on Parallel Processing for Scientific Computing***
- ***Some Compiler and Parallel Computing Conferences***
- ***IEEE International Parallel & Distributed Processing Symposium***
- ***ACM Symposium on Parallelism in Algorithms and Architectures***
- ***Some Theoretical Computer Science Conferences***

Papers on Parallel Scientific Computing.

- ***Estimating Execution Time of Distributed Applications by Maciej Drozdowski***
- ***pARMS: A Package for Solving General Sparse Linear Systems on Parallel Computers by Y. Saad and M. Sosonkina***
- ***Fast Scheduling and Partitioning Algorithm in the Multi-processor System with Redundant Communication Resources by Eryk Laskowski***
- ***Evaluation of Parallel Programs by Measurement of Its Granularity by Jan Kwiatkowski***
- ***Parallel Triangular Decompositions of an Oil Refining Simulation by Xiaodong Zhang***
- ***Parallel Algorithms for Arrangements by Richard Anderson, Paul Beame and Erik Brisson***
- ***Efficient parallel computation of arrangements of hyperplanes in d dimensions by Torben Hagerup, Hermann Jung and Emo Welzl***
- ***Recursive Array Layouts and Fast Parallel Matrix Multiplication by Siddhartha Chatterjee, Alvin R. Lebeck, Praveen K. Patnala and Mithuna Thottethodi***
- ***A Quantitative Comparison of Parallel Computation Models by Ben H.H. Juurlink and Harry A.G. Wijshoff***
- ***Towards Efficiemlcy and Portability: Programming with the BSP Model by Mark Gouclreau, Kevin Lang Satish Rao, Torsten Suel and Thanasis Tsantilas***

Papers on Parallel Symbolic Computing.

- ***A BSP Parallel Model for the Göttfert Algorithm over F2 by Fatima Abu Salem***
- ***Fast Rectangular Matrix Multiplications and Improving Parallel Matrix Computations by Xiaohan Huang and Victor Y. Pan***
- ***NC2 mathend000# Computation of Gcd-free Basis and Application to Parallel Algebraic Numbers Computation by T. Gautier and J.L. Roth***
- ***Processor-Efficient Parallel Matrix Inversion over .Abstract Fields: Two Extensions by W. Eberly***
- ***Complexity of the Wu-Ritt decomposition by Á. Szántó***
- ***Distributed Partial Evaluation by Michael Sperber, Herbert Klaeren and Peter Thiemann***

- ***Processor Efficient Parallel Solution of Linear Systems over an Abstract Field by E. Kaltofen and V. Pan***
- ***On the Parallel Cbmplexity of Matrix Factorization Algorithms byMauro Leoncini, Giovanni Manzi and Luciano Margara***
- ***Multidimensional, Multiprocessor, Out-of-Core FFTs with Distributed Memory and Parallel Disks by Lauren M. Baptist and Thomas H. Cormen***
- ***Multithreaded Algorithms for the Fast Fourier Transform by Parimala Thulasiraman, Kevin B. Theobald, Ashfaq A. Khokhar and Guang R. Gao***
- ***How much can we speedup Gaussian Elimination with Pivoting? by M. Leoncini***
- ***Communication Efficient Matrix Multiplication on Hypercubes by Himanshu Gupta and P. Sadayappan***
- ***Distributed Data Structures and Algorithms for Gröbner Basis Computation by SOUMEN CHAKRABARTI and KATHERINE YELICK***
- ***On the Correctness of a Distributed Memory Gröbner Basis Algorithm by SOUMEN CHAKRABARTI and KATHERINE YELICK***
- ***Extended Parallelism in the Gröbner Basis Algorithm by Stephen A. Schwab***
- ***A Fine-Grained Parallel Completion Procedure by Reinhard Bündgen, Manfred Göbel and Wolfgang Küchlin***
- ***Parallelization of The Sparse Modular GCD Algorithm for Multivariate Polynomials on Shared Memory Multiprocessors by Mohamed Omar Rayes, Paul S. Wang and Kenneth Weber***
- ***Parallel Computation of Modular Multivariate Polynomial Resultants on a Shared Memory Machine by H. Hong and H. W. Loidl***
- ***A Parallel Completion Procedure for Term Rewriting Systems by Katherine Yelick and Stephen Garland***
- ***Programming Models for Irregular Applications by Katherine Yelick***
- ***Data Structures for Irregular Applications by Katherine Yelick***
- ***Strategy-Accurate Parallel Buchberger Algorithms by G. Attardi and C. Traverso***
- ***Work Efficient Parallel Solution of Toeplitz Systems and Polynomial GCD by John H. Reif***
- ***Processor-Efficient Parallel Matrix Inversion over .Abstract Fields: Two Extensions by W. Eberly***
- ***A New Approach to Parallel Computation of Polynomial GCD and to Related Parallel Computations over Fields and Integer Rings by Victor Y. Pan***
- ***Processor-Efficient Parallel Computation of Polynomial Greatest Common Divisors by Erich Kaltofen***

- ***About this document ...***

Next Up Previous

Next: ***About this document ...***

Marc Moreno Maza
2007-09-03

8.4 Letter from Professor Rob Corless

August 30, 2007

To the Promotion and Tenure Committee (Computer Science),

Re: Marc Moreno Maza

I have known Marc now since 2002, especially as a highly valued colleague in ORCCA. Marc is an extremely hard-working person, always willing to do more than his share of the work. His work ethic is, indeed, a bit intimidating – it often seems that all he does is work! It is my hope that the Committee will understand that Marc's singular research accomplishments – which include an amazing number of top-ranked research papers, especially at ISSAC, the premiere computer algebra conference, and also include an incredible 250,000 line software library of implementations of the most advanced algorithms for algebraic problems – are of the best quality. In the field of Regular Chains, for example, Marc is a world leader, and his presence here helped to attract our Canada Research Chair, Eric Schost.

But it is his teaching that I wish to comment on, most. Marc delivers more attention to his grad students than anyone else I have ever known. He is always with them, in the lab, inspiring and leading them to excellence. It is no accident that his student Yuzhen Xie won an ISSAC Best Student Author award – Marc gets more out of his students than they ever could have imagined. He is the kind of supervisor all the students want: he really teaches them something, and gets them to give their best.

This characteristic shows up in his course teaching, as well. I have had significant opportunity to observe his teaching when he coordinated a graduate course in Computer Algebra, where I attended some of his lectures. He *always* has notes prepared well in advance: orderly, clear, detailed and to the point. His delivery of his lectures is also impressive. He is personable and takes the time to engage the class, using projects that rely on shared knowledge to focus the students' attention. It is a very effective technique. He responds well to questions, and is perfectly happy to admit when he doesn't know something (which is rare – he is a truly multi-disciplinary person).

In short, Marc is an excellent professor and I recommend wholeheartedly that he be promoted and granted tenure immediately.

Sincerely,

Robert M. Corless
Distinguished University Professor

8.5 Letter from Professor Éric Schost

Éric Schost
Department of Computer Science
Middlesex college
The University of Western Ontario
London, Ontario, Canada, N6A 5B7

1 519 661 2111 ext. 86994
eschost@uwo.ca
http://www.csd.uwo.ca/~eschost/

August 31, 2007

Prof. Marc Moreno Maza.

Dear Marc,

In this letter, I wish to comment on your course material for Computer Science 424 / 556, Foundations of Computational Algebra.

This is a very rich course. It contains a wealth of material for students. Those who wish to discover computer algebra will get in your course an excellent overview of the problems computer algebra can solve; those who will pursue graduate work in this domain will use the content of the course as a long-standing reference.

I have attended your lectures on several occasions; they were always clear, well-prepared and with a very strong content. I could see that the students were interested and active.

The lecture notes are thorough and useful, even for researchers, as they contain important material which is however hard to locate elsewhere in the literature. The quizzes and assignments are well adjusted; I can testify of their usefulness for students. The projects are well chosen and always related to important aspects of the course; they present an appropriate level of difficulty.

Finally, the web page for the course is very complete, offers a vast quantity of downloadable material and an extensive bibliography.

To summarize, this course is extremely well designed. It perfectly plays its role of laying soundly the foundations of computer algebra, and will be useful to any student, specialist of not.

Éric Schost
Assistant Professor and Canada Research Chair
the University of Western Ontario

8.6 Letter from Professor Mike Bauer

Marc:

Thank-you sharing your outline and notes for your Graduate Course *CS 855 - Parallel Scientific Computing: Models, Algorithms and Implementation.* This looks to be a very interesting course and a very timely one – given the growing interest in parallel computation and the computational advantages it offers. I think that the focus on Symbolic Parallel Computing is a good one given the number of students in our Department that are interested in this area. This is also a good area to start raising interest among students and in which to start raising the profile of the Department – it will be a growing area of importance in the near future.

This course might be of interest to students in other areas as well. One suggestion might be to broaden the scope of papers that students could read, especially if students could find papers in their disciplines that might be related to parallel computation. Alternatively, you might consider broadening the scope of the course to deal with "Scientific Computing" in the broader sense and perhaps offer a variation of this course as a 600-level graduate course and advertise it to Departments in Science, Engineering and Medicine. This could be quite an exciting course in parallel computation touching on many different disciplines.

Mike Bauer

8.7 Letter from Professor Mark Giesbrecht

Mark W. Giesbrecht, Associate Professor
David R. Cheriton School of Computer Science
Faculy of Mathematics
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1

Telephone: +1 519 888-4567 ext. 6582
Fax: +1 519 885 1208
E-mail: mwg@uwaterloo.ca
Internet: http://www.uwaterloo.ca/~mwg

August 30, 2007

Promotion and Tenure Committee
Department of Computer Science
University of Western Ontario
London, Ontario, N6A 5B7

Dear Promotion and Tenure Committee,

At the request of Dr. Marc Moreno Maza, I would like to offer a short evaluation of his course on compilers (CS 447 – Compiler Theory). I note that I have taught courses on compilers at UWO (CS 447 as well as a graduate course on compiler optimization) and at Waterloo. I have also served as director of the undergraduate programme at Waterloo (an associate director of the school of computer science) and thus have some experience with where this course fits within the CS curriculum.

A course on compilers is simultaneously extremely important, difficult to teach well, time consuming for all involved, and extremely rewarding. It is also difficult to fit the necessary material into the course to do justice to the full breadth of a modern compiler. While the course often has relatively few students, these are generally a self-selected elite group who will go on to graduate school and shape opinions on the CS programme.

Marc has done an excellent job in putting together a course which covers the fundamentals, but also gets to the "right" current topics in the field. The course notes themselves are clear and complete. Things move forward at an appropriate pace, yet the depth is more than sufficient. There are good examples in most sections, including lots of working code. I particularly liked the French-to-Javanese translator and English-to-French for Pascal translator, which are unique and effective!

One observation is that Marc makes room for code optimization in the syllabus. This is difficult to manage in a one-term course, but is the correct decision. It is an area that is of active current research, and one that brings together a lot of algorithmic skills the students have developed in their studies, such as graph algorithms. Marc's course is very strong in this section, and does a thorough introduction of dataflow analysis and associated optimizations. Again, the examples are good, and this is a topic where students can see the clear effects of sophisticated mathematical algorithms. It might have been nice to see something on recent topics such as just-in-time compilation and simple parallelization, but this would undoubtedly involve moving some necessary prerequisite topics earlier in the curriculum; for example, some of the parsing and lexical analysis could move into second year core.

The associated assignments are appropriate for the material, and clearly will be interesting and useful for the students. The midterm and final are also very carefully crafted, of appropriate difficulty, and reflect the course well.

In summary, this is a well-constructed and appropriate course for the topic of compilers at a senior level. It mixes the necessary background and "traditional" material with modern optimization techniques. I am sure that the students who take this course would be well-served by the contents and structure of the course, and challenged and rewarded by the materials.

Yours truly,

Mark Giesbrecht

8.8 Letter from Professor Roberto Solis Oba

August 17, 2007

Prof. Marc Moreno Maza
Department of Computer Science
The University of Western Ontario

Dear Marc,

I have reviewed your undergraduate course materials for Computer Science 210 and Computer Science 211. The website for CS211 is very well designed as it makes it very easy to find all sorts of information relevant to this course. I could not find the website for CS210, as you taught that course some time ago.

The course outlines follow university guidelines and clearly explain what the course is about, what the prerequisites are, what the course work is, and what is expected from each student. I found the programming assignments for CS210 particularly interesting. They are extremely detailed and not only explain what the students are required to do, but also how that work relates to what was taught in the classroom. It is nice that each assignment comes from a real application, as this helps the students understand the relevance of the material covered in the course.

I found the exams well designed and of appropriate length and level of complexity. The questions test the material covered in class and require a bit of ingenuity on the part of the students.

I think that these two courses are very well designed and I am sure that most of the students who took them must have enjoyed them.

Sincerely,

Roberto Solis-Oba

8.9 Letter from Doug Vancise, Undergraduate Chair

31 August 2007

Dr. Marc Moreno Maza
Assistant Professor
Department of Computer Science, UWO

Dear Marc:

You've requested feedback concerning your website for Computer Science 211b from the January 2007 term. I find your site straightforward, clean, and intuitive to navigate. All the requisite items (contact data, course outline, announcements, etc) are easy to locate and to understand. Assignments are well-written, appropriate for the course, and presented with sufficient detail so that students can clearly understand what's required of them.

Students appreciate that supplemental materials such as the lecture notes and exams from previous offerings of the course are available for download in convenient formats. I am impressed by the work you've done in preparing additional materials (shell script and C programs used in the notes, corrections and hints concerning exercises from the text, the convenient summary of C library documentation, the article on C pitfalls, etc), and with your permission, I'd like to incorporate many of these items into my website when I teach 211a this fall.

Sincerely,

D.A. Vancise
Undergraduate Chair, Department of Computer Science

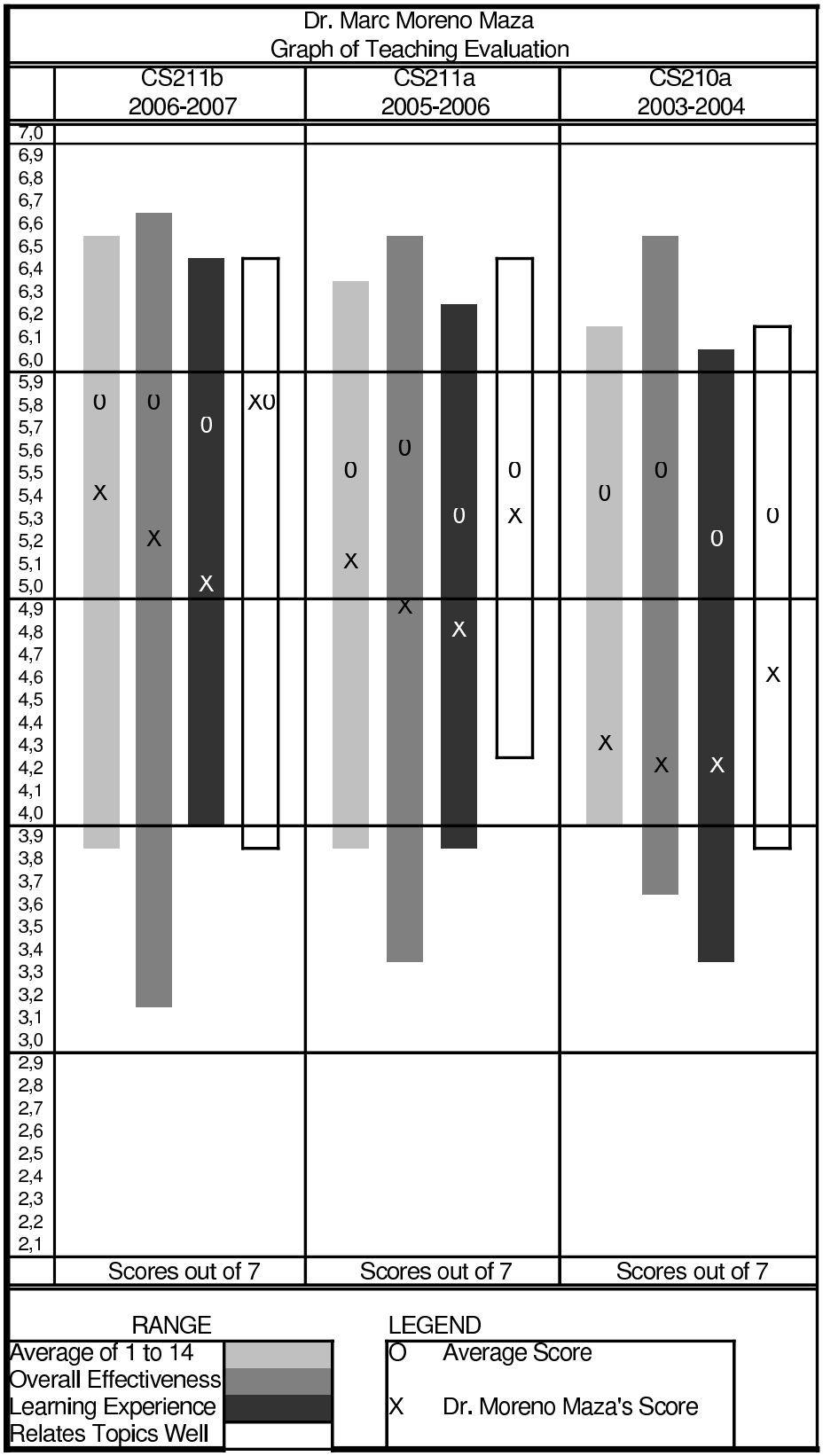
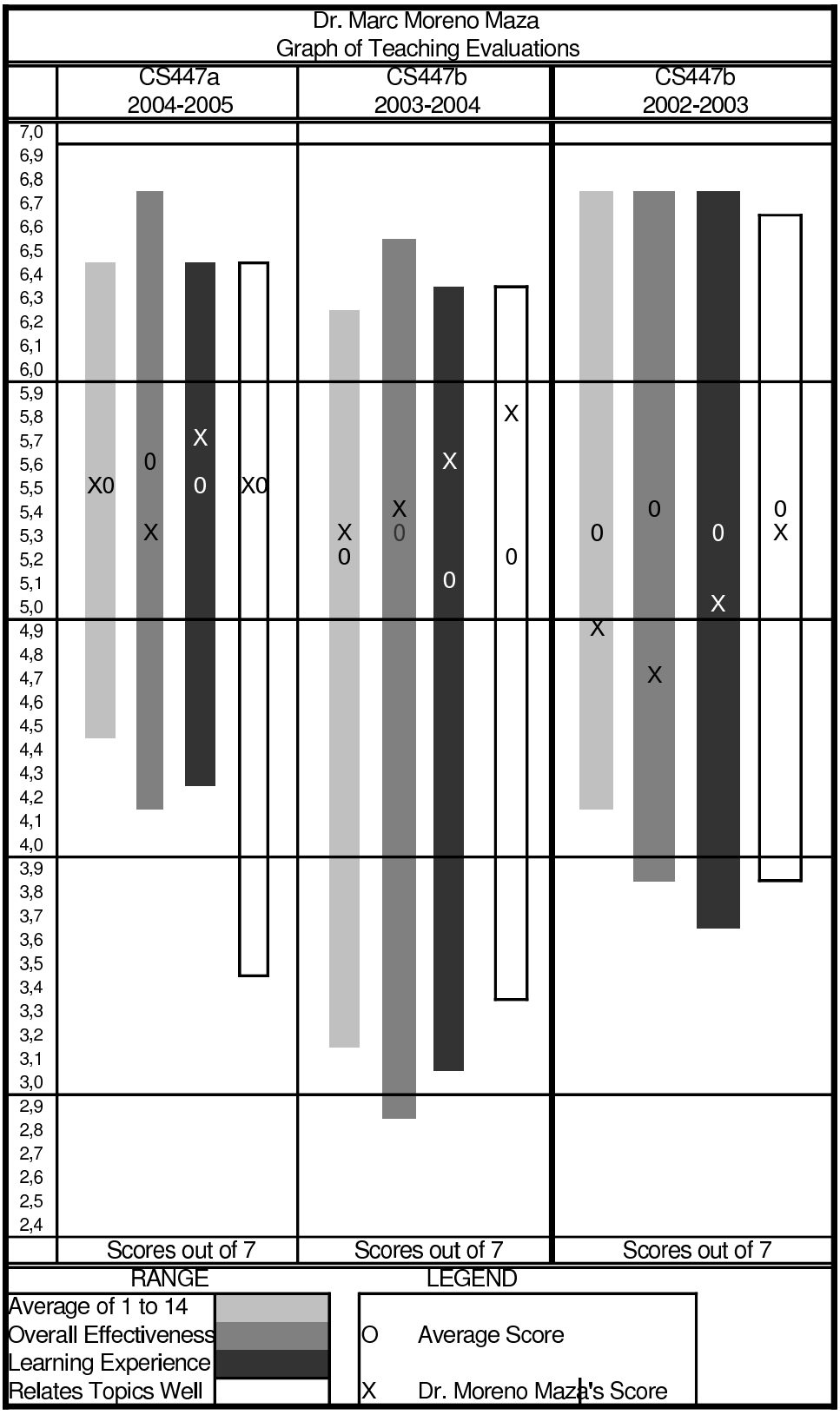


Figure 1: Student rating for 2nd year courses



The bars indicate the range of scores for all fourth year courses taught in the Computer Science Department in the year indicated.

Figure 2: Student rating for 4th year courses

	1	2	3	4	5
COURSE	CS545	CS545	CS556	CS556	CS556
TERM	W2003	W2004	W2005	W2006	W2007
#RESPONSE	6	5	4	3	4
Objectives clear	4,33	4,8	5	4,67	4,75
Expectations clear	4,33	4,4	5	4,67	5
Stimulate my interest	3,83	4,2	5	4,33	5
Evaluation appropriate	3,83	4,6	5	4,67	5
Workload appropriate	4,33	4,4	5	5	5
Instructor knowledgeable	4,5	4,8	5	5	5
Instructor available	4,5	4	5	4,67	5
Material covered	3,83	4,6	5	5	5
Level appropriate	3,67	4,2	5	4,67	5
Well organized	3,83	4,2	5	4,67	4,75
Overall effective	4,17	4,2	5	4,33	5
AVERAGE (out of 5)	4,11	4,4	5	4,7	4,95

	6	7	8	9	10	11
COURSE	CS652	CS855	CS855	CS867	CS874	AM583
TERM	W2004	W2006	W2007	W2005	W2003	F2003
#RESPONSE	7	3	5	9	5	6
Objectives clear	5	5	4,8	4,78	5	
Expectations clear	5	4,67	4,8	4,56	4,6	
Stimulate my interest	4,86	4,67	4,4	4,78	4,6	6,5
Evaluation appropriate	5	4,67	4,6	5	4,6	
Workload appropriate	5	4,33	5	4,89	4,8	6,83
Instructor knowledgeable	5	5	5	5	5	7
Instructor available	4,86	5	4,8	5	4,8	7
Material covered	4,71	4,67	5	4,78	4,4	6,67
Level appropriate	5	4,67	4,8	4,89	4,8	6,67
Well organized	4,86	4,67	4,6	4,44	5	6,67
Overall effective	5	4,67	4,8	5	4,8	6,83
AVERAGE (out of 5)	4,94	4,73	4,78	4,83	4,76	6.77/7

Figure 3: Student rating for graduate courses