On the Use of Simulation and Parallelization Tools in Computer Architecture and Programming Courses

Renato J. Figueiredo†, José A. B. Fortes†, Rudolf Eigenmann†, Nirav Kapadia†, Valerie Taylor††, Alok Choudhary††, Luis Vidal‡ and Jan-Jo Chen‡

†School of ECE ‡Dept. of Mathematics and Computer Science
Purdue University Chicago State University

††Department of EECS Northwestern University

Abstract

Computer architecture and programming are disciplines that require extensive experimentation with computer tools, such as simulators and compilers. At the authors’ universities, several tools are being incorporated in courses at the junior and senior levels by using a powerful, web-based network-computing system as a computational and educational resource. The educational content includes examples, manuals, homeworks and other related information. The web-based computing system (PUNCH: Purdue University Network Computing Hubs) provides transparent access to computers and tools from any machine capable of browsing the web. The result is a system that supports the integration of a large number of tools in undergraduate classes. The system is publicly accessible and, upon request, available for use by other universities and educators. This paper describes the existing system, discusses examples of tool-integration in existing classes and reviews the current status of the project. It also reports on experiences at the three institutions of the authors on the use of PUNCH and the inclusion of tool-based homeworks and content into undergraduate classes.

I. Introduction

As the complexity of the hardware and software of computing systems continues to grow, the use of software tools for computer architecture design and programming are essential to computer architects and software developers in industry. It is therefore necessary that computer engineering students be exposed to extensive experimentation with computer tools in undergraduate computer architecture and programming disciplines.

At the authors’ universities, several tools are being incorporated in courses at the junior and senior levels by using a web-based network computing system as a computational and educational resource. This system is called PUNCH (Purdue University Network Computing Hubs). PUNCH provides access to a pool of computers, installed tools (and their documentation) and educational material (“virtual laboratory” experiments and homeworks) from any machine capable of browsing the web. The set of installed tools include simulators of
caches, pipelined datapaths, multiprocessors, instruction sets, compilers, program analyzers and trace generators/analyzers developed by researchers from academia and industry.

Access to the network-computing system, including account requests, document retrieval and actual execution of tools, is obtained entirely through standard, web-based interfaces. The emphasis is on exposing the students to the functionality and nature of tools, while eliminating the need for time spent in securing access to machines, accounts, documentation, and learning unfriendly interfaces. The result is a system that supports the integration of a large number of tools in undergraduate classes, while minimizing the overheads of installing and learning a tool and finding resources to run it.

Our work is part of an NSF-funded project on combined research and curriculum development. This paper describes the computational and educational infrastructure currently available for computer architecture and parallel programming tools in the existing system. It also discusses examples of tool integration in existing classes and the inclusion of tool-based homeworks and content into undergraduate classes, and reports student evaluations on the use of PUNCH.

The paper is organized as follows. Section II describes the computational infrastructure available in the Computer Architecture and Parallel Programming Hubs of PUNCH. Section III describes the educational resources that are available in the fore-mentioned hubs. Section IV presents a detailed example of all steps involved in deploying a class assignment, from account creation to student assignment. Section V describes the integration of the architecture and programming tools with undergraduate computer engineering curricula in the authors’ universities. Section VI discusses the availability of the computational and educational resources to other universities and educators. Section VII presents concluding remarks.

II. Computational infrastructure

Research performed in many universities and industries continues to provide public-domain and commercial tools for the design, evaluation and programming of high-performance processors and computing systems. Due to their nature and complexity, such tools most often demand complex installation and maintenance procedures, and plentiful computing resources to achieve acceptable performance. The overheads associated with satisfying these requirements are addressed by the PUNCH network-computing infrastructure.

Many representative research tools in computer architecture and parallel programming have been installed onto suitable computing resources as part of this project. These tools, corresponding documentation, and educational material are being integrated into computer education curricula. This section describes this computational infrastructure currently available in the Computer Architecture and Parallel Programming hubs of PUNCH.

Tables 1 and 2 show the set of computer architecture and parallel programming tools currently available in the system. For each tool, the table lists the user interface to the tool provided by PUNCH, and related course topics. For tools with native graphical/interactive user interfaces, PUNCH provides the tool’s native interface to the user through remote display technologies such as Virtual Network Computing [15] connection. For other types of tools, PUNCH provides a customized interface based on HTML forms. Figure 1 depicts examples of the HTML and graphical interfaces for the tools Cachesim5 and DLXview, respectively.
Figure 1: Graphical (left) and HTML (right) user interface examples (from DLXView and Cachesim5, respectively).

The computer architecture hub contains simulation tools that model processor pipelines, memory hierarchies, and multiprocessor architectures, as well as tool-building packages that allow the development of customized tracing/simulation tools.

The current parallel programming hub contains a number of compilers for parallel machines and a tool that lets one analyze the maximum parallelism available in a given application. The goal of this hub is to provide students, researchers and developers of parallel computer applications with all the tools necessary to develop parallel programs, to study their characteristics, and to tune their performance. The compilers include Polaris [16], SUIF [6], and the Triman compiler infrastructure [8]. The tool for analyzing maximum program parallelism is MaxP [9] developed at Purdue University.

Future extensions to the parallel programming hub include the installation of interactive tools. Performance analyzers are among the most important parallel programming tools. We are currently working towards an implementation of the tools UrsaMinor [14], Guide-View [10], and CIAT [1].

III. Documentation, educational modules, and exercises

In addition to the computational infrastructure provided by PUNCH, integrating tools to existing computer engineering classes requires the availability of extensive educational material. This includes manuals, tool documentation, answers to frequently-asked questions, examples, tutorials and homeworks. This section discusses the development of such educa-
Table 1: Simulation/Tracing tools currently available in PUNCH’s Computer Architecture Hub. The terms GUI, ILP, DSM and SMP refer to graphical user interface, instruction-level parallelism, distributed shared-memory and symmetric multiprocessors, respectively. The topics shown in boldface in the rightmost column are typically covered in undergraduate courses and can benefit from integration with one or more tools.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Purpose</th>
<th>User interface</th>
<th>Related course topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>CacheSim5 [5]</td>
<td>Cache simulator</td>
<td>HTML forms</td>
<td>Memory hierarchy</td>
</tr>
<tr>
<td>Dinero-IV [7]</td>
<td>Cache simulator</td>
<td>HTML forms</td>
<td>Memory hierarchy, tracing</td>
</tr>
<tr>
<td>DLXView [2]</td>
<td>Processor pipeline simulator</td>
<td>GUI, interactive</td>
<td>Pipelines, ILP</td>
</tr>
<tr>
<td>LSU Proteus [3]</td>
<td>Multiprocessor simulator</td>
<td>HTML forms, GUI</td>
<td>DSM, coherence, ILP, consistency</td>
</tr>
<tr>
<td>Shade [5]</td>
<td>Tool-building libraries</td>
<td>HTML forms</td>
<td>Instruction set, assembly programming</td>
</tr>
<tr>
<td>WW1-II [12]</td>
<td>Multiprocessor simulator</td>
<td>HTML forms</td>
<td>DSMs, SMPs, coherence</td>
</tr>
<tr>
<td>XSPIM [7]</td>
<td>MIPS assembly simulator</td>
<td>GUI, interactive</td>
<td>Instruction set, assembly programming</td>
</tr>
</tbody>
</table>

Table 2: Programming tools currently available in PUNCH’s Parallel Programming Hub.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Purpose</th>
<th>User interface</th>
<th>Related course topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxP [9]</td>
<td>Parallelism analyzer</td>
<td>HTML forms</td>
<td>Parallel programming</td>
</tr>
<tr>
<td>Polaris [16]</td>
<td>Parallelizing compiler</td>
<td>HTML forms</td>
<td>Parallel programming</td>
</tr>
<tr>
<td>SUIF [6]</td>
<td>Parallelizing compiler</td>
<td>HTML forms</td>
<td>Parallel programming</td>
</tr>
<tr>
<td>Trimaran [8]</td>
<td>Parallelizing compiler</td>
<td>HTML forms</td>
<td>Parallel programming</td>
</tr>
</tbody>
</table>

Educational content for computer architecture and programming tools.

Educational content on the Computer Architecture and Parallel Programming hubs of PUNCH is available as a collection of on-line documents accessible to students and instructors via PUNCH’s web-based interface. Such documents can be categorized as follows:


<table>
<thead>
<tr>
<th>Laboratory</th>
<th>CacheSim5</th>
<th>DineroLV</th>
<th>DLXView</th>
<th>HPAM-Sim</th>
<th>Pecos</th>
<th>RSIM</th>
<th>Shade</th>
<th>SimpleScalar</th>
<th>WWf-2</th>
<th>XSpin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache Simulation</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Education</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction Level Parallelism</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiprocessor Simulation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network Simulation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallelization Tools</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uniprocessor Simulation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Assignment of tools to virtual laboratories.

1. **Original tool documentation.** All documentation that is provided in the tool’s distribution package.

2. **Local tool documentation.** Includes description of extra features and/or limitations that are specific to the interface of the tool provided by PUNCH.

3. **Questions and Answers.** Answers to frequently-asked questions (FAQs). PUNCH provides e-mail links to people in charge of maintenance of each installed tool, allowing questions and feedback to be sent through its web-based interface.

4. **Educational modules and examples.** Include example files that can be accessed by PUNCH users, first-time user’s guide (tutorials introducing the tool’s interface on PUNCH), and educational modules (tutorials introducing the tool’s capabilities and functionality).

5. **Exercises.** Documents with tool-based experiments (exercises and open-ended projects) that can be assigned in classes.

Items 4 and 5 are designed to facilitate the integration of tools with computer engineering courses. Educational modules are introductory tutorials on the usage of a tool; their purpose is to guide the student through experiments that illustrate the tool’s functionality. Educational modules are self-contained documents that allow students to become familiar with tools by independently experimenting with the tool, and checking results and conclusions against the expected ones described in the module. Students are not expected to turn in assignments based on educational modules, but rather to familiarize themselves with the tool(s) that are used in exercises and course projects.

Exercises and open-ended projects are documents that are intended to be used in (graded) class assignments. They assume that students have acquired the necessary background knowledge of the subject and related tool(s) from class material and the educational modules, and typically involve more challenging experiments than the ones of introductory educational modules.
The structure of educational modules and exercises/projects is similar; they differ on the scope/complexity of the experiments and on the availability of results, answers and conclusions to students. These documents contain five sections, as follows:

1. Introduction. Brief introduction to the problem being addressed by the module or exercise.

2. Experimental methods and models. It defines which aspects of the material discussed in Section 1 will be covered by the experiment (and are thus modeled by the tool used in the experiment), and explains the experimental methodology. In open-ended projects, the definition of methods and models may be left as part of the student’s assignment.

3. Experiment setup. This section determines the model parameters that will be varied/observed by the student during the experiment and the specific tool options to be used. As in Section 2, for open-ended projects the definition of these parameters may be left as part of the student’s assignment.

4. Analysis. This section provides guidelines to the analysis of the data gathered after the experiments have been conducted. The purpose of this section is to aid in the interpretation and understanding of the results obtained from the experiment. For educational modules, results, answers and conclusions are provided in this section to allow students to check their work. For exercises and projects, results and answers are provided in a separate document whose access is granted by PUNCH to registered instructors only.

5. References. This section contains bibliographical references to all published material that is cited throughout the entire document.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Type</th>
<th>Topic covered by module/exercise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cachesim5</td>
<td>Module</td>
<td>Performance study of single-level cache memory system designs...</td>
</tr>
<tr>
<td></td>
<td>Module</td>
<td>Performance study of two-level cache memory system designs.</td>
</tr>
<tr>
<td>MaxP</td>
<td>Module</td>
<td>Analyzing the maximum parallelism in programs.</td>
</tr>
<tr>
<td>Polaris</td>
<td>Module</td>
<td>Techniques of a parallelizing compiler.</td>
</tr>
<tr>
<td>RSIM</td>
<td>Module</td>
<td>Parallel speedup of single-issue and ILP processors.</td>
</tr>
<tr>
<td>WWT-2</td>
<td>Module</td>
<td>Parallel speedup of SMPs.</td>
</tr>
<tr>
<td></td>
<td>Module</td>
<td>Parallel speedup of DSMs.</td>
</tr>
<tr>
<td></td>
<td>Module</td>
<td>Memory hierarchy of DSMs.</td>
</tr>
<tr>
<td></td>
<td>Exercise</td>
<td>Performance scalability of SMPs and DSMs.</td>
</tr>
</tbody>
</table>

Table 4: Computer architecture and parallel programming educational modules and exercises.

Table 4 lists the educational modules and exercises developed for computer architecture and parallel programming tools.
IV. Educational experiment example

This section presents an example of the integration of a tool-based experiment (and associated educational content) into a computer architecture class using PUNCH. It describes all steps involved in this process, from account creation to student assignment, from an instructor’s perspective. The example includes actual PUNCH screen-shots to illustrate the capabilities of the system’s web interface. The purpose of the example is to illustrate how the system facilitates the integration of tools to existing curricula. A discussion on this subject follows the example at the end of this section.

Without loss of generality, the example considers a scenario of an instructor teaching fundamental concepts on the architecture of shared-memory multiprocessors, possibly in a senior-level computer architecture class. In this scenario, the instructor wishes to assign a homework where students are to quantitatively study and compare the performance of bus-based and directory-based multiprocessors. It is intended to be the only assignment on the subject for this class, and should not require more than 2-3 weeks to be completed, since multiprocessor architecture is only one out of many other topics that are covered in the class.

Without the availability of an infrastructure like PUNCH, this task would require: (1) the installation of a suitable multiprocessor simulator tool (and benchmark programs) on local machines, and (2) that students learn the simulator’s user interface. The overhead in setting up the computing infrastructure and documentation, and in the students’ learning curve of the simulator’s interface and usage are likely to discourage the development of this assignment for the time frame intended for this assignment.
The availability of PUNCH’s computational and educational infrastructure reduces the overheads for both instructor and students. The remaining of this section describes the steps involved in setting up a virtual “lab assignment” using PUNCH.

1. The instructor requests an account via PUNCH’s web interface. In addition, she requests that a new class be added to PUNCH’s database, giving the students in this class access to the needed tools; this class will be referenced by its code (e.g., EE-401) by the students when they request class accounts.

2. Students request individual PUNCH accounts via the web. Figure 2 shows an example of the account request procedure.

3. In a handout, the instructor refers the students to the Multiprocessor Lab of the Computer Architecture Hub (via its web URL). Figure 2 shows the current organization of this lab on PUNCH. Assume the instructor has selected the Wisconsin Wind Tunnel-2 simulator for this assignment. In this handout, the instructor asks the students to:
   (a) read the basic documentation associated with the tool (Figure 3, left),
   (b) follow the first-time user’s guide to become familiar with the tool’s interface (Figure 3, right),
   and (c) go through two educational modules: parallel-speedup experiments with both SMP (Figure 4, left) and DSM multiprocessors. The students are not required to turn in experimental results or answers to questions for this handout. However, by the time they have completed the assignment, they will be familiar with the tool’s capabilities.

Figure 3: Available on-line documentation for the WWT-2 tool (left) and its first-time user’s guide (right).
interface and the experimental methodology that they will need in order to complete the homework assigned in step 5. Figure 4 shows the main page of the WWT-2 simulator’s HTML interface that is introduced to the students via the first-time user’s guide and the educational modules.

4. (Optional) The instructor has access to the documents for the homework exercise and solutions; she may adjust the homework to reflect particular needs of the class (e.g., extra questions, experiments, etc) and request that the updated version of the homework document be available to the class.

5. In a second handout, the instructor refers students to the exercise on the performance comparison between the two multiprocessor architectures (Figure 5, left). The students perform the experiments described in this document, collect and summarize outputs (Figure 5, right); for this assignment, they are required to turn in their homeworks to be graded.

Steps (1) and (2) are required only once at the beginning of the semester to provide access to one or more of the tools installed on PUNCH. These steps typically require 1 to 2 weeks, which can overlap with other activities in the first weeks of class.

Step (3) is expected to take 1 to 2 weeks; students can pace themselves in learning the tool and performing initial experiments, and since there is no homework to be turned in, the assignment can be overlapped with other class activities. The duration of the activities in step (5) depends on several factors, including expected simulation time to perform
the experiments and the level of difficulty involved in the analysis of the resulting data. Simple experiments can be designed for a 1-2 week time frame, while more challenging exercises and/or projects may require longer periods. The authors expect that, as exercises are developed, enhanced and posted by instructors in Step (4), instructors will be able to leverage from an on-line library of assignments of various difficulty levels when deciding on the assignments that they will reuse and/or adapt to their classes.

V. Integration with computer engineering curricula

The computer architecture and programming tools available on Punch have been used in several undergraduate- and graduate-level classes at the authors’ universities. Table 5 summarizes the usage of the system for the years of 1998 and 1999. The remaining of this section presents summaries of findings from student evaluations.

Purdue University

Evaluation results are available from experiments that used the Polaris compiler of the parallel programming hub. This PUNCH lab assignment was part of a entry-level graduate course in compilers. The main points of the students’ findings are as follows:

- The PUNCH interface is easy to learn. Accessibility via the Web was most important. Many students found the interface to be convenient for regular use. However the UNIX environment is still generally preferred over the PUNCH environment.

Figure 5: Experimental setup for the UMA/NUMA performance exercise (left) and sample program output file of a 16-processor WWT-2 simulation (right).
Table 5: Computer architecture and programming classes that used tools available on Punch for student assignments.

- The system response time and availability was considered always satisfactory.
- Students accessed the tools both from Purdue computing environments and from home. Among the platforms were PCs, Macintosh systems, Sun Workstations, HP machines, and various other platforms.
- The students pointed out that the PUNCH system was generally very helpful in learning the tool-specific aspects of their assignments. The multiple levels of help files made it easy to learn the tool.
- Most criticism was received with regard to the user interface for tool input and output. The current, simple Web forms were not considered adequate. Users would like to be able to transfer input and output to/from their favorite text editor or through pre- and post-processing steps similar to Unix pipes.

**Northwestern University**

Evaluation results are available from experiments that used the SimpleScalar simulator of the computer architecture hub. This PUNCH lab assignment was part of a entry-level graduate course in computer architecture (ECE D52).

A summary of the evaluation forms handed out to students showed an average rating of 4.5/5.0 for the effectiveness of PUNCH with aiding in understanding the concept of ILP. Students found the tool and the web interface easy to use.
Chicago State University

The Chicago State University students found Punch’s interface to the Cachesim5 cache simulator tool very easy to use. It provides the flexibility to set up their own one-level or two-level cache memory system for hands-on experimentation of the concepts learned in class.

The students also found that the navigation through Punch’s web interface was very easy to follow. They found that the handouts distributed in class, together with an introductory explanation of Punch and the CacheSim5 tool by the instructor, were very helpful in explaining the usage of the simulator via the web interface.

The procedure for requesting Punch accounts was very easy to follow; student accounts were set up and available as soon as students requested them. Thus, the students were able to start using tools right away.

The Punch laboratory was very appealing to the computer science students at CSU, and thus they feel that more computer organization tools should be installed in the Punch virtual laboratory and make them available to CS and engineering students. They felt that the virtual laboratory will be the laboratory of the future in computer science classes.

VI. Availability

The computational and educational infrastructure provided by PUNCH is publicly accessible, and available for use by other universities and educators, upon request. The system can be accessed through any WWW browser via the URL http://www.ece.purdue.edu/labs/punch. This entry point to the PUNCH infrastructure provides a web portal interface that allows current and new users to access documentation, request accounts, execute tools and contact PUNCH personnel.

VII. Conclusions and outlook

A large percentage of computer engineering graduates will have to use computer-based tools in their jobs. This paper describes an innovative approach to the integration of computer architecture and parallel programming tools into existing undergraduate and graduate-level curricula via network-computing technology, that reduces or eliminates the overheads involved in locating, testing, learning, using and maintaining tools.

In addition to the network-computing infrastructure, integrating tools into existing classes requires the availability of extensive educational material. The development of educational modules and class assignments and their integration into the infrastructure are also described in this paper via a detailed example of a “virtual lab” assignment using PUNCH.

This paper also reports on usage of the system in classes at the authors’ universities, and on student evaluations of their experience with the web-based interface.

Planned extensions to the system include support for a text-based, terminal interface to the infrastructure. This interface will enable fast execution of text-based interactive tools, such as debuggers, as well as a Unix-like interface to user file management. Another planned extension is to support controlled program execution. Current users of the Parallel Programming hub can compile and analyze their programs on PUNCH, but then have to post-process and run their executables on a system external to PUNCH. We will develop execution modes, such that the parallel program developer can perform all steps of the development within the PUNCH environment. This will require the installation of code-generating compilers, execution commands, and debuggers. This environment will support both machine-specific
and transparent modes. In a machine-specific mode the user develops and executes code of a particular machine type whereas in generic mode the target architecture is not relevant.

VIII. Acknowledgments

The authors thank the developers of the tools integrated in the Computer Architecture and Parallel Programming hubs for making them available. This work was partially funded by the National Science Foundation under grants EIA-9872516 and EIA-9975275, and by an academic reinvestment grant from Purdue University.

Bibliography


Mr. RENATO FIGUEIREDO is currently a doctoral candidate at Purdue University. Renato Figueiredo received both B.S.E.E. and M.S.E.E. degrees from the State University of Campinas (UNICAMP), Brazil, in 1994 and 1995.

Dr. JOSÉ A. B. FORTES is currently a Professor and Assistant Head for Education of the School of Electrical and Computer Engineering at Purdue University. Dr. Fortes received a B.S.E.E. degree from Univ. de Angola in 1978, a M.S.E.E. degree from Colorado State University in 1981, and a Ph.D. in Electrical Engineering and Systems from the University of Southern California in 1984.

Dr. RUDOLF EIGENMANN is currently an Associate Professor of Electrical and Computer Engineering at Purdue University. Dr. Eigenmann received a Diploma in Electrical engineering from ETH Zürich, Switzerland in 1980, and a Ph.D. in Computer Science from ETH Zürich in 1988.

Dr. NIRAV KAPADIA is currently a researcher at Purdue University. Nirav Kapadia received his B.E. degree in Electronics and Telecommunications from Maharashtra Institute of Technology (India) in 1990. He subsequently worked at the institution for a year as a lecturer. He received his M.S.E.E. and Ph.D. degrees from Purdue University in 1994 and 1999, respectively.

Dr. VALERIE TAYLOR is currently an Associate Professor of Electrical and Computer Engineering at Northwestern University. Dr. Taylor received a B.S.C.E.E. degree with highest honors from Purdue University in 1985, a M.S.E.E. degree from Purdue University in 1986, and a Ph.D. degree in Electrical Engineering and Computer Science from the University of California at Berkeley in 1991.

Dr. ALOK CHOUDHARY is currently an Associate Professor of Electrical and Computer Engineering at Northwestern University. Dr. Choudhary received a B.E. degree in Electrical Engineering from Birla Institute of Technology and Science in 1982, a M.S. degree in Electrical and Computer Engineering from the University of Massachusetts in 1986, and a Ph.D. in Electrical and Computer Engineering from the University of Illinois in 1989.

Dr. LUIS VIDAL is currently an Associate Professor of Mathematics and Computer Science at Chicago State University. Dr. Vidal received a B.S. degree in Mathematics from the University of Trujillo in 1970, M.S. degrees in Computer Science and Mathematics from Washington State University in 1981 and 1983, and a Ph.D. degree in Computer Science from the Illinois Institute of Technology in 1980.

Dr. JAN-JO CHEN is currently an Assistant Professor of Mathematics and Computer Science at Chicago State University. Dr. Chen received a B.S. degree in Architecture from Tunghai University, Taiwan in 1985, a M.S. degree in Computer Science from the Illinois Institute of Technology in 1990, a M.S. degree in Mathematics, Statistics and Computer Science from the University of Illinois at Chicago in 1996, a M.S. degree in Management Information Systems from the University of Illinois at Chicago in 1997, and a Ph.D. degree in Mathematics, Statistics and Computer Science from the University of Illinois at Chicago in 1997.