July 2017

Volume 8 Issue 2

JoCsE

Journal Of Computational Science Education

Promoting the Use of Computational Science Through Education

ISSN 2153-4136 (online)

Volume 8, Issue 2 **July 2017**

JOCSE

Journal Of Computational Science Education

The Journal Of Computational Science Education (JOCSE), ISSN 2153-4136, is published quarterly in online form, with more frequent releases if submission quantity warrants, and is a supported publication of the Shodor Education Foundation Incorporated. Materials accepted by JOCSE will be hosted on the JOCSE website, and will be catalogued by the Computational Science Education Reference Desk (CSERD) for inclusion in the National Science Digital Library (NSDL).

Subscription: JOCSE is a freely available online peer-reviewed publication which can be accessed at `http://jocse.org`.

Contents

Introduction to Volume 8 Issue 2

Steven I. Gordon
Editor
Ohio Supercomputer Center
Columbus, OH
sgordon@osc.edu

Forward

In this issue, Sharma describes a model scientific computing course at a liberal arts college. The course focuses on scientific data analysis and data visualization using Mathematica and a variety of open source tools for molecular visualization. It provides a description of the hands-on exercises and student assessments of the experience.

The article by Shamir describes a model for an undergraduate research experience in computer science that has been implemented as his institution. The program has resulted in about 40% of the undergraduate majors participating in the research experience across a wide range of topics.

The other articles in this issue detail the internship and research experiences of five students and their faculty mentors. Alford and Toth describe a project that compared the cost, power consumption, and computational efficiency of ARM-CPU systems with a regular cluster performing virtual screening with AutoDock Vina.

Haddad et.al. tested a new algorithm for the computation of Bayesian networks that generates multiple networks in parallel to help remove the bias of other approaches. They demonstrated a major performance gain through 64 processors after which the communications overhead resulted in wasted computational resources.

Isokpehi and his students used the Blue Waters Supercomputer to analyze the information from microbial genomes. They then used several data analytics tools to visualize the similarities and differences across a 547 annotation files for the Rhizobiales genome.

Peterson et.al. describe the contents of a computer science course for high school students with a capstone project of building an n-Body simulation. They report a number of challenges faced by students in the course along with the results of the final project.

Finally, Yu and Mark describe their model for the interplanetary low-thrust trajectories from Earth to Mars for spacecrafts supplying necessary cargo for future human-crewed missions. They were able to achieve significant speedup in the application after parallelization.

A model Scientific Computing course for freshman students at liberal arts Colleges

Arun K. Sharma
Wagner College
Department of Chemistry & Physics
1 Campus Road
Staten Island, NY 10301
aksharma@wagner.edu

ABSTRACT

This paper describes a course called, "Introduction to Scientific Computing" that has been developed for freshman students at Wagner college, a private national liberal arts institution. The course trains students in computational thinking and involves hands-on learning of typical work-flows in scientific data analysis and data visualization. Students develop proficiency in the symbolic computing platform, Wolfram Mathematica®, to apply functional programming to develop data analysis and problem solving skills. The course presents computational thinking examples in the framework of various scientific disciplines. This exposure helps students to understand the advantages of technical computing and its direct relevance to their educational goals. The students are also trained to perform molecular visualization using open source software packages, such as Avogadro and Visual Molecular Dynamics, to understand secondary and tertiary protein structures, construct molecular animations, and to analyze computer simulation data. These experiences stimulate students to apply these skills across multiple courses and their research endeavors. Student self-assessment data suggests that the course satisfies a unique niche in undergraduate education. We have provided a sample syllabus, homework assignments, and examples of student work to aid in the design and implementation of similar courses at other institutions.

CCS Concepts

•**Applied computing** → **Education;** *Chemistry; Physics; Computer-assisted instruction; Computer-managed instruction;* Mathematics and statistics; Collaborative learning;

Keywords

Undergraduate; Freshman Year; Scientific Computing; Education; Visualization

1. INTRODUCTION

DOI: https://doi.org/10.22369/issn.2153-4136/8/2/1

Computational Science is a relatively new phenomenon in the long march of scientific disciplines and pursuits. From their invention in 1940s[1] to large scale supercomputers that are advancing fundamental research in all disciplines, computing has a fascinating history[2]. However, a typical liberal arts student gets very little to no exposure to such inspirational and powerful uses of computers and computational thinking. Most students consider the usage of computers for routine purposes (browsing, social media, document processing, etc.) as being "technologically literate". However, this is a very limited form of literacy and educators need to raise the level of technological competency to enhance the career prospects of our students[3, 4].

The STEM education community has focused its attention on introducing students to computational thinking and computation as distinct from routine computer usage [5, 6, 3]. The physics education community has been instrumental in adopting computer aided instruction and full fledged curricula and majors in Computational Physics exist at many institutions[7, 8, 9, 10, 11, 12]. Similar attempts to introduce computing in the chemistry curriculum have also been reported in the literature [13, 14, 15, 16, 17, 18]. These curricular level transformations are important milestones in modernizing and improving educational outcomes and skills of future graduates. However, an institution without the resources to engage in curricular overhaul may not be able to take advantage of such approaches. Particularly, smaller liberal arts colleges with limited faculty size and infrastructure are in an especially unfavorable position to embed computing holistically in an entire discipline or program.

A feasible alternative is to modernize the introductory computer course to highlight computational thinking and move beyond routine usage of computers. However, this approach lacks cohesiveness and students may not see the computing as directly relevant to their studies. We advocate the alternative approach to embed computing in a freshman level course in the natural sciences and provide students with an exposure to computational thinking in the framework of their course. A striking example of such an approach in an introductory mechanics course was recently reported [19]. That course required students to learn elementary programming skills to solve physics problems using VPYTHON programming environment. These students did not possess formal computer science coursework exposure and a majority of the class successfully completed the evaluation. These approaches are warranted to ensure that future STEM grad-

uates can converse with computers with reasonable fluency in the future. We highlight an approach to inject computational thinking and molecular visualization at the freshman level. The importance of visualization in disciplines like chemistry and biology cannot be overstated[20, 21] and the power of computational thinking to assist students in physics and other disciplines has been well documented[19, 7, 8, 16, 17]. Our course provides students with a toolbox of computational thinking and molecular visualization and editing tools that can be advantageously applied to a variety of courses offered in the STEM disciplines. The easy availability and widespread acceptance of these tools by students has the potential to increase the richness and depth of coursework in many disciplines.

2. METHODOLOGY

We designed and implemented a Scientific Computing course to teach elementary computational thinking in the framework of STEM disciplines and to highlight applications of computing to the natural sciences. The course aims to develop specific competencies of computational thinking, molecular visualization and editing, and data analysis. The course also aims to provide students with an introduction to functional programming, large scale data visualization and representation. A Scientific Computing course is commonplace in engineering disciplines. Such courses are typically focused on applications of numerical methods and their applications in solving scientific and engineering problems. There are many texts available on scientific computing with this focus [22, 23, 24, 25]. However, these books and corresponding courses are designed for an audience familiar with some computer programming language and/or advanced mathematics courses like calculus or linear algebra. Such courses are generally offered by Computer Science faculty and are typical coursework for engineering concentrations or Computational Physics majors.

However, there is no such parallel at liberal arts colleges. A standard scientific computing course as previously described would not be applicable to the student population at liberal arts colleges. Wagner College, like many liberal arts colleges, requires a semester of a Computer Science affiliated course to provide students with an exposure to "technological skills". Sadly, such courses typically do not delve into high-level computational skills or computational thinking and generally provide instruction in using Microsoft Office® products and rudimentary worldwide web concepts. These courses and approaches were probably valuable a decade ago when computing devices were not quite as prevalent. However, in today's world these courses appear outdated and do not provide relevant skills to the modern undergraduate student. We devised a course called, "Introduction to Scientific Computing", with the express intention to include all potential STEM students in the course. The course is designed to be inclusive toward students with all levels of preparation and previous exposure to computers and mathematics preparation. Most importantly, the course has no computer science pre-requisites. The course pre-requisites are any introductory class in the natural science disciplines taken during the first semester freshman year. Thus, the course is designed to be appropriate and most advantageous for second-semester freshman students.

The broad spectrum of the course allows it to serve a variety of students pursuing multiple STEM career paths. The course intention is to highlight the ubiquitous thread of computing that permeates modern scientific endeavors. The course goal is not to design and produce efficient programs or code. The course goal is to reach out to students who may have never considered computers as an ally to perform the science and learning that interests them. An important course goal is to highlight the role and power of computational thinking in solving problems. The course uses the freely available text, An Elementary introduction to Wolfram language[26], by Stephen Wolfram, the creator of the Wolfram programming language which is at the core of Mathematica®.

We chose to use the Mathematica platform to reduce the entry barrier and also to take advantage of the immense variety of applications and in-built functions in Mathematica that span the STEM domains and beyond. Python programming language is similar in many respects and we think it could be suitable to construct a similar course. However, we wanted to steer clear of the notion that this is a computer programming course and eschewed the Python programming language. We wanted to ensure that we could reach the maximum number of students and even those students who had previous negative experiences with computer programming. A similar course could be constructed using Maple®, Matlab®, or other software packages based on instructor familiarity and availability. The most important consideration is that there should be an element of higher level programming and scripting that allows students to engage with the code and to construct analysis tools using functional programming ideas. Thus, we would argue that spreadsheet based packages like Microsoft Excel®, etc. are not appropriate as the primary course platform.

The course can be divided into two broad sections: Mathematica technical platform and molecular visualization and editing. The Mathematica portion familiarizes students with the basic data structures and operations. We also highlight the applications of curated databases and their integrations into solving scientific problems. Students are introduced to various forms of data presentation including static and dynamic plots and charts. Finally, we introduce students to molecular editing and visualization using open source tools like Avogadro[27] and Visual Molecular Dynamics(VMD)[28]. We combine the elements of data analysis and visualization by presenting students with molecular dynamics simulation trajectory and carrying out its analysis as a classroom exercise. The next few sections will describe each of these modules in more detail.

3. MATHEMATICA MODULE

The course starts by introducing students to the various domains that can be accessed through Mathematica. This is achieved by highlighting in-built functions in Mathematica. We start by plotting trigonometric functions and three-dimensional plots. Mathematica recently added curated databases to the program. These allow a look-up of data from domains like chemistry, physics, biology, financial markets, economic data for countries, engineering data, mortality rates, weather data, etc. For many students this is their first ever encounter with such data. The goal of the first lecture is to ensure that students feel empowered and experience that they can write the code that can access such high level tasks. We impress upon students that the course is not about nuts and bolts of programming, but learning

technical platforms that can be used to address high level questions.

The course follows a workshop model in which the instructor projects the notebook (Mathematica interface) on a large screen and students write down the code on their individual computers. This is important to get the first practice of writing down the code and to increase student familiarity and confidence with the interface. The basic data structure in Mathematica is a list. We spend at least three lectures on familiarizing students with the list (array) structure and performing operations. Standard operations like reversing the list, calculating mean, standard deviations, etc. are carried out to develop baseline competency. We perform simple algebra on lists and solve problems that require setting up lists for solutions. Such problems abound in scientific domains and provide a review of scientific concepts as well as the skill of interpreting scientific problems into a computational framework. We necessarily choose problems from different domains, physics, chemistry, etc. to highlight broad usage of the platform and to inspire students to use their newly acquired skill across other courses.

Students also learn to visually represent different types of data. The standard mathematical functions, dynamic plots, parametric plots, etc. are also explored in the course. Bar charts and histograms are also covered to provide students with an overview of different aspects of data presentation. The students appreciate different data representations and their appropriateness to highlight certain aspects of the data. Lively discussions on choosing data representation are commonplace in the classroom. In all cases the data originates from solution of a problem from one of the scientific disciplines. This builds up on previous knowledge of the core principles of Mathematica. A glimpse of the diversity of examples that we execute in class and homework assignments for this module is provided below. These examples take advantage of Mathematica's curated research databases and the coding principles taught in the course. This approach enables students to use computational skills in their other courses and research assignments.

1. Plotting density and temperature of water to determine the temperature of maximum density of liquid water
2. Construction and analysis of dice games
3. Plotting density of elements and analyzing trends
4. Identifying the 10 most populated cities in the USA and data representations
5. Determination of the number of airports in the G8 countries
6. Plotting temperature data for various cities and analyzing weather trends
7. Atomic radii of elements in the second and third period of the Periodic Table
8. Interactive histograms of distributions of randomly selected integers
9. Interactive plots of mathematical functions

We will highlight the example of a die throwing game to demonstrate the readable and expansible nature of the code. The algorithm is simply described as:

1. Create a list of 6 integers for a regular six-sided die
2. Pick an integer randomly to represent a particular face of the die
3. Accumulate the counts of each face over a number of die throws
4. Plot the distribution of each face for the specified number of trials

A sequential construction of this example is provided in the following code snippet

```
rolls = Table[RandomInteger[{1,6}],{i
    ,1,100000}];
Histogram[rolls]
```

However, these can be merged into a one-line code eliminating the need to create and store a variable. This approach where functions are input to other functions is a powerful feature of functional programming and we stress this repeatedly during the course. This feature enables students to chain complicated tasks into simple and easy to read code fragments.

```
Histogram[Table[RandomInteger[{1,6}],{i
    ,1,100000}]]
```

This can be easily extended to two dice games and plotting the histograms of sums of two dice throws. The following command simulates million throws of two dice and adds the output. It then plots a histogram to highlight the distribution of the recorded sum.

```
Histogram[Table[RandomInteger[{1, 6}] +
    RandomInteger[{1, 6}], {i, 1,
    1000000}]]
```

As students develop their skills in the language and computational thinking, we revisit this example and create a custom function to carry out these operations. The dice example is used to highlight the power of repeated trials and the need for large numbers of trials to understand stochastic phenomena. Students construct the scenario of a comparison between an unloaded regular dice and a dice loaded to favor the face bearing the numbers four or six three times over other faces. This example impresses upon students the need for repeated trials and also the role of random numbers in stochastic processes. The output of this comparison of a loaded die with a regular die is shown in Fig. 1. The result clearly shows that a loaded die behavior can be detected with few trials, however, the regular die output needs many trials to verify the equal probability of each outcome. The code snippet highlights the advantage of using Mathematica and the ease with which students progress in their knowledge of coding.

```
regularDice[throws_] :=
Module[{out = Table[RandomChoice[{1, 2,
    3, 4, 5, 6}], {throws}]},
 Histogram[out, PlotLabel -> throws "
     Unloaded Die Throws"]]

loadedDice[throws_] :=
Module[{out =
   Table[RandomChoice[{0.1, 0.1, 0.1,
       0.3, 0.1, 0.3} -> {1, 2, 3, 4,
```

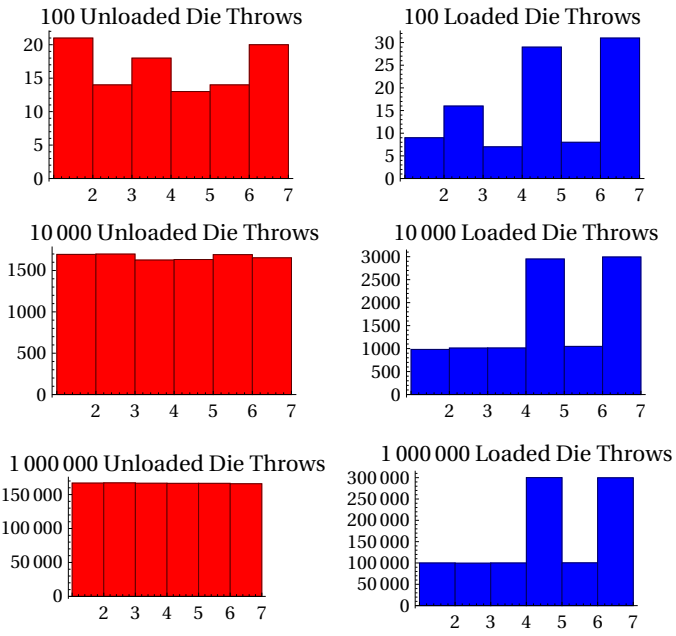


Figure 1: Histograms of outputs of an unloaded and loaded die. The behavior of a loaded die is clear even at low number of trials.

```
        5, 6}], {throws}]},
  Histogram[out, PlotLabel -> throws "
     Loaded Die Throws"]]

GraphicsGrid[{{regularDice[100],
   loadedDice[100]}, {regularDice[10000],
   loadedDice[10000]}, {regularDice
      [1000000], loadedDice[1000000]}}]
```

The focus of this module is to empower students with the skills of computational thinking and its application in various domains. The students are reminded constantly that Mathematica would be a great help for them in other courses, laboratory reports, and homework assignments. Students are also trained to write short programs (called Modules) that chain multiple built-in functions. This experience introduces students to the power of programming and its applicability to their daily activities. However, the course is not devoted to programming, rather its intention is to highlight the power and ease of functional programming and the advantages it can provide to students. The examples have been deliberately chosen to improve student understanding of pre-calculus and calculus, data analysis, data fitting, data visualization, and elementary physical science concepts. In the next section we describe the molecular visualization module.

4. MOLECULAR VISUALIZATION

Molecular editing and visualization is also embedded in the course to allow students to experience the applications of computing beyond numerical calculations. Students use an open-source application Avogadro[27] to create and edit molecular structures. Students also use Avogadro's features to create carbon nanotubes, aromatic compounds, and polypeptide sequences. This activity of building different molecules

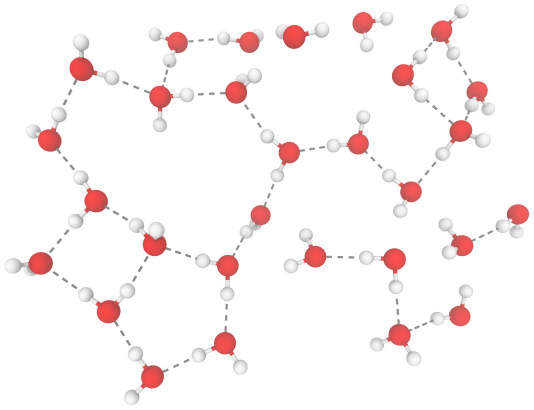


Figure 2: A snapshot of water arrangement in Avogadro. The dashed lines represent hydrogen bonding interactions.

is also helpful for students to develop skills of interaction with Graphical User Interface (GUI) of typical scientific applications. We apply Avogadro's molecular mechanics[29] capabilities to highlight intermolecular interactions and hydrogen bonding between water molecules. We also highlight Avogadro's application to analyzer stereochemistry and 3-D molecular structures. Students have informally reported after progressing to organic chemistry course that Avogadro was a very helpful tool for that course.

A signature activity that we perform with Avogadro is the study of arrangement and orientation of water molecules in a small cluster. The students follow along in this activity on their personal computers. We utilize the Auto Optimization feature of Avogadro for this exercise. This feature allows a continuous optimization of molecular geometry and arrangement using molecular mechanics. We start with 20-25 randomly placed water molecules on the screen. The classical mechanics force field MMFF94[30]is chosen to represent intermolecular interactions. This force field recognizes hydrogen bonding interactions and the resulting arrangement of water molecules in the cluster is a result of the hydrogen bonding propensity of these molecules. Students are asked to pick and drag molecules around the screen. The system responds instantaneously and rearranges other molecules in the vicinity. The program also displays the potential energy at each instant and students observe that potential energy fluctuations correspond to favorable or unfavorable local and global arrangements of water molecules. Specifically, the system of water molecules attempts to optimize the hydrogen bonding interactions. A snapshot of animation with hydrogen bonds is shown in Fig.2. An animation of this process is supplied in the Supporting Information.

The course utilizes Visual Molecular Dynamics (VMD)[28] to study three-dimensional structures of proteins and to introduce students to a tool widely used in computational chemistry and biology research laboratories. VMD is a powerful tool to visualize structural properties, perform molecular dynamics simulation setup and data analysis for the NAMD[31] simulation engine. VMD allows fine-grained control of structural representation and can also be used to visualize a simulation trajectory run in any of the popular molecular dynamics simulation engines like GROMACS[32],

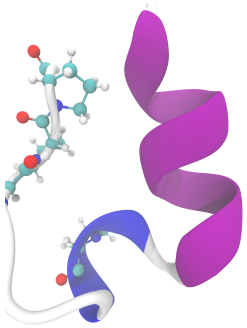


Figure 3: 1l2y mini-protein with proline residues highlighted in CPK representation. The secondary structure is emphasized by using the NewCartoon representation in VMD.

NAMD[31], LAMMPS[33], etc. This enables us to introduce the Protein Data Bank (PDB)[34] and to provide an overview of protein structure, stability, and function.

Students learn to browse the PDB resource, load molecules into VMD and extract information from the structures. They are trained to represent different molecular representations like CPK[35], Van der Waals, and secondary structure representations that are commonly seen in the literature and biology textbooks. For example, we study the structure of a small synthetic mini-protein trp-cage (PDB code: 1L2Y)[36]. The structure is shown in Fig.3 with the proline residues in CPK representation and the rest of the molecule in NewCartoon representation to highlight the α-helix motif. This also allows students to observe the general rule that proline residues do not participate in the formation of α-helix secondary structure motifs[37].

Similarly, we highlight structure of a β-barrel membrane protein (PDB code: 1G90)[38] located in the outer membrane of Gram-negative bacteria. This protein is chosen to highlight the β-sheet secondary structure element and the β-barrel morphology. Students learn to visualize the hydrogen bond connections between the sheets and to represent the structure in multiple ways. A representation of this protein is depicted in Fig.4. The red dashed lines represent hydrogen bonding interactions.

An important element of the course is to develop critical thinking skills of students and to encourage them to question their own beliefs about concepts learned in other courses. Hydrogen bonding is one of the most important concepts that students of Chemistry and Biology (the major population of our course at Wagner College) need to understand. A hydrogen bond is simply an electrostatic interaction between a hydrogen atom connected to a highly electronegative atom like fluorine, oxygen or nitrogen and another electronegative atom. Hydrogen bonds are commonly understood to be a key factor in protein structural stability, stability of DNA, RNA, and the unique properties of water. However, most students do not possess an intuitive understanding of how an interaction would be labeled as a hydrogen bond in the pictures seen in their biology or chemistry textbooks. Students use VMD options to display hydrogen bonds in protein structures. They also modify the default distance and angle cutoff to study the impact of such restrictions on the number of hydrogen bonds reported in the structure. The

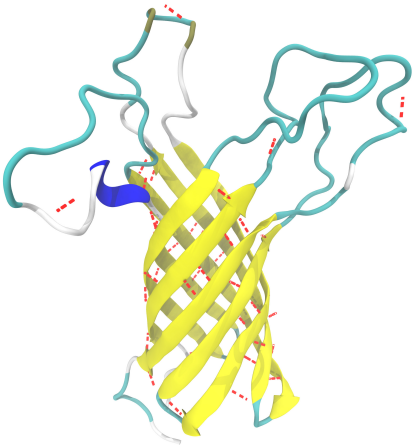


Figure 4: A β-barrel membrane protein. The red dashed lines represent hydrogen bonding interactions.

default distance criterion is 3.3 Angstrom and default angle criterion is that the Donor-Hydrogen-Acceptor angle is less than 20 degrees. Students discover that modifying these definitions leads to a change in the number of interactions flagged as hydrogen bonds and also study hydrogen bonds in different regions of proteins. This model of instruction enables students to appreciate the role of visualization and also the type of decisions that are implicit in visualization engines that study biomolecules. This exercise helps students to understand that pictures seen in books, textbooks, and research articles are the result of many decisions that must be understood from a computational and physical perspective.

The interplay of visualization with data analysis is highlighted in the next signature activity that we perform with the students. We apply VMD to visualize the trajectory of a short molecular dynamics simulation of liquid water and Mathematica to perform data analysis on observables recorded during the simulation. Such an activity would not be possible to perform with freshman level students outside of this course.

5. ANALYSIS OF MOLECULAR DYNAMICS SIMULATION

We perform the visualization and analysis of a short molecular dynamics simulation of liquid water at room temperature. We execute the simulation beforehand and project the simulation trajectory for visualization to the class. We then display hydrogen bonds during the simulation frames. Usually, a lively discussion follows about the behavior of liquid water during the simulation and how they could use VMD[28] to display the hydrogen bonds and use different representations of the simulation trajectory. At this point in the curriculum students realize that a molecule in bulk water attempts optimization of its hydrogen bonding network. The trajectory is processed using GROMACS[39] tools beforehand and students are presented with text files containing measurement of temperature, potential energy, kinetic energy, etc. during the simulation.

Students are provided a brief overview of molecular dy-

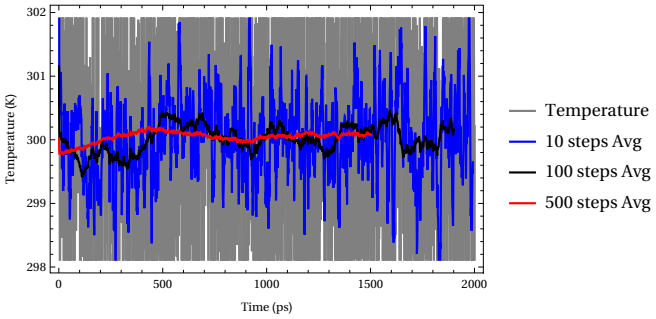


Figure 5: Moving averages to compute the average temperature during a molecular dynamics simulation. The fluctuations in the measurement are highly suppressed as the width of averaging window increases.

namics simulation and a brief explanation of the NPT ensemble[40] used to carry out the simulations. They calculate the number of water molecules and volume of simulation box to achieve appropriate density. This helps to build an appreciation of the scale of molecular simulations. A particularly illuminating aspect of this exercise is the concept of average value of a measurement over a time period. The simulation provides an output of temperature measurement and other physical observables at every picosecond during the simulation. Thus, for a 2 nanosecond simulation there are 2000 recorded temperature values. Students import the raw data into Mathematica and visualize the measurement. They notice that the temperature fluctuates and is not truly fixed at 300 K for the entire simulation. The students then perform moving averages to understand the temperature measurement. As they perform moving averages over measurement values, they realize that instantaneous measurements may differ significantly from the average. They perform a time average using a duration of 10 ps, 100 ps, and 500 ps each. They observe that with larger averaging duration the temperature measurement is very close to the preset 300 K for the simulation. Fig.5 depicts a plot that students construct using the raw data. This exercise also provides an opportunity to impart the skill of importing data from text files into Mathematica and data processing for desired analysis. Students report that this is usually their first experience analyzing large amounts of data and visualizing fluctuations in the measurement of a commonly experienced physical observable such as temperature. The exercise also involves analyzing potential and kinetic energies and the number of hydrogen bonds during the simulation.

6. SMALL PROJECT EXPERIENCE

Students perform a small research project during the last six weeks of the course. These are group projects that incorporate themes from the course. We reserve a portion of class time for students to pursue these projects and to get assistance from the instructor and other classmates. A few student projects are described below to provide an overview:

1. Mathemusica: Students explained the mechanism of sound, action of sound waves, and programmed popular music and classical music compositions using Wolfram programming language in Mathematica. This was a highly unusual and creative application of programming and computing.

2. Global warming trends: Students downloaded datasets of global surface and sea temperatures from NOAA to analyze trends and to highlight the measures of global warming.

3. Gallery of 3-D structures and functions: Students created 3-D structures of proteins, their genetic information, and their properties using VMD and Mathematica's biological data functionalities.

4. Gallery of atomic orbitals: Students plotted mathematical expressions for atomic orbitals to depict shapes of atomic orbitals and their properties.

These projects highlight the diversity of students and their interests in the course. They also demonstrate the diverse range of functions and databases that are built into Mathematica and of student efforts to apply their skills in a variety of domains.

7. STUDENT ASSESSMENT

This course has been taught twice at Wagner College to a total audience of 30 students with a distribution of students from all levels. Student response to the course has been overwhelmingly positive. The freshman population in Spring 2015 was 50% and in Spring 2016 was 43%. The strong interest from students at all levels is highly encouraging. Although, the course would be most beneficial if students take it as early as second semester of freshman year. Additionally, it also points to a pressing need for this course. Students in their junior and senior years of College have expressed amazement and disbelief about the accomplishments of computational science and its potential impact on their career paths. The course has a strong enrollment for the Spring 2017 semester with 19 students (11 freshmen) registered for the course.

The assessment was carried out using anonymous on-line survey. The average of student responses is reported along with standard deviation in parentheses. The responses have been merged for the first two iterations of the course with 30 respondents. The first set of questions had responses on scale from Strongly Agree (5) to Strongly Disagree (1). These questions have a broad focus and students seemed intent on applying the skills from this course to other courses.

1. I have acquired a better understanding of applications of computing to Science. **4.50** (0.51)

2. I plan to apply software and skills acquired from this course to other courses. **4.17** (0.81)

3. I have acquired skills that will help me in my major. **4.23** (0.77)

The second set of questions had responses on a scale used for Wagner College Chemistry graduate exit survey: A great deal (5), A lot (4), Some (3), A little (2), Not at all (1). These questions address distinct skills and tools practiced in the course. The responses are overwhelmingly positive and the course seems to be addressing a unique niche in the undergraduate liberal arts curriculum.

1. My skill in manipulation of large datasets has increased. **3.97** (0.81)

2. My skill in graphical analysis of data has increased. **4.07** (0.87)

3. My skill in setting up and solving numerical problems has increased. **3.9** (0.80)

4. My skill in visualization and analysis of mathematical functions has increased. **4.07** (0.91)

5. My skill in visualization of structure of biomolecules has increased. **4.13** (0.86)

6. My skill in molecular drawing and editing has increased. **4.23** (0.82)

The positive course evaluations and student experiences have inspired us to share this work with a larger audience.

8. CONCLUSION

We have created a course called, "Introduction to Scientific Computing" to introduce students to technical computing and functional programming at liberal arts Colleges. The course focus is to present computing and interaction with data in the framework of STEM disciplines. This approach fosters the integration of computing into the educational goals of students from all STEM branches and provides a set of tools that can be used throughout their undergraduate education and well into the professional work-place or graduate school. We think that the course utility is the highest for freshman students because of the high impact of these skills and tools on their undergraduate education. However, student response from all levels of student population has been highly encouraging. Students learn the skills of data analysis, data visualization, functional programming, molecular visualization, and molecular editing. Students also apply these skills collaboratively in small group based research projects. The diversity of projects and assignments carried out in the course highlight the broad applicability of the course to STEM education. Our hope is to transform the student mindset into accepting scientific computing as a skill that is as integral to the practice of STEM disciplines as their laboratory skills. We hope that colleagues at other institutions will consider creation of a similar course to better prepare our future generations.

9. SUPPORTING INFORMATION

The following files are provided as supporting information:

1. A notebook with code samples described in section 3 in Mathematica notebook format and PDF

2. Animation of the molecular rearrangement activity described in Section 4

3. A popular music song, "All of me" recreated by students in Mathematica (in CDF, PDF and Mathematica format). The document can be opened with the freely available Wolfram CDF player https://www.wolfram.com/cdf-player/.

4. A sample course syllabus and assignments are provided to aid in creation of a similar course.

10. ACKNOWLEDGMENTS

The author wishes to express his gratitude to Dr. Shawn Sendlinger for his encouragement during the Computational Chemistry for Chemistry educators workshop [41] to disseminate this work.

11. REFERENCES

[1] N Metropolis. The beginning of the Monte Carlo method. *Los Alamos Science*, 15:125–130, 1987.

[2] H. L. Anderson. Scientific uses of the MANIAC. *Journal of Statistical Physics*, 43(5-6):731–748, 1986.

[3] Theresa Julia Zielinski Mary L. Swift. What Chemists (Or Chemistry Students) Need to Know about Computing. *Journal of Science Education and Technology*, 4(2):171–179, 1995.

[4] J S Friedman and A A DiSessa. What students should know about technology: The case of scientific visualization. *Journal of Science Education and Technology*, 8(3):175–195, 1999.

[5] Terry Hinton. Computer assisted learning in physics. *Computers & Education*, 2(1-2):71–88, 1978.

[6] Richard Hooper. Computers in science teaching-An introduction. *Computers and Education*, 2(1-2):1–7, 1978.

[7] Vinesh Chandra and James J. Watters. Re-thinking physics teaching with web-based learning. *Computers and Education*, 58(1):631–640, 2012.

[8] Norman Chonacky and David Winch. Integrating computation into the undergraduate curriculum: A vision and guidelines for future developments. *American Journal of Physics*, 76(4):327–333, 2008.

[9] David M. Cook. Introducing Computational Tools in the Upper-Division Undergraduate Physics Curriculum. *Computers in Physics*, 4(2):197, 1990.

[10] K.R. Roos. An Incremental Approach to Computational Physics Education. *Computing in Science & Engineering*, 8(5):44–50, 2006.

[11] Ruxandra M. Serbanescu, Paul J. Kushner, and Sabine Stanley. Putting computation on a par with experiments and theory in the undergraduate physics curriculum. *American Journal of Physics*, 79(9):919, 2011.

[12] Jaime R. Taylor and B. Alex King. Using computational methods to reinvigorate an undergraduate physics curriculum. *Computing in Science and Engineering*, 8(5):38–43, 2006.

[13] Sharona T. Levy and Uri Wilensky. Crossing levels and representations: The connected chemistry (CC1) curriculum. *Journal of Science Education and Technology*, 18(3):224–242, 2009.

[14] Sharona T. Levy and Uri Wilensky. Students' learning with the connected chemistry (CC1) curriculum: Navigating the complexities of the particulate world. *Journal of Science Education and Technology*, 18(3):243–254, 2009.

[15] Ned H. Martin. Integration of Computational Chemistry into the Chemistry Curriculum. *Journal of Chemical Education*, 75(2):241, 1998.

[16] Richard A. Paselk and Robert W. Zoellner. Molecular Modeling and Computational Chemistry at Humboldt State University. *Journal of Chemical Education*,

79(10):1192, 2002.

[17] Mike Stieff and Uri Wilensky. Connected Chemistry: Incorporating Interactive Simulations into the Chemistry Classroom. *Journal of Science Education and Technology*, 12(3):285, 2003.

[18] David L. Cedeño, Marjorie A. Jones, Jon A. Friesen, Mark W. Wirtz, Luz Amalia Ríos, and Gonzalo Taborda Ocampo. Integrating Free Computer Software in Chemistry and Biochemistry Instruction: An International Collaboration. *Journal of Science Education and Technology*, 19(5):434–437, 2010.

[19] Marcos D. Caballero. Computation across the curriculum: What skills are needed? In *2015 Physics Education Research Conference Proceedings*, pages 79–82. American Association of Physics Teachers, 2015.

[20] National Academies. *Integrating Discovery-Based Research into the Undergraduate Curriculum*. National Academies Press, Washington, D.C., 2015.

[21] Loretta L. Jones, Kenneth D. Jordan, and Neil a. Stillings. Molecular visualization in chemistry education: the role of multidisciplinary collaboration. *Chemistry Education Research and Practice*, 6(3):136, 2005.

[22] Ionut. Danaila. *An introduction to scientific computing: twelve computational projects solved with MATLAB*. Springer, 2007.

[23] D Gruntz. *Symbolic Computation of Explicit Runge-Kutta Formulas*, pages 281–297. Springer, Berlin, Heidelberg, 2004.

[24] Alfio Quarteroni, Fausto Saleri, and Paola Gervasio. *Scientific Computing with MATLAB and Octave*, volume 2 of *Texts in Computational Science and Engineering*. Springer, Berlin, Heidelberg, 2014.

[25] Joseph L. Zachary. *Introduction to Scientific Programming: Computational Problem Solving Using Maple and C*. Springer New York, 1996.

[26] Stephen Wolfram. *An Elementary Introduction to the Wolfram Language*. Wolfram Media, 2015.

[27] Curtis D E Hanwell M.D. Lonie D.C., Vandermeersch, T., Zurek E., Hutchison, G. and Curtis D E Hanwell M.D. Lonie D.C., Vandermeersch, T., Zurek E., Hutchison, G. Avogadro: An Advanced Semantic Chemical Editor, Visualisation, and Analysis Platform. *Journal of Chemical Informatics*, 4.17:1–17, 2012.

[28] William Humphrey, Andrew Dalke, and Klaus. Schulten. VMD: Visual molecular dynamics. *Journal of Molecular Graphics*, 14(1):33–38, 1996.

[29] David C. Young. *Computational chemistry : A practical guide for applying techniques to real world problems*. Wiley, 2001.

[30] Thomas A. Halgren. Merck Molecular Force Field. *J. Comput. Chem.*, 17(5-6):490–519, 1996.

[31] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kalé, and Klaus Schulten. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry*, 26(16):1781–1802, 2005.

[32] H. J C Berendsen, D. van der Spoel, and R. van Drunen. GROMACS: A message-passing parallel molecular dynamics implementation. *Computer Physics Communications*, 91(1-3):43–56, 1995.

[33] Steve Plimpton. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *Journal of Computational Physics*, 117(1):1–19, 1995.

[34] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The Protein Data Bank. *Nucleic acids research*, 28(1):235–242, 2000.

[35] Robert B. Corey and Linus Pauling. Molecular models of amino acids, peptides, and proteins. *Review of Scientific Instruments*, 24(8):621–627, 1953.

[36] Jonathan W. Neidigh, R. Matthew Fesinmeyer, and Niels H. Andersen. Designing a 20-residue protein. *Nature Structural Biology*, 9(6):425–430, 2002.

[37] C N Pace and J M Scholtz. A helix propensity scale based on experimental studies of peptides and proteins. *Biophysical journal*, 75(1):422–427, 1998.

[38] Lukas K. Tamm, Heedeok Hong, and Binyong Liang. Folding and assembly of beta-barrel membrane proteins. *Biochimica et Biophysica Acta - Biomembranes*, 1666(1-2):250–263, 2004.

[39] David Van Der Spoel, Erik Lindahl, Berk Hess, Gerrit Groenhof, Alan E. Mark, and Herman J C Berendsen. GROMACS: Fast, flexible, and free. *Journal of Computational Chemistry*, 26(16):1701–1718, 2005.

[40] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*. Academic Press, San Diego, second edition, 2002.

[41] Shawn C. Sendlinger and Clyde R. Metz. Computational Chemistry for Chemistry Educators. *The Journal of Computational Science Education*, 1(1):28–32, 2010.

Authentic computer science undergraduate research experience through computational science and research ownership

Lior Shamir
Lawrence Technological University
21000 W Ten Mile Rd.
Southfield, MI, USA
lshamir@mtu.edu

ABSTRACT

Research experience has been identified as a high-impact intervention for increasing student engagement and retention in STEM. However, authentic undergraduate research leading to primary authorship peer-reviewed publications is a challenge due to the relatively short time the students work on their capstone projects, and the insufficient preparation of the students as researchers. The challenge is further magnified in the field of computer science, where the absence of "traditional" labs limits the opportunities of undergraduate students to participate in research. Here we present a novel approach to authentic computer science undergraduate research, based on interdisciplinary computational science and student ownership of their research projects. Instead of the traditional role of undergraduate research assistant, the students select their own research topic based on their personal interests, and with the assistance of a faculty complete all stages of their research project. The uniqueness of the approach is its ability to lead to scientific discoveries and peer-reviewed publications such that the primary author is the student, while allowing the student to experience the entire research process, from defining the research question through analysis of the experimental results. In three years the model led to a dramatic increase in the number of undergraduate students who publish primary-author peer-reviewed scientific papers. The intervention increased the number of peer-reviewed student-authored publications from none to a very high rate of about one third of the students, in many cases publishing in the top outlets in their field.

Categories and Subject Descriptors

K.3.2 [**Computers and education**]: Computer and Information Science Education

General Terms

Experimentation, Human Factors

1. INTRODUCTION

Research experience has been becoming increasingly important in undergraduate education, and a primary tool for attracting and retaining students in the Science, Technology, Engineering, and Mathematics (STEM) disciplines [32, 29]. One of the goals of undergraduate research experience is to improve student learning [18], as it has been shown that active learning is superior to "traditional" teaching methodologies [12], also leading to higher grades [20, 3]. Another goal is student preparation and ability to make connections among seemingly disparate pieces of information, evaluate evidence, and bring the requisite expertise to address complex issues [1].

In addition to enhancing the learning experience, research-based education has been proven to be a powerful intervention for engaging undergraduate students [37], and consequently retaining students in STEM [16, 20, 25, 9, 29]. The possibility of making a discovery and the participation in authentic STEM research has substantial impact on student engagement and learning compared to classical experiment or "cookbook" style laboratory exercises that reproduce known results [32, 29].

In particular, research-based education was found effective for attracting underrepresented minority students to STEM [3, 48]. Other proven interventions for underrepresented minorities in STEM include attending conferences, presenting at conferences, and faculty mentorship [48], which are also activities related to undergraduate research.

However, undergraduate research in computer science introduces several obstacles, making it more challenging than undergraduate research in many other STEM disciplines. Fields such as physics, chemistry, or biology offer a variety of hands-on opportunities for undergraduate research. Undergraduate students can join research labs and participate in experiments by executing protocols, preparing materials, or operating basic research equipment. Such research labs provide opportunities for undergraduate students to become familiar with the environment of a research lab. As a result, undergraduate students have noticeable presence in research labs in these disciplines, and are often authors on scientific peer-reviewed publications. Although the experiments are normally designed by more senior researchers, undergraduate students can take part in the research and benefit greatly from their presence at the lab and their work as part of a

research team in a discovery-driven environment.

Research in computer science, on the other hand, normally requires deep knowledge of the concepts being studied and familiarity with the state-of-the-art literature [30], making it more difficult for an undergraduate student to make a contribution.

Therefore, undergraduate research experience in computer science can be broadly divided into two primary categories: The first is student independent capstone projects, driven by student ideas and the personal interest of the student. Students working on these projects often receive the assistance of a faculty or industry supervisor, but the project is normally owned by the student, who leads it through all stages of the development [28]. The downside of these capstone projects is that they rarely lead to peer-reviewed scientific publications and authentic scientific discoveries.

The second category is student research assistantship [49, 5], in which the student joins a research project led by a faculty or another senior person, and assists them with the research. As a research assistant, the student is exposed to authentic research, but does not lead or own the research [19]. Another disadvantage of student research assistantship is that it is limited by available funding [31] and faculty attention [34], and therefore while some undergraduate students in computer science have the opportunity to join faculty or research labs, the majority of undergraduate students completing a four-year degree in computer science do not participate in authentic research or become authors on a peer-reviewed scientific paper. In some cases research opportunities are available for the most qualified and motivated students, and allowing all students to work on research requires a compromise on the research quality and expectations [4]. It should also be noted that undergraduate research experience in the form of research assistantship is far more common in research universities compared to other institution of higher education [44, 33], and therefore students in smaller universities have less opportunities experience research.

Here we describe a model of undergraduate research experience that combines student ownership with authentic research that leads to peer-reviewed publications such that the primary author is the student. The project is owned by the student and driven by the student's area of interest, while leading to authentic scientific discoveries and peer-reviewed publications. The model provides research opportunities to all undergraduate students, and significantly improved student engagement. It also dramatically increased the number of students publishing peer-reviewed papers from none to one third of the total number of graduating students.

2. STUDENT RESEARCH ASSISTANT COMPARED TO STUDENT RESEARCHER

The typical way by which an undergraduate student is exposed to authentic STEM research is through the role of student research assistant. As a research assistant, the student joins a faculty or lab, and performs tasks related to the research under the direction of the primary investigator or another senior member in the lab. Although there is clearly high educational value in being part of a research team, the research assistantship model is imperfect for providing research experience to large numbers of students as part of their curricula.

Firstly, due to the close supervision that mentoring an undergraduate student in computer science requires, the primary investigators are limited by the number of undergraduate students they can mentor in their lab, and therefore just few of the undergraduate students have the opportunity to participate in research. Less senior members such as PhD candidates also have commitments that limit their ability to mentor students, and being trainees themselves they lack the training and experience to effectively mentor students and lead them to scientific discoveries. Undergraduate students working in labs often receive stipend for their work, making the number of students also limited by the availability of funding. Because of the limited number of available research assistantship positions, these positions are sometimes competitive, and students are selected based on their academic achievements, making the research experience inaccessible to those who are not at the top of their class.

Another downside of the student research assistant model is that the students are required to join an existing research project designed and led by a faculty, and therefore do not select their research topic by themselves. That limitation is magnified in smaller institutions of higher education, where the number of research programs is limited and the student has even less research options to choose from. Working on a research program led by a faculty also diminishes the aspect of ownership of the research, which is an important element of the undergraduate research experience [29, 2]. Another disadvantage of the student research assistantship model is that the student often performs specific tasks defined by the primary investigator or other supervisors, and therefore does not earn hands-on experience in performing the entire research process, from the definition of the research problem to the analysis of the experimental results.

According to the undergraduate research experience model proposed in this paper, the undergraduate research is performed such that the student serves as the researcher, and the faculty assists the student and provides the required knowledge to successfully complete the project. The student selects the research topic of his or her interest, defines the scientific question, designs the research, performs the experiments, and then analyzes the experiential results, while the faculty mentor assists the student and provides the knowledge required for the completion of each step of the research. That experience exposes the student to all stages of the research, and embraces the idea of student ownership rather than assisting the research agenda of a more senior person. The ability of the student to work on a research topic of their choice helps to engage the student in the research, and also attracts students who would not otherwise work on a research project. It can also lead to peer-reviewed scientific papers on which the undergraduate student is the primary author.

Table 1 shows a summary of the differences between the role of a student research assistant and the proposed model of student researcher.

Table 1: Student research assistant vs. student researcher

	Student research assistant	Student researcher
Accessibility	Some (best) students	All students
Owner of the research	Faculty	Student
Selection of research topic	Faculty	Student
Student responsibilities	Defined tasks	Entire research project
Paper authorship	For some, as co-authors	Yes (if paper is published)
Role of faculty mentor	Direct the student	Assist the student

3. IMPLEMENTATION OF THE STUDENT RESEARCHER MODEL

In the proposed program, the student projects normally span over two semesters, during which they earn three credit hours in each semester. The two project courses are mandatory for all students, although students are not required to perform research, and are free to choose to work with an industry partner, develop a video game, or work on their own technological invention that is not necessarily considered research. As will be described later in this paper, the number of students choosing to work on research increased dramatically since the described undergraduate research experience model was implemented.

The students do not work on existing research programs, and therefore the number of students that can perform research is not limited by the number of research labs or open research assistantship positions. However, supervising and mentoring students is still a time-consuming task for the faculty. As will be described in Section 4, the experience of implementing the program showed that one faculty can mentor the research programs of about 10 students. Since the ratio between faculty and senior students is normally lower than 1:10, the implementation of the program is feasible even in smaller institutions of higher education, or colleges where substantial part of the teaching is performed by part-time faculty members.

One of the key features of the scheme is that the students can choose their own research topics, based on their interest. In the beginning of the semester the student meets with the faculty mentor to discuss the topics the student is interested in, and the student and faculty collaboratively design a research plan that is as close as possible to these fields of interest. That is done in the first two weeks of the project. Examples of research projects will be described in Section 4.

One of the important aspects of the research topics is computational science and interdisciplinarity [7]. Most Undergraduate students do not have deep knowledge in the different sub-fields of computer science, and therefore the requirement to expand the state-of-the-art in a core computer science sub-discipline might not be in agreement with the preparation of the student and their ability to perform that type of research. In other cases research in these sub-disciplines of computer science might not be aligned with the interests of the student. Fortunately, computer science has application to many other fields, and the increasing availability of scientific data in these disciplines allows effective research and substantial scientific discoveries through computational science. Biology has long been a field with strong ties to computer science in the form of bioinformatics and computational biology, but other fields such as astronomy and geoscience are also in the process of establishing strong links to computer science, leading to interdisciplinary sub-fields. Since many of the students who choose computer science are also interested in astronomy, computational astronomy is a field of study that can attract undergraduate students to research. Other fields that are of high student interest can be zoology, art, music, literature, and sport. Allowing the students to choose a topic of research in these disciplines can engage the student in research, and more importantly, can attract students who would not otherwise participate in authentic scientific research. Students can also choose research topics related to their culture or ethnicity, and express their identity through computing.

The availability of scientific data that the students can process is often a critical requirement for completing the research, and therefore the student and faculty mentor should verify that the data are publicly available or can be obtained within a reasonable period of time. That ensures that the beginning of the project is not delayed because the data are not available. The analysis of the data can be done by modifying existing open source data analysis tools and adjusting them to the specific needs of the data analysis project.

When the students are provided with the option to select the research topics regardless of the existing research programs available on their campus, it is expected that some students would choose to study topics that the faculty mentor is not familiar with. To satisfy the expectations of the student, the faculty mentor can learn the new field with the student, leading to the expansion of the research topics that the faculty can mentor. Therefore, it can be expected that the implementation of the program will lead to a gradual increase in the variety of research programs the students can choose from.

The downside of the student self-selection of the research projects is that students are geared towards their own interests so that each student tends to pick the project by the topic, and not necessarily by the other students that they want to work with. The priority of the research topic over the research team results in much less teams of students, and some students who work on research tend to work by themselves. In fact, just about 25% of the students who chose to work on research worked in teams. That can be very different from non-research projects such as video games, where the vast majority of the students worked in teams. In that case, the students first select the team they want to work with, and only then they discuss the specific video game they wish to develop.

Two semesters is a short time for completing a research project and submitting a scientific paper. Unlike graduate students who spend most of their time working on their thesis, undergraduates spend substantial part of their time taking courses, and sometimes have other commitments such as work, and therefore their progress is limited by the time they can commit to their research. For that reason it is important to meet the student at least once a week, as well as maintaining a channel of communication by email or other technologies (e.g., Skype) so that the student gets the support as soon as they need it, without delaying their progress. It is also the responsibility of the faculty to monitor the progress of the students and make sure the students use their resources efficiently, and make the correct implementation decisions during the semester. The level of student-faculty interaction therefore goes beyond the regular office hours. From the faculty perspective it means spending longer hours on campus meeting with students, and becoming available to the students at almost any time.

The open communication with the faculty mentor is also important for supporting the student. Undergraduate students who have no experience in research are naturally concerned about not being able to complete their project, and consequently failing the course. The communication with the faculty and the availability of the faculty assures the students that they receive the support they need to complete the project, and that the faculty is aware of their efforts and the solutions they develop as they attempt to solve the scientific question at hand. Working as a team with their faculty mentor can drastically reduce the anxiety, and make the research project a positive experience. For the same reason it is important to make a clear statement that the project is not graded by the ability of the student to provide a solution to the scientific problem by the end of the semester, but by the way the student studies the topic and approaches the scientific question in attempt to solve it. Such statement reduces the natural anxiety of the students who are asked to perform tasks they never attempted before, and might be worried about the academic consequences in case the problem they chose to work on does not cooperate with their solutions. Another reason is to encourage the student to choose more challenging research projects, and not necessarily the simpler problems that they know they can solve and secure a passing grade.

4. RESULTS

The program described above was implemented in our department, which has ∼120 undergraduate students, and about 25-30 senior students. Before the implementation of the program the undergraduate students were rarely involved in research. In three years the program increased the number of students working on research from practically none to about 40% of the students. Students in the video game concentration have to develop a video game as part of their degree requirements, and cannot choose to work on a research project (for academic credits), so when excluding these students about half of the undergraduate students chose to work on a research project, while the others preferred to develop creative computer applications or work on industry-oriented projects. Table 2 shows the distribution of the type of projects selected by the students. As the table shows, the number of students who chose to work on research increased dramatically when the program was first implemented in 2011, allowing the students to work on their own research projects. The drop in the number of students in 2014 can be associated with the financial crisis that hit the Detroit area in 2009-2011, affecting the number of senior students about four years later.

Table 2: Student selection of capstone projects. The new approach to student capstone projects was started in 2011.

Year	Video games	Research	Other
2010	14	1	38
2011	16	22	25
2012	12	24	21
2013	11	21	19
2014	7	12	15
2015	6	18	14
2016	8	21	16

The increase in the number of students who participate in authentic research is also reflected by the number of students who become authors on peer-reviewed scientific papers. Before the implementation of the program none of the undergraduate students submitted papers to peer-reviewed journals or conferences. After the first year of the program two papers were published [47, 42], three papers in 2013 [36, 13, 39], seven in 2014 [14, 43, 17, 11, 26, 46, 21], six in 2015 [8, 24, 35, 22, 15, 27], and seven in 2016 [23, 38, 41, 45, 40, 50, 10].

The number of submitted papers peaked in the spring semester of 2013, where out of 21 students eight papers were submitted with 11 student authors, more than half of the students in that semester. These papers were published during 2013 through 2015.

The student engagement is also reflected by the number of students who keep working on their project and meet regularly with the faculty after they graduate. A student who continues to work on their research makes an indirect statement about the level of engagement and commitment to their scholarly work. About one quarter of the students who work on research projects continue to come to campus and meet with their mentor faculty for at least one semester after they graduate.

One of the surprising observations of the experiment is the student response to publishing scientific papers. The experiment revealed a very positive attitude of undergraduate students toward publishing papers, and expressed willingness to put efforts in the preparation of papers also after they graduate. In some cases the students return to campus to work on revising their papers, as the report of the reviewers and editors is normally received after the student graduates. Another expression of student enthusiasm about publishing papers is explicit statements made by the student expressing their expectation to publish a paper, even before they chose their research topic. In addition to student pride and motivation, the communication of the student research results through peer-reviewed papers helps to defend the overall quality and impact of the student research, and helps to justify the engagement of students in research.

An important aspect of the results observed in the first four years of the program is the impact of research experience on underrepresented minorities. Research experience has been demonstrated as one of the most powerful tools to attract and retain underrepresented minority groups in STEM [6, 48, 3], and the findings of this experiment are in strong agreement with these reports. Our department has a relatively small population of underrepresented minority students, but the effect was measured by the participation of women in research. Like many other computer science programs, our student population is dominated by male students. In 2013 the department had 114 male students and just 12 female students. However, the proportion of female students is completely different when considering the student-authored papers published so far. Out of 27 student-authored papers published so far, about 25% were authored by female students [47, 42, 26, 43, 24, 50, 10], much higher than the proportion of female student in the entire student population, which is ∼9%. That proportion remains consistent also among the papers that are currently under review. It should be noted that the sample size is still too small to be statistically significant.

The presence of female students in research is also felt by news items in the mainstream media. Since 2011 we had four student research projects [42, 14, 43, 23] featured on the premier national and international popular press (e.g., NBC News, Fox News, CBS, NPR, Discovery Channel, Scientific American, The Atlantic, etc'), including press interviews with the students about their research. The exposure through the mainstream media elevated the research experience to a new level of pride and excitement for the students, their friends, and their families. Of these four students two were women. Although it is clear that the test group is far too small for making a conclusion, it is a result of female student engagement in the research, as the research topic is strongly related to the student's personal interests. Also, out of the 27 students that were authors on scientific papers, 10 continued to graduate school.

As mentioned in Section 3, the research project of each student starts with a meeting in which the student describes their interests to the faculty mentor, and then the topic of the research is defined by both the student and the faculty. Since different students have different topics of interest, supporting the interest of the students requires a broad range of research topics on which the students can perform their studies, and new topics and disciplines are added every year based on the student request. The first interdisciplinary program was bioinformatics and medical informatics [36, 26, 24, 41], and based on the request of students was enhanced with astroinformatics [17, 11, 21, 35, 22, 23]. Other programs that followed were Zoology [47, 43], art [42], music [14, 15, 13], literature [39], sports [46, 45, 50], and human aspects of computing [8, 38]. All of these programs were added based on student interest and their expressed desire to perform research in these disciplines. One example is a student who was also a volunteer in a bird preservation society, and chose a project which applied computational science to study birds behavior and preservation [47]. A student who is an amateur artist chose to apply computational science to analyze art [42], a semi-professional rock musician used computational methods to analyze music [14], a football fan applied computational science to the analysis of football coach decisions [46], and a soccer fan chose to analyze the salaries of soccer players [50].

4.1 Transferability of the model

Working with every student and the need to learn new disciplines based on the student interest requires substantial efforts from the faculty. Therefore, the return should be weighted against the time investment to make the model transferable. Incentives for faculty to implement the model include publications and opportunities for external funding. Additionally, the opportunity to learn new disciplines can also have a certain value, and the engagement in research through education can be appealing to faculty at institutions that mostly focus on education.

This project was started without institutional funding, but led to several external grants directly or indirectly related to the work. For instance, an NSF grant (CNS-1157162) to fund a computing facility was based primarily on computational research performed by students. Another NSF grant (IIS-1546079) was received with substantial help from the work of students who were interested in computational astronomy. A grant from the AAC&U was given partially for the work on computational analysis of art [42], and funded the implementation of this model in art history courses.

Peer-reviewed publications can also be an incentive for the faculty, as publications in competitive outlets often add to the reputation of their authors. However, the diverse nature of the papers and outlets does not necessarily lead to a solid career development path, and therefore career development in the sense of peer-reviewed publications is not a primary incentive.

5. CONCLUSION

Research experience is an effective intervention for attracting and retaining undergraduate students in STEM, and develop creativity and critical thinking skills [1]. Due to the deep knowledge required to perform research in computer science, as well as the limited open positions for student research assistants in labs, most computer science undergraduate students do not participate in authentic research before they graduate, or become authors on scientific papers.

Here we propose a model of interdisciplinary computer science undergraduate research that can provide research opportunities for undergraduate students. The model is based on student selection of their research topic and student ownership of the research, while the faculty assists the students to perform all stages of the research project, from the definition of the scientific question to the analysis of the experimental results. That model is different from the role of the student research assistant, in which the student joins an existing research program and follows the directions of the faculty supervisor. The primary advantages of the proposed model is its accessibility to all students, its ability to engage students by ownership of their research, and expose the students to hands-on experience in all stages of a research project.

The results show dramatic increase in the number of students participating in research, and consequently the num-

ber of peer-reviewed papers authored by students. These results are achieved without the need for investment in more research labs or research assistantship positions, and without additional funding to support student stipends. However, the implementation of the program requires intensive interaction and frequent meetings with the student, and availability of the faculty far beyond the regular office hours.

An important requirement for successful implementation of the program is a broad range of interdisciplinary research topics that can engage and motivate students, and attract computer science students who would otherwise preferred to work on other projects that are not necessarily research. These research programs are different from the "establishment"' research programs and topics normally expected from computer science faculty members. Therefore, the development of such programs also requires redefinition of the expected research achievements for the purpose of career decisions such promotion and tenure. While normally decisions regarding promotion or tenure are based on teaching and scholarship achievements, the model proposed here combines the two and makes it difficult to make a clear line that separates between them. Therefore, the "traditional" teaching-research-service scheme of faculty assessment for tenure and promotion might need to be adjusted to a model that has substantial overlap and strong link between teaching and research responsibilities, but also requires a different research agenda from the faculty who is interested in the implementation of such program.

6. ACKNOWLEDGMENTS

This work is supported in part by the TIDES program of the AAC&U, with financial support from the Helmsley foundation. It was also supported by NSF grants IIS-1546079 and CNS-1157162.

7. REFERENCES

[1] American Association for the Advancement of Science. Vision and change - a call to action, 2009.

[2] L. C. Auchincloss, S. L. Laursen, J. L. Branchaw, K. Eagan, M. Graham, D. I. Hanauer, G. Lawrie, C. M. McLinn, N. Pelaez, S. Rowland, et al. Assessment of course-based undergraduate research experiences: a meeting report. *CBE-Life Sciences Education*, 13(1):29–40, 2014.

[3] A. E. Barlow and M. Villarejo. Making a difference for minorities: Evaluation of an educational enrichment program. *Journal of Research in Science Teaching*, 41(9):861–881, 2004.

[4] M. Beckerleg and J. Collins. Producing research from undergraduate projects. In *Proceedings of the 18th Conference of the Australian Association for Engineering Education*, pages 9–13, 2007.

[5] L. A. Beninson, J. Koski, E. Villa, R. Faram, and S. E. OâĂŹConnor. Evaluation of the research experiences for undergraduates (reu) sites program. *CUR Quarterly*, 32(1):43–48, 2011.

[6] J. A. Bianchini. Expanding underrepresented minority participation: America's science and technology talent at the crossroads. *Science Education*, 97(1):163–166, 2013.

[7] J. Blackmer. The gesture of thinking: Collaborative models for undergraduate research in the arts and humanities. plenary presentation at the 2008 cur national conference. *CUR Quarterly*, 29(2):8–12, 2008.

[8] B. Bock and L. Shamir. Assessing the efficacy of benchmarks for automatic speech accent recognition. In *8th International Conference on Mobile Multimedia Communications*, 2015.

[9] J. M. Braxton, A. S. Hirschy, and S. A. McClendon. *Understanding and Reducing College Student Departure: ASHE-ERIC Higher Education Report*, volume 30. John Wiley & Sons, 2011.

[10] A. Burcoff and L. Shamir. Computer analysis of pablo picassoâĂŹs artistic style. *International Journal of Art, Culture and Design Technologies*.

[11] L. Dojcsak and L. Shamir. Quantitative analysis of spirality in elliptical galaxies. *New Astronomy*, 28:1–8, 2014.

[12] S. Freeman, S. L. Eddy, M. McDonough, M. K. Smith, N. Okoroafor, H. Jordt, and M. P. Wenderoth. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, page 201319030, 2014.

[13] J. George and L. Shamir. Computer-based approaches to music research. In *Network Detroit: Theory and Practice*, 2013.

[14] J. George and L. Shamir. Computer analysis of similarities between albums in popular music. *Pattern Recognition Letters*, 45:78–84, 2014.

[15] J. George and L. Shamir. Unsupervised analysis of similarities between musicians and musical genres using spectrograms. *Artificial Intelligence Research*, 4(2):61, 2015.

[16] W. v. Hippel, J. S. Lerner, S. R. Gregerman, B. A. Nagda, and J. Jonides. Undergraduate student-faculty research partnerships affect student retention. *The Review of Higher Education*, 22(1):55–72, 1998.

[17] C. Hoehn and L. Shamir. Characteristics of clockwise and counterclockwise spiral galaxies. *Astronomische Nachrichten*, 335(2):189–192, 2014.

[18] A.-B. Hunter, S. L. Laursen, and E. Seymour. Becoming a scientist: The role of undergraduate research in students' cognitive, personal, and professional development. *Science Education*, 91(1):36–74, 2007.

[19] M. Jonas. Capstone experience: lessons from an undergraduate research group in speech at unh manchester. In *Proceedings of the 2011 conference on Information technology education*, pages 275–280. ACM, 2011.

[20] D. H. Kinkel and S. E. Henke. Impact of undergraduate research on academic performance, educational planning, and career development. *Journal of Natural Resources & Life Sciences Education*, 35(1):194–201, 2006.

[21] E. Kuminski, J. George, J. Wallin, and L. Shamir. Combining human and machine learning for morphological analysis of galaxy images. *Publications of the Astronomical Society of the Pacific*, 126(944):959–967, 2014.

[22] E. Kuminski and L. Shamir. Computer analysis of digital sky surveys using citizen science and manual

classification. In *American Astronomical Society Meeting*, volume 225, 2015.

[23] E. Kuminski and L. Shamir. A computer-generated visual morphology catalog of 3,000,000 sdss galaxies. *The Astrophysical Journal Supplement Series*, 223(2):20, 2016.

[24] E. Lixie, J. Edgeworth, and L. Shamir. Comprehensive analysis of large sets of age-related physiological indicators reveals rapid aging around the age of 55 years. *Gerontology*, 61(6):526–533, 2015.

[25] D. Lopatto. Undergraduate research experiences support science career decisions and active learning. *CBE-Life Sciences Education*, 6(4):297–306, 2007.

[26] S. Manning and L. Shamir. Chloe: A software tool for automatic novelty detection in microscopy image datasets. *Journal of Open Research Software*, 2(1):e25, 2014.

[27] I. Model and L. Shamir. Comparison of data set bias in object recognition benchmarks. *IEEE Access*, 3:1953–1962, 2015.

[28] B. Olsson, M. Berndtsson, B. Lundell, and J. Hansson. Running research-oriented final year projects for cs and is students. *ACM SIGCSE Bulletin*, 35(1):79–83, 2003.

[29] PCAST. Report to the president - engage to excel: Producing one million additional college graduates with degrees in science, technology, engineering, and mathematics, 2012.

[30] J. A. Polack-Wahl and K. Anewalt. Learning strategies and undergraduate research. In *ACM SIGCSE Bulletin*, volume 38, pages 209–213. ACM, 2006.

[31] M. Ramirez, J. McNicholas, B. Gilbert, J. Saez, and M. Siniawski. Creative funding strategies for undergraduate research at a primarily undergraduate liberal arts institution. *CUR QUARTERLY*, 36(2):5–8, 2015.

[32] S. H. Russell, M. P. Hancock, and J. McCullough. Benefits of undergraduate research experiences. *Science*, 316(5824):548–549, 2007.

[33] S. H. Russell, M. P. Hancock, J. McCullough, J. D. Roessner, and C. Storey. Evaluation of nsf support for undergraduate research opportunities: Survey of stem graduates. *Evaluation*, 2005.

[34] A. Schmolitzky and T. Schümmer. Patterns for supervising thesis projects. In *EuroPLoP*, 2008.

[35] A. Schutter and L. Shamir. Galaxy morphology-an unsupervised machine learning approach. *Astronomy and Computing*, 12:60–66, 2015.

[36] E. Schwartz and L. Shamir. Correlation between brain mri and continuous physiological and environmental traits using 2d global descriptors and multi-order image transforms. *Journal of Medical Imaging and Health Informatics*, 3(1):12–16, 2013.

[37] E. Seymour, A.-B. Hunter, S. L. Laursen, and T. DeAntoni. Establishing the benefits of research experiences for undergraduates in the sciences: First findings from a three-year study. *Science Education*, 88(4):493–534, 2004.

[38] L. Shamir, D. Diamond, and J. Wallin. Leveraging pattern recognition consistency estimation for crowdsourcing data analysis. *IEEE Transactions on Human-Machine Systems*, 126(944):959–967, 2015.

[39] L. Shamir and C. R. Everett, M. Computationally classifying literature with frequency analysis and pattern recognition. In *Digital Humanities Theory and Practice*. Network Detroit, 2013.

[40] L. Shamir and E. Kuminski. Image-based query-by-example for big databases of galaxy images. In *American Astronomical Society Meeting Abstracts*, volume 229, 2017.

[41] L. Shamir and J. Long. Quantitative machine learning analysis of brain mri morphology throughout aging. *Current aging science*, 4(9):1–7, 2016.

[42] L. Shamir and J. A. Tarakhovsky. Computer analysis of art. *ACM Journal on Computing and Cultural Heritage*, 5(2):7, 2012.

[43] L. Shamir, C. Yerby, R. Simpson, A. M. von Benda-Beckmann, P. Tyack, F. Samarra, P. Miller, and J. Wallin. Classification of large acoustic datasets using machine learning and crowdsourcing: Application to whale calls. *The Journal of the Acoustical Society of America*, 135(2):953–962, 2014.

[44] R. D. Slocum and J. D. Scholl. Nsf support of research at primarily undergraduate-institutions (puis). *Council on Undergraduate Research Quarterly*, 34(1):31–40, 2013.

[45] J. T. Soares and L. Shamir. Quantitative analysis of penalty kicks and yellow card referee decisions in soccer. *American Journal of Sports Science*, 4:84–89, 2016.

[46] R. Strange and L. Shamir. Prediction of football plays using pattern recognition. *International Journal of Computer Science in Sport*, 3(1):12–16, 2014.

[47] S. Svatora and L. Shamir. Improving eastern bluebird nest box performance using computer analysis of satellite images. *Computational Ecology & Software*, 2(2), 2012.

[48] L. Tsui. Effective strategies to increase diversity in stem fields: A review of the research literature. *The Journal of Negro Education*, pages 555–581, 2007.

[49] K. Ward. Research with undergraduates: a survey of best practices. *Journal of Computing Sciences in Colleges*, 21(1):169–176, 2005.

[50] L. Yaldo and L. Shamir. Computational estimation of football player wages. *International Journal of Computer Science in Sport*, In Press.

Energy-Efficient Virtual Screening with ARM-CPU-Based Computers

Olivia Alford[1] and David Toth[2]
Centre College
600 West Walnut Street
Danville, KY 40422
olivia.alford@centre.edu, david.toth@centre.edu

ABSTRACT
We attempted to find a more sustainable solution for performing virtual screening with AutoDock Vina which uses less electricity than computers using typical x64 CPUs. We tested a cluster of ODROID-XU3 Lite computers with ARM CPUs and compared its performance to a server with x64 CPUs. In order to be a viable solution, our cluster needed to perform the screen without sacrificing speed or increasing hardware costs. The cluster completed the virtual screen in a little less time than our comparison server while using just over half the electricity that the server used. Additionally, the hardware for the cluster cost about 38% less than the server, making it a viable solution.

CCS Concepts
• Applied computing~Chemistry.

Keywords
Electricity consumption; ARM CPUs; virtual screening; AutoDock Vina.

1. INTRODUCTION
The technique of virtual screening is used in the drug discovery process to reduce the time required to discover new drugs and the costs associated with that task [1]. Virtual screening is the process of using computers to simulate how well a molecule will bind to a protein or another target, using a molecular docking program, such as AutoDock Vina [2, 3]. Using virtual screening, millions of molecules can be "tested" by computers in a relatively short time, eliminating most of them from the pool of potential cures for a particular disease. This reduces the number of molecules to test in a wet lab to a very small number and thereby reduces costs of drug discovery significantly [1]. Due to the importance of virtual screening in the drug discovery process, it is critical to complete virtual screens quickly, and thus, large compute clusters and supercomputers are often used to conduct virtual screens [4].

For years, the focus of supercomputers was on increasing their computational power without concern about other aspects of supercomputing, such as electricity consumption [5]. However, the electricity consumption of supercomputers has become a large concern over the past years [5, 6]. This concern has led to GPUs and coprocessors being used in supercomputers to achieve more processing power while using less electricity [7, 8]. Additionally, the Green500 list was created to rank supercomputers in terms of energy efficiency [9].

DOI: https://doi.org/10.22369/issn.2153-4136/8/2/3

2. RELATED WORK
The Mont-Blanc project is exploring the potential for Advanced RISC Machines (ARM) CPUs to be used to build next-generation supercomputers [10, 11, 12]. Advantages of ARM CPUs include lower electricity consumption and cost. However, ARM CPUs are slower than x64 CPUs. Thus, the question of whether supercomputers and clusters built from x64 CPUs or ARM CPUs can provide a lower electricity consuming method of performing the same scientific computations in the same amount of time for similar cost is important. The Mont-Blanc project has built a prototype with system on chip (SoC) computers and compared its performance to the MareNostrum III supercomputer's performance. The Mont-Blanc prototype was slower than MareNostrum, but in some cases, more energy efficient and the prototype showed potential [11].

Toth et. al. conducted performance measurements of the programs AutoDock Vina and Dock6 on various computer-on-board products with ARM CPUs [13]. The measurements were compared to two computers with x64 CPUs. In this work, one of the computers with an ARM CPU outperformed both systems with x64 CPUs, consuming less electricity for a given task. That computer also was predicted to be able to complete the same task in the same amount of time using hardware costing less money. However, the work had two shortcomings. The first issue was that instead of conducting a full screen, the number of compounds that could be screened in 24 hours was measured.

To remove the variability of time to screen compounds, which could have led to unfair results, a single compound was screened repeatedly for 24 hours. The second issue was that the performance measurements were for a single device, which ignored the extra costs and electricity consumption of a cluster and the potential issues that would occur on a cluster, rather than a single system. These issues include slowdown from network communication between nodes and using a shared file system using the network file system (NFS). It also ignored the cost and electricity consumption of a dedicated controller node for the cluster which doesn't perform any work for the virtual screen, but just assigns tasks to each worked node and is responsible for managing the NFS shared folder.

[1] Undergraduate Student

[2] Corresponding Author

Work by Keipert et. al. compared the performance of the computational chemistry application GAMESS on x86 and 32-bit and 64-bit ARM CPUs [14, 15]. They found that the 32-bit ARM CPUs were more energy efficient than the x86 CPUs for completing tasks.

3. METHODS

To compare the two types of systems, we ran the same virtual screen on each type of system and recorded both the electricity and time required to conduct the screen.

3.1 Hardware

Our comparison systems were an x64 server and a cluster of ODROID-XU3 Lite SoC computers with ARM processors. The server contained four AMD Opteron 6378 processors. Each processor has 16 2.4 GHz x64 CPU cores for a total of 64 CPU cores. The server also had 64 GB of RAM, two 500 GB hard disks, and a 1000-watt power supply. The ODROID-XU3 Lite computers each have a Samsung Exynos5422 processor, which contains two quad-core processors (a CortexTM-A15 1.8 GHz quad-core processor and a CortexTM-A7 1.3 GHz quad-core processor) [16]. The ODROID-XU3 Lite computers have 10/100 Mbps Ethernet and 2 GB of RAM and are powered by a 5V4A power supply. The ODROID computers do not have persistent storage built in, and thus we used SanDisk 16 GB class 10 microSD cards for their storage. In addition, our cluster consists of a 48-port 10/100 Mbps Ethernet switch and Ethernet cables, a 240 GB SSD connected by USB 3 and an externally powered SSD enclosure, 5 power strips, and 280 1" metal standoffs. The ODORID cluster is shown in Figure 1. The cost breakdown of the cluster is shown in Table 1. To measure the electricity consumed by the virtual screens, we used P3 International P4400 Kill a Watt Electricity Usage.

3.2 Software

The server ran Ubuntu Linux 14.04.5 LTS and the computers in the cluster ran Ubuntu Linux 14.04.1 LTS. On all the computers, the graphical user interface was disabled. For the server, we ran one task per CPU core using gnu parallel [17]. On the cluster, we ran the SLURM clustering software to allow us to submit jobs so each CPU core in the cluster was always processing a molecule [18]. We used a slightly customized version of AutoDock Vina that outputs only the best score, rather than all the data that Vina outputs by default, to minimize file I/O [3]. That version is what we use to conduct virtual screens on supercomputers, servers, and clusters to improve the performance and allow for faster processing of the results of virtual screens. For the trials, we screened the full_nci_ALL_TAUTOMERS_2011 compound library from the ZINC Database, which contains 316,179 molecules [19].

4. RESULTS

We compared the data from running the virtual screen on the server with 64 x64 CPU cores to the data from running the virtual screen on the cluster of ODROID computers with ARM CPUs. The results of the virtual screens on the two platforms were identical. We compared the electricity consumed to complete the virtual screen, the time required to complete the virtual screen, and the cost of the hardware of both options. The results are summarized in Table 2. The ODROID cluster was the better solution in all three categories we measured. The electricity usage of both systems is shown in Figure 2. The cluster used only 51.8% of the electricity the server used to conduct the virtual screen. The cluster was able to complete the virtual screen in less time than the server, requiring only 94.3% of the time that the server required. The time required by each system to complete the virtual screen is shown in Figure 3. The cost of each system is shown in Figure 4. The cluster only cost 61.29% of the price of the server. Show cost parts in table here, too, to show totals.

Table 1 - Cluster Costs

Item	Cost/Unit	Units	Cost
ODROID-XU3 Lite	$96.80	25	$2,420.00
10/100 Mbps Ethernet cable	$0.91	25	$22.75
240 GB SSD	$79.99	1	$79.99
SSD enclosure	$39.99	1	$39.99
Power strips	$24.99	5	$124.95
280 metal standoffs	$114.25	1	$114.25
16 GB class 10 microSD card	$8.25	25	$206.25
10/100 Mbps Ethernet switch	$139.99	1	$139.99
Total cluster cost			$3,148.17

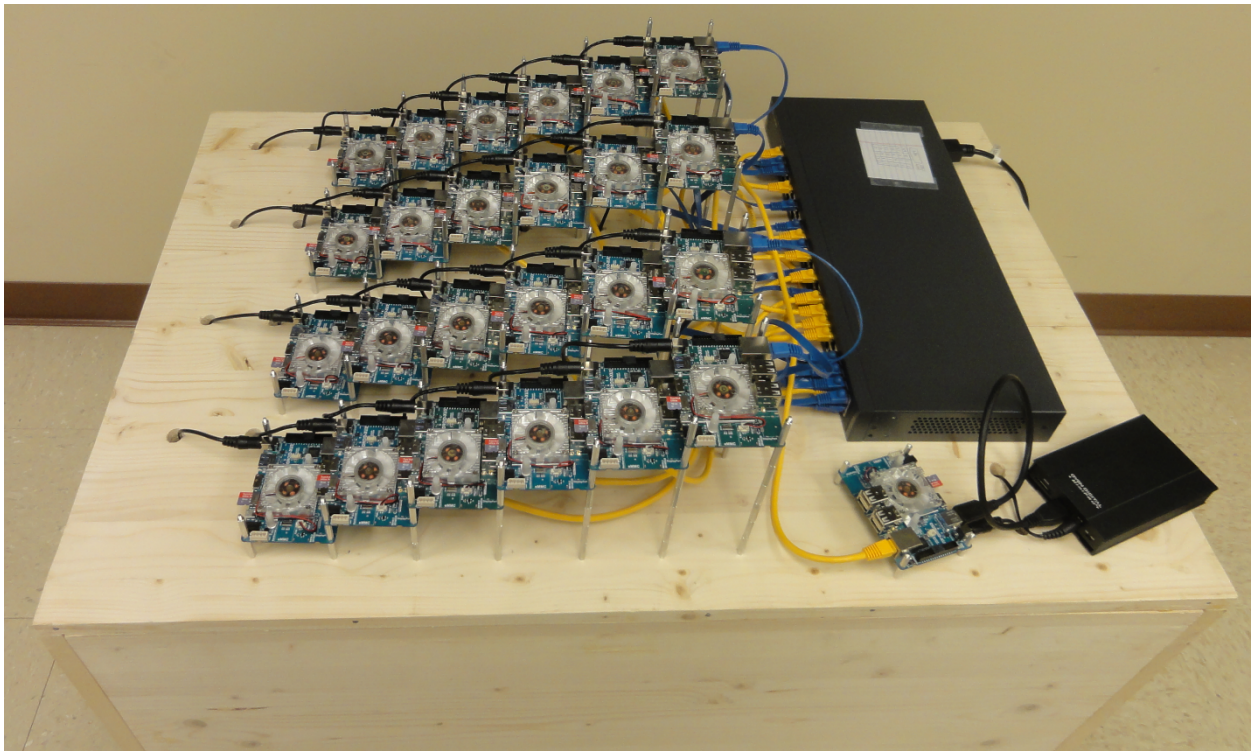


Figure 1 - The ODROID Cluster

Table 2 - Summary of Results

	Server	ODROID Cluster	Cluster's Resource Usage As Percent of Server's Resource Usage
Electricity Consumed (KWh)	86.10	44.56	51.8%
Time to Conduct Virtual Screen (sec)	513,364	484,356	94.3%
Hardware Cost	$5136.93	$3101.92	61.29%

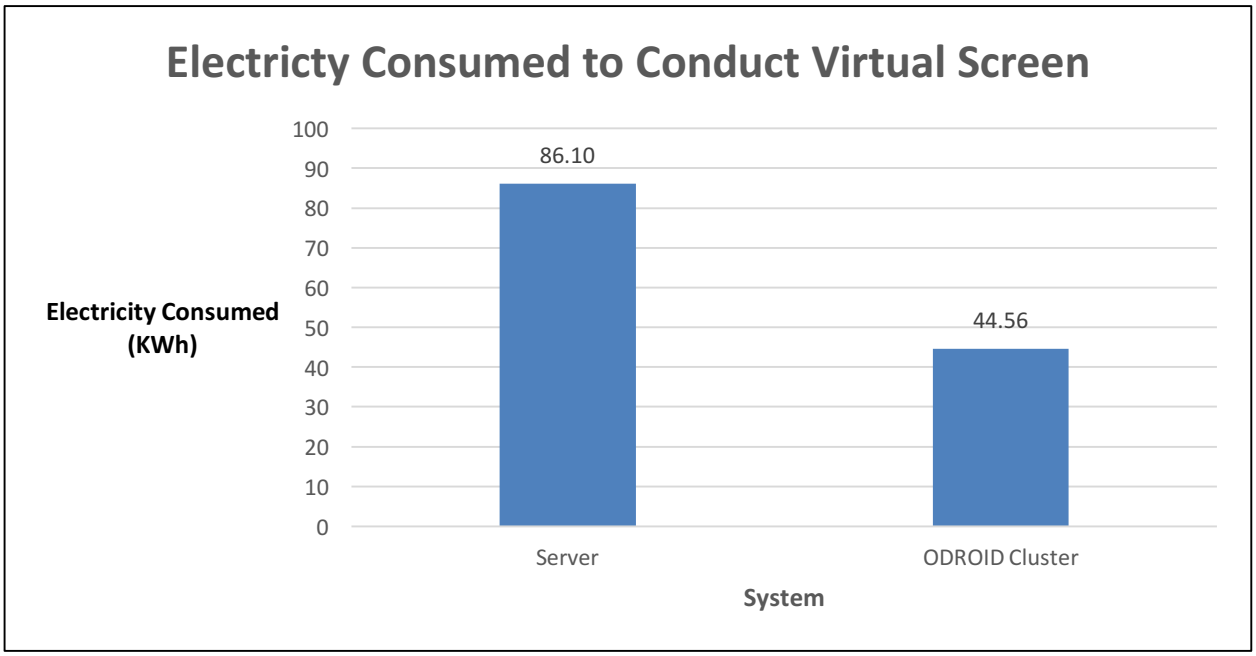


Figure 2 - Electricity Consumed by the Virtual Screen

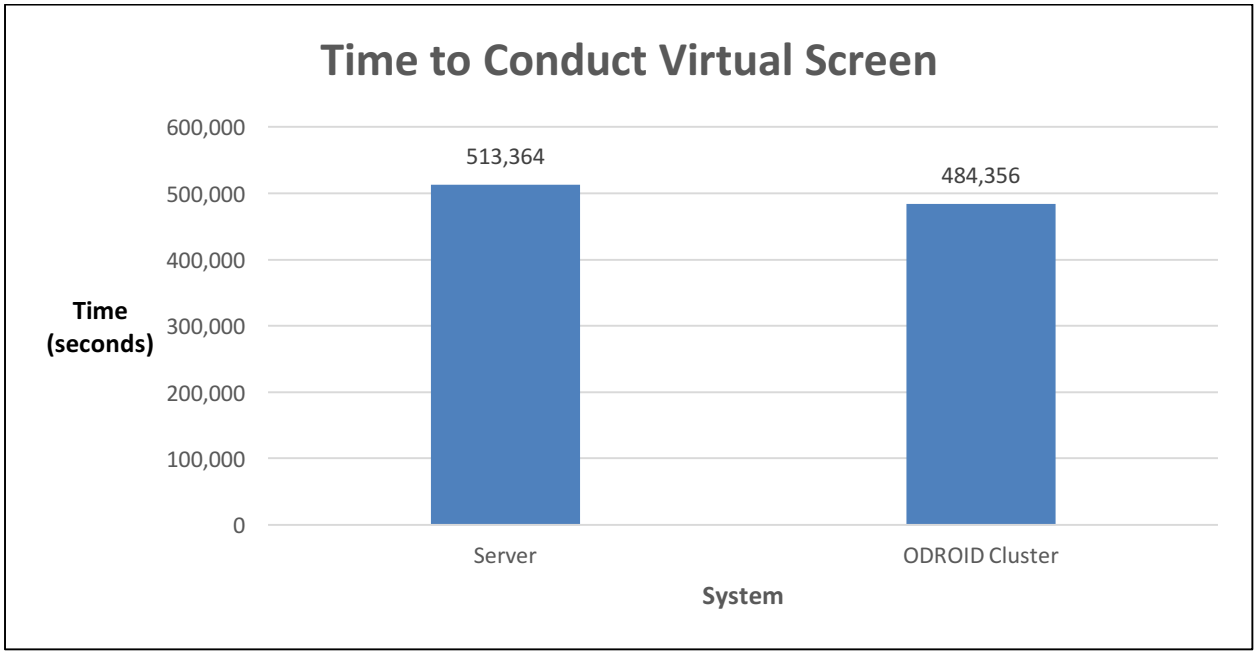


Figure 3 - Time Required to Conduct the Virtual Screen

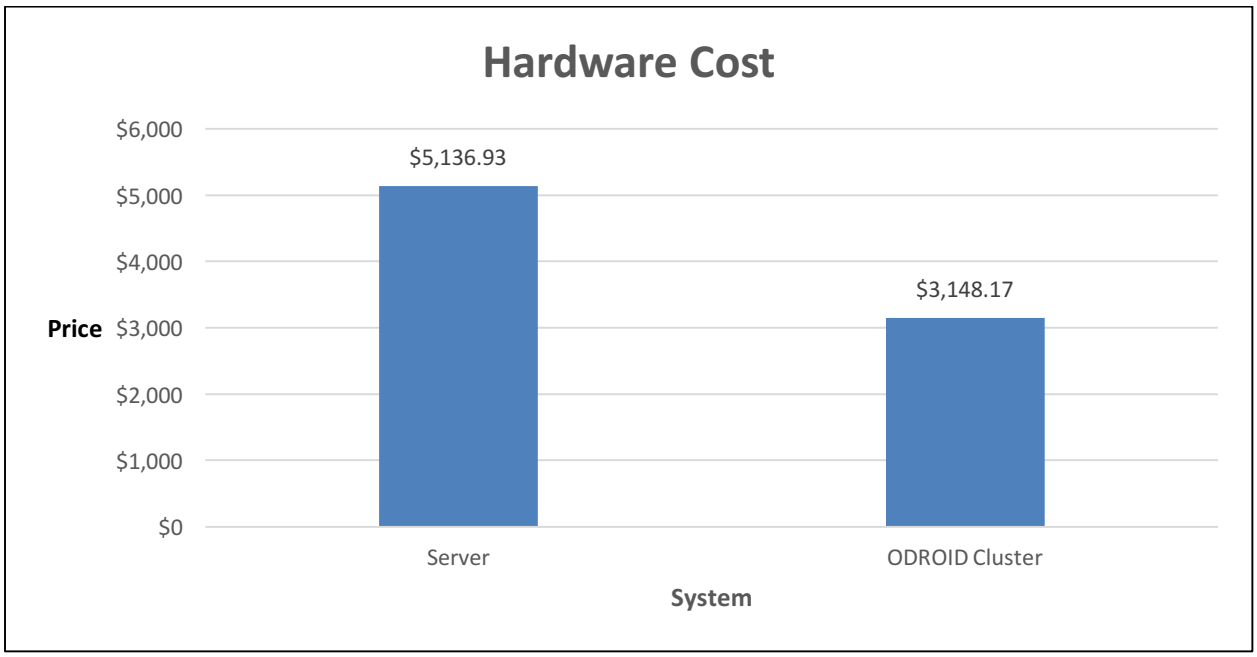


Figure 4 - The Cost of the Systems

5. CONCLUSIONS AND FUTURE WORK

We demonstrated that a cluster of ODROID-XU3 Lite computers with ARM CPUs can perform a virtual screen with AutoDock Vina using 48.2% less electricity than our x64 server. Additionally, the screen was completed using 5.6% less time and the hardware to perform the virtual screen cost about 38% less than the server. Based on our findings, clearly ARM processors have the potential to be used for scientific computing for tasks that require little memory per CPU core, like virtual screening.

However, we note that one drawback of using our cluster of OROID-XU3 Lite computers was that we needed to flash a microSD card with the operating system installed on it for each computer. Flashing the microSD cards for an entire cluster is very time-consuming and would be a problem when producing a large cluster of SoC nodes. We are currently exploring options to avoid this process.

We also note that while the process of virtual screening worked well with our cluster, virtual screening requires only a small amount of memory per molecule being screened, so it works well on the systems with small amounts of memory per CPU core. For other scientific problems that require more memory per CPU core used, this solution may not work as well.

Additionally, virtual screening doesn't write significant amounts of output to files, so we did not run into any problems using NFS for our cluster. However, it would be interesting to see if other programs that produce more output cause issues due to the NFS share. Newer ODORID SoC computers also have Gigabit Ethernet, so that may help keep NFS from becoming a bottleneck. A hierarchical configuration where all the nodes of a particular portion of the cluster write to one NFS share while nodes in other portions of the cluster write to other NFS shares could also help prevent this from becoming a problem.

6. REFLECTIONS

I have been taking computer science courses since I was in high school, so I've known that I have a knack for the type of analytical thinking required in computer science for a while. Since I was a sophomore in high school, I have loved using computers to build things and solve problems. By the time I got to college, I was pretty set on furthering my education in computer science and eventually pursuing a job in the field, though at the time I assumed that school was the immediate concern and making a difference in the field would come post-graduation. I had never imagined that my opportunity to advance and explore the field would come while I was still an undergraduate.

Through a mentor who is very dedicated to the education of his students both in and out of the classroom, I received the opportunity to participate in the Blue Waters internship program. Once we had discussed the details, I knew that I would be spending my summer doing research, and I couldn't have been more excited. I was excited to have the opportunity to explore problems that were more applicable and important to the "real world" than those I had been working on for my classes. In addition to this, I looked forward to expanding my knowledge base beyond what the traditional class room had to offer.

The summer that I spent doing the research was memorable for many reasons. It began with my trip to University of Illinois at Urbana–Champaign for the Blue Waters Petascale Institute. This two-week workshop provided me with the unique opportunity to meet other students who excel in the field of computer science, learn new skills to bring back to my own project, and see, as well as work with, the Blue Waters supercomputer. These opportunities were unique to the Petascale Institute and something that I could never have hope to experience without this internship.

When I returned to school to begin my project I had a new expanse of knowledge with which to work, and couldn't have been more excited. The project was engaging and gave me the

opportunity to test some of my new skills in an independent setting.

Overall, I could not have asked for more from a summer experience. I was given the chance to learn new skills and put them to the test on a project that was manageable, yet challenging. Through this experience I discovered that I have an interest in hardware which I would not have otherwise discovered in my undergraduate studies. This new-found interest led me to pursue a more hardware-based internship this last summer. I found that I thoroughly enjoy writing embedded software and have chosen to pursue a master's degree in computer engineering at the University of Louisville while continuing work in embedded software after I graduate in the spring. Without this internship, it is quite possible that I would have simply stuck to software for the remainder of my college career and never discovered my interest in hardware.

7. ACKNOWLEDGMENTS

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. We thank the Blue Waters Student Internship Program for providing Olivia with this opportunity. Finally, we thank Centre College, which provided Olivia with housing for the summer and provided us with the equipment we used for this work.

8. REFERENCES

[1] Zhu, G (ed.). 2012. *NMR of Proteins and Small Biomolecules*. Springer Science & Business Media, Heidleberg. DOI: https://doi.org/10.1007/978-3-642-28917-0.

[2] Shoichet BK. 2004. Virtual screening of chemical libraries. *Nature*. 432(7019) (Dec. 2004), 862-865. DOI: https://doi.org/10.1038/nature03197.

[3] Trott, O. and Olson, A. J. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading. *Journal of Computational Chemistry* 31 (2010) 455-461. DOI: https://doi.org/10.1002/jcc.21334.

[4] Toth, D., Franco, J., and Berkes, C. 2013. Attacking HIV, tuberculosis and histoplasmosis with XSEDE resources. *In Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery*. (San Diego, CA, July 22-25, 2013). XSEDE '13. ACM, New York, NY. DOI: https://doi.org/10.1145/2484762.2484766.

[5] Sharma, S., Hsu, C., and Feng, W. Making a case for a green500 list. *In Proceedings of the 20th international conference on Parallel and distributed processing*. (Rhodes Island, Greece, Apr. 25-29, 2006). IPDPS'06. IEEE Computer Society, Washington, DC, USA. DOI: https://doi.org/10.1109/ipdps.2006.1639600.

[6] Hsu, J. Supercomputer 'Titans' Face Huge Energy Costs. http://www.livescience.com/18072-rise-titans-exascale-supercomputers-leap-power-hurdle.html.

[7] New top supercomputer dumps cores and increases power efficiency. http://royal.pingdom.com/2012/11/13/new-top-supercomputer-dumps-cores-and-increases-power-efficiency/.

[8] Power Optimization in HPC, Enterprise and Mobile Computing. http://www.scientificcomputing.com/article/2013/02/power-optimization-hpc-enterprise-and-mobile-computing.

[9] Green500 | Top500 Supercomputer Sites: https://www.top500.org/green500/.

[10] Francesquinia, E., Castroc, M., Pennae, P., Duprosf, F., Freitase, H., Navauxc, P., and Méhautg, J. On the Energy Efficiency and Performance of Irregular Application Executions on Multicore, NUMA and Manycore Platforms. *Journal of Parallel and Distributed Computing* 76 (2015) 32-48. DOI: https://doi.org/10.1016/j.jpdc.2014.11.002.

[11] Rajovic, N., Rico, A., Puzovic, N., Adeniyi-Jones, C., and Ramirez, A. Tibidabo: Making the case for an ARM-based HPC system. *Future Generation Computer Systems* 36 (2014) 322–334. DOI: https://doi.org/10.1016/j.future.2013.07.013.

[12] Rajovic, N., Rico, A., Mantovani, F., Ruiz, D., Vilarrubi, J., Gomez, C., Backes, L., Nieto, D., Servat, H., Martorell, X., Labarta, J., Ayguadé, E., Adeniyi-Jones, C., Derradji, S., Gloaguen, H., Lanucara, P., Sanna, N., Mehaut, J-F., Pouget, K., Videau, B., Boyer, E., Allalen, M., Auweter, A., TAFANI, D., Brayford, D., Brömmel, D., Halver, R., Meinke, J., Beivide, R., Benito, M., Vallejo, E., Valero, M., Ramirez, A. The Mont-Blanc prototype: An Alternative Approach for HPC Systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. (Salt Lake City, UT, Nov. 13 - 18, 2016). SC '16. DOI: https://doi.org/10.1109/sc.2016.37.

[13] Toth, D., Kamperman, F., and Paige, E. Making Drug Discovery More Sustainable by Reducing the Electricity Consumption and Cost of Virtual Screening. *In Proceedings of the Second International Conference on Computer Science, Computer Engineering, and Education Technologies*. (Kuala Lumpur, Malaysia, Sept. 8-10, 2015). CSCEET 2015.

[14] Keipert, K., Mitra, G., Sunriyal, V., Leang, S., Sosonkina, M., Rendell, A., and Gordon, M. Energy-Efficient Computational Chemistry: Comparison of x86 and ARM Systems. *Journal of Chemical Theory and Computation* 11 (2015) 5055-5061. DOI: https://doi.org/10.1021/acs.jctc.5b00713.

[15] Tiwari, A., Keipert, K., Jundt, A., Peraza, J., Leang, S., Laurenzano, M., Gordon, M., and Carrington, L. Performance and energy efficiency analysis of 64-bit ARM using GAMESS. In Proceedings of the 2nd International Workshop on Hardware-Software Co-Design for High Performance Computing. (Austin, TX, Nov. 15, 2015). Co-HPC '15. ACM, New York, NY. DOI: https://doi.org/10.1145/2834899.2834905.

[16] ODROID | Hardkernel: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G141351880955&tab_idx=2.

[17] Gnu parallel: https://www.gnu.org/software/parallel/.

[18] Slurm Workload Manager: http://slurm.schedmd.com/.

[19] Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G., ZINC: A Free Tool to Discover Chemistry for Biology. *Journal of Chemical Information and Modeling*

57 (7) (2012) 1757-1768. DOI:
https://doi.org/10.1021/ci3001277.

An Implementation of Parallel Bayesian Network Learning

Joseph S. Haddad
The University of Akron
302 E Buchtel Ave
Akron, OH, 44325, United States
jsh77@zips.uakron.edu

Timothy W. O'Neil
The University of Akron
302 E Buchtel Ave
Akron, OH, 44325, United States
toneil@uakron.edu

Anthony Deeter
The University of Akron
302 E Buchtel Ave
Akron, OH, 44325, United States
aed27@zips.uakron.edu

Zhong-Hui Duan
The University of Akron
302 E Buchtel Ave
Akron, OH, 44325, United States
duan@uakron.edu

ABSTRACT

Bayesian networks may be utilized to infer genetic relations among genes. This has proven useful in providing information about how gene interactions influence life. However, Bayesian network learning is slow as it is an NP-hard algorithm. K2, a search space reduction, helps speed up the algorithm but may introduce bias. The bias arises from the fact that K2 enforces topologies which makes it impossible for subsequent nodes to become parents of previous nodes while the algorithm builds the network. To eliminate this bias, multiple Bayesian networks must be computed to ensure every node has the chance to be a parent to every other node. The purpose of this paper is to propose a hybrid algorithm for generating consensus networks utilizing OpenMP and MPI. This paper evaluates the parallelization of network generation and provides commentary on learning and implementing OpenMP and MPI. The OpenMP and MPI accelerations are implemented in a single library and can be switched on or off. These accelerations are for computing multiple Bayesian networks simultaneously. Methods are developed and tested to evaluate the results of the implemented accelerations. The results show generating networks across multiple cores results in a linear speed-up with negligible overhead. Distributing the generation of networks across multiple machines also introduces linear speed-up, but results in additional overhead.

1. INTRODUCTION

Inferring relations among genes requires a significant amount of data. Bayesian networks may be used to correlate this data and extract relationships among the genes [12]. We do not know what this relationship is, but we do know it has a high likelihood of existing. These relationships can then be used to make testable hypotheses to determine how gene interactions influence life in organisms or humans. As a result, tests can be performed in the lab with more confidence and a reduced chance of wasting time and resources.

This concept has been applied to smaller data sets and shows promising results [12], however remains too slow to be applied to a larger problem. It is our objective to decrease the runtime required to form a network which may reveal genetic interactions. Bayesian network learning, however, is inherently slow because it is an NP-hard algorithm [4]. Search space reduction algorithms may be utilized to reduce the computational complexity. K2 is a great example of a search space reduction algorithm, and is our algorithm of choice. However, it introduces a new problem. K2 restricts the parent hierarchy of genes within the network [4], and thus introduces bias in the computed relations. To achieve high confidence in the generated networks, an abundance of Bayesian networks need to be computed using random search space restrictions. These random search space restrictions (or topologies) remove the bias and provide results which can be interpreted at various levels of confidence.

By eliminating one problem and introducing another, consensus networks enable the ability of parallelization by requiring multiple units of work rather than just one faster unit of work. Other authors describe parallel implementations that can increase the speed of Bayesian network learning [2] [8]. However, no libraries exist which compute multiple Bayesian networks concurrently. This project examines the value of Bayesian network learning within a parallel environment in order to reduce the time needed to generate consensus networks using many topological inputs. This examination is performed through implementation of the said algorithm, exploring methods available such as OpenMP and MPI.

Results from running experiments with varying number of cores and machines are examined and it is found our parallelization has a positive impact. There are a couple caveats, however, such as the over provisioning of resources which leads to waste and potential introduction of latency from cluster parallelism. When the resources are appropriate for the problem size, OpenMP and MPI substantially reduce the time to generate a consensus network. The reduction in runtime appears to be linear, more so after accounting for introduced latency and overhead.

This paper is an extension to the initial analysis performed

DOI: https://doi.org/10.22369/issn.2153-4136/8/2/4

on the algorithm and explains the thought processes behind the implementation. The preceding publication shows why the algorithm needs to be sped up, as an increase in samples causes linear growth of the problem and introduction of additional genes causes exponential growth of the problem [5]. After reading this paper, the reader should have a sense of why and how the parallelization was reasoned about and implemented to achieve optimal efficiency.

2. BACKGROUND

2.1 Bayesian Networks

Bayesian networks capture qualitative relationships among variables within a directed acyclic graph (or DAG). Nodes within the DAG represent variables, and edges represent dependencies between the variables [6] [11]. Bayesian networks have a search space which grows exponentially when introducing new nodes and not placing restrictions on the structure of the network. This complication can be overcome by using the K2 algorithm. The K2 algorithm reduces the computational cost of learning by imposing restraints on parent node connections via topological ordering [4]. Here, a topology refers to a hierarchical structure of parenthood that the K2 algorithm will utilize to reduce overall computational complexity while scoring data relationships. Restricting the parent ordering, however, creates an issue of bias, which is inherent within a constraint-based search space reduction [12]. Sriram [12] proposed a solution to this issue by creating a consensus network, or the combination of multiple Bayesian networks derived from several topological inputs. To eliminate the bias created by these restraints, many randomly generated topologies are used. By increasing the number of topological inputs, the consensus network has a greater chance of reflecting the true nature of the gene interactions with higher levels of confidence.

2.2 OpenMP

OpenMP or (Open Multi-Processing) is a cross-platform, multilingual application programming interface (API) which enables shared-memory parallel programming on a single machine. The OpenMP specification consists of compiler directives and library functions used to parallelize portions of a program's control flow [10]. The most rudimentary example of OpenMP would be to distribute a for-loop across multiple threads.

An advisory board of top entities in computation controls its specification [1] which can be implemented by various compilers to target specific system capabilities and architectures. The specification includes language-specific APIs, compiler directives, and standardized environment variables [10]. The model of OpenMP is comparable to the fork-join model, but provides additional convenience (cross-platform) features through compiler directives. These directives consist of, but are not limited to, barriers, critical regions, variable atomicity, shared memory, and reductions [10].

OpenMP enables parallel code portability at a level which would not be achievable while retaining an ideal code climate. OpenMP, by nature allows simple and straight-forward parallelization of loops with a compiler directive that targets the system for which the program is compiled on. Without OpenMP, the program would have to include many different libraries and routines to achieve parallel code across different systems. The result of this would be a program which only works on a specific set of machines, or a code base which is hard to maintain and debug when changes are made to the underlying algorithm.

2.3 MPI

MPI (or Message Passing Interface) is a standard which outlines network-routed (a)synchronous communication between machines [9]. MPI enables executing programs across multiple machines in a cluster and passing messages between them to schedule work or share information.

Execution of a program which utilizes MPI is most often performed with a tool. This tool is responsible for forwarding appropriate parameters to each program in order to specify the information required for the processes to communicate. Upon program start, the MPI execution environment must be initialized using the MPI library methods [9]. The initialization sequence results in augmented program arguments (to remove arguments passed by the execution tool) and the rank of the program in the MPI environment [9]. This information allows the program to proceed as normal while being a small part in a larger sum.

3. METHODOLOGY

Testing was performed on the Blue Waters petascale machine at the University of Illinois at Urbana-Champaign. The facility is maintained by Cray and consists of 22,640 Cray XE6 machines and 3,072 XK7 machines, which are CPU-only and GPU-accelerated machines respectively. The XE6 machines consist of two 16 core AMD processors with 64 GBs of RAM. The XK7 machines consist of a single 16 core AMD processor with 32 GBs of RAM and a NVIDIA K20X GPU [7].

Cray XE6 machines were used to perform all tests utilizing purely synthetic data. OpenMP and MPI were implemented by the Cray Compiler, Cray C version 8.3.10. The synthetic data is in the form of a gene-by-sample matrix consisting of the presence or absence of each gene within the sample. This data was generated according to a model we defined. We then ensured the result of the consensus network(s) matched our model to validate functionality and evaluate a degree of correctness for our algorithm. Each test was run five times with the mean, standard deviation, and standard error calculated to measure runtime consistency.

The library being used to run the tests is available online [3]. This library was implemented as described in this paper.

3.1 Processors

The first natural step in parallelizing computation is to attempt to use multiple cores (or threads) simultaneously on the machine. This can be done by running multiple instances of the program, or by implementing code which takes advantage of multiple threads. Analyzing the program reveals a couple potential places for parallelization. There are many for-loops which perform actions which are independent from one another. The for-loops identified for inspection are the generation of topologies and the iteration over the topologies to generate networks.

The generation of topologies results in a a predetermined number of topologies filled into an array. This operation can be easily parallelized across multiple cores as they are independent. The appropriate tool to perform this parallelization is OpenMP. OpenMP was implemented with a simple compiler directive which sped up computation.

```
#pragma omp parallel for
for (...) { }
```

Iterating over the topologies to generate networks can also be parallelized. The creation of Bayesian networks are independent from one another, and thus, networks can be asynchronously generated. Implementation of this parallelization is straight-forward as Bayesian network computation does not mutate its data set. This prevents us from having to replicate the memory and increase the space complexity of the algorithm. OpenMP was implemented again as shown above. Additionally, within the parallel for, the resulting network must be appended to the consensus network. The consensus network, however, is not thread-safe and must be operated on within a critical section. A critical section specifies that the code can only be executed on one thread at a time.

```
#pragma omp critical
for (...) { }
```

This ensures the networks are properly summed together, otherwise, an addition may be lost. For example, if `Thread A` and `Thread B` attempt to increment a variable at the same time, they may both access the value before the other commits the new value. This will result in a lost operation, as the threads are not aware of one another.
To measure the resulting computational runtime decrease, multiple tests were performed with varying number of processors. A single set of synthetic data was used which consisted of 10 genes and 10,000 samples. Using an exclusively reserved machine, tests were run by varying the number of processors (up to 32) and measuring the algorithm performance for the creation of 160 Bayesian networks per gene 1600 total). We have reached the resource limits on the systems which we have access to, and cannot test beyond 32 cores. The selection of 10 genes and 160 Bayesian networks was arbitrarily chosen as sufficient means to measure computation time.

3.2 Cluster Parallelism

Distributing work across multiple machines requires a different approach than that of OpenMP. OpenMP cannot share memory across machines so it cannot be applied to this situation. MPI is optimal for this situation as it allows machines to send messages back and forth to share memory and communicate their responsibilities and results. Distributing the Bayesian network learning process across multiple machines doesn't make much sense because each step is dependent on the previous, so the result would be a slower computation since calculations couldn't happen in parallel and there would be added network latency. The main candidate for distribution would be the computation of a Bayesian network (or the iteration over the topologies), because networks are computed independent of one another and there is a large backlog of networks which need to be computed. Distributing the work with MPI is surprisingly simple, as the topologies are randomly generated. This means there is no communication required prior to beginning computation. Upon initialization, each machine must determine its rank and role by augmenting the arguments, this may be done like so.

```
int main(int argc, char **argv) {
  int forkIndex = 0, forkSize = 1;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &forkIndex);
  MPI_Comm_size(MPI_COMM_WORLD, &forkSize);

  ...
}
```

Each machine can then determine how much work it needs to do by dividing the number of requested topologies per gene by the number of machines in the swarm.

```
int top_d = topologies / forkSize;
int top_r = topologies % forkSize;
if (forkIndex < top_r) ++top_d;
topologies = top_d;
```

When the machines complete their share of the computation they communicate to coalesce the computed networks into a consensus network. The master machine then saves the consensus network to the disk and completes any other required computations which are simple enough not to require being distributed across machines.
Tests are conducted to measure the impact on runtime when multiple machines are used. The same data is used from the above (processors) test. Tests were run on dedicated machines utilizing 16 processors and computing 60 Bayesian networks per gene (600 total). The selection of 10 genes and 60 Bayesian networks was arbitrarily chosen as sufficient means to measure computation time.

4. RESULTS AND DISCUSSION

In the following tables, the standard deviation is represented by the letter `s` and the standard error is denoted by `se`. This standard deviation and error is in regards to the algorithm runtime, not the accuracy of the algorithm.

4.1 Processors

When increasing the number of processors, the resulting runtime decrease appears to be linear. The linear nature of the results removes the necessity for further testing between the number of cores tested. Figure 1 illustrates that as the number of processors increase, the runtime decreases at approximately the same rate. Exact results may be seen in Table 1.

Table 1: Runtimes for the program across increasing numbers of processors.

Cores	Mean Time	*s*	*se*
1	396.348	3.192	1.427
2	269.023	0.530	0.237
4	137.359	0.629	0.281
8	76.169	0.220	0.090
16	40.359	0.307	0.137
32	22.172	0.144	0.064

This linear decrease is consistent with how OpenMP distributes its work. OpenMP distributes the task of an independent Bayesian network computation across multiple threads simultaneously. These independent tasks are non-blocking and do not lock one another, and thus have very little contention. There is one lock after each computation which appends the network to the consensus network, but is negligible to the total time taken to compute the Bayesian

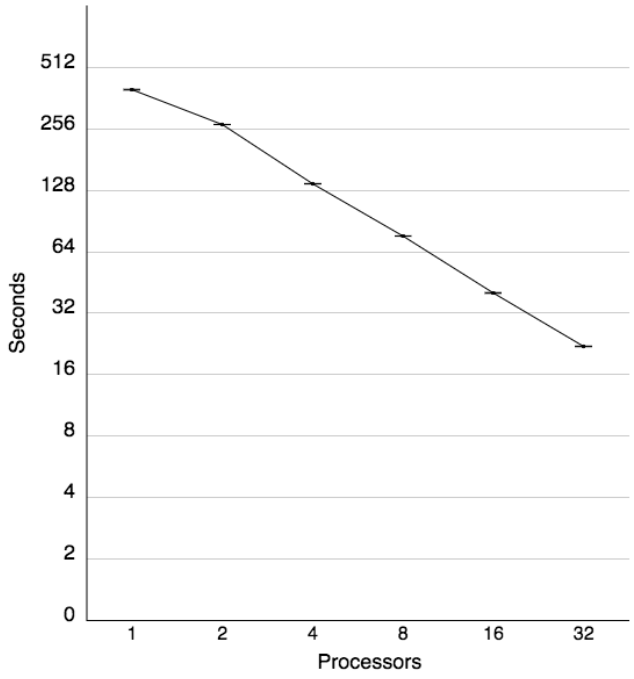


Figure 1: Illustrates runtime decrease as the number of processors increase. The decline is nearly linear.

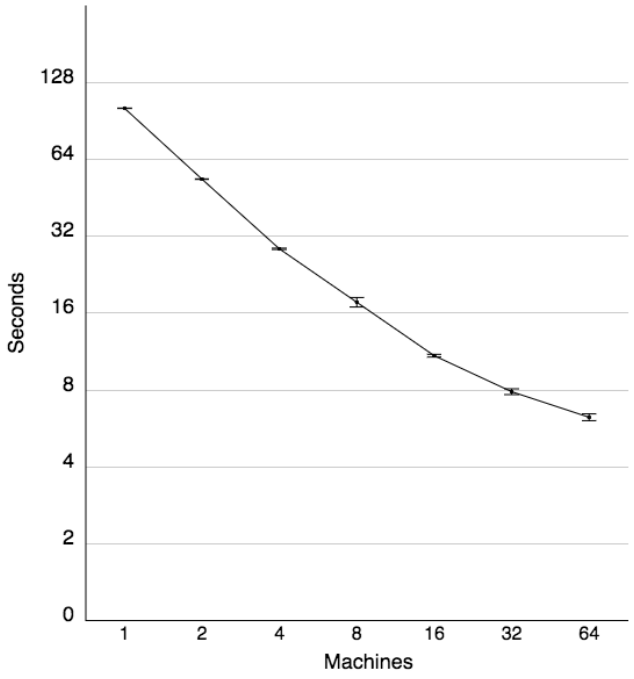


Figure 2: Illustrates runtime decrease as the number of machines increase. The decline is nearly linear.

networks. OpenMP results in such low runtime standard error because it works with memory within the program and requires no network communication like MPI. The reduction of standard error as the number of threads increase may be due to the kernel. The kernel is responsible for scheduling threads and ensuring other work on the system gets done. The increase in threads means there are more threads which may go uninterrupted by the kernel scheduling something else from the operating system.

4.2 Cluster Parallelism

The resulting runtime decrease also appears to be linear while increasing the number of machines. However, as the number of machines increase, overhead also increases. Figure 2 demonstrates that as the number of machines increase, there is much more variation introduced and overhead in the runtime.

Observing 64 machines and leading up to 64 machines, it can be noted that the reduction in runtime becomes less and less and then starts increasing. This increase in runtime happens when the inflection point has been reached for the given set of data. At some point, it takes longer to send the data over the network than it would be to simply compute more data on fewer machines. There are some potential modifications which can be made to mitigate this overhead (such as asynchronous coalescing), but it cannot be eliminated completely. It is important to note that an increase in resources does not necessarily mean an increase in performance, nor always one for one; see Table 2 for test results.

The standard error generally increases with the increase in machines, but this is not always true. There does not seem to be a correlation between an increase or decrease in machines with an increase or decrease in standard error, except for the general rule stated above. This is consistent with the fact that networks are very unpredictable. Pings may vary wildly depending on other network traffic and the route which packets decide to take. Additionally, there may be other noisy peers on the network hogging bandwidth and causing slower transmissions. On clusters across the world wide web, traffic may have to travel through geographical displacement and suffer packet loss or increases in latency. The only thing consistent with the standard error is that it is not consistent.

5. CONCLUSION

By generating a consensus network out of many Bayesian networks, researchers may screen and infer new gene interactions. This allows researchers to feel more confident about testing hypotheses in the lab, such that their resources and time will not be wasted.

We have concluded that utilizing parallelization through means of OpenMP and MPI substantially reduces the time to generate a consensus network. However, as demonstrated in the graphs above, an increase in resources must be tailored to the problem at hand. Increasing the resources too significantly becomes detrimental, resulting in costly waste; see Table 2.

Future work may involve parallelizing the coalescing of consensus networks in effort to reduce the overhead introduced when increasing cluster parallelism. Additionally, all matrix operations are currently done on a single-thread. These operations (in some cases) contain thousands of rows and columns being applied to an expensive mathematical function. These operations are ideal for the GPU as it can perform the arithmetic across several thousand of threads simultaneously. As such, the motivation for this is that CUDA (or other

Table 2: Runtimes for the program across increasing numbers of machines.

Nodes	Mean Time	*s*	*se*
1	102.204	0.361	0.161
2	53.451	0.272	0.122
4	28.656	0.383	0.171
8	17.8	1.812	0.810
16	10.917	0.327	0.134
32	7.862	0.462	0.207
64	6.259	0.444	0.198
128	6.739	0.430	0.193
256	7.904	1.110	0.496
512	7.241	0.246	0.110
1024	8.845	1.105	0.494

means of GPGPU acceleration) has the potential to speed the algorithm up by several orders of magnitude.

6. REFLECTIONS

Working on this project gave me a massive amount of experience, which far surpassed what I thought it would. I gained experience in professional writing for journal publications and renewed my skills in proofreading. I also gained exposure to a whole new aspect of project organization which I was not used to: meetings with advisors, progress reports, and demos. I feel like this has really helped foster my professional identity and prepared me more for higher education and the workforce. Additionally, I flexed my problem solving skills while implementing the algorithm and begun refactoring. The refactoring had to be done in such a fashion to allow for parallelization. This presented some challenges because there were also memory considerations to make things sharable over the network (MPI). Overall, I learned many invaluable skills which will be applied to my future education and work. Notably, I performed my first publication [5] and gave a presentation at the associated conference, then proceeded to present a poster version of the paper at GLBIO 2016 to draw attention to the work.

7. ACKNOWLEDGMENTS

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.
Additional financial support was provided by the Buchtel College of Arts and Sciences at The University of Akron. Further support was provided by a grant from the Choose Ohio First Bioinformatics scholarship.
The data, statements, and views within this paper are solely the responsibility of the authors.

8. REFERENCES

[1] About the OpenMP ARB and OpenMP.org: *http://openmp.org/wp/about-openmp/*.
[2] Altekar, G. et al. 2004. Parallel metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. Bioinformatics.
[3] Bayesian Learning source code: *https://github.com/Timer/bayesian-learning*.
[4] Cooper, G.F. and Herskovits, E. 1992. A Bayesian method for the induction of probabilistic networks from data. Machine Learning.
[5] Haddad, J.S. et al. 2016. Analysis of Parallel Bayesian Network Learning. Proceedings of the 31st International Conference on Computers and Their Applications.
[6] Korb, K. and Nicholson, A. 2003. *Bayesian artificial intelligence*. Chapman and Hall/CRC.
[7] Lessons Learned From the Analysis of System Failures at Petascale: The Case of Blue Waters: *https://courses.engr.illinois.edu/ece542/sp2014/finalexam/papers/bluewaters.pdf*.
[8] Misra, S. et al. 2014. Parallel Bayesian network structure learning for genome-scale gene networks. International Conference for High Performance Computing, Networking, Storage and Analysis.
[9] MPI: A Message-Passing Interface Standard: *http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf*.
[10] OpenMP Application Program Interface: *http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf*.
[11] Pearl, J. 1998. *Probabilistic inference in intelligent systems*. Morgan Kaufmann Publishers.
[12] Sriram, A. 2011. *Predicting Gene Relations Using Bayesian Networks*. MS thesis, University of Akron.

Interactive Analytics for Complex Cognitive Activities on Information from Annotations of Prokaryotic Genomes

Raphael D. Isokpehi, Kiara M. Wootson
College of Science, Engineering and Mathematics
Bethune-Cookman University
640 Dr. Mary McLeod Bethune Blvd.
Daytona Beach, Florida 32114, USA
isokpehir@cookman.edu

Dominique R. Smith-McInnis
Environmental Science PhD Program
Jackson State University
1400 JR Lynch Street
Jackson, Mississippi 39217, USA

Shaneka S. Simmons
Jarvis Christian College
P. O. Box 1470
Hawkins, Texas 75765, USA

ABSTRACT
Several microbial genome databases provide collections of thousands of genome annotation files in formats suitable for the performance of complex cognitive activities such as decision making, sense making and analytical reasoning. The goal of the research reported in this article was to develop interactive analytics resources to support the performance of complex cognitive activities on a collection of publicly available genome information spaces. A supercomputing infrastructure (Blue Waters Supercomputer) provided computational tools to construct information spaces while visual analytics software and online bioinformatics resources provided tools to interact with the constructed information spaces. The Rhizobiales order of bacteria that includes the *Brucella* genus was the use case for preforming the complex cognitive activities. An interesting finding among the genomes of the dolphin pathogen, *Brucella ceti*, was a cluster of genes with evidence for function in conditions of limited nitrogen availability.

General Terms
Big Data, Human-Computer Interaction, Microbiology, Visualization.

Keywords
Bacteria; *Brucella*; Cognitive Activities, Genomics; Stress Response; Universal Stress Protein, Visual Analytics

1. INTRODUCTION
The automated annotation of genome sequences of bacteria and archaea produces diverse types of data sets including multivariate data on predicted protein-coding genes [1-4]. Examples of variables annotated for protein-coding genes are genome unique identifier, genome name, unique gene identifier (locus tag), coordinates of the start and end position, product description, Enzyme Commission identifier, length of gene sequence, and location of gene on positive or negative strand.

Several microbial genome databases [1, 3, 4] provide collections of thousands of genome annotation files in formats (such as tab delimited) suitable for importing to computational environments that support the performance of complex cognitive activities. In complex cognitive activities (such as analytical reasoning, decision making, knowledge discovery, learning, planning, problem solving, sense making and understanding), humans interact with information to support their information-intensive thinking processes [5-7].

DOI: https://doi.org/10.22369/issn.2153-4136/8/2/5

The goal of the research reported in this article was to develop interactive analytics resources to support the performance of complex cognitive activities on a collection of publicly available genome information spaces. A genome annotation file containing protein-coding genes of a bacterial (eubacteria and archaebacteria) genome could be described as an information space which can be compared or integrated to other information spaces. The complex genomic information space presents diverse opportunities for knowledge generation on microbial genomes that combines the affordances from both the human cognitive system and computing system. The goal of our research was to obtain potentially biologically relevant insights from the microbial genomic information space. Therefore, we have combined (i) the use of a supercomputing environment (Blue Waters Supercomputer) [8] to construct information spaces; (ii) the use of visual analytics software to interact with the constructed information spaces; and (iii) online bioinformatics resources on microbial genomes.

Visual analytics affords humans to analyze huge information spaces in order to support complex cognitive activities such as decision making and data exploration [9]. The interaction with information through visual representations provides a human-centered approach to the performance of cognitive activities [10, 11]. This human-centered approach lowers the barriers to knowledge generation from genome information spaces. In addition, there is potential to increase the number of undergraduate students who are able to engage in genomics research.

An example of genome information space is the PATRIC Bioinformatics Resource, which provides collection of thousands of genome annotation files available for download at ftp://ftp.patricbrc.org/patric2 [4]. The first objective of this research study was to construct an information space on the count of genes assigned to strands [positive (+) or negative DNA strand (-)] in the thousands of genome annotation files. This objective will lead to a reduction in the complexity of the information space for subsequent complex cognitive activities with desktop visual analytics software. The second objective was to perform complex cognitive activities on genomic information from multiple sources. Though, we recognize that some complex cognitive activities often done without clear distinctions.

These objectives are important to our investigation of stress responsive gene clusters that include genes which encode the universal stress proteins (pfam00582) [12, 13]. The genomes sequenced from bacteria in the order Rhizobiales were used to accomplish the research study objectives. Rhizobiales is a diverse order of bacteria that include nitrogen-fixing bacteria associated with leguminous plants and lichens as well as intracellular

pathogens of animals and plants [14, 15]. Examples of genera in the order Rhizobiales (alphaproteobacteria) are *Bartonella*, *Beijerinckia*, *Bradyrhizobium*, *Brucella*, *Cohaesibacter*, *Hyphomicrobium*, *Methylobacterium, Microvirga*, *Methylocystis*, *Phyllobacterium*, *Rhizobium*, *Rhodobium, Rhodopseudomonas* and *Xanthobacter* [16]. Finally, the interactive views can provide opportunities for learning about the genomes of bacteria.

2. METHODS

2.1 Source of Genome Annotation Files

The genome annotation files (with file extension RefSeq.cds.tab) were downloaded from the PATRIC Bioinformatics Resource at ftp://ftp.patricbrc.org/patric2/genomes_by_species/ to the Blue Waters Supercomputer. Each file is expected to contain a header row and records with annotation for each gene including genome unique identifier, genome name, unique gene identifier (locus tag), coordinates of the start and end position, product description, Enzyme Commission identifier, length of gene sequence, and location of gene on positive or negative strand.

Three additional files (genome_lineage, genome_metadata and genome_summary) were obtained from ftp://ftp.patricbrc.org/patric2/current_release/RELEASE_NOTES/Feb2016/. These files contain fields that can be used accomplish complex cognitive activities. The genome_lineage file includes taxonomic annotation of genomes including kingdom, phylum, order, genus and National Center for Biotechnology Information (NCBI) Taxonomy Identifier. The genome_metadata includes data on habitat, gram stain category and temperature of the microbial isolate source of the genome sequence. The genome_summary file includes data on genome length, gene count and genome sequencing status (e.g. Whole Genome Sequencing, Plasmid and Complete).

2.2 Construction of Information Space on Strand Location of Genes

The genome annotation files include annotation on the transcription direction of the gene (location of gene on the positive (+) or negative (-) strand). A set of computer scripts were developed on Blue Waters Supercomputer [17] to extract the transcription direction of each gene in the genome annotation files. The output file was formatted as a tab delimited file with Genome ID, Gene Count for Strand, the Genome Name and the Transcription Direction. This method allowed us to accomplish our objective to construct an information space on the distribution of genes in genome annotation files by transcription direction [location of gene on positive or negative strand].

2.3 Development of Interactive Analytics for Complex Cognitive Activities

We developed interactive analytics using guidelines provided for designing interactive visual representations for complex cognitive activities [10, 18]. Therefore to design human-information interaction tools for decision making, the interaction features in the design are expected to include the following action patterns: blending, filtering, linking/unlinking, measuring, sharing and translating [7].

A software for visual analytics, Tableau Desktop Professional (Tableau Software Inc. Washington, USA), was used to design the views for accomplishing the following activities: (i) to identify biases in gene distribution across genomes [sense making]; (ii) to decide on which bacteria genome to investigate based on annotated comments [decision making]; and (iii) to determine the arrangement and functions of a cluster of genes that are transcribed together [analytical reasoning].

3. RESULTS

3.1 Information Space on Strand Location of Genes

A total of 21,139 genome annotation files were downloaded from the PATRIC Bioinformatics Resource and processed on the Blue Waters Supercomputer. The collection of files provides a data resource for the performance of data analytics. Each file had 16 fields and number of records corresponding to the protein-coding genes annotated for the genome. The total number of gene records obtained from PATRIC was 74,991,894. The derived information space consisted of four fields: Genome ID, Genome Name, Strand and the Gene Count (assign to each strand).

3.2 Interactive Analytics for Sense Making on Protein-Coding Genes in Rhizobiales

A total of 547 Rhizobiales genome annotation files were evaluated because of our interest in *Rhodopseudomonas palustris* [19]. Figure 1 shows the number of protein-coding genes (RefSeq annotation) assigned to the strands of the genomes of *Brucella ceti*, a *Brucella* species that cause chronic diseases in marine mammals such as dolphins and whales [20]. The visualizations in Figure 1 and Figure 2 allow for the difference in count of genes assigned to the genome strands to be calculated.

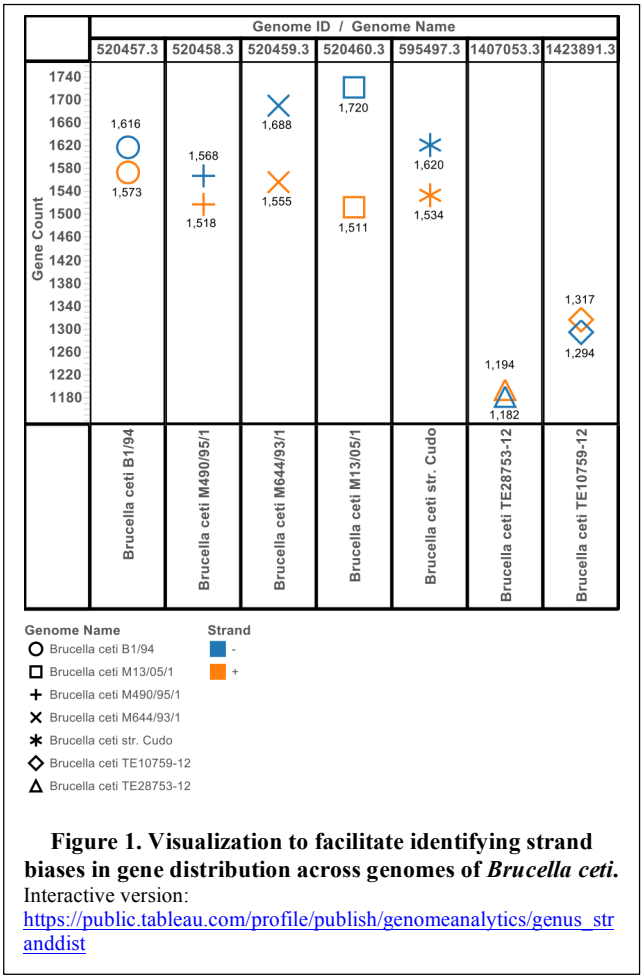


Figure 1. Visualization to facilitate identifying strand biases in gene distribution across genomes of *Brucella ceti*. Interactive version: https://public.tableau.com/profile/publish/genomeanalytics/genus_stranddist

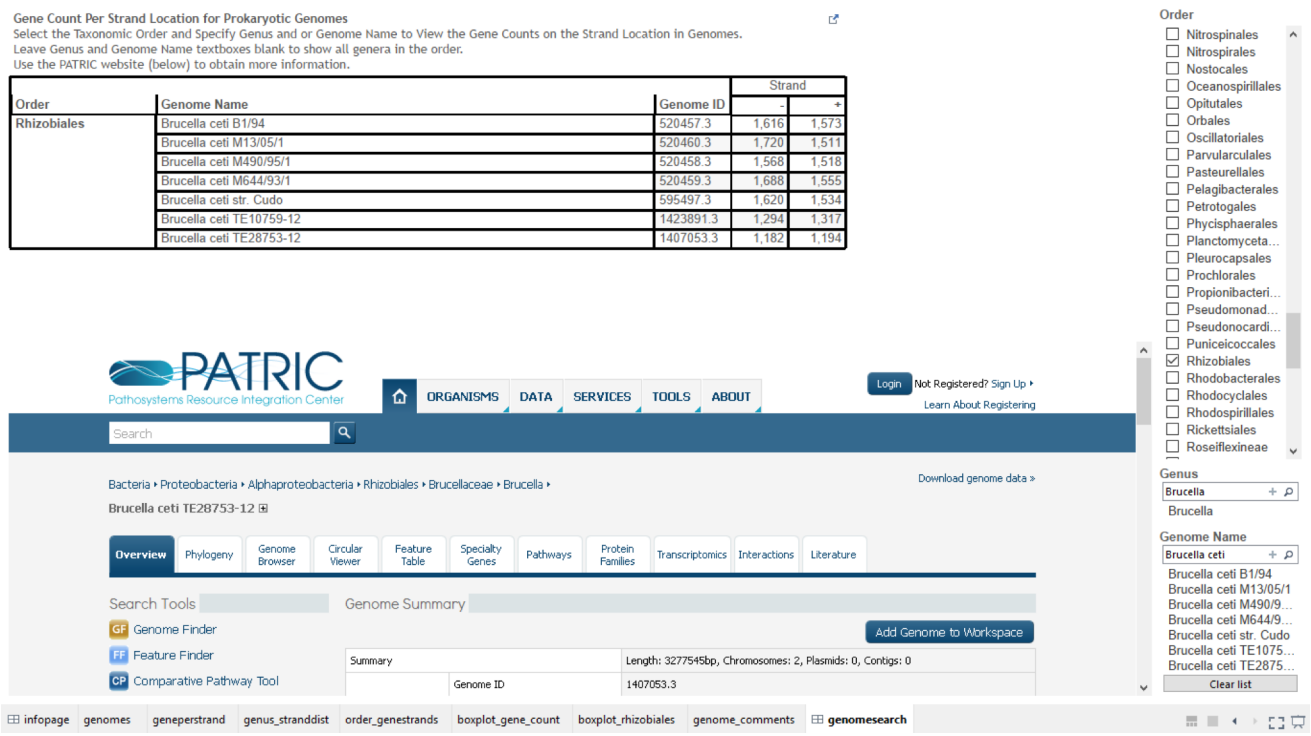


Figure 2. Dashboard providing access to a bioinformatics resource as well as integrating information on the number of genes assigned to chromosomal strand locations for prokaryotic taxonomic and genome categories.

Interactive Analytics resource at: https://public.tableau.com/profile/publish/genomeanalytics/genomesearch. User of the resource can perform activities such as sense making and decision making through selection or specifying the taxonomic order, genus or genome name to view the gene counts on the strand location in genomes. Additional information could be obtained through the Pathosystems Resource Integration Center (PATRIC) website. The dashboard can also be used as a resource for learning the distribution of genes to strand location. In the example, the genomes of *Brucella ceti* are the focus of sense making, decision making and learning activities.

3.3 Interactive Analytics for Sense Making on Protein-Coding Genes in Rhizobiales

Sense making "is concerned with developing a mental model of an information space about which one has insufficient knowledge" [7]. We used the Box plot visualization technique to compare multiple distributions of the gene counts for genera (*Agrobacterium*, *Bartonella*, *Beijerinckia*, *Brucella*, *Methylobacterium*, *Nitrobacter* and *Rhodopseudomonas*) in the Rhizobiales (Figure 3). Interactive figure is available at https://public.tableau.com/profile/publish/genomeanalytics/boxplot_rhizobiales

The design of the visualization involves blending data fields from (i) the genome_lineage file (contains taxonomic information); (ii) the genome_summary file (contains plasmid count); and (iii) the constructed information space on the strand location of genes. The interactive version allows user to specify the bacteria taxonomic family or families to compare.

In the case of the Rhizobiales genomes, examining the box plot revealed genomes with outlier protein coding sequence within the genus. Outlier values in the box plot were annotated for selected genomes. For example, *Brucella ceti* TE10759-12 has 2,376 protein-coding genes in the RefSeq genome annotation file. The missing genes of TE10759-12 provides a user with information to generate testable hypotheses.

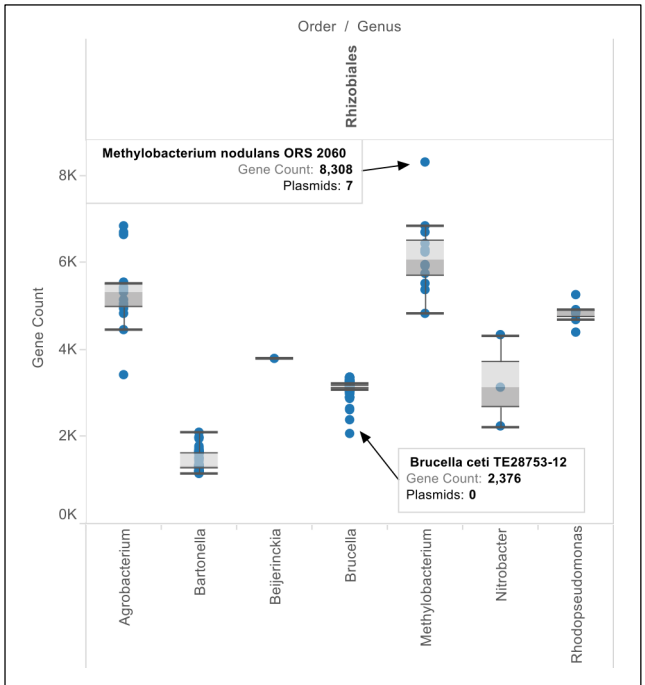


Figure 3. Visual representation (box plot) to facilitate sense making of protein-coding gene counts for selected genomes of Rhizobiales bacteria.

Interactive version: https://public.tableau.com/profile/publish/genomeanalytics/boxplot_rhizobiales

3.4 Interactive Analytics for Decision Making on Genomes for Investigation

In decision making "the attention that is drawn to emergent features may facilitate the choice of one among a number of alternatives within the information space" [21]. We developed a view from the genome_metadata file to display the comments associated with eight *Brucella ceti* genomes. Four categories of comments were identified (Table 1).

Table 1. Categories of Comments on *Brucella ceti* genomes

***Brucella ceti* Strains**	**Comment Category**
B1/94, M13/05/1, M490/95/1, M644/93/1	This strain will be used for comparative analysis with other *Brucella* species.
B1/94	Sequencing of *Brucella* species for qPCR assay development.
str. Cudo	*Brucella ceti* Cudo was isolated from a bottlenose dolphin (*Tursiops truncatus*). The genome sequence of this organism will provide interesting insights into the evolution of this species.
TE10759-12, TE28753-12	… The aim of the study is the deep characterization of the isolates …

https://public.tableau.com/profile/publish/genomeanalytics/genome_comments (Interactive version of genome comments).

The comment "*Brucella ceti* Cudo was isolated from a bottlenose dolphin (*Tursiops truncatus*)" facilitated our decision to further conduct gene neighborhood analysis of the universal stress proteins of *Brucella ceti* Cudo. Universal stress proteins contain the protein family (Pfam) domain with Pfam Identifier as PF00582 or pfam00582 [22]. We obtained a list of 1377 genes predicted as encoding universal stress proteins in 348 *Brucella* genomes. The Locus Tags for *Brucella ceti* Cudo universal stress proteins (USP) are BCETI_1000312, BCETI_3000327, BCETI_5000106 and BCETI_7000519. Only BCETI_7000519 was annotated as located on the positive strand (+) location.

We subsequently obtained and used the image of gene neighborhood of each USP gene using the BioCyc Database Collection [23]. The comparison of the gene neighborhood images would help us to confirm the transcription direction and also discover the functions adjacent to the *Brucella* genes for universal stress proteins (Figure 4). We found that BCETI_1000312 USP gene is at the beginning of a four-gene transcription unit (operon) (Figure 4). The other genes (BCETI_1000313, BCETI_1000315 and BCETI_1000316) respectively encode for tryptophanyl-tRNA synthetase (trpS), integral membrane protein (MviN) [renamed Peptidoglycan biosynthesis protein MurJ], and protein-P-II uridylyltransferase (glnD). The gene BCETI_1000311, adjacent to the USP gene BCETI_1000312, encodes a nitrogen fixation related protein. BCETI_1000311 is not predicted to be in same transcription unit with the USP gene (BCETI_1000312).

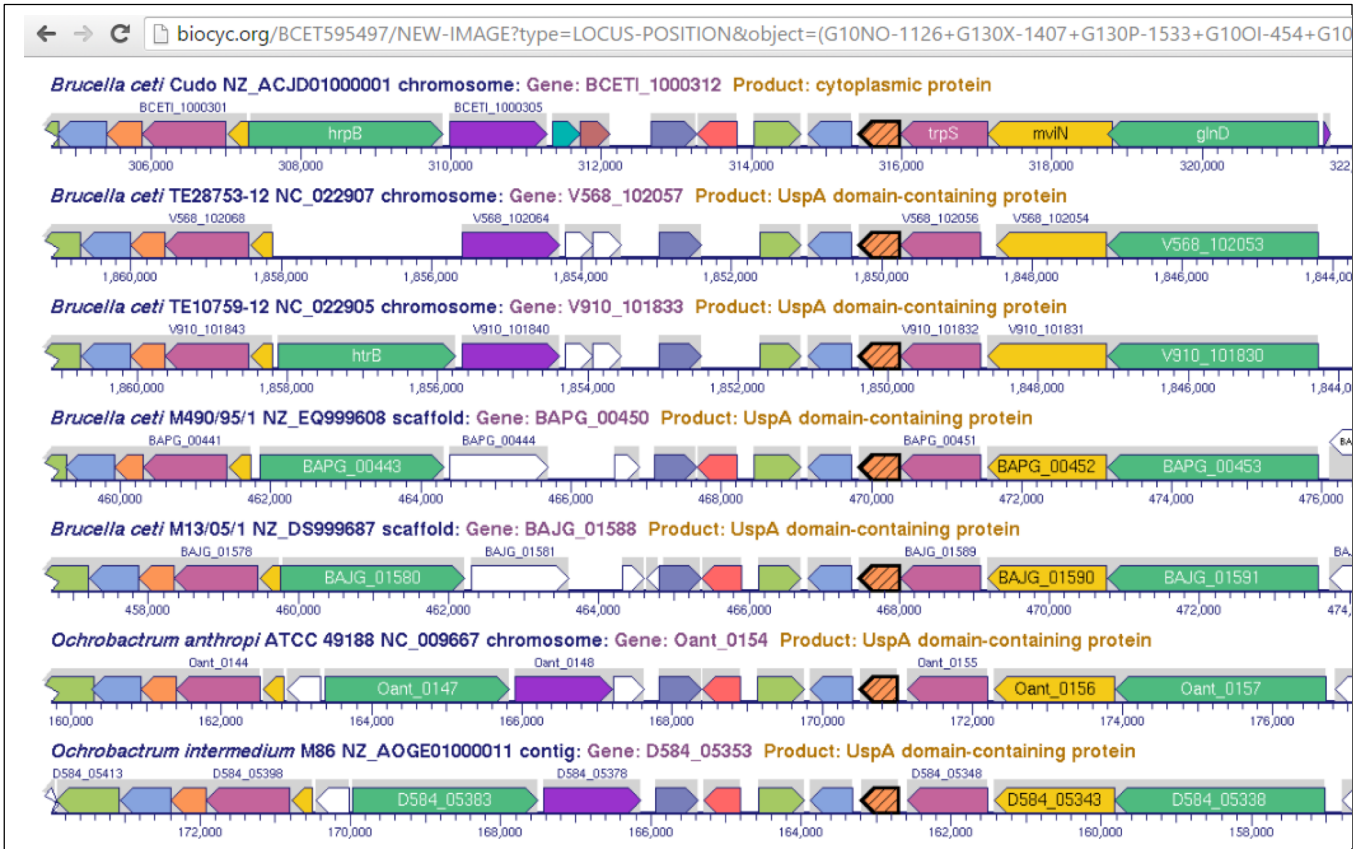


Figure 4. Multi-Genome alignment of the gene neighborhood of predicted genes for universal stress proteins in genomes of *Brucella ceti* and *Ochrobactrum* species. The genes for universal stress proteins have diagonal lines.

3.5 Interactive Analytics for Analytical Reasoning on *Brucella ceti* Transcription Units containing Gene for Universal Stress Protein

Analytical reasoning "is based on rational, logical analysis and evaluation of information" as well as "a structured, disciplined activity" [7]. We performed analytical reasoning on the multi-genome alignment of the gene neighborhood of 37 *Brucellae* genomes in BioCyc. The interactive alignment is can be constructed at BioCyc.org. We used the *B. ceti* Cudo four-gene transcription unit as template to analyze the presence and composition of transcription units and subsequently evaluate the level of conservation of the genomic region between *Brucella ceti* and *Ochrobactrum* genomes (Figure 4). The finding that BCETI_1000312 and BCETI_1000311 are not an operon was confirmed with a multi-genome alignment of the gene neighborhood. Among the *Brucella ceti* genomes, strain Cudo is unique for having the 4-gene transcriptional unit, which consists of genes for universal stress protein, tryptophanyl-tRNA synthetase, peptidoglycan biosynthesis protein and protein-P-II uridylyltransferase, a regulator of nitrogen status of *Escherichia coli* [24].

4. DISCUSSION

4.1 Information Space on Strand Location of Genes

We developed a computational workflow that led to a reduction in the complexity of 21,139 genome annotation files from 16 fields to 4 fields. This complexity reduction process implemented involved algorithmic operations including sorting and comparisons that required high performance computing resources. There is growing need for use of supercomputing resources and cloud computing in bioinformatics [25, 26]. The derived information space enabled a variety of complex cognitive tasks to be performed with desktop visual analytics software as well as online bioinformatics software.

Our research used the RefSeq genome annotation files. PATRIC bioinformatics resource includes re-annotated versions of microbial genomes [4]. Therefore, the computational protocols that we have developed on the Blue Waters Supercomputer [17] for deriving new information space on strand location of genes can be adapted for the PATRIC genome annotation files (with extension PATRIC.cds.tab). We expect to obtain additional genomes and gene loci. For example, our information space included 547 Rhizobiales genome annotation files. Based on statistics available at the PATRIC website (patricbrc.org), we expect to have at least 1441 Rhizobiales genomes. A web-based

4.2 Interactive Analytics for Sense Making on Protein-Coding Genes in Rhizobiales

As shown in Figure 1, among the seven *Brucella ceti* strains, 3 strains had excess of at least 50 genes mapped to the negative strand. The M13/05/1 strain has the largest difference in number of mapped genes, at 209 genes. This may indicate that certain genes have been recently duplicated, or that groups of genes were transferred from one strand to another, thereby providing a user with information to generate testable hypotheses.

The integration of information space on strand location with other annotation files enabled us to make sense of the distributions of the gene counts for genera in the Rhizobiales (Figure 3). We chose to use the box plot technique since the technique is suitable to visually summarize and compare groups of data [27]. A finding from the box plot visual representation (Figure 3) is that methanol-oxidizing *Methylobacterium nodulans* ORS 2060, the legume (*Crotalaria*) root-nodule-forming and nitrogen-fixing bacteria [28], has at least 7 sequenced plasmids [29]. The possession of an intact 120kb megaplasmid correlated with ability of *Methylobacterium extorquens* DM4 to utilize dichloromethane as sole source of carbon and energy [30]. Comparative analysis of the genes in the plasmids of *Methylobacterium* species could improve understanding of methylotrophy and nitrogen-fixation.

Rhodopseudomonas palustris TIE-1 has an upper outlier gene count among the *Rhodopseudomonas*. Further research could investigate the function of the additional genes in the iron oxidizing *R. palustris* strain [31].

4.3 Interactive Analytics for Decision Making on Genomes for Investigation

The comments associated with eight *Brucella ceti* genomes (Table 1) helped us decide to further investigate the genome of *Brucella ceti* Cudo, a dolphin associated *Brucella* [32, 33]. In the BioCyc pathway databases, a transcription unit is a set of one or more genes that are transcribed to produce a single messenger RNA [34]. Our research interest is in multi-gene transcription units which include at least one gene for universal stress protein. Four genes for universal stress proteins were observed in the genome of *B. ceti* Cudo. We have not observed reports describing the function of the *B. ceti* USPs. Therefore, this report provides new insights into the organization of transcription units and possible function of *B. ceti* USPs. [35]. The decision making then led to analytical reasoning of the gene neighborhood of *B. ceti* USP transcription units.

4.4 Interactive Analytics for Analytical Reasoning on *Brucella ceti* Transcription Units containing Gene for Universal Stress Protein

Among the *Brucella ceti* genomes, strain Cudo is unique for having the 4-gene transcriptional unit, which consists of genes for universal stress protein, tryptophanyl-tRNA synthetase, (pfam 00579), peptidoglycan biosynthesis protein (pfam03023) and protein-P-II uridylyltransferase (pfam08335) (Figure 2). There is a need for research studies to confirm the existence of the 4-gene transcription unit as well as the role of each gene. A common annotated function of the proteins encoded by the transcription unit is metabolism of nitrogen. The universal stress proteins are induced in response to stress conditions including nitrogen starvation [36-38]. Tryptophanyl-tRNA synthetase (TrpRS) ensures the translation of the genetic code for tryptophan, a nitrogen containing amino acid, by catalyzing the activation of tryptophan by adenosine triphosphate (ATP) and transfer to the tryptophanyl-tRNA ($tRNA^{Trp}$) [39].

The peptidoglycan biosynthesis protein in *Escherichia coli* is a lipid II flippase essential for cell wall peptidoglycan synthesis [40]. The protein-P-II uridylyltransferase (GlnD) is involved in glutamine metabolism and primary sensor of nitrogen [41]. In *Mycobacterium tuberculosis*, an intracellular pathogen as *Brucella* species, L- glutamine is a major component of the cell wall [42] and a source of nitrogen in *Brucellae* [43]. An immune response in mammalian cells for the control of intracellular pathogens includes the gamma interferon induced production of indoleamine 2,3-dioxygenase (IDO), an enzyme for the degradation of tryptophan [44]. The transcription direction of the four genes is conserved in the two *Ochrombactrum* genomes (Figure 3). Furthermore the functions for peptidoglycan synthesis and

nitrogen sensing exist as a transcription unit in both *Ochrombactrum* genomes and four of the five *B. ceti* genomes. In summary, there is evidence that the function of the transcription unit in the *Brucella ceti* Cudo genome that contains the gene BCETI_1000312 is for nitrogen stress response.

5. CONCLUSIONS

The goal of the research reported in this article was to develop interactive analytics resources to support the performance of complex cognitive activities on a collection of publicly available genome information spaces. Our expectation is that the information spaces and interactive views present opportunities for learning about the microbial genomes. An overview of the resources developed is presented in the figure in the Appendix section.

A supercomputing infrastructure (Blue Waters Supercomputer) provided computational tools to construct information spaces while visual analytics software and online bioinformatics resources provided tools to interact with the constructed information spaces. The Rhizobiales order of bacteria that includes the *Brucella* genus was the use case for preforming the complex cognitive activities. An interesting finding among the *Brucella ceti* genomes was that strain Cudo is unique for a predicted four-gene transcriptional unit that contain genes known to respond to limited nitrogen availability.

6. REFLECTIONS

6.1 Dominique Smith-McInnis, PhD Candidate in Environmental Science at Jackson State University, Mississippi.

The main goal of my doctoral research is to generate knowledge on the biological processes in *Brucella* species that include universal stress proteins. I am a recipient of career development fellowships from the Institute for Infectious Animal Diseases, a Department of Homeland Security Science & Technology Center of Excellence at Texas A & M University. I have conducted biological research using computational techniques and resources. This manuscript shows examples of knowledge on universal stress proteins of *Brucella* species that were generated using bioinformatics and visual analytics tools. The learning experiences from my doctoral research has equipped me for a career in K-12 education and higher education.

6.2 Kiara Wootson, Undergraduate Student in the College of Science, Engineering and Mathematics, Bethune-Cookman University, Daytona Beach, Florida.

I was an intern in Blue Waters Internship Program from May 2015 to April 2016. At the beginning of the internship I attended the two-week Blue Waters 2016 Petascale Institute held at the University of Illinois Urbana-Champaign (UIUC) from May 24th to June 5th 2015. I gained an introduction to high performance computing. During my mentored internship at Bethune-Cookman University I became familiar with command-line instructions for performing computing actions. The internship training has helped me to better understand microbial genomes as well as data visualization techniques. I have a clearer understanding of career pathways that incorporate computational science.

7. ACKNOWLEDGMENTS

National Science Foundation Grant Award: HRD-1435186. KMW and RDI acknowledge the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. RDI, DRSM and SSS acknowledge funding support from the U.S. Department of Homeland Security Science and Technology Directorate: 2011-ST-062-000048. DRSM acknowledge Fellowship from National Center for Foreign Animal and Zoonotic Disease Defense and the Environmental Science PhD Program. **Disclaimer:** "The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the funding agencies".

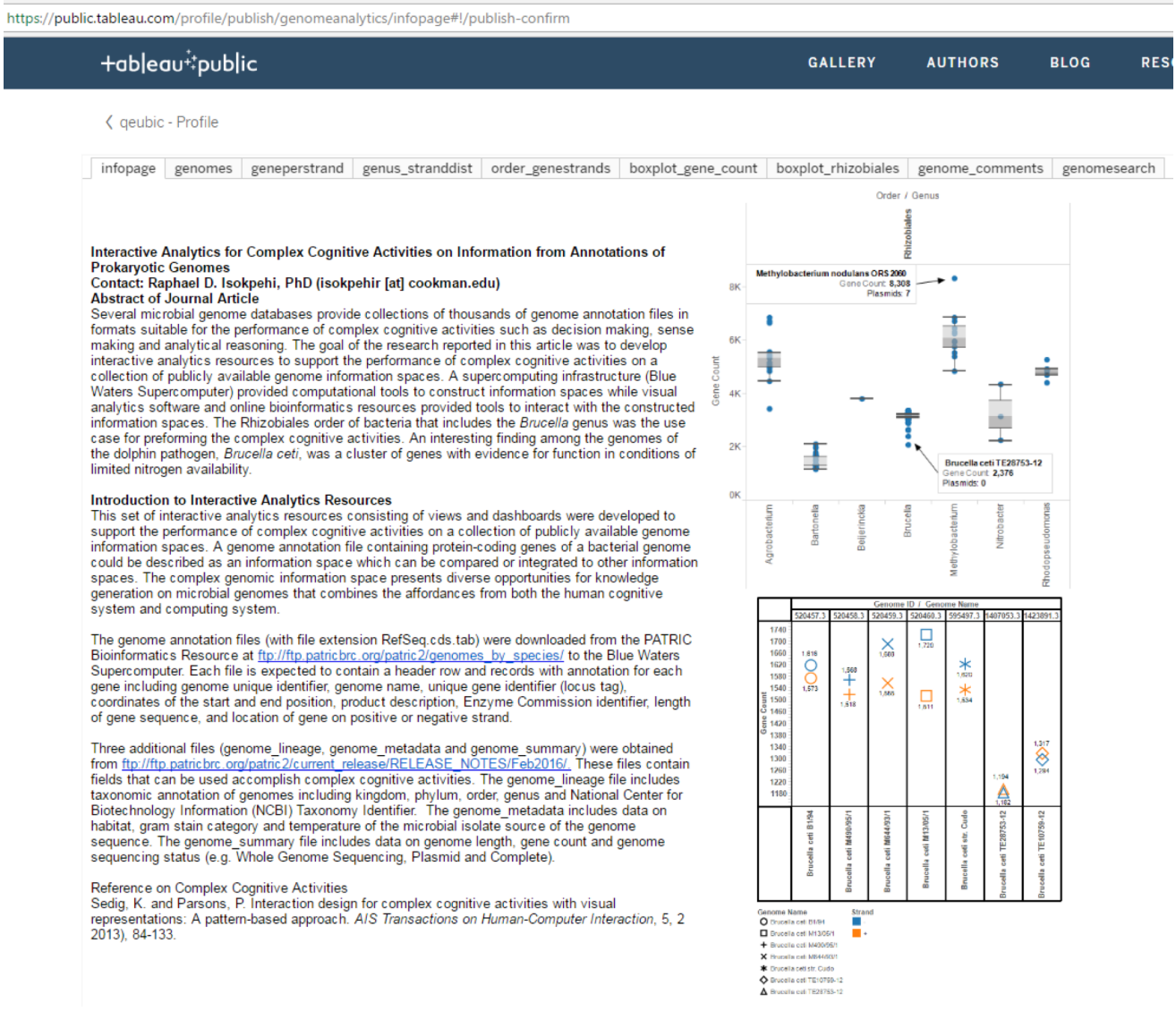
8. REFERENCES

1. Chen I-MA, Markowitz VM, Palaniappan K, Szeto E, Chu K, Huang J, Ratner A, Pillay M, Hadjithomas M, Huntemann M: **Supporting community annotation and user collaboration in the Integrated Microbial Genomes (IMG) system**. *BMC Genomics* 2016, **17**:307.
2. Mavromatis K, Ivanova NN, Chen I-MA, Szeto E, Markowitz VM, Kyrpides NC: **The DOE-JGI Standard operating procedure for the annotations of microbial genomes**. *Standards in Genomic Sciences* 2009:63-67.
3. Tatusova T, Ciufo S, Fedorov B, O'Neill K, Tolstoy I: **RefSeq microbial genomes database: new representation and annotation strategy**. *Nucleic Acids Research* 2013:D553-D559.
4. Wattam AR, Abraham D, Dalay O, Disz TL, Driscoll T, Gabbard JL, Gillespie JJ, Gough R, Hix D, Kenyon R: **PATRIC, the bacterial bioinformatics database and analysis resource**. *Nucleic Acids Research* 2013:D581-D591.
5. Albers MJ: **Human–Information interaction with complex information for decision-making**. *Informatics* 2015, **2**(2):4-19.
6. Parsons P, Sedig K: **Distribution of information processing while performing complex cognitive activities with visualization tools**. In: *Handbook of Human Centric Visualization.* Springer; 2014: 693-715.
7. Sedig K, Parsons P: **Interaction design for complex cognitive activities with visual representations: A pattern-based approach**. *AIS Transactions on Human-Computer Interaction* 2013, **5**(2):84-133.
8. Di Martino C, Kalbarczyk Z, Iyer RK, Baccanico F, Fullop J, Kramer W: **Lessons learned from the analysis of system failures at petascale: The case of Blue Waters**. In: *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on: 2014*: IEEE; 2014: 610-621.
9. Sacha D, Stoffel A, Stoffel F, Kwon BC, Ellis G, Keim DA: **Knowledge generation model for visual analytics**. *Visualization and Computer Graphics, IEEE Transactions on* 2014, **20**(12):1604-1613.
10. Sedig K, Parsons P: **Design of visualizations for human-information interaction: A pattern-based framework**. *Synthesis Lectures on Visualization* 2016, **4**(1):1-185.
11. Sedig K, Parsons P, Dittmer M, Haworth R: **Human-centered interactivity of visualization tools: Micro-and macro-level considerations**. In: *Handbook of Human Centric Visualization.* Springer; 2014: 717-743.

12. Isokpehi RD, Udensi UK, Simmons SS, Hollman AL, Cain AE, Olofinsae SA, Hassan OA, Kashim ZA, Enejoh OA, Fasesan DE: **Evaluative profiling of arsenic sensing and regulatory systems in the human microbiome project genomes**. *Microbiology Insights* 2014, **7**:25-34.
13. Mbah AN, Isokpehi RD: **Application of universal stress proteins in probing the dynamics of potent degraders in complex terephthalate metagenome**. *BioMed Research International* 2013, **2013**:196409.
14. Carvalho FM, Souza RC, Barcellos FG, Hungria M, Vasconcelos ATR: **Genomic and evolutionary comparisons of diazotrophic and pathogenic bacteria of the order *Rhizobiales***. *BMC Microbiology* 2010, **10**:37.
15. Erlacher A, Cernava T, Cardinale M, Soh J, Sensen CW, Grube M, Berg G: **Rhizobiales as functional and endosymbiontic members in the lichen symbiosis of *Lobaria pulmonaria* L**. *Frontiers in Microbiology* 2015, **6**:53.
16. Parte AC: **LPSN—list of prokaryotic names with standing in nomenclature**. *Nucleic Acids Research* 2014, **42**(D1):D613-D616.
17. Bode B, Butler M, Dunning T, Gropp W, Hoe-fler T, Hwu W-m, Kramer W: **The Blue Waters Super-System for Super-Science. Contemporary HPC Architectures, Jeffery Vetter editor**. In.: Sitka Publications, November; 2012.
18. Lurie NH, Mason CH: **Visual representation: Implications for decision making**. *Journal of Marketing* 2007, **71**(1):160-177.
19. Simmons SS, Isokpehi RD, Brown SD, McAllister DL, Hall CC, McDuffy WM, Medley TL, Udensi UK, Rajnarayanan RV, Ayensu WK: **Functional annotation analytics of *Rhodopseudomonas palustris* genomes**. *Bioinformatics and Biology Insights* 2011, **5**:115-129.
20. Moreno E, Guzmán-Verri C, Gonzalez-Barrios R, Hernandez G, Morales JA, Barquero-Calvo E, Chaves-Olarte E: ***Brucella ceti* and brucellosis in cetaceans**. *Frontiers in Cellular and Infection Microbiology* 2012, **2**:3.
21. Parsons P, Sedig K: **Common visualizations: Their cognitive utility**. *Handbook of Human Centric Visualization* 2014:671-691.
22. Isokpehi RD, Simmons SS, Cohly HH, Ekunwe SI, Begonia GB, Ayensu WK: **Identification of drought-responsive universal stress proteins in viridiplantae**. *Bioinformatics and Biology Insights* 2011, **5**:41-58.
23. Caspi R, Billington R, Ferrer L, Foerster H, Fulcher CA, Keseler IM, Kothari A, Krummenacker M, Latendresse M, Mueller LA: **The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases**. *Nucleic Acids Research* 2016, **44**(D1):D471-D480.
24. Tøndervik A, Torgersen HR, Botnmark HK, Strøm AR: **Transposon mutations in the 5′ end of glnD, the gene for a nitrogen regulatory sensor, that suppress the osmosensitive phenotype caused by otsBA lesions in *Escherichia coli***. *Journal of Bacteriology* 2006, **188**(12):4218-4226.
25. Dumancas GG: **Applications of supercomputers in sequence analysis and genome annotation**. *Research and Applications in Global Supercomputing* 2015:149-175.
26. Singh P: **Big genomic data in bioinformatics cloud**. *Applied Microbiology: Open Access* 2016, **2**(1000113):2.
27. Williamson DF, Parker RA, Kendrick JS: **The box plot: a simple visual method to interpret data**. *Annals of Internal Medicine* 1989, **110**(11):916-921.
28. Jourand P, Giraud E, Béna G, Sy A, Willems A, Gillis M, Dreyfus B, de Lajudie P: ***Methylobacterium nodulans* sp. nov., for a group of aerobic, facultatively methylotrophic, legume root-nodule-forming and nitrogen-fixing bacteria**. *International Journal of Systematic and Evolutionary Microbiology* 2004, **54**(6):2269-2273.
29. Marx CJ, Bringel F, Chistoserdova L, Moulin L, Haque MFU, Fleischman DE, Gruffaz C, Jourand P, Knief C, Lee M-C: **Complete genome sequences of six strains of the genus *Methylobacterium***. *Journal of Bacteriology* 2012, **194**(17):4746-4748.
30. Gäll R, Leisinger T: **Plasmid analysis and cloning of the dichloromethane-utilization genes of *Methylobacterium sp.* DM4**. *Microbiology* 1988, **134**(4):943-952.
31. Jiao Y, Kappler A, Croal LR, Newman DK: **Isolation and characterization of a genetically tractable photoautotrophic Fe (II)-oxidizing bacterium, *Rhodopseudomonas palustris* strain TIE-1**. *Applied and Environmental Microbiology* 2005, **71**(8):4487-4496.
32. Wu Q, McFee WE, Goldstein T, Tiller RV, Schwacke L: **Real-time PCR assays for detection of *Brucella* spp. and the identification of genotype ST27 in bottlenose dolphins (*Tursiops truncatus)***. *Journal of Microbiological Methods* 2014, **100**:99-104.
33. Setubal C, Brettin T, Sobral BW, Boyle SM, Tsolis J, Munk C, Tapia R, Han C, Detter J, Bruce D: **Genomes reveals *Brucella* analysis of ten**. *Journal of Bacteriology* 2009, **191**(11):3569.
34. Caspi R, Billington R, Foerster H, Fulcher CA, Keseler I, Kothari A, Krummenacker M, Latendresse M, Mueller LA, Ong Q: **BioCyc: Online Resource for Genome and Metabolic Pathway Analysis**. *The FASEB Journal* 2016, **30**(1 Supplement):lb192-lb192.
35. Williams BS, Isokpehi RD, Mbah AN, Hollman AL, Bernard CO, Simmons SS, Ayensu WK, Garner BL: **Functional annotation analytics of *Bacillus* genomes reveals stress responsive acetate utilization and sulfate uptake in the biotechnologically relevant *Bacillus megaterium***. *Bioinformatics and Biology Insights* 2012, **6**:275-286.
36. Gumber S, Taylor DL, Marsh IB, Whittington RJ: **Growth pattern and partial proteome of *Mycobacterium avium* subsp. paratuberculosis during the stress response to hypoxia and nutrient starvation**. *Veterinary Microbiology* 2009, **133**(4):344-357.
37. Kvint K, Nachin L, Diez A, Nyström T: **The bacterial universal stress protein: function and regulation**. *Current Opinion in Microbiology* 2003, **6**(2):140-145.
38. Gustavsson N, Nyström T: **The universal stress protein paralogues of *Escherichia coli* are co-ordinately regulated and co-operate in the defence against DNA damage**. *Molecular Microbiology* 2002, **43**(1):107-117.
39. Doublié S, Bricogne G, Gilmore C, Carter Jr CW: **Tryptophanyl-tRNA synthetase crystal structure reveals an unexpected homology to tyrosyl-tRNA synthetase**. *Structure* 1995, **3**(1):17-31.
40. Ruiz N: **Bioinformatics identification of MurJ (MviN) as the peptidoglycan lipid II flippase in *Escherichia coli***. *Proceedings of the National Academy of Sciences* 2008, **105**(40):15553-15557.
41. Yurgel SN, Rice J, Kahn ML: **Transcriptome analysis of the role of GlnD/GlnBK in nitrogen stress adaptation by**

Sinorhizobium meliloti **Rm1021**. *PloS One* 2013, **8**(3):e58028.
42. Pashley CA, Brown AC, Robertson D, Parish T: **Identification of the *Mycobacterium tuberculosis* GlnE promoter and its response to nitrogen availability**. *Microbiology* 2006, **152**(9):2727-2734.
43. Ronneau S, Moussa S, Barbier T, Conde-Álvarez R, Zuniga-Ripa A, Moriyon I, Letesson J-J: ***Brucella*, nitrogen and virulence**. *Critical Reviews in Microbiology* 2014:1-19.
44. Schaible UE, Kaufmann SH: **A nutritive view on the host–pathogen interplay**. *TRENDS in Microbiology* 2005, **13**(8):373-380.

9. APPENDIX



Interactive Analytics Resources for Complex Cognitive Activities on Information from Annotations of Prokaryotic Genomes
Website: https://public.tableau.com/profile/publish/genomeanalytics/infopage
This set of interactive analytics resources consisting of views and dashboards were developed to support the performance of complex cognitive activities on a collection of publicly available genome information spaces.

How to Build a Fast HPC n-Body Engine From Scratch

Eric Peterson[1]
Marmion Academy
1000 Butterfield Rd
Aurora, IL 60502
epeterson@marmion.org

Max Kelly[2]
Rose Hulman Institute of Technology
5500 Wabash Ave
Terre Haute, IN 47803
maxwellrkelly@gmail.com

Dr. Victor Pinks II[3]
Marmion Academy
1000 Butterfield Rd
Aurora, IL 60502
vpinks@marmion.org

ABSTRACT
Communicating and transferring computational science knowledge and literacy is a tremendously important concept for students at all levels of education to understand. Computational knowledge is especially important due to the tremendous impact that computer programming has had on all scientific and engineering disciplines. As technology evolves, so must our educational system in order for society to evolve as a whole. We undertook direct instruction of a computational science course, and have developed a curriculum that can be expanded upon to provide students entering technical disciplines with the background that they need to be successful. The course would provide insight to the C programming language as well as how computers function at a more basic level. Students would undertake projects that explores how to program simple tasks and operations and ultimately ends in a final project aimed at assessing the knowledge accumulated from the course.

CCS Concepts
• *Social and professional topics* • *Social and professional topics~Computing education* • Social and professional topics~K-12 education

Keywords
Computing education, K-12 Education

1. INTRODUCTION
Computational science as a discipline uses modern tools of computer science combined with a mathematical approach to problem solving to tackle issues that are relevant to science. Computational science requires knowledge from three disparate fields: computer and information sciences, which is used to develop the software and data structures to solve computationally interesting problems, numerical and non-numerical approaches to modeling, which can be used to represent data for scientific problems, and computing infrastructure that the software can be run on.

A problem of scientific interest is usually first understood in terms of a model that predicts or explains observation, which serves as a template for software engineers to develop a computer program that allows the model to be studied, and then is finally executed on a computing platform. Virtual modeling helps to see the expected outcome of experiments and ideas without real world execution. This helps save time and money. Today companies utilize computational science to do just that. Other uses of computational science and virtual modeling are to view objects and interactions that are difficult to view under normal circumstances. Particle interaction is the best example of this. The class that this paper will help outline was all about modeling a particle, something that cannot be seen by the naked eye, as it moved through space.

Computational Science is not widely taught. There are schools that will cover one of the fields of computational science in great detail, but will provide little or no detail on the other two. The goal of this course was to provide a strong foundation in all three areas.

DOI: https://doi.org/10.22369/issn.2153-4136/8/2/6

2. RELATED WORK
The approach that was developed at Marmion Academy was heavily based upon the instruction that the SHODOR educational foundation developed for use in the Blue Waters Student Internship and the Petascale Institute. The approach of the institute combined lectures on the theory of parallel programming and high-performance computing with practical exercises that reinforced the concepts. We sought to adapt this approach for use at Marmion Academy, and used the XSEDE training roadmap available from HPC university as a starting point to developing computational literacy in our students.

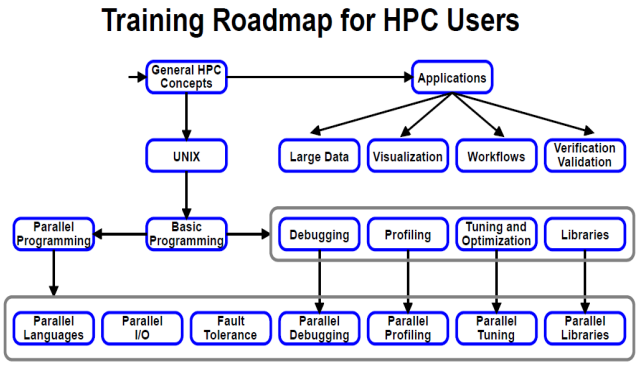


Figure 1. XSEDE Training Roadmap

3. CHALLENGES

There are some significant challenges that are faced when introducing a computational science curriculum into a high school environment. The primary challenge faced is the lack of computing backgrounds amongst the majority of the course enrollees. Most students taking the course do not have any familiarity with programming languages, none do they have any exposure to a Linux operating system, and few have taken the mathematic and scientific coursework necessary to understand the scientific models that will be covered. Also, the high school format with 45 minute class periods limits the amount of time that instructors have to cover new material. With crowded student course schedules, students have very little time outside of class to self-study or prepare.

The students' lack of knowledge and available time greatly hindered the results of the course. Several were able to learn the material, but almost all struggled in applying the material even at the end of the year. Students lost interest in the class as the year progressed. Additionally, majority of the students enrolled in the course were seniors and became afflicted with senioritis making it difficult for them to learn and pay attention. As stated previously, students had problems applying concepts that had been introduced throughout the entire year. The same questions were constantly being asked by students showing their lack of ability to grasp what the class was teaching.

Several students had difficulty with the language as well. The C programming language has a lot of rules and syntax to it. Students were very confused with how to structure their code and irked the students once they figured out how "dumb" a computer really is. Several students that did express interest in the class say that the difficulty of the language being used dissuaded them from pursuing a career in computational science in the future. The environment they were programming in, Cygwin, also confused the students because it used keystrokes that the students were unfamiliar with. It caused a lot of frustration for the students since they might accidentally delete a line of code due to the unfamiliar key strokes.

4. COURSE OUTLINE

Using the roadmaps available from HPC University, we developed a prototype curriculum for the course. The curriculum was developed with the express goal of introducing high school students with no computing background. We began with a conceptual introduction to the ideas of high performance computing, computer architectures, parallel programming, and data visualization. We would then proceed with a tutorial of the Linux command-line interface, along with the basic skillset needed to run and submit jobs on the Blue Waters system. The program used for this was Cygwin due to its ability to emulate a Linux environment and because of the instructor's familiarity with it. Next, we would introduce a computer programming language that the students would use when implementing the algorithms that model solutions to scientific problems. The language that we chose for this course was the C programming language, due to the large codebase of examples from the Petascale Institute and the language's high degree of hardware optimization. Throughout the course, topics relevant to software development would be introduced, such as best practices, debugging, libraries, and profiling. This was done through lectures and class examples as well as projects that the students would work on throughout the week. The projects would implement the topics discussed in class in order to help students understand their importance. The difficulty of the projects was extremely low due to the constant difficulty in understanding the C programming language and issues with the Cygwin environment. After the students were confident in their knowledge of basic programming, we would then proceed to parallel programming concepts and techniques using resources such as MPI. Due to time constraints and overall difficulty in the instruction the students were not able to receive instruction in parallel programming. The course would conclude with a project that would encompass all they had learned throughout the course. We decided that the final project should be an N-Body simulation of a particle's position on a Cartesian Plane in order for the students to demonstrate a proper amount of knowledge from the course. This project would involve computing the continual position of the particle using loops and dynamically updating the variables involved with the particle's position as well as printing to the screen. The final project would be graded based on how well constructed the student's code was able to output the results, the accuracy of the results, the student's understanding after a small Q&A, and finally if the code was well documented. All percentages for appropriate grading scales as well as a more documented step by step walkthrough of how the students would be taught are shown in the attached course syllabus after the acknowledgements and references.

5. COURSE INSTRUCTION

The course ran for 40 weeks, with one 45-min session every day. We utilized a project-based teaching method, through which the students were graded based on their ability to work through and complete in-class practical projects and materials. Each week consisted of a combination of the following: lectures on new concepts and new materials, practical lectures that the students could follow along with, or in-class projects and exercises. We also needed a development environment that would simulate the Blue Waters development environment, and for this role we used Cygwin. Cygwin allows us to simulate a Linux environment on a Windows system, and allows for the student to practice the basics of the Linux command-line, which is the only interface that they would be exposed to on Blue Waters. Resources from the Petascale Institute were used to familiarize the students with job processing on Blue Waters, include the "Time to Science" demonstration intended to demonstrate the performance benefits of increasing node size on a job. A basic workflow guide was also created by the course instructors for use as a simple in-class tutorial on basic operations on the Blue Waters system.

The first semester was extremely different from the second. The instructors decided that the students would be graded on completion rather than an assessment. This changed at the end of the first semester because the students were prioritizing other classes since they were finding the class extremely easy. It was found that students were not learning the material and constantly asking questions that they should have been able to answer. Second semester saw a lot of grading on projects that was not seen previously. While this was met with a lot of frustration from the

1 - Undergraduate Student

2 - Undergraduate Student

3 - Principal Investigator

students, they did begin to take the class more seriously and focused on learning and becoming more proficient with the C language.

6. COURSE RESULTS

The final project for the course was for the students to create an n-body simulator using the skills they had learned in the class throughout the year. The students were given three weeks to work on the simulator. The goal was to accept input for a time function and then track the position of a particle on a standard Cartesian plane. The students were required to print out a graphic inside of a terminal showing the coordinates of the particle in relation to the origin. The equations that the students used to model the particle were explained. Debugging and algorithm help were also provided as well.

As stated in Section 3, the students had difficulty throughout the course. Due to the ease of use with modern technology, students had difficulty grasping the basic implementations and syntax of the C programming language. Their lack of knowledge in programming greatly hindered the progress of the course. Questions were continuously asked throughout the year about topics that were extremely basic and demonstrated in every project. Examples include the scope of a variable, variable assignment, and syntax for both for and while loops.

The project proved very challenging for several of the students. The most issues became clear when the students had to figure out how to graph the particle's position. Students were familiar with printing out to the screen by the end of the course, but were unsure how to do it with continuous updates to an object as it changed position. Others had problems figuring out how to use the C programming language to accomplish what the project required and were much slower to finish.

Overall the project was a moderate success. The students had the most problems with learning the C language because of its foreign nature. Students constantly struggled with basic concepts such as creating variables or managing the scope of a program. The students constantly asked the same questions such as "How do I make this?" for extremely basic concepts such as object and variable creation or Boolean logic. After interviewing some of the students, it was determined that most of the issues encountered were because of the approach taken in the first semester where grading was not weighted as heavily. Instruction could not be slowed down without sacrificing time to teach other concepts that would be needed for use in the final project. Students were also extremely unfamiliar with programming syntax. Several students were able to overcome this obstacle, but others struggled up until the very end of the class.

These faults do not hinder the results of the class. Of the original twelve students that were enrolled in the course, only a single student dropped the course. The student who dropped did not drop for academic purposes, but for issues with scheduling. All the other students who completed the course finished with very good grades due to their ability to finish the projects assigned and demonstrate all knowledge required of them. All of the students were still able to complete the final project and all had extremely good results. Several of the students also became extremely adept at utilizing the C programming language for their own use. Additionally, several of the students expressed interest in going into an engineering or programming related field, which require computational science knowledge. Students expressed extreme satisfaction in being able to learn programming since it will almost definitely help them as they are transitioning to college.

7. CONCLUSIONS

Throughout the course there were several issues that had to be overcome and others that could not be accounted for. Were this course to be taught in the future, we have listed several issues that we encountered and solutions that we feel would be most effective.

- We feel that the course instruction has been informative on how to best approach future computational science instruction. To improve and extend our instruction into the future, we feel that more emphasis be placed upon the applications of computational models, and less emphasis be placed upon computer programming instruction.
- Computer programming instruction requires a significant amount of time to adequately prepare the student, and thus is not a proper use of limited class time when the focus is to prepare students for the applications of computational science. It is therefore more appropriate to require the student to be comfortable with a programming language prior to entering the course, or to learn how to program outside of class time.
- Another option is to use a language that is simpler to use and requires less class time to attain familiarity with. Our primary candidate for a simple language included Scientific Python.
- Based on frustrations programming in a command-line environment, we recommend the use of an integrated development environment (IDE) when developing applications on a local system, and the use of command-line tools only on remote systems.
- A more rigorous grading methodology should be used in order to properly motivate the students and encourage active student participation.

8. REFLECTIONS

I feel that this experience has been very beneficial to me in my academic and career development, and I will take away a great deal from this work. The Petascale Institute especially was a tremendous opportunity to expand and polish my knowledge using the resources that SHODOR had available. The chance to work with colleagues from other institutions and learn from experts in the field of computational science education was an enlightening experience. I look forward to the chance to continue using the skills I gained from Blue Waters in my future academic work.

9. ACKNOWLEDGMENTS

I would like to thank Marmion Academy for all their help and support, especially Dr. Victor Pinks II for his guidance and expertise. I would also like to thank the teaching staff from SHODOR, whose codebase and teaching materials heavily inspired this course. Acknowledgements would not be complete without Max Kelly, whose help authoring this publication was invaluable. This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National

Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.

10. REFERENCES

[1] *Blue Waters Petascale Institute,* Available at: *http://shodor.org/petascale/workshops/bw2015/*

[2] *Cygwin Project,* Available at: https://www.cygwin.com

[3] *Unix Tutorial*, Available at: *http://www.tutorialspoint.com/unix/*

[4] *XSEDE Roadmap Training*, Available at: http://hpcuniversity.org/RoadmapSite/index/

[5] Steve Qualline (1997) *Practical C Programming*, 3 edn., : O'Reilly Media.

[6] Dr. Victor Pinks (2015), *Fall 2015 Marmion Academy CT-STEM Syllabus*

[7] Eric Peterson (2015) *A Very Basic Blue Waters Workflow Guide*

[8] Aaron Weeden (2015) *Time to Science Instructions,* Available at:

http://shodor.org/~aweeden/TimeToScience.pdf

[9] *BCCD Tutorials,* Available at: http://bccd.net/wiki/index.php/Tutorials

Appendix 1. Fall 2015 CT-STEM Syllabus

1st Semester - Fall 2015

Computational Science & Engineering (CT-STEM) Syllabus & Guidelines

Section 1

Textbook: Curricular outline will coincide with materials from the Computational Science Institute (http://www.computationalscience.org/) , the CT-STEM program designs at https://osep.northwestern.edu/projects/ct-stem , and from resources on HPC University (http://hpcuniversity.org/)

This is an introductory level course on the application of computational thinking to the solution of general science problems guided by the inherent processes of the Scientific Method. This course draws on the three pillars of science – theory, computation and experiment. Computational elements will be performed using the programming language of C that will be taught at an introductory level. Physical experiments will be performed to enhance computer simulation experiments as needed. This course is an introduction of computational thinking principles in a problem-based learning environment. A major piece of the learning process will be focused on using high performance computing (HPC) as the platform for computational science & engineering modeling via the Blue Waters system (see: http://www.shodor.org/petascale/). Some prototyping of HPC code will be performed on the Marmion Academy Raspberry Pi HPC cluster. Grading is formative.

Note: *Pre-requisites: Pre-Calculus with Trigonometry (completed or currently enrolled)*

- **Topic by Order**
- What is Computational Science and why?
- Federal, State and Local efforts to promote computational science
- Basic pre-calculus and basic linear algebra math review
- The Scientific Method as built on the three pillars of science (theory, computer simulation, and experiment)
- Outlining the n-body simulation semester project (why and how)

- General HPC concepts
- Basic Linux/Unix overview and command-line interface tutorial
- Overview of computer programming and C language basics (more advanced elements are taught in context of the project)
- How to log into and use HPC resources like Blue Waters and the Raspberry Pi HPC cluster
- Test run sample C code

- Introduction to parallel programming and parallel programming languages like MPI, OpenMP, CUDA
- Debugging, profiling, and optimization of serial and parallel code

- Creating a single particle (1-body) 3D simulation (step-by-step)

  - Introducing the creation of computer representations of a single classical particle in a box
  - Moving the particle through numerical integration
  - Moving the particle in 1D then a single particle in 3D
  - Applying Periodic Boundaries to the box
  - Running the simulation with C and visualizing the output

- Creating a many particle (n-body) 3D simulation (step-by-step)
  - Moving from 1D to 3D simulations – considerations of force
  - "It's all about forces" lecture
  - Realism of computer simulations – "If the forces are realistic, the simulation will be realistic" lecture
  - How we use physical experiments to improve computer simulation realism
  - Running the 3D n-body simulation for :
    - Planetary bodies and astronomical size simulations
    - Monatomics (homogeneous and heterogeneous systems)
    - Hard spheres to simulate macroscopic objects in our 'big' world
    - Chemical simulations and experiments (how they compare)
    - Biological simulations and experiments (how they compare)

- Biological simulation of wolf and sheep populations
- Chemical statistics of motion as applied to the Boltzmann distribution of velocities

All programming projects and physical experiment labs are defended with an oral presentation and examination immediately after completion of the write-up. All lab reports will follow the MLA research report standards. Bad grammar, plagiarism and spelling will also be checked /scored

Grading

Grading is calculated against a formative project-based model. Emphasis is placed on quality and timely completion of assignments. Homework grades are reduced 10% for each day late past the assigned due date. Procrastination and/or poor time management can have the most devastating effect on your grade. Stay on task. It's about following the path and doing the work. You will be given multiple attempts (except on exams and quizzes) to make them acceptable. If you do the work to the level of quality that I ask, you will get a good grade.

> *All students will be held accountable for their behavior according to the Policies outlined in the Marmion Academy Student/Parent Handbook 2015 – 2016. Verbal bullying such as negative characterization of other students or negative characterization of other student's behavior will be considered on par with physical bullying and not tolerated.*

Quizzes (online; in class; note quizzes; C quizzes)	15%
Quarter Exam (comprehensive)	15%
Homework & Projects (online homework; in class projects)	20%
Labs (computer simulation analyses and/or physical laboratory experiment)	20%
Semester Exam (comprehensive test + successful 3D n-body simulation project)	30%

If you need to contact us: E-mail vpinks@marmion.org or epeterson@marmion.org

Appendix 2. A Very Basic Blue Waters Workflow Guide

A Very Basic Blue Waters Workflow Guide

Author: Eric Peterson, Marmion Academy

Created: June 2015

Last Modified: May 2016

Notes: Commands are in bold and are preceded by a dollar sign **$**. [ ] tokens should be replaced with the appropriate information, minus the [ ]. Text that is to be inserted into files is in italics. Text in the following font will be actual output from the system: `example system output`

1. First, SSH into Blue Waters (steps are different depending on type of account, either training or regular account):

   $ssh [your username]@bwbay.ncsa.illinois.edu or
   $ssh [your username]@bw.ncsa.illinois.edu

   `Password: [enter your password here]`

2. Use basic commands to change directories, display directories/files, move and copy files and directories, edit files (use basic CLI tutorial by Mubeen)

3. Requesting resources on Blue Waters, either XE or XK nodes:

   a. There are two different ways to request resources:
      i. Interactively (allows you to submit jobs and see output immediately):

         $ **qsub –I [resource list]**

      ii. Batch:

         $qsub [batch script file].pbs

   b. Resource lists:
      i. This is where you request the number of nodes, type of nodes, number of processors, amount of time required, and more. One example for an interactive mode:

         **$qsub -I –l nodes=1:ppn=[32 / 16]:[xk / xe] **
         -l walltime=1:00:00

      ii. For batch files:

         #PBS -l [resource list]

   c. Batch files: (Allows you to request resources that will run without direct intervention and waste less of your computing time)
      i. Basic structure of a PBS file:

         #!/bin/bash
         cd $PBS_O_WORKDIR
         #PBS -l [resource list]

#PBS -o [pathname of output file]
#PBS -e [pathname of error file]

aprun ./[name of program]

4. Running your job:
   a. On an interactive session:

      $aprun –n [number of processes] ./[name of program]
   b. In a batch file:

      aprun –n [number of processes] ./[name of program]

5. After the job has been submitted, you will see the job ID of your job appear at the command line. Example of a job submission (through batch):

```
Job submitted to account: jt4
1817863.nid11293
```

6. Checking the status of your jobs will be critical (It also allows you to check what job ID you received):

   $qstat -u [your username] or
   $qstat -u `whoami` (if you don't know your username)

7. When your job has finished (Q in stat indicates a queued job, R indicates a running job, and C indicates a completed job):

   a. In batch:
      You will receive files (hopefully in the same directory as the program) with the name : [program name].pbs.[e / o][job ID], unless you chose different names for the output and error files.

   b. In Interactive mode:
      You will see your results through standard output (the screen) unless re-directed.

8. Compiling your programs:
   a. Compiling your programs may depend on if you are using CUDA, MPI, openMP, openACC, or none of those, but the basics are as follows:
      i. Once you have finished editing your .c file, use the CRAY C compiler to compile your program with the following flags:

         $cc -o [program name] [program name].c –l[libraries]

      ii. This will create a file named [program name].
   b. If you are using openACC, you may need to add:

      $Module load craype-accel-nvidia35
      –h pragma=acc (add this to compiler line before -o flag)

   c. When using openMP, remember to set the number of threads:
      export OMP_NUM_THREADS=[number of threads]

d. If you are using CUDA, you may need to compile your CUDA section separately, then link the CUDA to your main program

$nvcc -o [program name].o -c [program name].cu
$CC -o [program name] [program name].o [program name].c

9. Makefiles:
   a. Makefiles are a way of automating the compilation process, and ensuring that all dependencies and flags are set properly every time you compile.
   b. Basic structure of a makefile: (example):

all:
    make
[program name]: [program name].c
    CC –o [program name] [program name].c
clean:
    rm –rf [program name]

10. Performance Testing:
    a. This guide will talk briefly about the GNU Profiler (gprof). First swap the environment to the GNU programming environment:

$module swap PrgEnv-cray PrgEnv-gnu

    b. Compile a program with the –pg and –g options enabled: (this enables profiling)

$gcc –pg –g –o [program name] [program name].c

    c. Now we actually run our program to generate the profiling info:

$ ./[program name]

    d. This generates a file called "`gmon.out`". Now we use gprof to visualize the profiling data, additionally providing it another run of the program to give it more data:

$gprof ./[program name] gmon.out > profiling_data

    e. Then, we can see the final data by using less:

$less profiling_data

11. Editing a file using a text editor (vi):
    a. vi is a simple text editor that is available on blue waters and all Unix-based systems. It can invoked simple by typing:

$vi or
$vi [file name]

    b. vi has two basic modes:
        i. Command mode, keyboard presses will be executed as commands. The default mode that vi starts in. To enter command mode, press **ESC**.

ii. Insert mode, keyboard presses will be insert or edit text in the file. To enter insert mode, press '**i**' on the keyboard.

c. In command mode, you can use the direction keys to move through the file, or use:
   i. **Ctrl-f** to move up one screen through the file.
   ii. **Ctrl-b** to move down one screen through the file.
   iii. '**gg**' to move to the top of the file.
   iv.
d. In command mode, you can use keyboard presses to edit:
   i. '**dd**' will remove an entire line
   ii. '**yy**' will yank a line, and p will place the yanked line where your cursor is located.
   iii. '**u**' will undo the last change you made

e. In command mode, to close and save files:
   i. '**:w**' will write the file to disk but not quit
   ii. '**:wq**' will write the file to disk and quit
   iii. '**:q!**' will quit without saving (useful if you made unwanted edits)

f. In command mode, you can navigate to a line by pressing

   '**:[line to navigate to]**'

g. In command mode, searching for a pattern is relatively painless:

   '**/[pattern to search for]**'

Parallelized Model of Low-Thrust Cargo Spacecraft Trajectories and Payload Capabilities to Mars

Wesley Yu
Department of Aerospace Engineering and Engineering Mechanics
The University of Texas at Austin
Austin, TX 78712
+1(915)309-7972
yuwesley@utexas.edu

Hans Mark
Department of Aerospace Engineering and Engineering Mechanics
The University of Texas at Austin
Austin, TX 78712
+1(512)471-5077
hmark@mail.utexas.edu

ABSTRACT

As a Blue Waters Student Internship Program project, we have developed a model of interplanetary low-thrust trajectories from Earth to Mars for spacecrafts supplying necessary cargo for future human-crewed missions. Since these cargo missions use ionic propulsion that causes a gradual change in the spacecraft's velocity, the modeling is more computationally expensive than conventional trajectories assuming instantaneous spacecraft velocity changes. This model calculates the spacecraft's time of flight and swept angle at different payload masses with other parameters kept constant and correlates them with known locations of the planets. With parallelization using OpenMP on Blue Waters, its runtime has decreased from 10.55 to 1.53 hours. The program takes a user-selected Mars arrival date and outputs a given range of dates with maximum payload capabilities. This parallelized model will greatly reduce the time required for future mission design projects when other factors like spacecraft solar panel power output may vary with new mission specifications. The internship experience has enhanced the intern's ability to manage a project and will impact positively on his future graduate studies or research career.

CCS Concepts

•Applied computing → Physical sciences and engineering →Aerospace

Keywords

Computational Modeling; Orbital Mechanics; Low-Thrust Trajectories

1. INTRODUCTION

Sending humans to Mars has presented difficult problems to solve in the past few decades since the Moon landings. With the advent of new technology and launch methods, such as NASA's Space Launch System (SLS) and SpaceX's reusable first stage, the possibility of reaching Mars within the next few decades is becoming a reality.

There are still many aspects of a crewed mission that need work, like the effect of galactic cosmic radiation on the crew's health, and the ability to get all necessary equipment and backup supplies to the Martian surface to keep the crew safe there. The latter problem leads into the focus of this study.

Ionic engines can be the safe and efficient source of propulsion [1,2,3] for cargo missions to supply the Martian surface with food, water, habitats, scientific instruments, and transportation equipment. With the low-thrust characteristics of ionic engines, we cannot model their trajectories with instantaneous velocity changes [2]. Instead, the calculations must take into account the continuous velocity change along the entire trajectory at infinitesimally small time steps. This study explores the use of parallel computing to speed up the calculation of these low-thrust cargo transfers.

2. BACKGROUND

The possibility of travelling to Mars hinges largely on our ability to support astronauts with necessary sustenance and equipment for the length of their stay on the surface. This requires consideration of backup safety factors that call for multiple cargo launches to ship the necessary supplies to the Martian surface [3]. The eventual goal would be continual shipments from Earth to Mars to setup surface stations and provide a growing community of shelter and supplies for the astronauts to utilize. These cargo missions would be unmanned and much more frequent than the crewed missions.

Each individual cargo mission itself does not need to reach Mars quickly because of the continual nature of the shipment process. Therefore, we do not require current high-thrust chemical propulsion to complete Earth to Mars transfer in the usual eight months [2]. In order to increase safety and conserve fuel, we can utilize ion propulsion for this operation. This would lengthen the trip to a year and a half, but each shipment would be arriving within a few months of each other, giving the astronauts a steady supply, much like the current supply system for the International Space Station.

The main difference between ionic and chemical propulsion is the method in which the thrust is applied. In chemical engines, cold gas is ignited to cause an expulsion of propellant at a very high mass flow rate. The thrust, in turn, is very high but fuel runs out within a few minutes [4]. Thus, chemical engines are fired only

DOI: https://doi.org/10.22369/issn.2153-4136/8/2/7

during a very short period of time, and can be modeled assuming an instantaneous change in velocity, ΔV [5]. Using solar panels to provide electric power, ionic engines operate by using energetic electrons to bombard and ionize noble gases, such as argon and xenon [2,3,6]. A potential difference then develops, accelerates the ions, and emits a positively charged beam as the exhaust. Unlike conventional chemical engines, the mass flow rate is very low [7,8], so the thrust must be applied throughout the entire transfer. These two different propulsion systems using ionic and chemical engines would produce two contrasting types of orbital transfers, known as the Ward spiral and Hohmann transfer respectively (see Figure 1). Modeling the Ward spiral trajectory requires a calculation of velocity change at each point along the spacecraft's path. Optimizing these low-thrust trajectories for a proper launch date becomes computationally expensive, and would benefit from a parallelized computational model.

There are two main questions we are trying to answer in this study. The first is the relationship between the time of flight and payload mass of the spacecraft. The second is the effect of launching at non-ideal times on the payload mass capabilities of a certain spacecraft. Having a model to calculate this will help determine safety factors and contingencies in a mission design project.

3. METHODS

The main model in this study centers on a low-thrust Ward spiral, with the inputs being the thrust and mass of the spacecraft. After accounting for the inclination change between Earth's and Mars' orbits, we correlate the positions of the two planets in the coming decades, called ephemerides, to the possible trajectories. These ephemerides tables are obtained from the NASA Jet Propulsion Laboratory's (JPL) HORIZONS online tool [9] by inputting a start and end time, then specifying the time step size. Since we are testing all the launch dates over a five-year period as well as altering the payload mass levels, parallelizing the code would speed up the process significantly.

3.1 The Ward Spiral

Using a method developed for satellite orbit lowering maneuvers that employ low thrust ion engines [10,11], we reversed the direction of travel to an orbit raising trajectory [10]. Since Earth and Mars are on slightly different orbital planes, the change in inclination also adds additional ΔV to the trajectory, so we had to consider this in our calculations as well. This allowed us to model a spiral of gradually increasing orbital radius that reached from Earth to Mars. The Ward spiral starts with a circular orbit where the orbital velocity, v, is calculated by the equation:

$$v = \sqrt{\frac{\mu}{r}}$$

where μ is the gravitational parameter of the Sun and r is the spacecraft's distance from the Sun [4]. From the inputted thrust, F, of the spacecraft we can calculate the power, P, of the spacecraft:

$$P = Fv = F\sqrt{\frac{\mu}{r}}$$

and correlate it to the vis-viva energy [4], defined as the total energy of the spacecraft in a circular orbit, E:

$$E = -\frac{\mu m}{2r}$$

where m is the inputted spacecraft mass. Since the time derivative of energy is power, we can equate $\dot{E}$ to P and therefore:

$$F\sqrt{\frac{\mu}{r}} = \frac{\mu m}{2r^2}\dot{r}$$

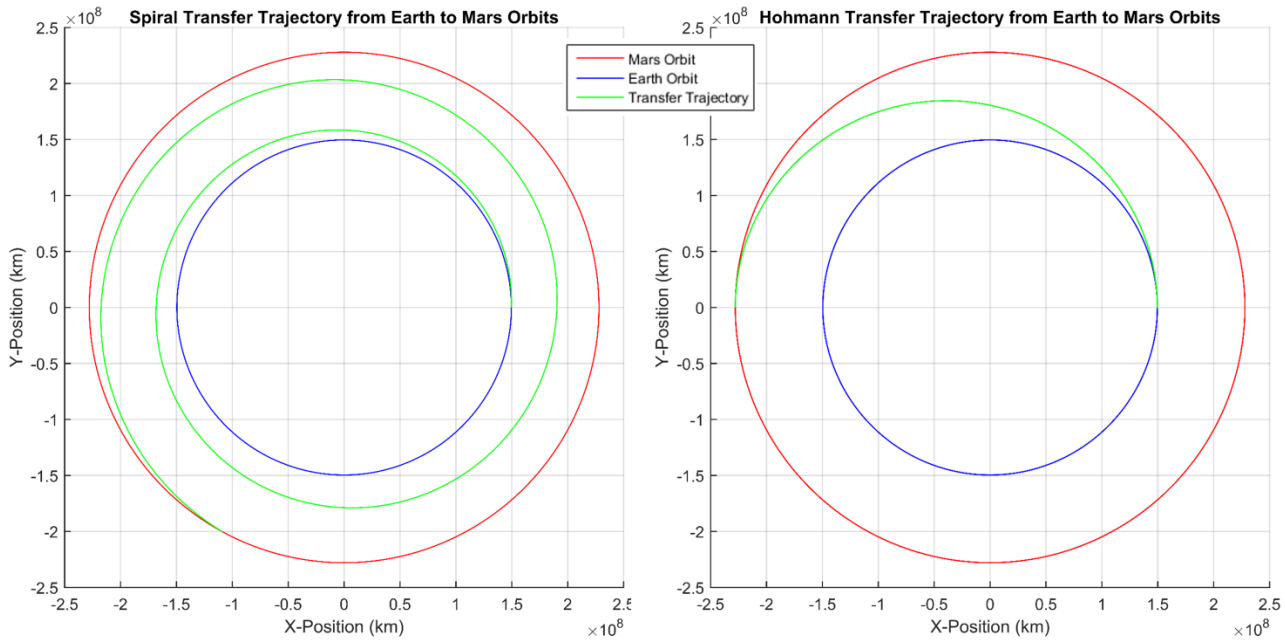


Figure 1. Comparison between a low-thrust Ward spiral and conventional Hohmann transfer from Earth to Mars. The blue circle represents Earth's orbit around the Sun, red is Mars' orbit, and green is the transfer trajectory.

We now have a first-order differential equation that directly relates the time derivative of the orbital radius and the thrust of the spacecraft's ion engine, with initial conditions of $r = r_0$ (Earth's orbital radius around the Sun), at $t = 0$.

$$\frac{dr}{dt} = \frac{2F}{\sqrt{\mu}m} r^{3/2}$$

Solving the above differential equation gives:

$$r = \frac{r_0}{\left(1 - \frac{F}{m}\sqrt{\frac{r_0}{\mu}}t\right)^2}$$

This calculates the orbital radius at any given time, t.We can now use the initial and final orbital radii, r_0 and r_1, to find the time of flight, t_f, and integrate the stored orbital radius and velocity values to find the total swept angle, θ, of the transfer orbit:

$$t_f = \frac{m}{F}\sqrt{\frac{\mu}{r_0}}\left(1 - \sqrt{\frac{r_0}{r_1}}\right)$$

$$\theta = \int_0^{t_f} \frac{v(t)}{r(t)} dt$$

With these relations it is straightforward to code a model for a single trajectory, with an array of time and an inputted thrust and mass. The next part of the model calculates the inclination change between Earth's and Mars' orbits. Just as energy is required to change the orbital radius, the change in orbital inclination requires an amount of ΔV given by:

$$\Delta V = 2v \sin\left(\frac{\Delta i}{2}\right)$$

where Δi is the change in inclination. Since the orbital velocity v is constantly decreasing as the orbital radius increases, the total ΔV is a summation of all the inclination changes along the trajectory [4].

3.2 Trajectory Optimization

3.2.1 Time of Flight

For conventional propulsion, the optimization is conducted with ΔV as the main objective. Minimizing the ΔV in a maneuver will minimize the fuel needed for a certain transfer, thus lowering the overall cost of the mission. [5,6]. However, with a low-thrust trajectory, the ΔV is determined by the difference in orbital velocities of Earth and Mars around the Sun, and is therefore a constant. So regardless of how long it takes to reach Mars on a certain trajectory, the spacecraft needs the same amount of ΔV. Therefore the main optimizing objective is the time of flight of the transfer, which is mostly dependent on the locations of the planets at a given time [12]. Although there will be a limit to how fast the spacecraft can reach Mars due to the extremely low thrust of the ion engines, we still would like a reasonably quick transfer. This would minimize exposure of food/water and even some scientific equipment to the harmful effects of galactic cosmic radiation.

3.2.2 Computational Approach

With the two inputs of thrust and payload mass, we chose to keep thrust constant for each run, as an assumption that we are using one engine with readily available electrical power. The thrust of 0.023 N is used for our computations based on the NASA-released information on the NEXT ion thruster [3]. We then varied the payload mass from 1 to 20 metric tons (t) with increments of 0.1 kg, and calculated the time of flight and swept trajectory angle for each mass value. This range was chosen based on payload masses in NASA heritage missions as a lower bound, up to anticipated future necessary mass levels as an upper bound [3]. With these values stored in a text file, we correlated the data with ephemerides to find potential launch windows for missions with different payload masses.

As we make the payload mass increments smaller to increase the accuracy of time of flight calculations, the number of data points also increases, making the text files larger in size. To minimize the inconvenience of loading large text files we separated the outputs into smaller organized files labeled with the first mass value in the file. The main use for the files is to create a table that relates mass and time of flight so that when needed, one can extract the values on a desktop computer, without having to access Blue Waters. The most important application would be a mission plan to arrive on Mars by a certain date, given the maximum payload mass and necessary thrust. This would require many calculations of the time of flight versus mass, and with the tabulated values for a given thrust, it will be much easier to read through them instead of calculating the time of flight each time.

All the code that was run on Blue Waters to calculate time of flight was written in C, and compiled using cc in the default craype/2.5.0 programming environment with Cray Linux Environment (CLE 5.2). The programs for further payload capability analysis were also written in C and compiled with gcc/4.9.2 in a Cygwin 6.3 terminal installed on a Windows 10 desktop computer. We used gnuplot 5.0 to generate payload capability figures and MATLAB 2015b to generate the three-dimensional Earth-Mars trajectory plots.

3.2.3 Maximum Payload

Since the tabulated values are separated by thrust level, one can also start with a time of flight and arrival date, and use the tables to find the maximum payload mass capability for a given thrust. The model takes the required time of flight and associates it to a payload mass, and if the ephemerides suggest that the launch cannot be completed in time, the model will go to the next thrust level.

This application takes advantage of the tabulated values extracted from the parallelized code run on Blue Waters. The user can input the desired arrival date and a maximum time of flight, and the program can find a range of launch dates with their corresponding payload mass and present them in a plot. For mission design projects this would be a very useful tool to visualize the relationship between an actual launch date and the possible payload mass.

Another application is to lock the launch date at the ideal case, and then match it with the user's needed arrival date. Then we can vary the arrival date within a range around the user-selected date to see its effect on the maximum payload mass. This allows us to test the effect of launching at non-ideal launch windows on the maximum payload capability (including the mass spacecraft mass) of a mission. Using this model can help test contingency and develop safety margins for the mission design.

3.3 Launch Date Calculation

In order to calculate the proper launch dates we require the ephemerides of Earth and Mars for the coming decades. This allows the determination of the initial and final positions of the

planets during the transfer orbit, which can be correlated with the outputted times of flight to determine the necessary launch date of the cargo shipment. We generated the ephemerides from JPL's HORIZONS software [9] and loaded them into text files. Using a 50-year range from January 1, 2018 to December 31, 2067, we have the positions of both planets for the next five decades.

To avoid overly long or unrealistically short times of flight, we set an upper and lower bound on the times of flight calculated from the varying payload mass. Then the times were matched with their corresponding arrival dates. Using the known ephemerides we could work backwards to find the appropriate launch dates. This allowed us to identify "windows of opportunity" and optimal launch dates.

3.4 Parallelization

3.4.1 Parallelization Method

Finding the times of flight from varying payload masses represented the bulk of the computations needed in this project. The time required for extraction and analyses of the resulting data was negligible. For simplicity, we separated the parallelized time of flight calculations and tabulations using Blue Waters from the subsequent payload and launch dates analyses that can be run by a user on a typical desktop computer.

To parallelize the time of flight computations, we used OpenMP in the default Cray compiler on Blue Waters. The programming environment was the default craype/2.5.0, and we coded in C on OpenMP v3.0 to parallelize the for loops calculating the vector of different transfer orbits. The code was then compiled on Blue Waters using gcc/4.9.2, running on XE compute nodes.

We assigned one potential transfer orbit to each thread, with eight cores total for the parallel runs. With the thrust kept constant, each orbit had a different payload mass assigned to it and produced a time of flight. Then the trajectory's time of flight was correlated to the different launch dates with the same window for travel. The program would print out the time of flight, swept angle, and corresponding payload mass so that extracting the data later would be easier. Both the serial and parallel versions of the code are available upon request.

3.4.2 Measuring Speed Increase

The most convenient way to track the run times of all our tests is to use the 'time' function for all tasks we run on Blue Waters. We ran several small data sets with around 1000 payload mass inputs for testing, where hardly any speedup was observed. We also ran 100 large datasets with millions of payload mass inputs to increase the accuracy in time of flight calculations, where more significant speed increase is expected due to the smaller percentage of overhead.

4. RESULTS

The products of the model we created are three-dimensional plots of the trajectories and possible launch dates for the mission. These are helpful in visualizing and planning missions, where the plots can show us the general shape and characteristic of the trajectory, the possible launch dates give us concrete numbers on the frequency and viability of the missions.

The more interactive product is a program that can determine the payload capability of the spacecraft based on given launch or arrival dates. The user can input a necessary Mars arrival date and time of flight, and use the model to find the maximum payload mass for a given thrust. If the mission does not meet the requirements for the given thrust, the model will find the next level suitable thrust level by adding additional engines to increase the thrust.

4.1 Plots of Trajectories

The plots take into account the inclination change of the spacecraft, and show the spacecraft's path from Earth's orbit around the Sun to Mars'. The inclination change is exaggerated in the plots for visual effect, but the actual change in inclination between Earth and Mars orbit is 1.85°. The x-y plane in this reference frame is the orbital plane of Earth around the Sun. Examples of trajectory plots are shown in Figures 2 and 3.

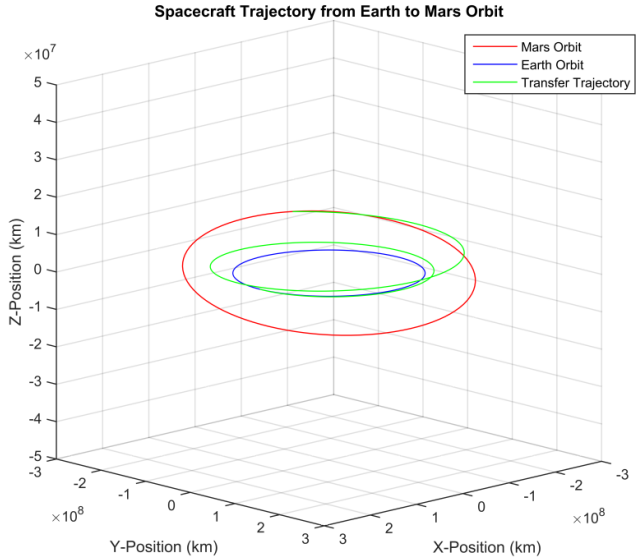


Figure 2. A typical Ward spiral from Earth to Mars, a trajectory that would take two years to complete for a 15t payload with a single NEXT engine.

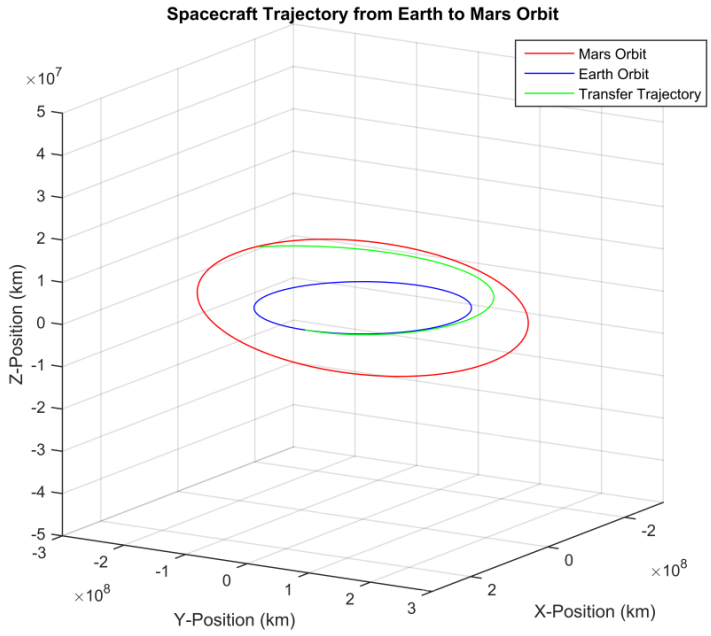


Figure 3. A much faster trajectory (10 months transit time) than in Figure 2, representative of a small probe's trajectory, where the payload would be relatively light.

4.2 Effective Launch Dates

As mentioned earlier, the model outputs possible launch dates given a Mars arrival date, which is very helpful in mission planning. The model can predict the date of launch, or even the hour of the launch if the user chooses. We represent the date as a Julian Date, which is the number of days since noon Universal Time on January 1, 4713 BCE, to facilitate ease of calculation. Julian dates can be converted back to the common Gregorian calendar dates easily using the U.S. Naval Observatory Julian Date Converter [13]. Table 1 below shows two examples of the outputs that the program will give with a 14.9t payload and 0.023 N thrust, with two different user inputted arrival dates:

Table 1. Ideal and Latest Possible Launch Dates

	Mission 1	Mission 2
Inputted Mars Arrival Gregorian Date	5/2/2023	6/17/2045
Corresponding Julian Date	2460066.5	2468148.5
Ideal Launch Julian Date Based on Time of Flight	2459406.5	2467384.5
Ideal Launch Gregorian Date	7/11/2021	5/15/2043
Latest Launch Julian Date Adjusted for Ephemerides	2459415.5	2467416.5
Latest Launch Gregorian Date	7/20/2021	6/16/2043

Launching either earlier or later than the ideal date would require more fuel. However, an earlier launch can give the mission greater scheduling flexibility. Launching later would lower the safety margin, limiting the flexibility of the mission. Beyond the latest launch date, it will not be possible to reach Mars by the desired arrival date with the given payload mass and thrust level.

4.3 Maximum Payload

With the program that outputs payload capability from a given time of flight, we were able to plot the maximum payload mass against a range of launch dates. With the added constraint of the arrival date and maximum time of flight, we could use this tool to narrow down the search of ideal launch windows.

The outputs from this program are once again tabulated in a text file as maximum payload mass and launch date. Such outputs for Mission 1 in Table 1, with required arrival date of May 2, 2023 are plotted in Figure 4. At the ideal launch date of July 11, 2021, the maximum payload mass of 14.9t is attained, but the payload capability decreases parabolically as the launch date deviates from ideal.

When locking the launch date to the ideal case in the aforementioned model and using the user-inputted arrival dates as a reference, we plotted the maximum payload mass while varying the arrival date. Once again we can see that the maximum payload mass remains at the ideal arrival date, but the loss of payload capability is less than that generated by deviations from the ideal launch date.

This computational tool is useful for trade studies when determining the cost of a mission and its resulting trade-off with payload mass. We can now see the effects of delaying or moving up the timetable of a mission, and how it will limit certain equipment or resources to be transported on the mission. With this information we can effectively plan shipments and their fail-safe launch dates.

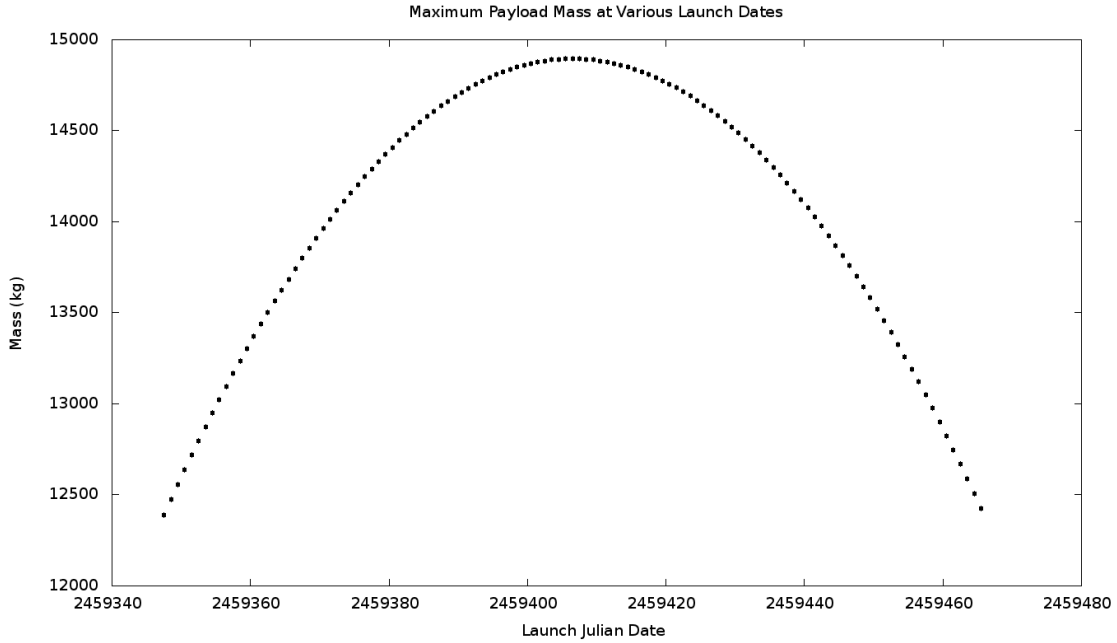


Figure 4. Maximum payload mass the spacecraft can carry at different launch dates with a constant time of flight. The suggested ideal date of July 11, 2021 (Julian 2459406.5, see Table 1) is on the apex of the graph.

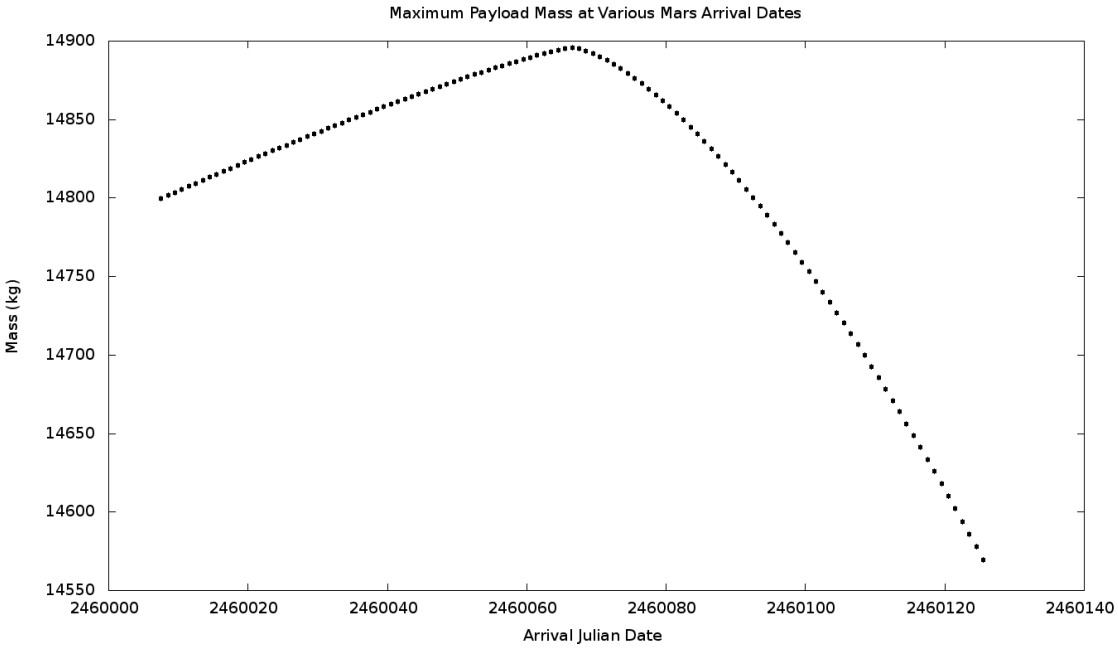


Figure 5. Maximum payload mass the spacecraft can carry at different arrival dates with the launch date kept at the original ideal from Table 1. Note the asymmetry of the decrease in payload capability on either side of the ideal arrival date. Due to the change in positions of Earth and Mars, arriving earlier versus later yields different payload capabilities.

4.4 Speedup After Parallelization

We ran both the serial and parallelized versions of the program on Blue Waters' Cray XE nodes, at a peak performance of 313.6 GFLOPS per node. As expected, when running small data sets of 1000 payload masses for initial testing, there was no noticeable speedup since the effects from overhead canceled out the parallelization. When running the parallelized version to calculate 10 million payload mass data points we used eight cores and the runtime decreased from 10.55 hours to 1.53 hours. This reduced runtime facilitated our development of the Ward spiral code without the need for overnight testing and outputting. The increased speed was utilized primarily to speed up the intensive calculations to prepare the tabulated values, allowing us to quickly start analyzing the resulting data for mission planning.

5. DISCUSSION

5.1 Usefulness of Parallelization

While using Blue Waters was extremely helpful in generating the time of flight tables, a user with the intent to input one mass and thrust value will see no difference in a single calculation of the trajectory versus a program that reads tabulated values. So the parallelization in this case did not affect the end user. However, with the case where the user is looking for maximum payload mass given a time of flight and arrival date, it would be very inconvenient to calculate the values serially on a laptop. Having the tabulated values to scroll through in this case would greatly reduce the computing time.

5.2 Viability of Low-Thrust Missions to Mars

From the trajectory plots, time of flights, and payload capabilities obtained from this study, we can see that the use of ionic propulsion is a viable method to provide support for future missions to Mars. The mission parameters that we have calculated are well within the reasonable range of required cargo shipment set by NASA standards. This will be key in future development of routine cargo missions when we continue our mission designing process.

5.3 Educational Impact of the Project

The project is a one-year Blue Waters Student Internship with an initial training workshop for C programming on the supercomputer, followed by required monthly reports submitted to the internship program. Throughout the course of the program, the intern has learned to structure a research project with a well-defined timeline to meet clear and realistic goals and steps. The experience has enhanced the intern's ability to plan and manage a research project and helped find solutions in a timely fashion for challenges such as parallelization I/O problems and memory-related segmentation errors encountered in this project. This will have a positive impact on the intern's future graduate studies or research career.

The main educational goal of this project is to expose the intern to methods of parallel computing. During the short session in Summer 2015, the intern learned the basics of various parallel programming languages like OpenMP and MPI, and also strengthened his skills in using C. Afterwards during the year, he was able to use this knowledge and apply it to trajectory modeling problems. While the integration of OpenMP into the trajectory model was essentially used to run more cases in the same time, it was a good introduction to parallel computing, and we hope to continue utilizing this tool for future studies of this nature.

Through this process the intern also greatly strengthened his ability to comfortably use Linux command terminals to communicate remotely with a supercomputer, a skill that will be extremely useful in upcoming research. Additionally, the intern has gained insight into trajectory modeling and general knowledge in spacecraft mission design and planning. The program generated for this project will be valuable in the intern's continuing research project to design a cost effective, human-crewed mission to Mars. The code (Version 1.0) was fully developed by the intern, and is available upon request.

6. CONCLUSION

Throughout the study we evaluated the use of parallel computing to assist in trajectory modeling. The most effective course of action was to increase the volume of trajectories we could model, and generate tables to use offline. This method is like a higher mission level tool similar to how ephemerides would be used to plan a trajectory. Parallelization greatly increased the efficiency of the developmental stage of the model, as it saved time by speeding up testing of the Ward spiral code and results were obtained quickly for payload and launch date analyses. The ability to handle large amounts of data and organize it on Blue Waters saved a lot of disk space and computing time on the desktop computer.

From the study it is now possible to use the model to plan cargo missions to Mars in the concept study phase. The model can be used on any machine as long as the data and ephemerides are present in the package. With the input of thrust and mass in text file form, additional parameters that alter those two main variables can be added on without disrupting the original code. We can build on this template for future, higher-fidelity models.

7. LIMITATIONS AND FUTURE PROJECTS

The main limitation in this study was the assumption of full available power to the spacecraft at all times. The model accounted for an ideal transfer with the maximum thrust of the NEXT engine present throughout the trajectory. However, in an actual mission, the orientation of the solar panels and other factors like system reliability can come into play and alter the available power levels, thus changing the thrust of the engine unexpectedly. In addition, minute gravitational perturbations are not accounted for, and with Jupiter's large gravitational influence, it can alter the trajectory enough to require thrust changes for course correction. These extra factors are subjects we can explore in future studies that add to this model. We can still use the thrust and mass as inputs, but append extra calculations that will alter the thrust based on power levels and course corrections.

8. ACKNOWLEDGEMENTS

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. We thank Therese Larson for her assistance in communication with Shodor and NCSA and Natalie Wolfenbarger for her help on generating ephemerides.

9. REFERENCES

[1] Brophy, J. R., Noca, M. 1998. Electric Propulsion for Solar System Exploration. *Journal of Propulsion and Power* 14, 5, 700-707.

[2] Curtis, H. 2013. *Orbital Mechanics for Engineering Students* 3rd edn. Butterworth-Heinemann, Oxford, United Kingdom.

[3] Wertz, J. R., Everett, D. F., Puschell, J. J. 2011. *Space Mission Engineering: The New SMAD*. Microcosm Press, Hawthorne.

[4] Bate, R. R., Mueller, D. D., White, J. E. 1971. *Fundamentals of Astrodynamics*. Dover Publications, Inc, New York, United States.

[5] Hughes, K. M., Edelman, P. J., Saikia, S. J., Longuski, J. M.,

Loucks, M. E., Carrico Jr., J. P., Tito, D. 2015. Fast Free Returns to Mars and Venus with Applications to Inspiration Mars. *Journal of Spacecraft and Rockets* 52, 6, 1712-1735.

[6] Colasurdo, G., Casalino, L. 2003. Characteristics of Electric Propulsion Systems for Optimal Interplanetary Trajectories. In : *54th Annual Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law*, Bremen, Germany, 1-7.

[7] Purvis, J. W. 1992. Closed Form Low-Thrust Trajectories for Mars Missions. In : *AIAA/SAE/ASME/ASEE 28th Joint Propulsion Conference and Exhibit*, Nashville, Tennessee, 1-4.

[8] Rayman, M. D., Fraschetti, T. C., Raymond, C. A., Russell, C. T. 2005. *Preparing for the Dawn Mission to Vesta and Ceres.*, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California.

[9] NASA Jet Propulsion Laboratory HORIZONS Web-Interface. In: *Jet Propulsion Laboratory Solar System Dynamics.* Available at: http://ssd.jpl.nasa.gov/horizons.cgi

[10] Rimrott, F. PJ., Cleghorn, W. L. 2002. Orbit Transfer by Means of a Ward Spiral. *Technische Mechanik* 22, 4, 283-290.

[11] Rimrott, F. PJ., Salustri, F. A. 2001. Open Orbits in Satellite Dynamics. *Technische Mechanik* 21, 3, 207-214.

[12] Nah, R. S., Vadali, S. R., Braden, E. 2001. Fuel-Optimal, Low-Thrust, Three-Dimensional Earth-Mars Trajectories. *Journal of Guidance, Control, and Dynamics* 24, 6, 1100-1107.

[13] U.S. Naval Observatory Julian Date Converter. In: *Naval Oceanography Portal.* Available at: http://aa.usno.navy.mil/data/docs/JulianDate.php

July 2017

Volume 8 Issue 2

TABLE OF CONTENTS

ISSN 2153-4136 (online)