Finalist Reports

2011-2012

www.supercomputingchallenge.org

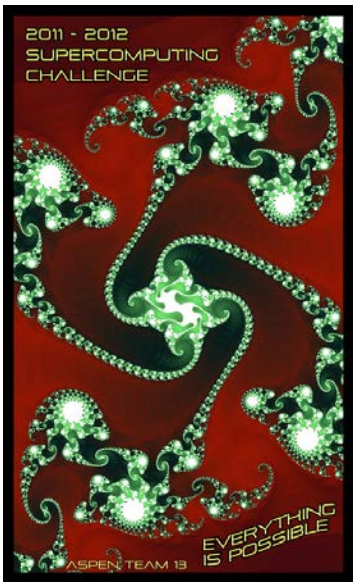
Printed in cooperation with
Los Alamos National Laboratory
High Performance Computing Group (HPC-3)
and
New Mexico Public Education Department

Cover: **The Intersection of Lines to Find Points**
Team 87 from Melrose High School
Randall Rush, Quinton Flores,
Jon Tello, and Kaleb Broome
Teachers Allen Daugherty and Rebecca Raulie
Winner in the Technical Poster Competition

Notification: These final reports are presented in an abridged form, leaving out actual code, color, and appendices where appropriate. Complete copies of most of the final reports are available from the archives of Supercomputing Challenge web site: http://www.supercomputingchallenge.org .

New Mexico Supercomputing Challenge
2011 – 2012 Finalist Reports

Table of Contents



Supercomputing Challenge Vision

The Vision of the Supercomputing Challenge is to be a nationally recognized program that promotes computational thinking in science and engineering so that the next generation of high school graduates is better prepared to compete in an information-based economy.

Supercomputing Challenge Mission

The Mission of the Supercomputing Challenge is to teach teams of middle and high schools students how to use powerful computers to analyze, model and solve real world problems.

About the Supercomputing Challenge

The Supercomputing Challenge (the Challenge) is an exciting program that offers a truly unique experience to students in our state. The opportunity to work on the most powerful computers in the world is currently available to only a very few students in the entire United States, but in New Mexico, it is just one of the benefits of living in the "Land of Enchantment."

The Challenge is a program encompassing the school year in which teams of students complete science projects using high-performance supercomputers. Each team of up to five students and a sponsoring teacher defines and works on a single computational project of its own choosing. Throughout the program, help and support are given to the teams by their project advisors and the Challenge organizers and sponsors.

The Challenge is open to all interested students in grades 6 through 12 on a nonselective basis. The program has no grade point, class enrollment or computer experience prerequisites. Participants come from public, private, parochial and home-based schools in all areas of New Mexico. The important requirement for participating is a real desire to learn about science and computing.

Challenge teams tackle a range of interesting problems to solve. The most successful projects address a topic that holds great interest for the team. In recent years, ideas for projects have come from Astronomy, Geology, Physics, Ecology, Mathematics, Economics, Sociology, and Computer Science. It is very important that the problem a team chooses is what we call "real world" and not imaginary. A "real world" problem has measurable components. We use the term Computational Science to refer to science problems that we wish to solve and explain using computer models.

Those teams who make significant progress on their projects can enter them in the competition for awards of cash and scholarships for the individuals and computer equipment for the school. Team trophies are also awarded for: Teamwork, Best Written Report, Best Professional Presentation, Electronic Search & Browse, Creativity and Innovation, Environmental Modeling, High Performance, Science is Fun and the Judges' Special Award, just to name a few.

The Challenge is offered at minimal cost to the participants or the school district. It is sponsored by a partnership of federal laboratories, universities, and businesses. They provide food and lodging for events such as the kickoff conference during which students and teachers are shown how to use supercomputers, learn programming languages, how to analyze data, write reports and much more.

These sponsors also supply time on the supercomputers and lend equipment to schools that need it. Employees of the sponsoring groups conduct training sessions at workshops and advise teams throughout the year. The Challenge culminates with an Expo and Awards Ceremony in the spring at Los Alamos National Laboratory.

History

The New Mexico High School Supercomputing Challenge was conceived in 1990 by former Los Alamos Director Sig Hecker and Tom Thornhill, president of New Mexico Institute of Mining and Technologynet Inc., a nonprofit company that in 1985 set up a computer network to link the state's national laboratories, universities, state government and some private companies. Sen. Pete Domenici, and John Rollwagen, then chairman and chief executive officer of Cray Research Inc., added their support.

In 2001, the Adventures in Supercomputing program formerly housed at Sandia National Laboratories and then at the Albuquerque High Performance Computing Center at the University of New Mexico merged with the former New Mexico High School Supercomputing Challenge to become the New Mexico High School Adventures in Supercomputing Challenge.

In 2002, the words "High School" were dropped from the name as middle school teams had been invited to participate in 2000 and had done well.

In the summer of 2005, the name was simplified to the Supercomputing Challenge.

In 2007, the Challenge began collaborating with the middle school Project GUTS, (Growing Up Thinking Scientifically), an NSF grant housed at the Santa Fe Institute.

2011—2012 Challenge Awards

• Albuquerque Manzano High School student takes top award in 22nd New Mexico Supercomputing Challenge

Computer algorithm for image analysis of Alzheimer's plaques

LOS ALAMOS, New Mexico, April 24, 2012—Jordan Medlock of Albuquerque's Manzano High School took the top prize in the 22nd New Mexico Supercomputing Challenge for his computer algorithm that automates the process of counting and analyzing plaques in magnetic resonance images of persons diagnosed with Alzheimer's disease. The program vastly speeds up the process of identifying the very small and difficult to see plaques.
For his project, "Detection of Alzheimer's Disease Plaques in a Transgenic Mouse Brain Using Image Analysis of SPION-Enhanced Magnetic Resonance Images," Medlock received a check for $1,000.

Los Alamos Middle School student Cole Kendrick took second place for his project, "Computer Simulation of Dark Matter Effects on Galaxy Collisions." Kendrick received a check for $500. He also received the $100 Crowd Favorite Award chosen by teachers and students and the Best Presentation Award. Kendrick took the top prize in last year's Supercomputing Challenge for his computer program to model the rotation of a galaxy including dark matter.

The Albuquerque La Cueva High School team of Alexandra Porter, Stephanie Djidjev and Lauren Li took third place for their project "Optimizing Community Detection." The trio also received the Women in Science and Engineering Award. The third place team members each receive $250.

Seven other teams were finalists in the yearlong competition culminating in Tuesday's awards ceremony in Los Alamos. These finalist teams received plaques and banners for their school; team members also each received $50.

The Supercomputing Challenge is open to any New Mexico high-school, middle-school, or elementary-school student. More than 200 students representing about 60 teams from schools around the state spent the school year researching scientific problems, developing sophisticated computer programs, and learning about computer science with mentors from the state's national laboratories and other organizations.

The goal of the yearlong event is to teach student teams how to use powerful computers to analyze, model, and solve real-world problems. Participating students improve their

understanding of technology by developing skills in scientific inquiry, modeling, computing, communications, and teamwork.

Each finalist team received plaques for their school trophy cabinets plus a large banner for their gym and $50 for each student.

The New Mexico EPSCoR Climate Change/Water Resources award goes to Team 94 from the New Mexico School for the Arts. Their title is *Water You Waiting For Santa Fe?* Team members are Mohit Dubey, Samuel Thompson, Milada Guenther, and Noah Caulfield. Their sponsor is Acacia McCombs and mentor is Stephen Guerin.

Team 37 from Desert Academy won an award for Modeling an Ecosystem which was presented by the New Mexico Museum of Natural History and Science. Their title is *Depletion of Aquifer Levels in the Lower Rio Grande*. Team Members are Jeremy Hartse, Taylor Bacon. Their sponsors are Jocelyne Comstock and Jeff Mathis.

Team 102 from Pinon Elementary won the Visualization prize from New Mexico Institute of Mining and Technology, sponsored by Dr. Lorie Liebrock. Ryan Swart from Pinon Elementary won with his project Visualizing the Tree of Life.

Team 17 from Aspen Elementary School won the Best Agent Based Modeling Project award. Their title is *A Devil Worth Saving*. Team Members are Thomas Chadwick, Gabriel Holesinger, Tazler Smith, Sabio Thompson and Jack Vandenkieboom. Their sponsor was Mrs. Martens, and

mentors were Mark Chadwick, Jeff Hay, Terry Holesinger, Sabina Johnson, John Vandenkieboom and Lynn Wysocki-Smith.

Team 101 from Pinon Elementary received the Community Impact Award for their projected titled *The Bullying Effect*. Team Members are Jordan Bailey and Ruby Selvage. Their mentors are Alison Bailey and Kim Selvage.

Team 38 from Down to Earth School was awarded the Newcomer Award for being a exceptional first year team. Their project was titled *Polystyrene Versus Down to Earth*. Team Members are Ella Kirk, Simone Hill, Ruby Zeuner, Addie Clemens, and Hailey Manlowe. Their Sponsors are Maia Chaney, Shanon Muelhausen, and Nathan Shay.

Team 4 from AIMS@UNM won the Magellan Award for their multi-grade level collaboration. Their project is Modeling the Flow of the Interstellar Medium Within Localized Sectors of Space and Team Members are Louis Jencka, Randall van Why, Nico Ponder, Stefan Klosterman, and Jake Kileen. Their Sponsor is Mr. Harris. They also won the Cray High Performance Computing Award.

The Best Web-based Presentation of a Final Report award was shared between Teams 74 from Los Alamos Middle School for their project titled *Haptic Feedback: Controlling Robots with Touch*, with team members Connor Bailey and Nate Delgado and sponsor Pauline Stephens and mentor Rob Cunningham, and Team 101 from Pinon Elementary received the Community Impact Award for their projected titled *The Bullying Effect*, team Members are Jordan Bailey and Ruby Selvage. Their mentors are Alison Bailey and Kim Selvage.

Team 87 from Melrose High School won the Best Technical Poster Award. Their project is titled *The Intersection of Lines to Find Points*. Team Members are Randall Rush, Quinton Flores, Jon Tello, and Kaleb Broome. Their teachers are Allen Daugherty and Rebecca Raulie. Their graphic becomes the front cover for the 2012-2013 Final Reports which will be published for the Kickoff in October 2012.

Teamwork Award from CHECS, the New Mexico Council for Higher Education Computing/Communication Service, went to Team 58, Las Cruces Young Women in Computing and their title is *Utopia*. Their team members are Marie Ellis, Samantha McGuinn, Hiba Muhyi, Noor Muhyi, Cindy Yeh. Their sponsors were Rachel Jensen, Rebecca Galves, and they were mentored by Jen Dana and Stephanie Marquez.

The Sandia National Labs Creativity and Innovation Award goes to Team 36, Desert Academy, Language Acquisition in Computers. Team members are Megan Belzner, Sean Colin-Ellerin and the teacher sponsors are Jocelyne Comstock and Jeff Mathis. Their Mentor is Jorge Roman.

Drew Einhorn received the Service Award for his continued support in old and new endeavors including the schedule for the kickoff and the wiki implementation.

The Graphical Poster award went to Team 77, Los Alamos Mid School. This design will become next year's logo and will appear on t-shirts, the website and teacher bags. Their teacher sponsor is Pauline Stevens and her student is Claire DeCroix.

The Science Rocks award went home with Team 107, Rio Rancho Cyber Academy, Reverse Osmosis Team Members: Stuart Perara, Trinity Medley and their Sponsor: Harry Henderson.

Team 2, a joint team from Desert Academy, Santa Fe High School and Academy for Technology and the Classics received the LANL Environmental Modeling Award for their project, "The Impact of Forest Fires on Water Resources." Team members are Sara Hartse, Hugo Rivera, and Nico Cruz and their sponsor is Jocelyne Comstock.

Edgewood Elementary School Team 41 received the Best Researched award from the Council for Higher Education Computing/Communication Services for their report, "Duel of the Fuel." Team members include Ethan Hintergardt, Chase Podzemny, Emily Robinson, Keith Stevens, and Pete Talamante and their sponsors are Carol Thompson and Jennifer Wiggins, and their mentors are Wayne Bitner and Joaquin Roibal.

The Jeff Bingaman Middle School Award went to Los Alamos Middle School Team 73 for the project, "Simulation of Multi-Agent Based Scheduling of Algorithms for Waiting-line Queuing Problems." Team members are Steven Chen and Andrew Tang and their sponsor is Pauline Stephens and their mentor is Hsing-bung (HB) Chen.

The Python Programming Award went to Los Alamos High School Team 66 for their project "ExcellAnts." Team members are Peter Ahrens and Dustin Tauxe and their sponsor is Lee Goodwin and mentors are James Ahrens and Christine Ahrens.

This year's Teacher Recognition Awards are in memory of Peggy Larisch, pioneer Supercomputing Challenge teacher from Silver High School. Students nominate teacher sponsors for recognition. This year's winners are Maia Cheney, Down to Earth School, Silver, Jocelyne Comstock, Desert Academy and Rebecca Galves, a Director of the Young Women in Computing at New Mexico State University.

The Challenge's management team Consult honored UNM Grad student Reffat Sharmeen as she facilitated at the kickoff, reviewed proposals and interim reports online, attended three face to face evaluations and judged teams at the Expo.

Five attendees at the Awards Ceremony received crisp $100 bills that were given out as random door prizes.

This year the Challenge was given $10,000 from LANL's Division of Computer, Computational, and Statistical Sciences Division for scholarship awards. An additional $10,000 came from LANS, $5000 came from Abba Technologies/Hewlett Packard, $500 from the Challenge for the Willard Smith Scholarships and $15,700 was given by in-state colleges and universities. Students receiving scholarships were:

Name	High School	College
Daniel Washington	Sat Sci & Math Acad	MIT
Stephanie Djidjev	La Cueva	Berkeley
Jordan Medlock	Manzano	UNM
Megan Belzner	Desert Academy	MIT
Jennifer Hu	LC YWiC	NMSU
Cindy Yeh	LC YWiC	NMSU
Noor Muhyi	LC YWiC	NMSU
Willie Fong	ATC	NMT
Bethany Tanner	SODA	NMT
Randy Van Why	AIMS@UNM	NMT
Jeremiah Marquez	Artesia	ENMU
Josh Trujillo	Artesia	OU
Peter Ahrens	Los Alamos	Berkeley

Dustin Tauxe	Los Alamos	CSU
Alanna Tempest	Miyamura High	Stanford
Max Bond	Monte del Sol	UNM
Devin Hayes	CEPi1	CNM
Billy Amershek	CEPi1	CNM
Louis Jencka	AIMS@UNM	NMT
Stefan Klosterman	AIMS@UNM	NMT
Samantha McGuinn	LC WYiC	NMSU

NCWIT (National Center for Women and Information Technology) Aspirations in Computing recognized junior and senior girls for their computing-related achievements. Awardees were selected for their computing and IT aptitude, leadership ability, academic history and plans for post-secondary education. The winners in New Mexico are Taylor Bacon, Desert Academy, Santa Fe, Stephanie Djidjev, La Cueva High, Albuquerque, Sara Hartse, Desert Academy, Julissa Hunte, V. Sue Cleveland High, Rio Rancho, Alexandra Porter, La Cueva High, and Falisha Trujillo, Jemez Valley High School. Irene Lee, Challenge Board President and Principal Investigator for Project GUTS and Guts Y Girls at Santa Fe Institute, was named the Educator Winner. The Challenge was pleased to host these awards and noted that four of the six young women have participated in the Challenge.

Three special guests joined the celebration: Henry Neeman, Executive Director, Research Computing & Services Director, OU, Supercomputing Center for Education & Research (OSCER). The University of Oklahoma Information Technology Adjunct Assistant Professor, School of Computer Science. The audience Skyped with Uri Wilensky, Principal Investigator from Northwestern University's Center for Connected Learning and Computer-Based Modeling, the home of NetLogo. Team 12 from Artesia High won the Best NetLogo award for their emergency egress model. Senator Jeff Bingaman gave a congratulatory message to the participants via video that was played during the Awards Ceremony.

Now in to its 23rd year, the Challenge is open to any New Mexico high-school or middle-school student. Over the past year, teams from schools around the state researched scientific problems, developed sophisticated computer programs, learned computer science with mentors from the state's national laboratories and other organizations, and had the opportunity to run their programs on some of the world's most powerful computers.

The goal of the year-long event was to increase knowledge of science and computing; expose students and teachers to computers and applied mathematics; and instill enthusiasm for science in middle- and high-school students, their families and communities. Participating students improve their understanding of technology by developing skills in scientific inquiry, modeling, computing, communications and teamwork.

More information on the New Mexico Supercomputing Challenge can be found at http://www.supercomputingchallenge.org online, while final student reports are available at http://www.supercomputingchallenge.org/archive/11-12/finalreports online.

Sponsors

The Supercomputing Challenge is sponsored by Los Alamos and Sandia national laboratories and Los Alamos National Security, LLC.

Educational partners include The Center for Connected Learning/NetLogo, New Mexico Council for Higher Education Computing/Communication Service (CHECS), Eastern New Mexico University, MIT StarLogo, New Mexico Computing Applications Center, New Mexico EPSCoR, New Mexico Highlands University, New Mexico Institute of Mining and Technology, Northern New Mexico College, New Mexico Public Education Department, New Mexico State University, NMSU-Dona Ana Community College ,San Juan College, Santa Fe Community College, Santa Fe Complex, Santa Fe Institute, Swarm Development Group, the University of New Mexico and the UNM Center for Advanced Research Computing.

Abba Technologies/HP, Google RISE, Intel Corporation, Lockheed Martin, Los Alamos National Laboratory Foundation, Synergy Group, The Math Works, Vandyke Software Inc., and Wolfram Research, Inc. are "Gold" commercial partners. "Silver" commercial partners are, Gulfstream Group and bigbyte.cc and Technology Integration Group. "Bronze" commercial partners are *Albuquerque Journal***,** Cray Inc., Fourth Watch Software, Innovate Educate NM, Lobo Internet Services, *New Mexico Business Weekly*, New Mexico Technology Council, PY Multimedia Services, Redfish Group, and Sun Microsystems.

About Los Alamos National Laboratory (http://www.lanl.gov)

Los Alamos National Laboratory, a multidisciplinary research institution engaged in strategic science on behalf of national security, is operated by Los Alamos National Security, LLC, a team composed of Bechtel National, the University of California, The Babcock & Wilcox Company, and URS for the Department of Energy's National Nuclear Security Administration.

Los Alamos enhances national security by ensuring the safety and reliability of the U.S. nuclear stockpile, developing technologies to reduce threats from weapons of mass destruction, and solving problems related to energy, environment, infrastructure, health, and global security concerns.

Teams Finishing the Challenge and submitting final reports:

Team 1, Academy for Technology and the Classics/Monte del Sol/The MASTERS Program, *Cellular Automata Based Tsunami Simulation*
Team Members: Max Benjamin Bond, Erik William Nelson, Arlo James Barnes, William Christopher Fong
Mentor: Nick Bennett

Team 2, Desert Academy, Academy for Technology and the Classics, Santa Fe High School, *The Impact of Forest Fires on Water Resources*
Team Members: Sara Hartse, Hugo Rivera, Nico Cruz
Sponsor: Jocelyne Comstock

Team 4, AIMS@UNM, *Modeling the Flow of the Interstellar Medium Within Localized Sectors of Space*
Team Members: Louis Jencka, Randall van Why, Nico Ponder, Stefan Klosterman, Jake Kileen
Sponsor: Mr. Harris

Team 5, Alamogordo High, *Go With the Flow*
Team Members: Pascal Rößner, Seth Hollis
Sponsor: Mr. Simon, Mr. Myoshi, Mentor: Bob Robey

Team 12, Artesia High School, *DON'T PANIC!!! EMERGENCY EGRESS*
Team Members: Madison Mutchler-Ramsey, Joshua Trujillo, Jeremiah Marquez
Sponsors: Randall Gaylor, Jose Quiroz, Mentor: Nick Bennett

Team 14, Aspen Elementary School, *Roundabout Efficiency*
Team Members: Christopher Koh, Do Vo, Jesse Prime
Sponsors: Mrs. Thomas, Mrs. Vandenkieboom, Mentors: Mike Prime, Duc Vo

Team 15, Aspen Elementary School, *Self-recovery of a distributed system after a large disruption*
Team Members: Victor Popa-Simil, Henry Poston
Sponsor: Mrs. Thomas, Mentor: Liviu Popa-Simil

Team 16, Aspen Elementary School, *Handling Failures: Supercomputers playing Telephone*
Team Members: Alex Ionkov, Andrei Popa-Simil
Sponsor: Kathryn Thomas, Mentor: Latchesar Ionkov

Team 17, Aspen Elementary School, *A Devil Worth Saving*
Team Members: Thomas Chadwick, Gabriel Holesinger, Tazler Smith, Sabio Thompson, Jack Vandenkieboom
Sponsor: Mrs. Martens, Mentors: Mark Chadwick, Jeff Hay, Terry Holesinger, Sabina Johnson, John Vandenkieboom, Lynn Wysocki-Smith

Team 18, Aspen Elementary School, *Mom, are we there yet?*
Team Members: Ignacio Rougier, Joshua Vigil, Kye Jones, Donald Poston
Mentor: Esteban Rougier

Team 22, Creative Education Preparatory Institute #1, *Hydrogen Fuel Cell*
Team Members: Devin Hayes, Billy Amershek, Dillon Kuhr
Sponsor: Jerry Esquivel, Mentor: Mrs. Velarde

Team 31, Chaparral High, *Space Junk: Problem of the Future*
Team Members: Sofia Bali, Crystal Zamora, Gabriela Marchan, Susana Bali
Sponsors: Rebecca Galves, Hadi Sharifi

Team 35, Cleveland High School, *Modeling Changes in Aquifer Water-Levels in New Mexico Due to the Imbalance between Discharge and Recharge*
Team Members: Ben Fowler, Korbyn Spooner
Sponsor: Debra Loftin, Mentor: Nick Bennett

Team 36, Desert Academy, *Language Acquisition in Computers*
Team Members: Megan Belzner, Sean Colin-Ellerin
Sponsors: Jocelyne Comstock, Jeff Mathis, Mentor: Jorge Roman

Team 37, Desert Academy, *Depletion of Aquifer Levels in the Lower Rio Grande*
Team Members: Jeremy Hartse, Taylor Bacon
Sponsors: Jocelyne Comstock, Jeff Mathis

Team 38, Down to Earth School, *Polystyrene Versus Down to Earth*
Team Members: Ella Kirk, Simone Hill, Ruby Zeuner, Addie Clemens, Hailey Manlowe
Sponsors: Maia Chaney, Shanon Muelhausen, Nathan Shay

Team 41, Edgewood Elementary, *Duel of the Fuel*
Team Members: Ethan Hintergardt, Chase Podzemny, Emily Robinson, Keith Stevens, Pete Talamante
Sponsors: Carol Thompson, Jennifer Wiggins, Mentors: Wayne Bitner, Joaquin Roibal

Team 45, Escalante High, *Flu on a Plane*
Team Members: Aaron Edwards, Levi Dryden, Tanner Warren, Cameron Garcia, Jonathon Lamb, Lucas Dryden
Sponsors: Yolanda Koontz, LeAnne Salazar

Team 49, Freedom High School, *Water, Water Everywhere*
Team Members: Chris Marroquin, Seth Morgan, Jake Wright
Sponsor: Richard Foust, Mentor: Joe Vertrees

Team 50, Grants High School, *Project Troll*
Team Members: Nicholas Kemp, Devin Lowther, Jacob Alford, Tanner Bond
Sponsor: Samuel Daunt, Mentor: Peter Yanke

Team 53, Jackson Middle School, *Testing the Amount of Rainfall Affecting Animal Life*
Team Members: Randie Terrill, Adam Greenwood
Sponsor: Mrs. Karen Glennon, Mentors: Christopher Koch, Nick Bennett

Team 54, Jackson Middle School, *What are the effects of Massive rainfall in a desert environment?*
Team Members: Aidan O'Hara, Nolan Fisk
Sponsor: Mrs. Karen Glennon, Mentors: Nick Bennet, Roger Critchlow, Janet Penevolpe

Team 56, La Cueva High School, *Optimizing Community Detection*
Team Members: Alexandra Porter, Stephanie Djidjev, Lauren Li
Sponsor: Samuel Smith

Team 57, La Cueva High School, *Modeling of Predator-Prey Relationships*
Team Members: Robert McDaniels, Dennis Huang, Eli Echt-Wilson
Sponsor: Sam Smith

Team 58, Las Cruces Young Women in Computing, *Utopia*
Team Members: Marie Ellis, Samantha McGuinn, Hiba Muhyi, Noor Muhyi, Cindy Yeh
Sponsors: Rachel Jensen, Rebecca Galves, Mentors: Jen Dana, Stephanie Marquez

Team 59, Las Cruces Young Women in Computing, *9/11 AND ITS HAZARDOUS EFFECTS*
Team Members: Connie Hu, Jennifer Hu, Analyssa Martinez, Arianna Martinez, Marisa Salazar
Sponsors: Becca Galves, Rachel Jensen, Mentors: Janie Chen Alyssa Soliz

Team 60, Little Earth School, *Leatherback Sea Turtles: A Step to Protection*
Team Members: Rowan Dwyer, Fatima Gutierrez, Lynn Robey, Weston Smith
Sponsor: Juli Curtis, Mentors: Tom Robey, Susan Gibb

Team 61, Little Earth School, *Mexican Gray Wolf*
Team Members: Busayo Bird Maquebela, Sean Shepherd
Sponsor: Julie Curtis, Mentors: Tom Robey

Team 62, Los Alamos High School, *Threshold of Collapse*
Team Members: Samuel R. Baty, Peter J. Armijo
Sponsor: Lee Goodwin, Mentors: Dr. Don Tucker, Dr. Roy Baty

Team 63, Los Alamos High School, *Parallel Data Mining Using Multi-Core Computing*
Team Members: Samuel Wang, Daniel Wang, Xiaoyu Deng, Ben Liu
Sponsors: Lee Goodwin, Wyatt Dumas
Mentors: HB Chen, HsingHui Liu, Hailin Deng

Team 64, Los Alamos High School, *Warehouse Layout and Picking Simulation*
Team Members: Sudeep Dasari, David Murphy, Colin Redman
Sponsor: Lee Goodwin, Mentors: Elizabeth Cooper

Team 66, Los Alamos High School, *ExcellAnts*
Team Members: Peter Ahrens, Dustin Tauxe
Sponsor: Lee Goodwin, Mentors: James Ahrens, Christine Ahrens

Team 68, Los Alamos High School, *Using Genetic Algorithms to solve complex optimization problems*
Team Member: Alexander Swart
Sponsor: Mr. Goodwin, Mentors: Pieter Swart

Team 72, Los Alamos Middle School, *Computer Simulation of Dark Matter Effects on Galaxy Collisions*
Team Member: Cole Kendrick
Sponsor: Brian Kendrick, Mentor: Brian Kendrick

Team 73, Los Alamos Middle School, *Simulation of Multi-Agent Based Scheduling Algorithms for Waiting-line Queuing Problems*
Team Members: Steven Chen, Andrew Tang
Sponsor: Pauline Stephens, Mentor: Hsing-bung (HB) Chen

Team 74, Los Alamos Middle School, *Haptic Feedback: Controlling Robots with Touch*
Team Members: Connor Bailey, and Nate Delgado
Sponsor: Pauline Stephens, Mentor: Rob Cunningham

Team 77, Los Alamos Middle School, *Investigating the Use of Hydroelectric Power*
Team Member: Claire DeCroix
Sponsor: Pauline Stephens, Mentor: David DeCroix

Team 80, Los Alamos Middle School, *Investigating Post Los Conchas Fire Erosion and Potential Control Methods*
Team Member: Phillip Heikoop, Hayden Walker, Greyson Venhaus
Sponsor: Pauline Stephens

Team 81, Manzano High School, *Stayin' Alive*
Team Members: Brendyn Toersbijns, Spenser Gomez-Nelson
Sponsor: Karen Glennon, Mentor: Christopher Alme

Team 82, Manzano High School, *Detection of Alzheimer's Disease Plaques in a Transgenic Mouse Brain Using Image Analysis of SPION---Enhanced Magnetic Resonance Images*
Team Member: Jordan Medlock
Mentor: Laurel Sillerud

Team 83, Manzano High School, *Air Traffic Control: The Next Step!*
Team Members: Tommy Soudachanh, Khiem Tang, Ian Wesselkamper
Sponsor: Steve Schum

Team 85, Melrose High School, *Pasture-ization*
Team Members: Ethan Wright, William Boughan
Sponsors: Mrs. Raulie, Mr. Daugherty, Mentor: Mike Daugherty

Team 87, Melrose High School, *THE INTERSECTION OF LINES TO FIND POINTS*
Team Members: Randall Rush, Quinton Flores, Jon Tello, Kaleb Broome
Sponsors: Allen Daugherty, Rebecca Raulie

Team 89, Mesa Middle School, *The Effect of Immigration on American Economy*
Team Members: Leonel Herrera, Andon Jones, Karina Velazquez
Sponsors: Tracie Mikesell

Team 90, Miyamura High School, *New Methods for Electronic Security*
Team Members: Alanna Tempest, Joshua Tavares
Sponsor: Andrew Ng, Mentor: John Donahue

Team 91, Miyamura High School, *Measurement of Earthquakes*
Team Members: Tabitha Hallock, Sridivya Komaravolu, Kirtus Leyba, Jeffrey Young
Sponsor: Andy Melenchek

Team 92, Navajo Preparatory School, *The Variables Associated with Livestock and Their Relationship with Navajo Nation Grazing Policies Research*
Team Members: Bruce Wood Jr., Damon Clark
Sponsor: Mavis Yazzie

Team 94, New Mexico School for the Arts, *"Water You Waiting For," Santa Fe?*
Team Members: Mohit Dubey, Samuel Thompson, Milada Guenther, Noah Caulfield
Sponsor: Acacia McCombs, Mentor: Stephen Guerin

Team 101, Pinon Elementary, *The Bullying Effect*
Team Members: Jordan Bailey, Ruby Selvage
Mentors: Alison Bailey, Kim Selvage

Team 102, Pinon Elementary, *Visualizing the Tree of Life*
Team Member: Ryan Swart
Sponsor: Mrs. Chrien, Mentor: Pieter Swart

Team 103, Quemado High School, *San Agustin Water Crisis*
Team Members: Justin Miller, Sam Eberle, Hank Edwards, Nicole Martin
Sponsor: Tim Angelus

Team 104, Red Mountain Middle School, *Don't Waste Your Water Embrace Your Water*
Team Members: Cristian Sanchez, Jarrod Harrington, Damian Crabb
Sponsor: Guyla Miller

Team 106, Rio Rancho Cyber Academy, *Deadly Dose*
Team Members: Sierra Venegas, Jocelyn Tansey, Monika Nadzins
Sponsor: Harry Henderson

Team 107, Rio Rancho Cyber Academy, *Reverse Osmosis*
Team Members: Stuart Perara, Trinity Medley
Sponsor: Harry Henderson

Team 108, Robertson High, *Wind vs. Solar*
Team Members: Jacob Bakarich, Victoria Gomez, Jacob Ratzlaff
Sponsor: Mike Boyle

Team 109, Sandia Preparatory, *hybrid scramjet engine*
Team Member: Alex Burd
Sponsor: Neil McBeth

Team 118, Saturday Science and Math Academy, *Ant Colony Conundrum*
Team Members: Daniel Washington, Rachel Washington, Muhammad Musleh
Sponsor: Debra Johns, Mentor: Wayne Witzel

Team 121, School of Dreams Academy, *Rock, Paper, Scissors Analogy*
Team Member: Bethany Tanner
Sponsor: Creighton Edington, Mentor: Elizabeth Finley

Team 125, School of Dreams Academy, *Operation Mother Bird*
Team Members: Danielle Garcia, Kyle Wheeler
Sponsor: Creighton Edington, Mentor: Elizabeth Spenly

Team 131, St. Pius X High School, *The Perfect Poker Player?*
Team Members: James Bunch, Luis Rivera, Thien-Nam Dinh, Ethan Sabay
Sponsor: Diana Perea, Mentor: Christopher Alme

Team 132, Taos High School, *Impact of Air Force Training Flights Over Taos and Northern New Mexico*
Team Members: Rodolfo Garcia, Daniel Kroning, Melissa Pacheco, Christian Evans
Sponsor: Tracy Galligan

Team 142, Sandia Prep and La Cueva High School, *Model of a Human Bloodstream*
Team Members: Eli Echt-Wilson, Corey Miner
Sponsors: Nadine Miner, Randy Wilson, Mentor: John Miner

Team 1002, Academy for Technology and the Classics, *Pollution and Its Effects on the Santa Fe River*
Team Members: Sabrina A. Cook, Alicia Sandoval
Mentors: Su Gibbs, Irene Lee

Judges

Janeen Anderson, Acaji, Inc.
Ed Angel, Santa Fe Complex
Claudia Aprea, Northern New Mexico College
Dorothy Ashmore, Sandia National Laboratories
Nick Bennett, Grass Roots Consulting
Jon Brown, New Mexico Institute of Mining and Technology
Kent Budge, Los Alamos National Laboratory
Chuck Burch, University of New Mexico
Jesse Crawford, New Mexico Institute of Mining and Technology
Jorge Crichigno, Northern New Mexico College
Roger Critchlow, File Systems Lab
Shaun Cooper, New Mexico State University
Larry Cox, Los Alamos National Laboratory
Mike Davis, Cray Inc.
Sharon Deland, Sandia National Laboratories
Hailin Deng, Los Alamos National Laboratory
John Donahue, University of New Mexico/CAaNES
Juergen Eckert, University of South Florida
Catherine Ertz, National Center for Women & Information Technology
Drew Einhorn, Fourth Watch Software
Bruce Gale, Los Alamos National Laboratory
John Paul Gonzales, Santa Fe Institute
Jeff Grantham, New Mexico Institute of Mining and Technology
Stephen Guerin, Redfish Group
Terri Hansen, New Mexico State University
Lisa Harris, Los Alamos National Laboratory
Klaus Heinemann, University of New Mexico
Clint Hubbard, Albuquerque Police Dept
David Janecky, Los Alamos National Laboratory
Elizabeth Marie Kallman, University of New Mexico
Morris Kaufman, Department of Energy
Brian Keller, City of Clovis
Larry Kilham
Sue King, Los Alamos National Laboratory
Larry Landis, Fourth Watch Software
Carene Larmat, Los Alamos National Laboratory
Tom Laub, Sandia National Laboratories
Maximo Lazo, Science Applications International Corporation
Irene Lee, Santa Fe Institute/Project GUTS
Miguel Leyba, University of New Mexico
Lorie Liebrock, New Mexico Institute of Mining and Technology
Debbie Limback, San Juan College
Nico Marrero, New Mexico Institute of Mining and Technology
Shasta Marrero, New Mexico Institute of Mining and Technology

Cleve Moler, The MathWorks
Rocky Navarrete, New Mexico State University
Henry Neeman, Oklahoma University Supercomputing Center
Paul Nelson, City of Clovis
James Overfelt, Sandia National Laboratories
Kathy Pallis, Los Alamos National Laboratory
Alfredo Perez, Northern New Mexico College
Eunice Perez, New Mexico Institute of Mining and Technology
Dana Roberson, Department of Energy
Nayeli Ramirez, Eastern New Mexico University-Roswell
Eric Renz-Whitmore, New Mexico Technology Council
Doug Roberts, RTI International
Bob Robey, Los Alamos National Laboratory
Rachael Robey, Los Alamos High
Anthony Schroeder, Eastern New Mexico University
Shawn Secatero, New Mexico State University-Grants
Bilal Shebaro, University of New Mexico
Matthew Solano, New Mexico Highlands University
Ramesh Shakamuri, New Mexico Institute of Mining and Technology
Reffat Sharmeen, University of New Mexico
Willard Smith, Tennessee State University
Shannon Steinfadt, Los Alamos National Laboratory
Julianne Stidham, Los Alamos National Laboratory
Janice Stockard, Eastern New Mexico University
Patrick Talou, Los Alamos National Laboratory
Michael Trahan, Sandia National Laboratories
Dennis Trujillo, New Mexico State University
Greg Valdez, Sandia National Laboratories
Eleanor Walther, Sandia National Laboratories
Tim Warren, San Juan College
Jennifer Watkins, Los Alamos National Laboratory
David West, New Mexico Highlands University
Scott Wilson, University of New Mexico
Edwin Wuieve, New Mexico Institute of Mining and Technology
Peter Yanke, BX Internet
Janelle Zamie, Eastern New Mexico University

Do you want to become a supporter of the Supercomputing Challenge? Please email us at consult@challenge.nm.org for details.

Detection of Alzheimer's Disease Plaques in a Transgenic Mouse Brain Using Image Analysis of SPION-Enhanced Magnetic Resonance Images.

New Mexico Supercomputing Challenge

Final Report

April 4, 2012

Team 82

Manzano High School

Team Member: Jordan E. Medlock

Mentor: Laurel Sillerud, PhD, University of New Mexico, Department of Biochemistry and Molecular Biology

Correspondence: jordanemedlock@gmail.com

Running title: Image Analysis of Alzheimer's Plaques in MRIs of a Tg Mouse

TABLE OF CONTENTS

1. ABSTRACT

Alzheimer's disease is a debilitating and fatal brain disorder, which impacts millions of older adults. Currently, it is only definitively diagnosed using histological analysis of plaques and neurofibrillary tangles in the brain, post mortem. Magnetic resonance imaging techniques can be used to identify Alzheimer's plaques, but the identifiable plaques are very small and difficult to see in the image. This makes diagnosing, monitoring, and treating the disease very difficult. Contrast agents are being developed to increase the conspicuity of the plaques in magnetic resonance images, although this still means days of counting the plaques by hand. This computerized algorithm automates the process of counting and analyzing the plaques in magnetic resonance images. Alzheimer's plaques can be counted in approximately five seconds with this program, rather than a month of counting plaques manually.

2. INTRODUCTION

2.1 Alzheimer's Disease

Alzheimer's Disease (AD) is a fatal, degenerative brain disease that causes a reduction in both cognitive functioning and memory. People with AD also experience progressive changes to their personality, behavior and judgment. AD primarily impacts adults over the age of 65. According to The Alzheimer's Association (2011), AD is the most common form of dementia, with a prevalence of approximately 5.4 million people with the diagnosis in the United States as of 2011. AD not only affects the people who suffer from the disease but millions of family members are impacted by the necessity to provide care for their loved ones. On average, one in eight Americans over age 65 has been diagnosed with AD. As the U.S. population continues to age, due to improvements in medical care and environmental conditions, The Alzheimer's Association projects that by 2050 there could be 11 to 16 million people in the U.S. with AD.

Alzheimer's Disease is characterized by a loss of neurons in the cerebral cortex. Evidence of the increased formation of plaques and neurofibrillary tangles in the brains of patients with AD has been found under microscopy. These plaques and neurofibrillary tangles are generally found in the hippocampus, the entorhinal cortex, the basal forebrain, and the amygdala (Bouras et al. 1994). The AD plaques are deposits of regularly ordered fibrillar aggregates composed of amyloid-beta peptides (Aβ) of 36-43 amino acids located in the brain around neurons. Aβ is a peptide from an amyloid precursor protein (APP), a critical element for neuronal growth and post-injury repair (Turner et al. 2003). Neurofibrillary tangles are composed of tau proteins in a hyper-phosphorylated state (Shin et al. 1991). AD is a protein mis-folding disease, produced by abnormally folded Aβ peptides and tau proteins; however, the cause of this process is unknown (Hashimoto et al. 2003). The following figures show a representation of Aβ plaques near neurons and neurofibrillary tangles within the nucleus of the neuron, followed by a histological

image of the brain of a transgenic Alzheimer's mouse with the presence of visible plaques:

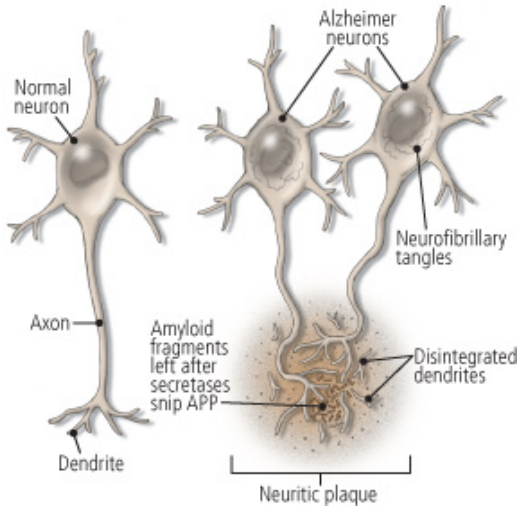


Figure 1. Representative image of Aβ plaques, from www.alz.org.

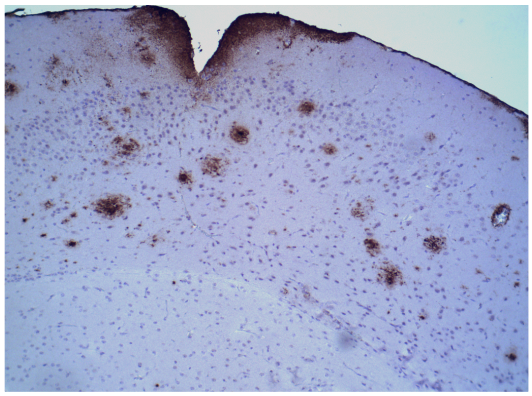


Figure 2. Histological image of Aβ plaques, from Sillerud et al.

Neuroinflammation is also present in the brains of patients with AD. Inflammation is found in many disease processes throughout the body and may be an indicator of tissue damage or an immunological response. Aβ plaques are encased in microglial cells; this is an indication that inflammation is present (Kreutzberg, 1995). The microglial cells are a type of macrophage found in brains; they are responsible for engulfing and digesting pathogens. The process by which these plaques and tangles cause atrophy and degeneration of the cerebral cortex may be related to neuroinflammation and microglial cells (Aloisi, 2001).

AD has been found to have a genetic cause in some individuals. Several genes have been found that produce proteins that enable AD development in humans. A gene has been located within chromosome 21 that is found to produce APP in different populations. A gene in chromosome 14 is transcoded into presenilin 1 (PS-1), and a gene in chromosome 1 is transcoded into presenilin 2 (PS-2). Mice were created that expressed APP, PS-1, PS-2, or tau protein genes. Most transgenic mice used in research are APP/PS-1 bigenic mice because they develop Aβ plaques and show symptoms similar to AD, such as progressive memory loss. However, these mice do not develop neurofibrillary tangles, which may impact the predictive ability of the research (Ashe, 2001). Transgenic mice that are created with tau protein genes exhibit neurofibrillary tangles but do not show signs of dementia.

2.2 Diagnosing Alzheimer's Disease

Currently, AD is diagnosed through clinical findings of neuropsycholgical symptoms, such as language and memory loss, using assessments of intellectual functioning. Other cerebral pathologies and possible causes for the dementia must be excluded to make a differential diagnosis of AD (Mendez, 2006). Neuroimaging, such as computed tomography, magnetic resonance imaging, single photon emission computed tomography, and positron emission tomography are used to exclude other causes for memory loss. In order to confirm the diagnosis, though, post-mortem brain tissue is analyzed histologically for signs of Aβ plaques. Neuroimaging techniques are not readily available to directly diagnose AD or to monitor the disease's progression, *in vivo*.

2.3 Magnetic Resonance Imaging

Magnetic Resonance Imaging (MRI) is a radiological technique used to visualize internal structures of the body. This technology employs a magnetic field to magnetize and align the magnetic poles of the nuclei within some cells in the direction of the MRI field. Tissues within the body return to their non-magnetized

state at different rates, called their relaxivity rate, which allows the MRI to translate the information into images that reflect identifiable tissues such as bone, muscle, and structures within the brain (Squire, 1997). Most often, MRIs use 1-3 Teslas (T), which is a unit of measuring magnet strength. T_1-weighted, T_2-weighted, and T_2*-weighted MRIs are standards in use for clinical and research purposes. T_1-weighted scans provide images with darker water and brighter fat, which provides good gray matter and white matter contrast for images of the brain. T_2-weighted scans show dark fat and bright water, which is well suited to show inflammation. T_2* scans increase the contrast for certain types of tissues (Squire, 1997).

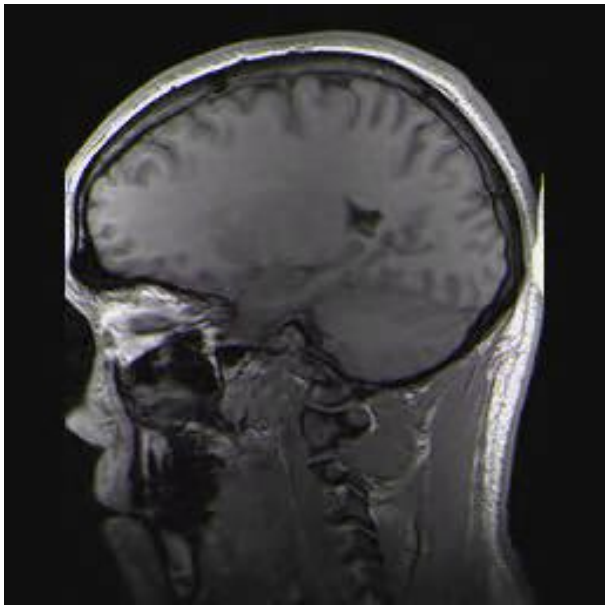


Figure 3. Para-sagittal MRI of the head, from Wikipedia Commons.

MRI has many practical uses in medicine, and it is frequently used to observe the structure of brains without harm to the patient. An MRI AD detection method would be greatly applicable for *in vivo* measurement of the progression of the disease. It has been possible to see the plaques in the brain *in vivo* with MRI; however, they are so small, around 50 micrometers (μm), that many hours, very high resolution imaging (<100 μm), and high magnetic fields (>9T) are required to identify Aβ plaques (Dhenain et al., 2009).

2.4 SPIONs

Superparamagnetic iron oxide nanoparticles (SPIONs) have been studied as a means of increasing the magnetic susceptibility of the plaques in the brain in order to view them in lower-powered MRI fields (Sigurdsson, 2008). SPIONs have an iron oxide core that is coated with a variety of other organic or inorganic materials. The superparamagnetic quality of SPIONs is characterized by a high relaxivity time, the time it takes to reach zero magnetization when not exposed to a magnetic field. This is characterized by a relaxation time τ:

$$\tau = \tau_0 e^{\frac{KV}{\kappa_B T}}$$

where τ is time (10^{-9}), K is the anisotropy energy (20,000J/m^3 for iron oxide), V is the volume of the particle, κ_B is the Boltzmann constant, and T is the temperature (Hofmann-Amtenbrink et al., 2009). The high relaxivity time of SPIONs compared to the brain's relatively low relaxivity time enables the MRI to readily visualize the differences in structures. When a SPION is coated with organic material, such as a peptide or protein, the median diameter of the nanoparticles is 50-160 nanometers (Hofmann-Amtenbrink et al., 2009). Anti-APP conjugated SPIONs and anti-Tau conjugated SPIONs selectively target Aβ plaques and bind to them in order to increase their conspicuity in MR images. These SPIONs have been shown to specifically recognize Aβ plaque in brains, which is consistent with histological studies (Sillerud et al.).

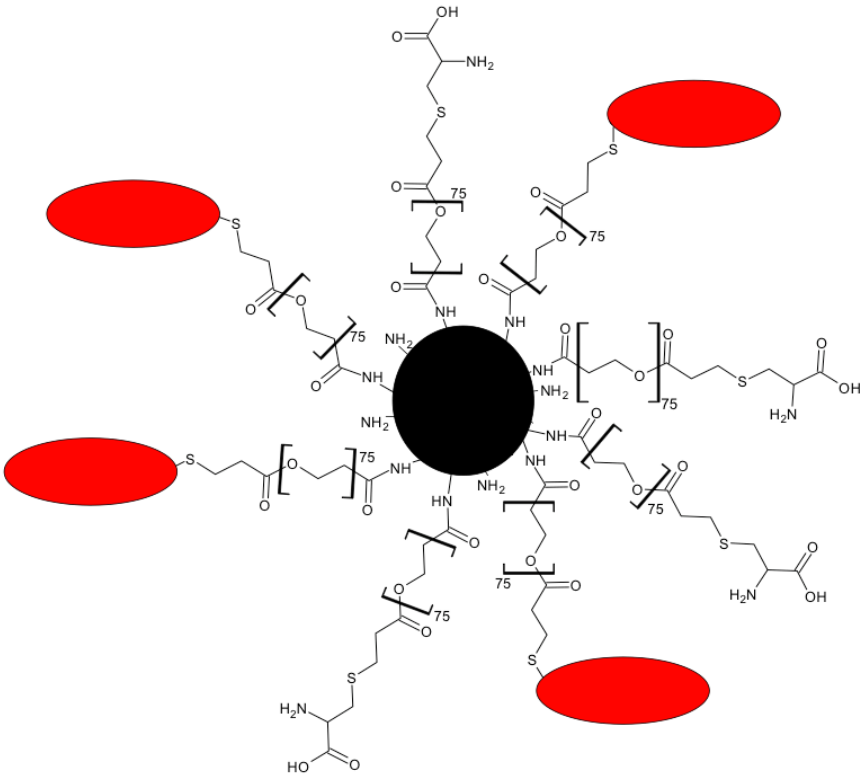


Figure 4. Structure of SPION particle coated with peptides, from Hofmann-Amtenbrink et al. 2009

Gadolinium contrast agents have been used successfully, *ex vivo*, to identify Aβ plaques; however, they leak toxic Gd^{3+} ions, which harm patients and cannot be used *in vivo* (Podulso et al., 2004). Highly invasive methods, such as injections of ozone, must also be introduced to allow the Gadolinium to cross the blood-brain barrier (BBB). Because neuroinflamation in the brains of patients with AD compromises the BBB, a contrast agent must be created that is small enough to cross the BBB without using these invasive techniques. SPIONs are non-toxic (Brambilla et al., 2011); the ferric iron in the SPION is a normally occurring element in the body and is not harmful when injected, and it is small enough to cross the BBB. This allows the SPIONs to be used to count plaques *in vivo,* in MR images.

SPION-treatment has allowed AD researchers to count plaques *in vivo*; however, counting the plaques is a long and labor-intensive process. In Sillerud et

al., the researchers took days to count plaques in an MR image of a single mouse brain. It was necessary for the researchers to find the difference between the intensities of the plaque and the brain. They computed the difference in standard deviations between the intensity of the suspected plaque and its surrounding pixels. If the standard deviation was less than 2.5, the region could not be considered plaque. The process of determining whether each darker region of the brain could be considered took many hours for each plaque. A process to automate the analysis of the image would allow both researchers and physicians to more quickly gather necessary information. Researchers could gather the data for studies related to the treatment of AD faster; improving the speed treatments are available to the people who need them most. Physicians treating people with AD could use an automated process to diagnose AD earlier and help monitor the process of the disease in individuals.

2.5 Image Analysis

Computerized algorithms are created to extract meaningful data from an image. Automating the image analysis can produce quantifiable and replicable data and reduce the time needed when compared to manual analysis. Image analysis algorithms utilize different methods including machine learning, digital geometry, and signal processing. Previous studies have used machine learning, a subcategory of artificial intelligence, to analyze MR images in order to locate plaque-like areas within the body (Nattkemper, et al. 2005). Artificial intelligence and machine learning require many data sets of recognizable points to extrapolate data from an image. In machine learning, plaques are first marked and enumerated in an image where the locations of the plaques have already been established manually. Then, the program creates parameters through trial and error until all of the regions specified by the algorithm match with the established plaques. However, when analyzing an image with less than approximately 10,000 reference points the correlation between manual and automated results is low. Artificial intelligence requires large data sets to learn to unambiguously identify plaques (Bishop, 2006).

When analyzing a single brain with 668 plaques, machine learning will not provide accurate information.

Signal processing uses arithmetic operations on discrete signals to create a desired output: the identification of plaques in the brain (Lim, 1995). The entire MR image is considered the signal, which is measured by the intensity of the pixels. Specifically, the plaques need to be differentiated from the brain and the noise that is overlaid on the image during the MRI process. Signal processing is a clear, mathematical system that provides exact data from complicated images. However, when using a signal processing image analysis algorithm with MR images complications arise. The intensities of the pixels are uneven throughout the image, which makes it difficult to use the same plaque-finding parameters. Because of the magnetic field used in the MRI, the images vary from top to bottom and from center to edges. With signal processing, though, the image can be flattened and the plaques can be thresholded and identified.

3. PROBLEM STATEMENT

Problems still exist with the visualization of Aβ plaques in humans, *in vivo,* to allow for the diagnosis and treatment of AD in the early stages. A variety of neuroimaging techniques have been used to try to visualize the plaques. In MR images, the Aβ plaques cannot be readily seen without a contrast agent. When a contrast agent is introduced (SPIONs) the plaques are numerous and time consuming to count by hand. It can take up to a month for a researcher to count the plaques in a single transgenic mouse brain. If a quick computational solution can be found to count and analyze plaques *in vivo,* in MR images, it would save many hours of time, which instead could be spent monitoring and treating AD. With a computational solution, many brains can be analyzed in a single day. My hypothesis is that the number of Aβ plaques found with a signal processing computer algorithm will correlate with the data found by manually counting the plaques in SPION-enhanced MR images of transgenic mouse brains.

4. METHODS

4.1 Subjects

All animal procedures were approved by the University of New Mexico (UNM) Institutional Animal Care and Use Committee. The animal procedures were completed by the UNM Biochemistry and Molecular Biology Department. The transgenic mouse used in this research was received from Jackson Laboratory in Bay Harbor, Maine. The mouse was a six-week-old, double transgenic Alzheimer's Disease mouse (B6C3-Tg (APPswe,PSEN1dE9) 85Dbo/Mmjax). There were two transgenes in the mouse: one was the mouse/human chimeric Aβ (A4) precursor protein (APPswe; K595N/M596L), and the other was a deletion of exon 9, which corresponds with early onset Alzheimer's Disease. After 20 weeks of age, Aβ peptide and human presenilin were detected in the mouse. By 12 months, astrocytosis was measured in the mouse, with significant cognitive impairment by 13 months. For 14 months, the mouse was given *ad libitum* access to food and water. Then, the mouse was treated with 7.07 μg of anti-tau SPION and 1.52 μg of anti-APP SPIONs by vein injection in the tail and killed 24 hours later. The brain was quickly harvested and fixed in buffered formalin for three days. The brain was stored in 2% agarose gel and 3mM NAN_3, then held at 4˚C until it was used for MRI analysis.

4.2 Magnetic Resonance Images

The MRI studies were conducted at the UNM Biomedical Research and Integrative Neuroimaging Center at 4.7 T. Dr. Laurel Sillerud provided the MR images to this team. The optimal sequence for plaque detection was a T_2-weighted image with slight T_2^*-weighting. A 192 x 1024 pixel (px) receive-only surface coil was used, and each pixel had a 60μm width. The MRI produced 32 coronal image-slices, each 120 μm wide.

4.3 Image Analysis

4.3.1 Programing Language

Many factors were considered in determining the programming language to use in processing the MR images for this study. MATLAB has a strong focus on matrix manipulations and a large set of image processing tools. Because the MR images are stored as a matrix of integer values, MATLAB has the ability to modify those images using its matrix manipulation tools. Established image-processing algorithms are available in MATLAB to locate object boundaries in binary images and extrapolate data from these boundaries, which would provide information regarding the edge and location of the Aβ plaques in the images. MATLAB, in contrast to C, C++, or Java, has boundary finding, matrix manipulation, and parallel processing tools without the need for a third party application-programming interface or the need to reinvent the processing tools by writing the entire code. In a field of multiple programming languages, MATLAB was determined to be the most applicable to this study.

4.3.2 Pre-Processing the Image

Thirty-two MR images were generated, each containing one MRI slice (1024x192 px), in the Digital Imaging and Communications in Medicine (DICOM) standard format. DICOM is a prominent means of formatting images in the scientific and medical communities. ImageJ, an image processing application, was used to convert the DICOM images into uncompressed Tagged Image File Format (TIFF), a recognizable MATLAB format, without any loss of resolution. After converting the images to TIFF, each file included 32 sub-files of MRI slices. MATLAB read each slice individually as a 16-bit unsigned integer matrix of gray scale pixels. This program then used a built-in MATLAB function to convert the 16-bit integer image into a 64-bit floating-point number matrix to enable operations that require precise outputs. A cell array was then created with the matrices from each MRI slice within it.

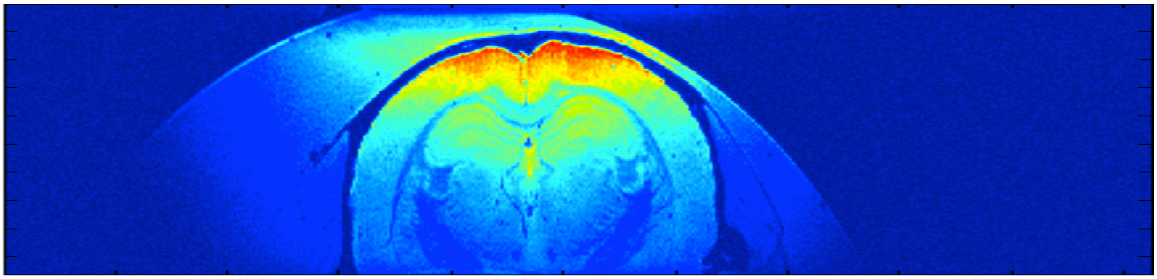


Figure 5. MR image of 10th slice in SPION treated Tg mouse brain

The first step in finding Aβ plaques within the images was to define the perimeter of the brain, thus ensuring any future findings are within the brain itself. A mask, which is a boolean image, was created to find the edge of the brain in order to differentiate the brain from the noise outside of the brain. To accomplish this, the image is displayed using the MATLAB image viewing and editing feature. Then, a freeform Region of Interest (ROI) tool was used to draw an outline around the brain. The selection is turned into an ROI object, then a mask. In this mask, the region outside of the ROI object is coded as zeros, while the region inside (the brain) is coded as ones. Later in the process this mask is used to filter out non-plaque findings.

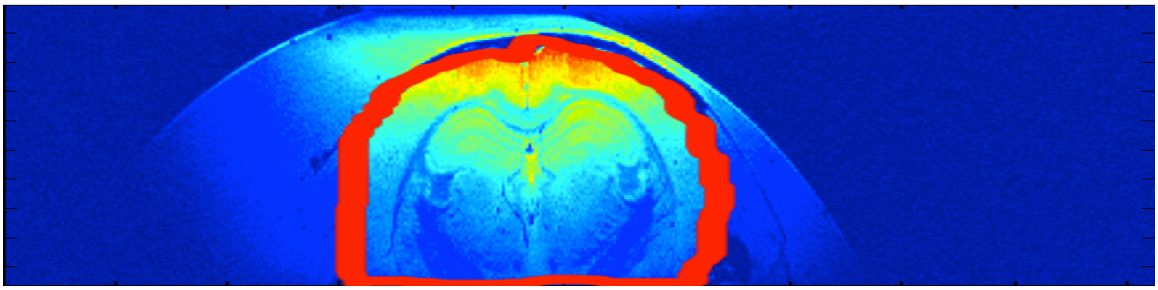


Figure 6. Freeform region of interest around brain in 10th slide

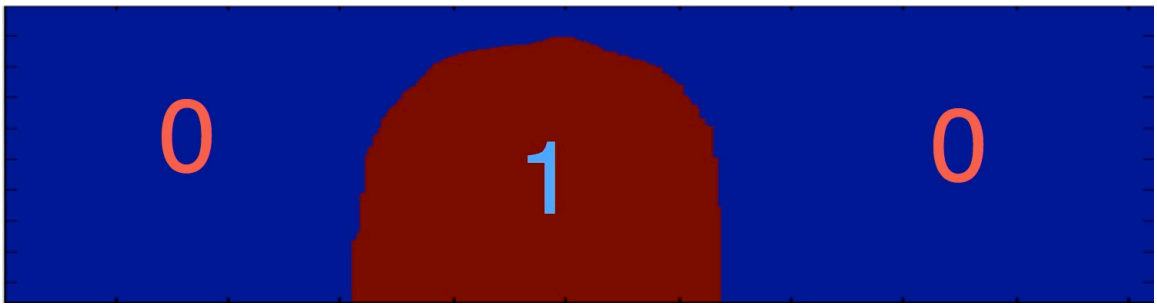


Figure 7. Boolean mask of brain in 10th slide

4.3.3 Processing the Image

On the MR image Aβ plaques appear as dark spots, or less intense regions, in the brain several pixels wide. However, there are numerous darker areas within the image that are not plaques. To differentiate plaques from non-plaque features or noise, a plaque-finding algorithm was created. Only plaques with a z-score greater than 5 standard deviations (σ) away from the mean are considered, because this ensures that there is no chance that a specific region is hypointense solely from noise. The z-score of each pixel is calculated by dividing the intensity of the pixel by the standard deviation of the noise. The standard deviation is the average distance from the mean pixel intensity, found using the formula below where n is the number of pixels, μ is the average intensity of the group of pixels, and x_i is the intensity of the pixel.

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2}$$

The standard deviation of the noise was found by defining a forty-by-forty grid of pixels on the upper left corner of the image, because they are the pixels furthest from the brain. The larger the standard deviation of the noise the less likely the pixel represents noise. Then, the image is divided by the standard deviation of the noise to create a z-score image, or an image where every pixel's intensity represents its difference from the average noise intensity. Finding the standard deviation allows the images to be standardized so that every brain can be analyzed using the same parameters.

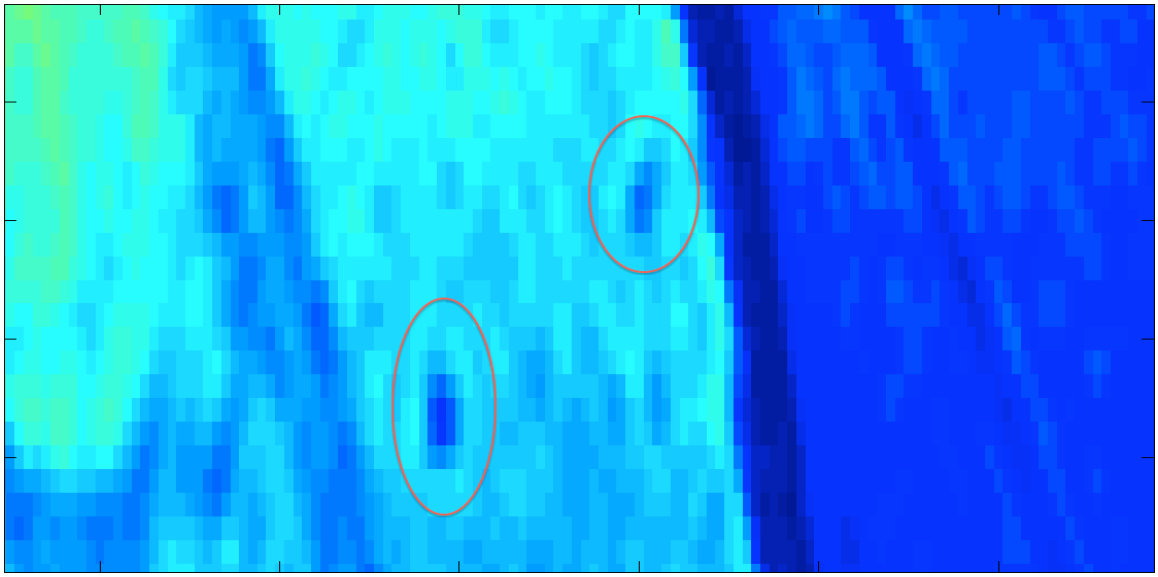


Figure 8. SPION-enhanced Aβ plaques in MR image of 10th slice

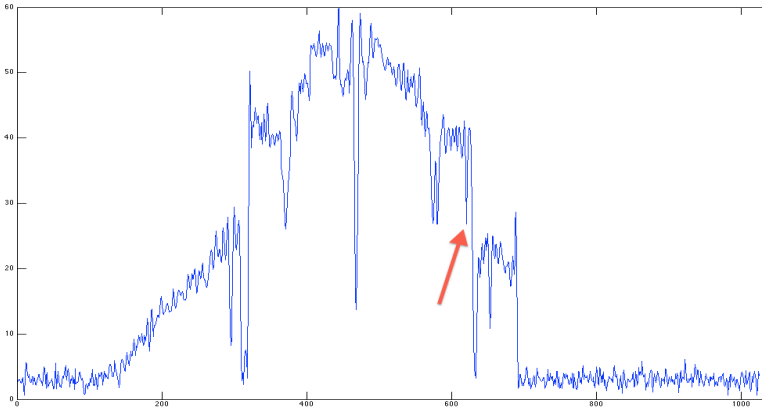


Figure 9. Z-score intensity graph of a row of pixels in 10th slice; red arrow points to small hypointense region/plaque (row runs through upper plaque in Figure 9)

The MR images have an uneven intensity distribution, so the image cannot be analyzed using the same parameters throughout the image without normalizing the intensities first. The intensity of the magnetization during the MRI is greatest at the top of the image and reduces with distance of the brain from the MRI machine. The standard deviation and intensity of the image at the top is different than it is on the bottom, as it is in the center versus the edges. Initial attempts to correct for this difference using a method of flattening the entire image increased the standard

deviation of the noise and resulted in no findings. To account for the uneven intensities, the program divided the image into one-pixel-wide rows and treated each row as a separate image; however, the center of the image it is still more intense than at the edges. It was determined that the intensity of the signal of the brain itself could be flattened without impacting the noise outside of the brain. The program used the mask to allow only the part of the row that contains the brain to be flattened, without flattening the noise. Then, the row was recombined with the noise, leaving the brain flat with the same original standard deviation while the noise is at a different intensity level.

The image of the brain was flattened using a Gaussian filter, which blurs the image and then subtracts the blurred image from the original. A Gaussian blur uses a Gaussian distribution to modify the intensity of each pixel based on the pixels around it. The equation for the Gaussian distribution is:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where G is the pixel intensity at (x, y), x and y are the horizontal and vertical locations of the pixel respectively, and σ is 10 so that the program could have a sufficiently blurry image to use as a background shape. The Gaussian blur image is just the background shape of the image, meaning that when the Gaussian blur image is subtracted from the original image the background shape is removed, leaving only the intensities as they relate to the background shape.

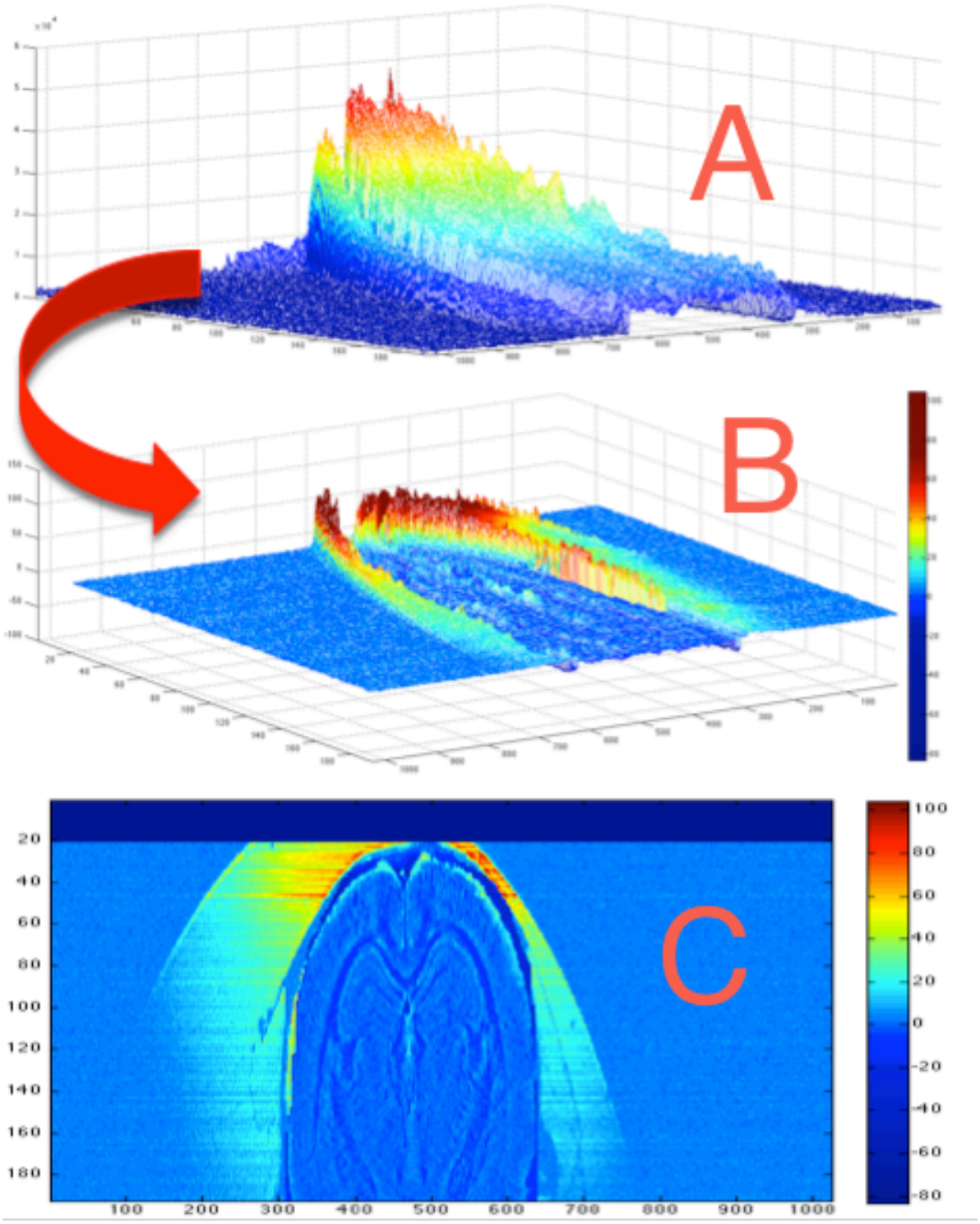


Figure 10. A: A mesh graph of intensity of the MR image before it is flattened

B: Mesh graph of the intensity of the MR image after flattening

C: Same as B but viewed as an image

The values of the intensities of the brain were then centered at zero, so the intensity of the pixels represents a difference from the average value of the brain. To accomplish this, the average value of the noise was found by using a built-in MATLAB mean-finding function. The resulting number is then subtracted from every value in the array, resulting in the noise being centered at zero. Using the mask, the program found the average intensity of only the brain, and not the noise.

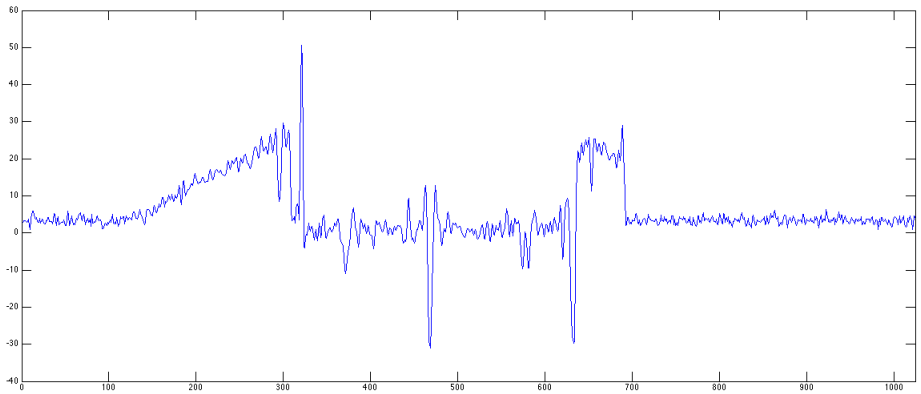


Figure 11. Z-score intensity graph of a row from 10th slice after the brain was centered at zero

Because the plaques have a lower intensity than the brain, the intensities were inverted so that the plaques will have greater values than the brain itself. The program then thresholded the image at 5, creating a binary image with values greater than 5 represented by 1 and values less than 5 are represented by 0. This binary image represents every pixel that has a distance of 5σ from the average value of the signal, based on whether the pixel value is 1 or 0. The areas that remain of this image are the plaques and the large areas in or outside the brain that are less intense than the brain itself.

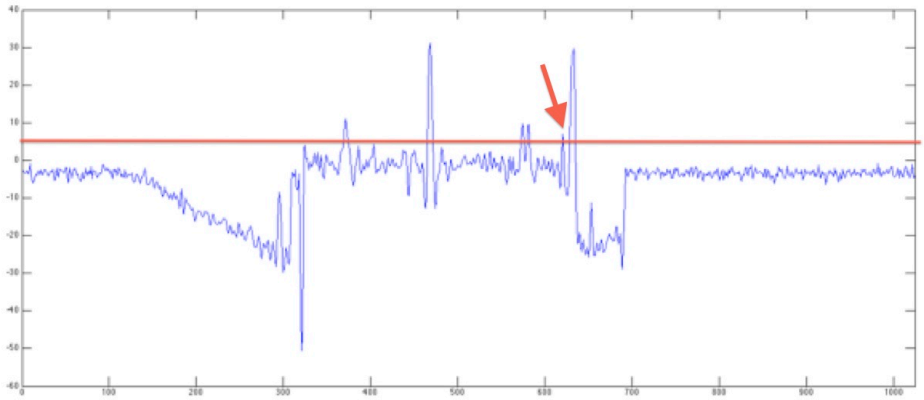


Figure 12. Inverted row with red line at 5 where the image will be thresholded; one plaque is shown above the threshold while there are multiple other spikes in the graph above 5

The binary image was then converted into a list of regions. The MATLAB function `bwboundaries` found the coordinates of every pixel inside a highlighted region and the area of that region. The program then used the coordinates found from `bwboundaries` to find the z-score of the corresponding area in the image. A list was then compiled containing each perspective plaque's coordinates in x, y, z, their z-score relative to the brain, their size in pixels, and a metric describing how close to circular they are, by calculating the ratio $\frac{P}{2\sqrt{\pi A}}$ where P is the perimeter of the finding and A is the area in pixels (See sample data set in Appendix).

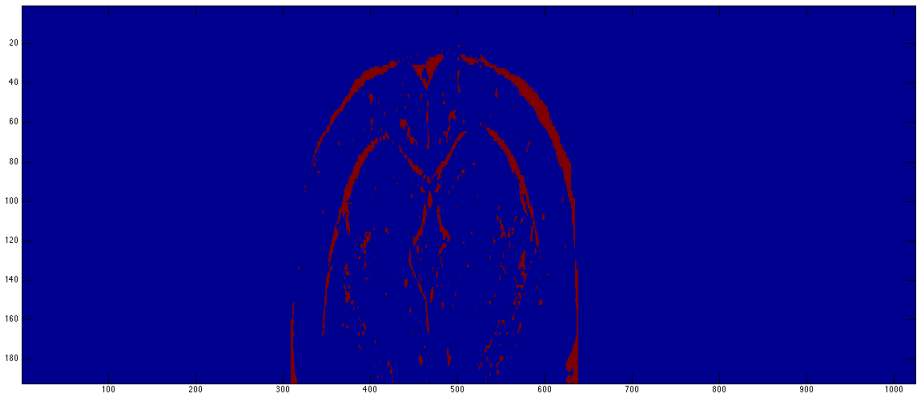


Figure 13. Binary image of findings in 10th slide before filtering non-plaque

4.3.4 Filtering Non-Plaque Findings

The compiled list of regions now contains many values that are not plaque, including components of the brain, which are dark relative to the rest of the brain. To distinguish plaques, the program filters the list of findings according to their size, z-score, and position. A parameter-sweeping script was used to find optimum upper and lower size thresholds as well as the minimum z-score. The parameter-sweeping program found that the optimum minimum size threshold is three pixels in area, the optimum maximum size threshold is ten pixels in area, and a minimum z-score threshold is 5σ. These parameters yielded the greatest possible correlation between the manual results and the automatic results created in this program.

4.3.5 Parallel Processing

In order to increase the speed and efficiency of the program, a built-in MATLAB structure `parfor` was used. This structure iterates over a set of values, running the contents simultaneously and independently on all processors and threads. The `parfor` structure allowed this program to run in around 5 seconds on a quad-core, 8-thread, Intel i7 processor.

5. RESULTS

To compare the manual results to the automated results, a MATLAB script was written that created a matrix of the number of plaques per slice for the manual results versus the number of plaques per slice by the program. From these arrays a scatter plot was made to compare the two data sets. The correlation coefficient that resulted from the comparison between the automatic results and the manual results is 0.68, which is a sufficiently large coefficient of correlation to be considered representative of the number of plaques in a brain. The automated results produced 1,723 possible plaques in the brain while the manual results contained 668 plaques, which is a difference of 1,055 plaques.

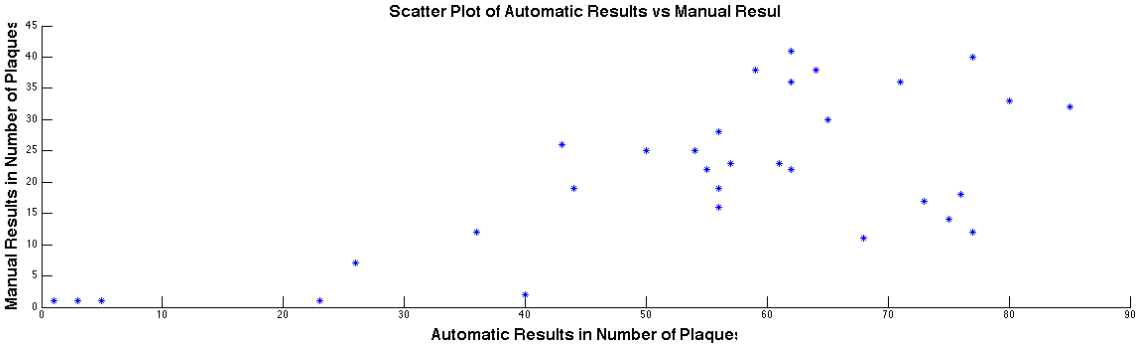


Figure 14. Scatter plot of Sillerud et al. manual results vs. automated plaques per slice

6. CONCLUSION

Alzheimer's disease is a deadly brain dysfunction that impacts millions of older adults. Currently, physicians diagnose AD by neuropsychological evaluation and the ruling out of other possible causes for dementia. A definitive diagnosis can only come after the patient's death with histological examination of the brain to identify plaques and neurofibrillary tangles. There are no diagnostic tools, in use, that can provide a diagnosis of AD while the patient is still alive. MRIs provide information regarding the volume and structure of the brain but the plaques are so small and difficult to identify that prohibitively strong MRI fields are necessary to view them. Recently, SPIONs have been introduced as a contrast agent that can be coated with organic material that will specifically bind to Aβ plaques to increase their conspicuity when viewing the MR images. When subjects are injected with the SPIONs, the plaques become visible on MRI and can be counted. Researchers at UNM are counting the plaques in MR images of transgenic mouse brains through a laborious manual process for each possible plaque. An automated means of locating and counting these plaques has not been available. Now, researchers and clinicians can use this signal processing image analysis algorithm to vastly decrease the time necessary to analyze each brain.

The correlation of 0.68 between the automated results and the manual results indicate a moderately high accuracy of the quantity, positions, intensities, sizes, and shapes of Aβ plaques in SPION-enhanced MR images of AD transgenic mouse brains. Because of this correlation, the relative amounts of plaques in the brain can be predicted from the number of plaques in the automated results. The results from this study can be used as a reasonable replacement for human labor when counting Aβ plaques in SPION-treated patients and comparing them between other automated results.

This signal processing algorithm found 1,723 plaques while the manual results found 668 plaques. The probable reason for an increase in plaques from the automated results is that the algorithm picked up more regions in the brain that have similar visual properties as plaques, but are not plaques, such as the ventricles

and the corpus colossum in certain slices. The additional findings may also be related to improved accuracy with a computerized algorithm in finding plaques that are difficult to visualize. The correlation of findings, however, indicates an accurate representation of the differences in quantity between slices. This allows for a true comparison using this algorithm between different brains. It would not provide an accurate comparison between the data gathered manually on a brain and the data gathered by this program on another brain. It is critical that the algorithm provide an accurate representation of the difference in the number of plaques per brain, while the absolute number is less important.

With the accuracy of these results, this program also ran in approximately five to ten seconds for a brain. When comparing this time to analyze an MR image to the days it is currently taking, it becomes more practical to use this computer algorithm. The program is a more viable tool for measuring plaque density than the previous method of marking and counting the plaques by hand. A program that can be run concurrently, in 10 seconds, as the MRI data is being collected would make the diagnosis and treatment of AD a quick and seamless process. This automated process can also give more quantitative information of not only the number of plaque and their intensities but also values that cannot be found by hand, such as size and roundness. These will give researchers and clinicians new data points to reference, more data that they can use to determine the efficacy of drugs or the progression of the disease in patients.

This program is the culmination of many researchers looking for faster and more efficient means of diagnosing, monitoring, and treating AD. Not only will faster results allow physicians to diagnose AD after an MRI, it will also allow for the physician to monitor the progress of the disease and efficacy of the treatment modalities. This program will also save many researchers the time and money that it would take to count plaques by hand, allowing for the expedited research of future drugs and treatments for AD. Giving researchers and physicians a tool to expedite this process is the most significant original achievement of this project.

7. DISCUSSION

Developing this program has been a trying and rewarding experience. I was able to use the knowledge I have gathered from learning a variety of programming languages including C++, Java, MATLAB, and others to develop a strong knowledge of what it takes to make a scientific application. Since beginning to participate in the Supercomputing Challenge, during my freshman year, I have learned that I enjoy coding and do it at every opportunity. At this time, I am working on several computationally intense coding projects to develop iOS and web applications. I am grateful that this challenge has opened up the opportunities to pursue coding and application development.

This particular program has failed many times. I tried over twenty different ideas for how the program should work; every time something went wrong, and I decided to make it simpler. I started in the middle of last summer with an extremely complex 300+ line algorithm that did not work. This experience showed me that a huge, complex algorithm was the wrong approach to take. Then, once my program seemed to be running perfectly I found the plaques per slice correlation compared to Dr. Sillerud's manually gathered results was negative, and I had to completely redesign the algorithm again. This project has taught me to be patient and start simply and work my way toward more complexity.

Next, I will work on applying this program to the MRI data that Dr. Sillerud is using to conduct his research on the inhibition NF-κB and its effects on AD as well as pancreatic cancer. I am also looking to acquire a larger data set to apply this program to. The number of brains I have available to analyze may increase my ability to use an element of machine learning in the algorithm to decrease the number of false positives. When this program is applied to multiple brains to show the benefits of drug protocols in decreasing plaque density, Dr. Sillerud is planning on helping me publish the results in a peer-reviewed journal.

I am also looking for other problems to apply my program to; medical processes that need a large number of objects to be counted and analyzed. I am consulting with Dr. Sillerud at the UNM Department of Biochemistry and Molecular

Biology to discuss possible other applications for this program. Possible applications could be identifying cancer cells, tumors, or other pathogens.

Working on this project has stimulated the idea of studying computer science and, more specifically, image processing and artificial intelligence because it is such a fascinating emerging field. Biochemistry was never a field I considered for my future but I have found the idea that my program may help alleviate personal suffering through research exciting. Completing this program in a medical field has opened up the possibility of pursuing a field like computational biology. There are so many applications of image analysis and artificial intelligence in the medical field, which could help solve the problems that are so important to people.

8. ACKNOWLEDGEMENTS

First, I would like to thank Dr. Laurel Sillerud, my mentor, who worked continuously with me developing the algorithm and teaching me the science behind what he, and myself, were working on. It has been an amazing learning experience for me. I understand what a huge opportunity it is for me to work in such a field this early in my school career. It has been an unparalleled privilege for me to be a part of his team.

Then, I would like to thank the entire research team at UNM Department of Biochemistry and Molecular Biology for their tremendous help and support throughout the project.

Also, I would like to thank my mom, Alyx Medlock, for her support and help as a sounding board in problem solving my program. Not to mention, the work she put into editing this paper to help me make it perfect. She was invaluable in helping me manage my time and the deadlines to get this project finished.

Lastly, I would like to thank my sponsor, Mr. Shanley, for helping with the grammar and editing of this paper.

9. BIBLIOGRAPHY

Aloisi, F. (2001). Immune function of microglia. *Glia, 36*, 165-179.

Alzheimer's Association. (2011). *Alzheimer's Disease Facts and Figures*. Alzheimer's Association, Washington, D.C.: Available at www.alz.org.

Ashe, K.H. & Zahs, K.R. (2010). Probing the biology of Alzheimer's disease in mice. *Neuron. 6*, 631-645.

Ashe, K.H. (2001). Learning and memory in trasngenic mice modeling Alzheimer's disease. *Learning and Memory. 8*, 301-308.

Bachstetter, A.D., Xing, B., De Almeida, L., Dimayuga, E.R., Watterson, D.M., & Van Eldik, L.J. (2011). Microglial p38α MAPK is a key regulator of proinflammatory cytokine up-regulation induced by toll-like receptor (TLR) ligands or beta-amyloid (Aβ). *Journal of Neuroinflammation. 8*, 79.

Benveniste, H. et al. (2007). Anatomical and functional phenotyping of mice models of Alzheimer's disease by microscopy. *Annals of the New York Academy of Sciences. 1097,* 12-29.

Bishop, C.M. (2006). *Pattern Recognition and Machine Learning.* New York, NY: Springer.

Bouras, C., Hof, P.R., Giannakopoulos, P., Michel, J.P., & Morrison, J.H. (1994). Regional distribution of neurofibrillary tangles and senile plaques in the cerebral cortex of elderly patients: A quantitative evaluation of a one-year autopsy population from a geriatric hospital. *Cerebral Cortex. 4*, 138-150.

Brambilla, D. et al. (2011). Nanotechnologies for Alzheimer's disease: Diagnosis, therapy, and safety issues. *Nanomedicine. 7*, 521-540.

Chamberlain, R. et al. (2009). Comparison of amyloid plaque contrast generated by T2-weighted, T2*-weighted, and susceptibility-weighted imaging methods in transgenic mouse models of Alzheimer's disease. *Magnetic Resonance Medicine. 61*, 1158-1164.

Dhenain, M. et al. (2009). Characterization of in vivo MRI detectable thalamic amyloid plaques from APP/PS1 mice. *Neurobiology of Aging. 30*, 41-53.

Fodero-Tavoletti, A.T., Villemagne, V.L., Rowe, C.C., Masters, C.L., Barnham, K.J., & Cappai, R. (2011). Amyloid-β: The seeds of darkness. *International Journal of Biochemical Cell Biology. 43*, 1247-1251.

Games, D. (1995). Alzheimer-type neuropathology in transgenic mice overexpressing V717F beta-amyloid precursor protein. *Nature. 373*, 523-527.

Granic, I., Dolga, A.M., Nijholt, I.M., Van Dijk, G., & Eisel, U.L. (2009). Inflammation and NF-κB in Alzheimer's disease and diabetes. *Journal of Alzheimer's Disease. 16*, 809-821.

Hashimoto, M., Rockenstein, E., Crews, L., & Masliah, E. (2003). Role of protein aggregation in mitochondrial dysfunction and neurodegeneration in Alzheimer's and Parkinson's diseases. *Neuromolecular Medicine. 4*, 21-36.

Herbert, L.E., Scherr, P.A., Bienias, J.L., Bennett, D.A., & Evans, D.A., (2003). Alzheimer's disease in the U.S. population: Prevalence estimates using the 2000 census. *Archives of Neurology. 60*, 1119-1122.

Hofmann-Amtenbrink, M., Von Rechenberg, B., & Hofmann, H. (2009). Superparamagnetic nanoparticles for biomedical applications. In M.C. Tan (Ed.), *Nanostructured Materials for Biomedical Applications* (pp. 120-149). Kerala, India: Hindawai Publishing Corporation.

Holmes, C. (2008). Long-term effects of Abeta-42 immunisation in Alzheimer's disease: Follow-up of a randomized, placebo-controlled phase I trial. *Lancet. 372*, 216-223.

Jack, C.R. et al. (2004). In vivo visualization of Alzheimer's amyloid plaques by MRI in transgenic mice without a contrast agent. *Magnetic Resonance Medicine. 52*, 1263-1271.

Kreutzberg, G.W. (1995). The first line of defense in brain pathologies. *Drug Research. 45*, 357-360.

Lim, J.S. (1990). *Two-dimensional signal and image processing*. Englewood Cliffs, NJ: Princtice Hall.

Lin, M.M., Kim, D.K., El Haj, A.J., & Dobson, J. (2008). Development of superparamagnetic iron oxide nanoparticles (SPIONs) for translation to clinical applications. *IEEE Transactions on Nanobioscience. 7*, 298-305.

Math Works (2012). www.mathworks.com

McKhann, G. et al. (1984). Clinical diagnosis of Alzheimer's disease. *Neurology. 34,* 939-944.

Mendez, M.F. (2006). The accurate diagnosis of early-onset dementia. *International Journal of Psychiatry Medicine. 36*, 401-412.

Mukherjee, S., Dudley, J.I., & Das, D.K. (2010). Dose-dependency of resveratrol in providing health benefits. *Dose-Response. 8*, 478-500.

Mundt, A.P et al. (2009). Targeting activated microglia in Alzheimer's pathology by intraventricular delivery of a phagocytosable MRI contrast agent in APP23 transgenic mice. *Neuroimage. 46*, 367-372.

Nattkemper, T.W. et al. (2005). Evaluation of radiological features for breast tumor classification in clinical screening with machine learning methods. *Artificial Intelligence in Medicine. 34,* 129-139.

Niolaev, A., McLaughlin, T., O'Leary, D., & Tessier-Lavigne, M. (2009). N-APP binds DR6 to cause axon pruning and neuron death via distinct caspases. *Nature. 457*, 981-989.

Podulso, J.F. et al. (2011). Targeting vascular amyloid in artierioles of Alzheimer disease transgenic mice with amyloid β protein antibody-coated nanoparticles. *Journal of Neuropathology & Experimental Neurology. 70*, 653-661.

Poduslo, J.F. et al. (2002). Molecular targeting of Alzheimer's amyloid plaques for contrast-enhanced magnetic resonance imaging. *Neurobiology of Disease. 11*, 315-329.

Rivere, C., Richard, T., Quentin, L., Krisa, S., Merillon, J.M., & Monti, J.P. (2007). Inhibitory activity of stilbenes on Alzheimer's beta-amyloid fibrils in vitro. *Bioorganic & Medicinal Chemistry. 15*, 1160-1167.

Selkoe, D.J. & Schenk, D. (2003). Alzheimer's disease: Molecular understanding predicts amyloid-based therapeutics. *Annual Review of Pharmacology & Toxicology. 43*, 545-584.

Shin, R.W., Iwaki, T., Kitamoto, T., & Tateishi, J. (1991). Hydrated autoclave pretreatment enhance tau immunoreactivity in formalin-fixed normal and Alzheimer's disease brain tissues. *Laboratory Investigation. 64*, 693-702.

Sigurdsson, E.M. et al. (2008). A non-toxic ligand for voxel-based MRI analysis of plaques in AD transgenic mice. *Neurobiology of Aging. 29*, 836-847.

Sillerud, L.O. et al. (unpublished). SPION-enhanced MRI shows that inhibition of NF-κB concomitantly lowers Alzheimer's plaque formation and microglial activation in transgenic mouse brain.

Sutcliffe, J.G., Hedlund, P.B., Thomas, E.A., Bloom, F.E., & Hilbush, B.S. (2011). Peripheral reduction of β-amyloid is sufficient to reduce brain β-amyloid: Implications for Alzheimer's disease. *Journal of Neuroscience Research. 89*, 808-814.

Taylor, R.M., Huber, D.L., Monson, T.C., Ali, A.S., Bisoffi, M., & Sillerud, L.L. (2011). Multifunctional iron platinum stealth immunomicelles: Targeted detection of human prostate cancer cells using both fluorescence and magnetic resonance imaging. *Journal of Nanoparticle Research. 13*, 4717-4729.

Tiraboschi, P., Hansen, L.A., Thal, L.J., & Corey-Bloom, J. (2004). The importance of neuritic plaques and tangles to the development and evolution of AD. *Neurology. 62*, 1984-1989.

Turner, P.R., O'Connor, K., Tate, W.P., & Abraham, W.C. (2003). Roles of amyloid precursor protein and its fragments in regulating neural activity, plasticity and memory. *Progress in Neurobiology. 70*, 1-32.

Wang, A., Das, P., Switzer, R.C., Golde, T.E., & Jankowsky, J.L. (2011). Robust amyloid clearance in a mouse model of Alzheimer's disease provides novel insights into the mechanism of amyloid-beta immunotherapy. *Journal of Neuroscience. 31*, 4124-4136.

Waring, S.C. & Rosenberg, R.N. (2008). Genome-wide association studies in Alzheimer disease. *Archives of Neurology. 65*, 329-334.

Zhang, F., Liu, J., & Shi J.S. (2010). Anti-inflammatory activities of resveratrol in the brain: Role of resveratrol in microglial activation. *European Journal of Pharmacology. 636*, 1-7.

Computer Simulation of Dark Matter Effects on Galaxy Collisions

New Mexico

Supercomputing Challenge

Final Report

April 4, 2012

Team Number 72

Los Alamos Middle School

Team Members:

Cole Kendrick

Teachers:

Brian Kendrick

Project Mentor:

Brian Kendrick

Summary

The main goal of this project is to develop a computer program to model two galaxies colliding including dark matter mass. The questions this project will answer are: How will dark matter effect colliding galaxies, how accurately can this be modeled, how big of an impact does dark matter have on colliding galaxies. Last year, a computer program was developed to model the effects dark matter had on a galaxy's rotational curve. That program was written in C, and this year that program has been modified to fit this year's problem. This year, the code has been modified to do two galaxies and include galaxy-galaxy interactions. Currently, for this model dark matter is being treated by a large mass point at the center of the galaxy. This model is also using the same computational methods as last year, the velocity Verlet method (2^{nd} order) is being used for Newton's laws of motion. Last year, a nearest neighbor method was implemented. This method is a much quicker way of solving a n-body calculation without doing full n-body computation. This method will be described in more detail later on. Andromeda and the Milky Way (MW) were the main focus of this project, 3 different cases were ran to determine how much dark matter really affected a collision. All simulations that were ran had 4000 stars (2000 per galaxy) and a time step (dt) of 200 years. The three cases that were ran are: (1) Normal mass – the realistic mass values we predict the galaxies to have, (2) MW has ¼ of Andromeda's dark matter mass, and (3) Normal mass with same rotational direction. My main results show that dark matter is needed to keep the galaxies stable throughout the collision. With case 2, the Milky Way galaxy was torn apart by Andromeda and eventually orbited it, with the normal mass cases 1 and 3, the galaxies eventually merge and become an elliptical galaxy over time. Without enough dark matter mass in each galaxy, the collision will not be stable. My results were compared to professional simulation results of the Andromeda and Milky Way collision as well as realistic mass distributions (what we observe).

Table of Contents

Introduction

What is dark matter and why do we need it?

No one knows what dark matter really is, although it has been one of the many theories that exist to help explain the flat rotational curve of all galaxies. Dark matter was first introduced in the 1930's but many people did not really support it until later in the

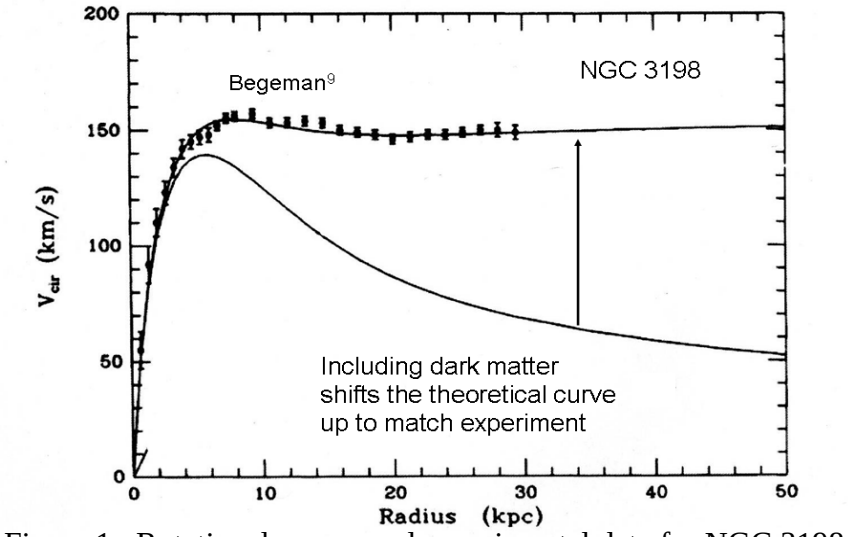


Figure 1: Rotational curves and experimental data for NGC 3198

1970's. In the 1970's people had begun collecting experimental data from galaxies by taking the brightness of the galaxy. They also made computer programs to test the dark matter theory. According to Newton's laws of motion, a galaxy rotation curve should dip down towards the edge of the galaxy. However, the experimental curve is relatively flat (see Fig. 1). Dark matter is one of the theories to resolve the difference between the two curves. Another theory that exists is called MOND (Modified Newtonian Dynamics) which changes Newton's laws of gravity to fit the flat rotational curve. By using the dark matter and MOND theory we can better understand how galaxies function and learn more about our tiny section of the universe.

How are galaxy rotational curves measured?

Galaxy rotational curves are measured from the Doppler shifts of the Hydrogen 21 cm line. The shift is broken down into red and blue, the red shift is going away and blue shift is coming towards you.

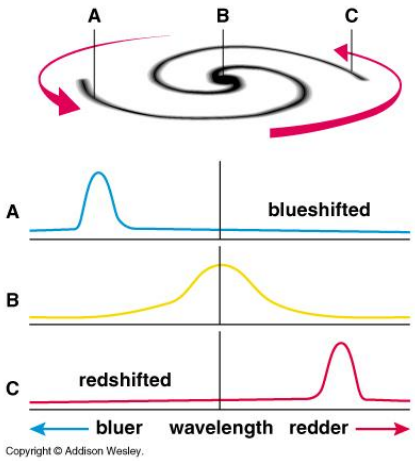


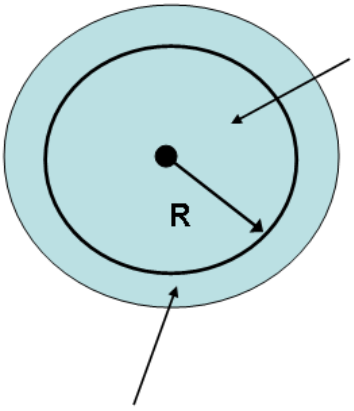
Figure 2: Schematic showing the blue and red Doppler shifts which are used to measure a galaxy's rotational velocity.

Problem

The two main questions my project will answer are: Can I successfully create a computer model to simulate the effects dark matter has on galaxy collisions? How accurate will my model be? Dark matter in my model is being treated by using the Navarro-Frenk-White (NFW) dark matter mass distribution. This distribution is a spherically symmetric static mass that encompasses the galaxy (Fig. 3). The same basic simulation methods that were used last year are also being implemented. Newton's law of motion F=m a (total force = mass times acceleration) will be used to move the visible matter ("stars") in all of my simulations. The force is due to gravity acting between all of the visible matter plus a large central force at the center of the galaxy due to dark matter. The force between any two mass points m_1 and m_2 is given by Newton's law of gravity $F=G\ m_1m_2/r^2$ where G is the universal gravitational constant and r is the distance between the two points.

Dark Matter Model

Dark matter mass distribution (the blue sphere) is static, spherically symmetric, and surrounds entire galaxy



The total gravitational force due to the dark matter mass M(R) enclosed inside radius R is the same as that of a **single mass point** (the black dot) of mass M(R) located at the center of galaxy

Newton's "Shell Theorem"

The gravitational force due to the dark matter mass outside radius R adds to zero

Galaxy Model

The same galaxy model is being implemented into this years code, although this year there is two galaxies instead of one. My galaxy model is very similar to our solar system, instead of planets orbiting the Sun, stars orbit the dark matter and black hole of the galaxy. Also, instead of 8 planets, there are trillions of stars. My galaxy model takes those stars and represents many stars as one because doing a trillion stars on single computer will take forever. My galaxy model calculates the mass of the stars by taking the total mass of the core or disk (Mcore or Mdisk) and dividing that by the number of stars in the core (N-N0) or disk (N0). The total number of stars is N. The formula in my

model is M = Mdisc/N0 or Mcore/ (N -N0) to get the mass of a star. The figure 4 below shows how my galaxy model is similar to a solar system.

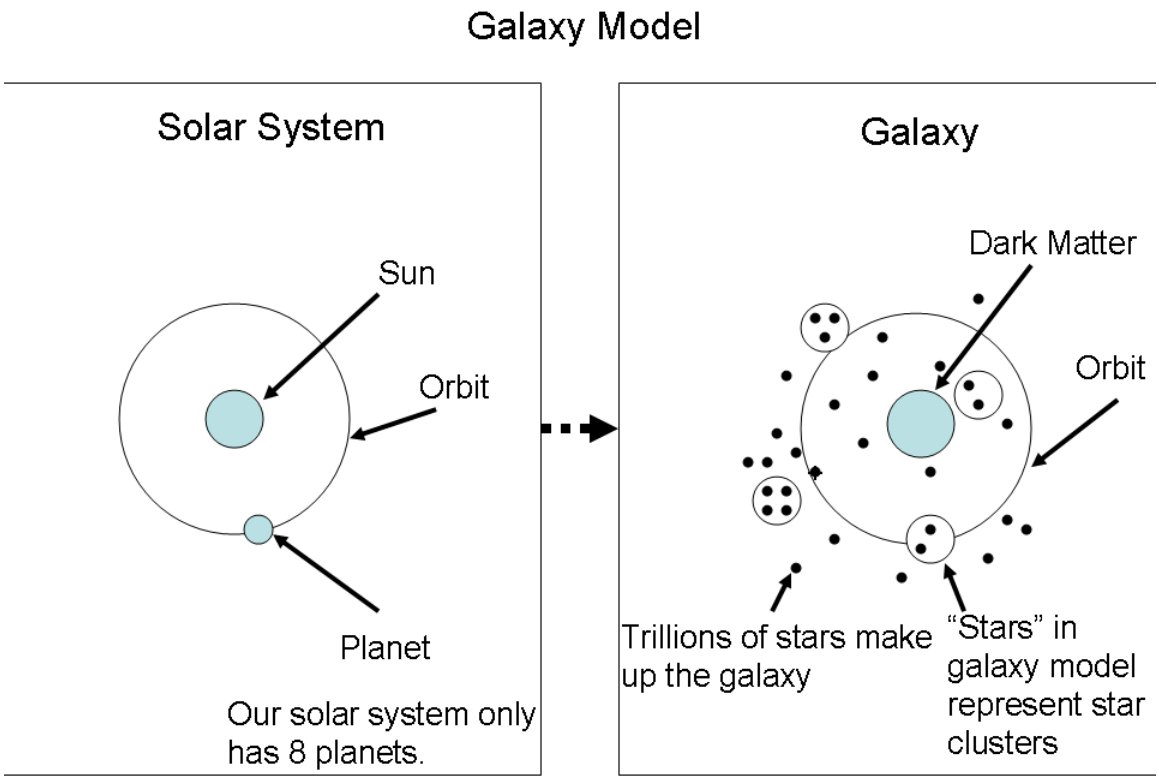


Figure 4: Schematic showing how my galaxy model resembles the solar system.

Nearest Neighbor

The nearest neighbor method is used in all of my calculations, and is a much quicker way of doing a n-body problem. Nearest neighbor creates a radius around each star and determines whether or not another star is in that radius. If the star is inside of the radius then the program stores that stars ID in an index and computes the gravitational force between it. The nearest neighbor radius size changes over the distance from the center of the galaxy. Therefore, the radius is small near the center of the galaxy because of the higher density of stars. On the edge of the galaxy the radius is larger because of the less dense areas. The optimal range is 1-11 kpc which was determined in my last year's project. (See fig. 5 below)

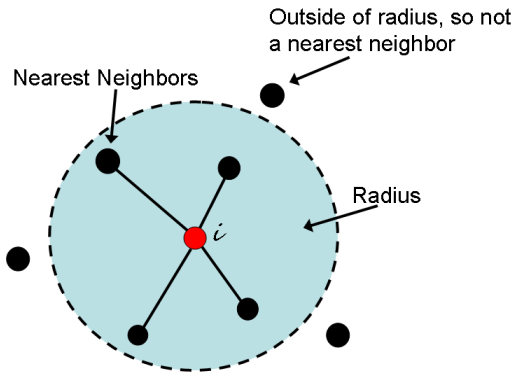


Figure 5: Schematic showing how the nearest neighbor method works.

Collision Model

This year, galaxy to galaxy interactions were added, which is the force behind the collisions. This year the program was also parallelized using MPI (see pg. 18) whereas last year the program was parallelized using OpenCL which utilizes the GPU instead of the CPU. The collision model shows how the interactions between galaxies are implemented in my model. Figure 6 (below) shows two galaxies colliding. The green arrow represents core to core interactions which are the dark matter halos. The blue arrow represents the core to star interactions, and the orange arrow represents the star to star interactions which is the nearest neighbor method. Each galaxy has internal interactions and external interactions. The nearest neighbor method that was above is used in both internal and external interactions.

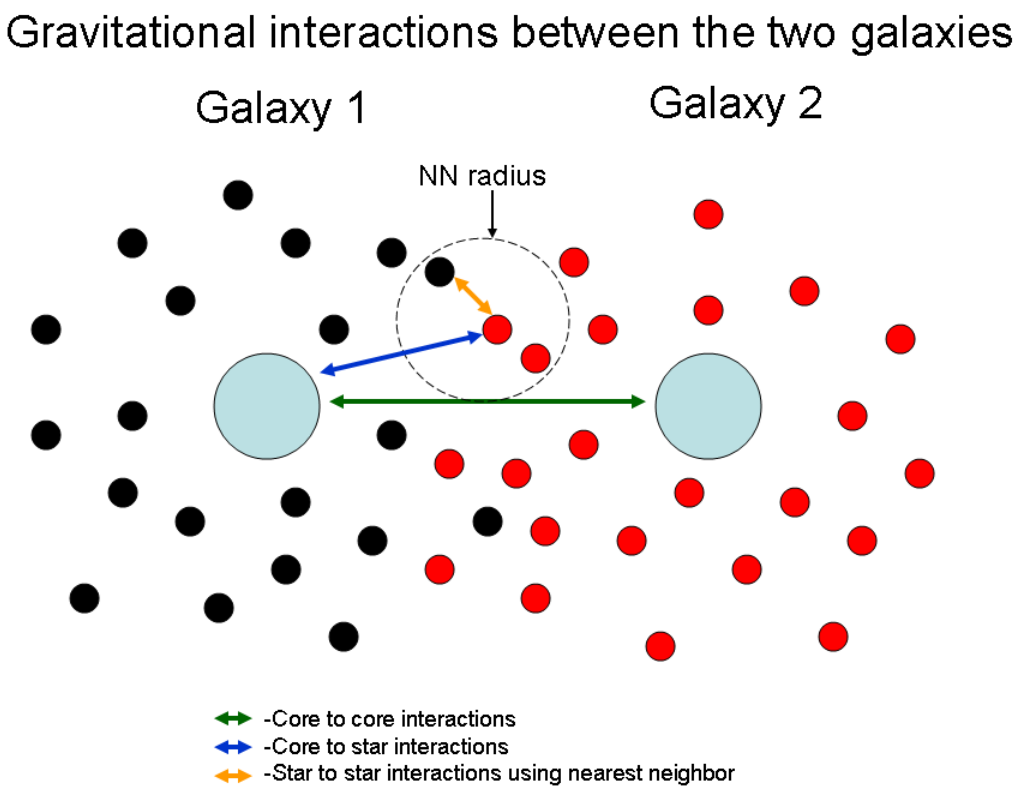


Figure 6: Interacting forces used in my model between colliding galaxies

Results

This project has mainly focused on the predicted future collision between the Andromeda and Milky Way galaxy. Three main simulations were ran: (1) Normal mass simulation (realistic mass values experimentally measured), (2) Milky Way has ¼ dark matter mass of Andromeda, (3) Normal mass both galaxies rotate in the same direction. Since the Milky Way is really far apart from Andromeda, another program placed a point (which represented the whole galaxy's mass) and ran a two point simulation until the two galaxies got within a certain distance (100 kpc). When the two galaxies reached that distance, the program will use those conditions and use them in the main program as initial conditions, place the galaxies, and start running the simulation. This method allows an easy assumption of the galaxies while they are far apart, which saves a lot of calculation time. Figure 7 (below) shows how the initial conditions were determined.

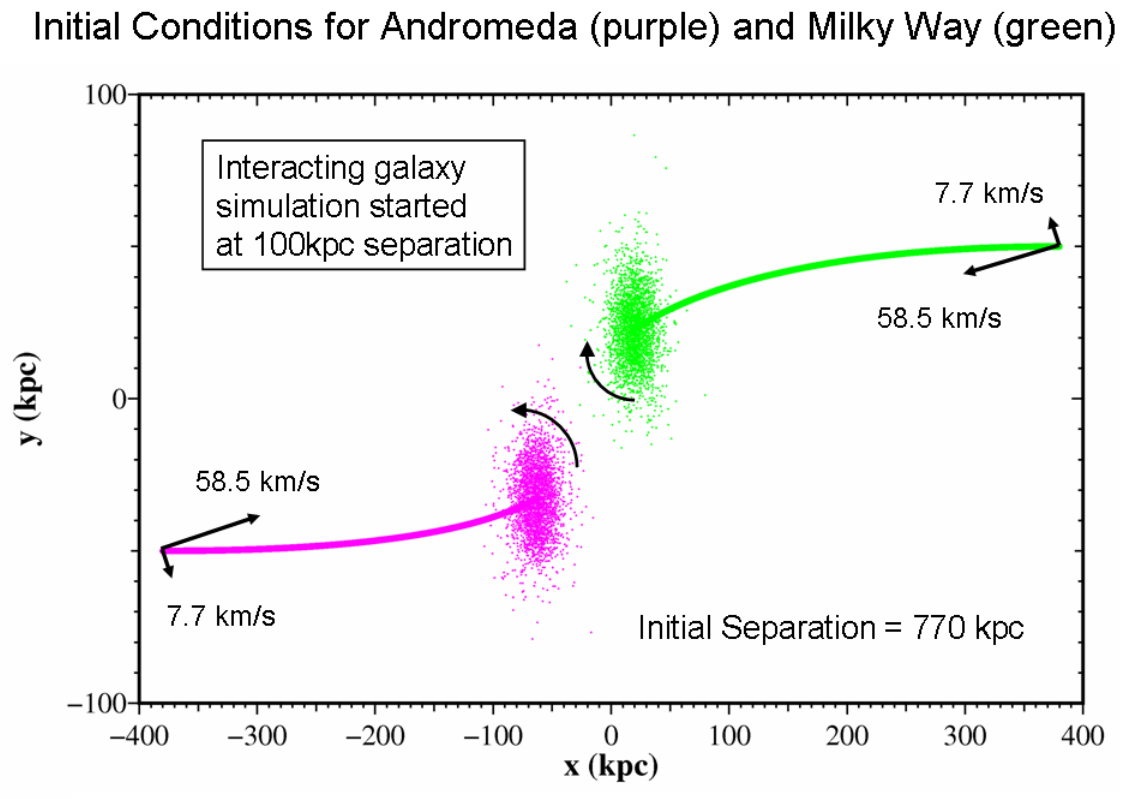


Figure 7: Shows initial separation between two galaxies and how starting conditions were obtained (MW – green, Andromeda – Purple)

All simulations that were run used a total of 4000 particles (2000 in each galaxy), a 200 year time step (dt), velocity Verlet method, and ran to about 12 billion years. Each simulation took about 2 days on a 3.0Ghz Intel Core 2 Duo. Figure 8-12 show results from the normal mass simulation. In the following figures, green represents the Milky Way and purple represents Andromeda.

Simulation 1

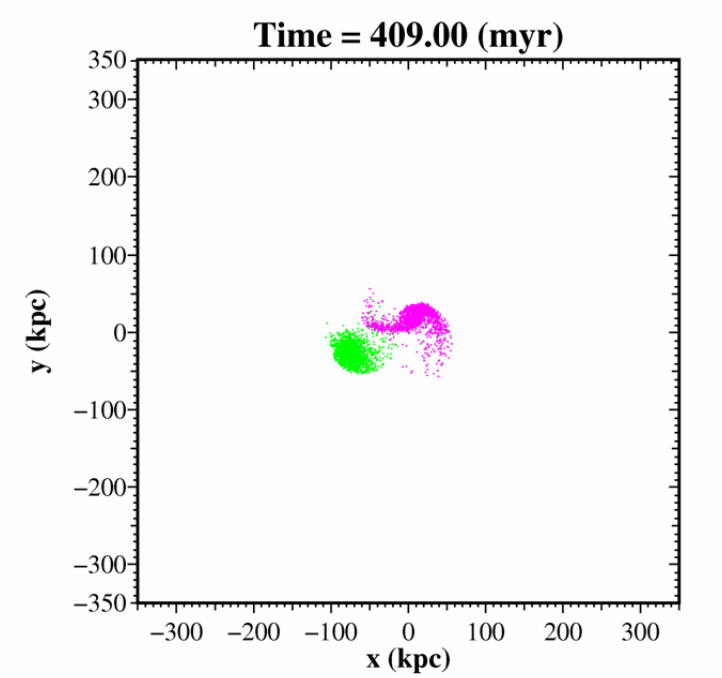


Figure 8: After initial collision, Andromeda has tails on the edges

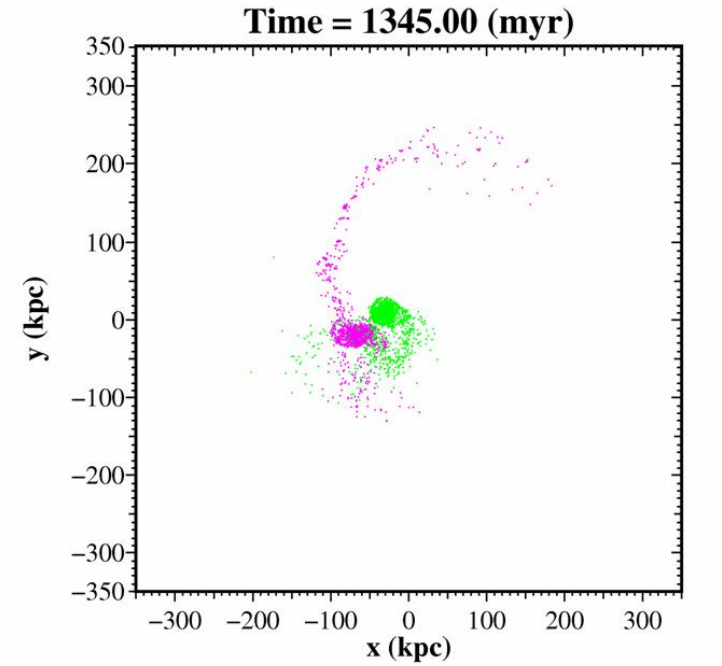


Figure 9: After several collisions, Andromeda has a large tail that extends out into space.

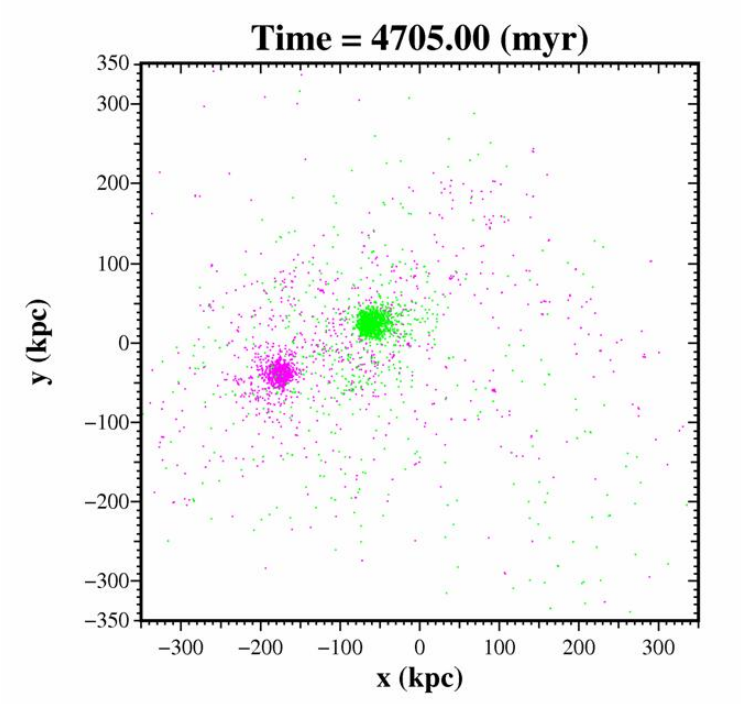


Figure 10: After several collisions, both galaxies slowly joining and slowing down.

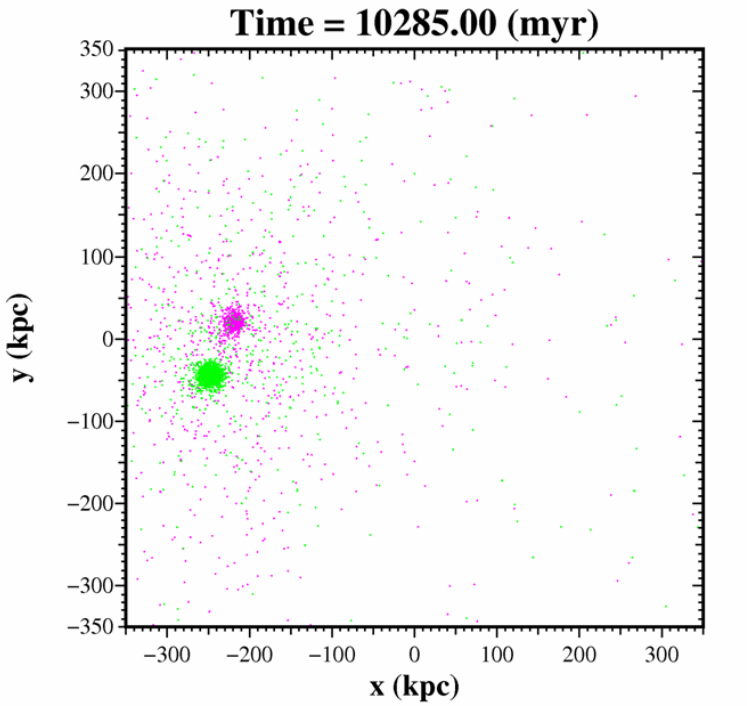


Figure 11: Near end of simulation, both galaxies have basically formed an elliptical galaxy.

Figure 8 shows the galaxies right after the initial collision, Andromeda starts to form two tails on either end of the galaxy. In figure 9, after several collisions, the tail extends out farther into space. This feature matches colliding galaxies viewed from observatories, this image below shows that this is a realistic feature.

Figures 10 and 11 shows the galaxies slowing down and accumulating stars around the two central forces which will lead to eventually causing the galaxies will merge and form an elliptical galaxy because they have about the same visible mass and dark matter mass. If one galaxy had less dark matter than the results will change. This was proven with simulation 2 where the Milky Way has ¼ of Andromeda's dark matter mass.

Simulation 2

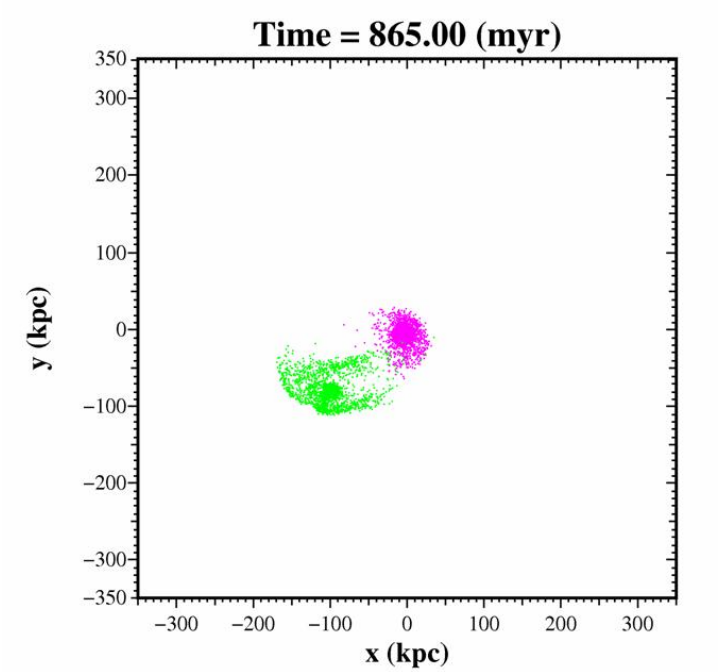


Figure 12: After initial collision, Milky Way (green) is getting thrown apart by Andromeda

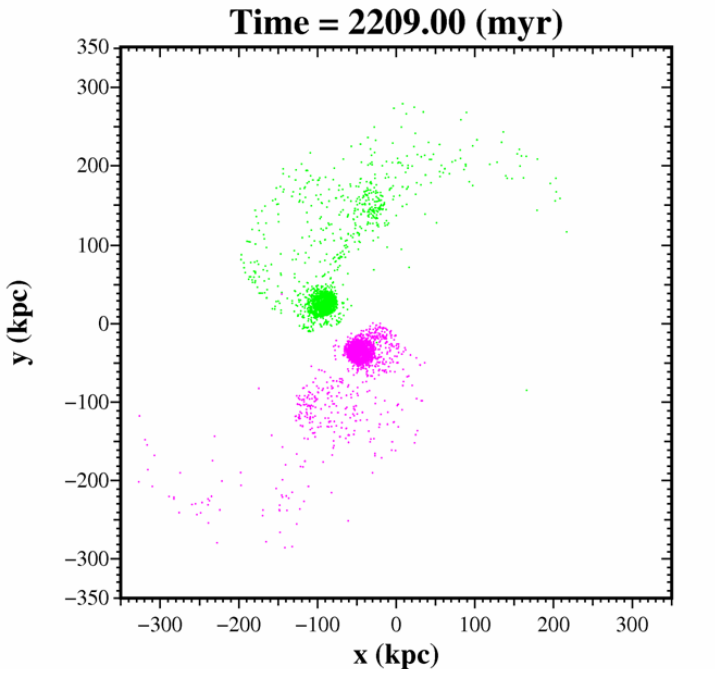


Figure 13: Several collisions have taken place, Milky Way has more matter slung out of the galaxy.

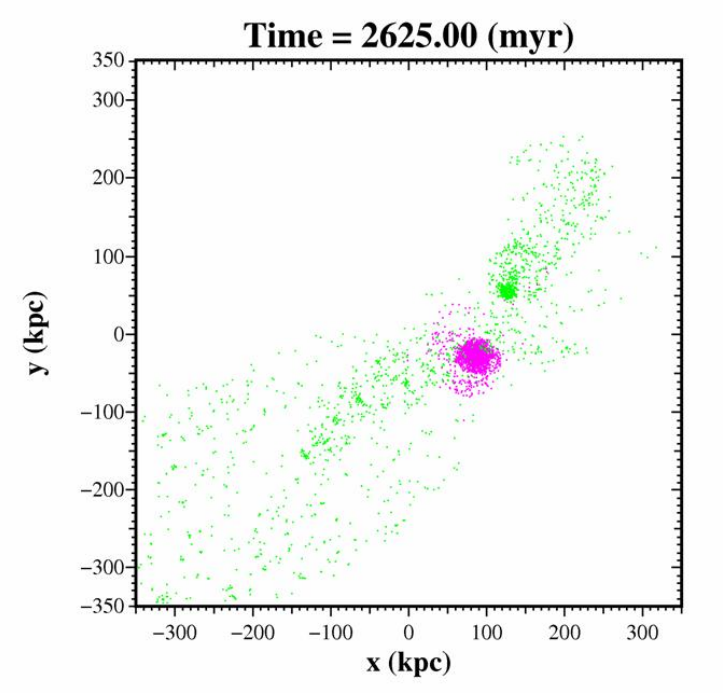


Figure 14: Milky Way has a smaller central mass, and has ‘clumps’ of stars further away. Andromeda stays compact and is not really affected.

In this simulation (figures 12-14), the Milky Way has ¼ the dark matter mass of Andromeda. Since the Milky Way has less dark matter mass, its stars are not as compact and do not remain intact throughout the collision. Early on in the simulation (figure 12) the Milky Way is already pretty much separated, and scattered. Andromeda remains stable and compact throughout the collision and is not impacted by the Milky Way at all. This simulation proves that in order to keep a galaxy stable throughout the collision, it has to have enough mass. Often if a collision has a large and small galaxy, the small galaxy will be absorbed or become a satellite galaxy. If two galaxy collide and they have about the same mass, they will both remain stable and eventually form a large elliptical galaxy. Both of the galaxies will orbit each other like a binary star. Below are the dark matter core trajectories for both galaxies. For simulation 1 (figure 15), the two galaxies orbit each other and become an elliptical like mentioned above, for simulation 2 (figure 16), Andromeda does not move a lot, it stays pretty much in a line whereas the Milky Way gets thrown around Andromeda and becomes a satellite galaxy to Andromeda. These two figures show how the galaxies move throughout the simulation.

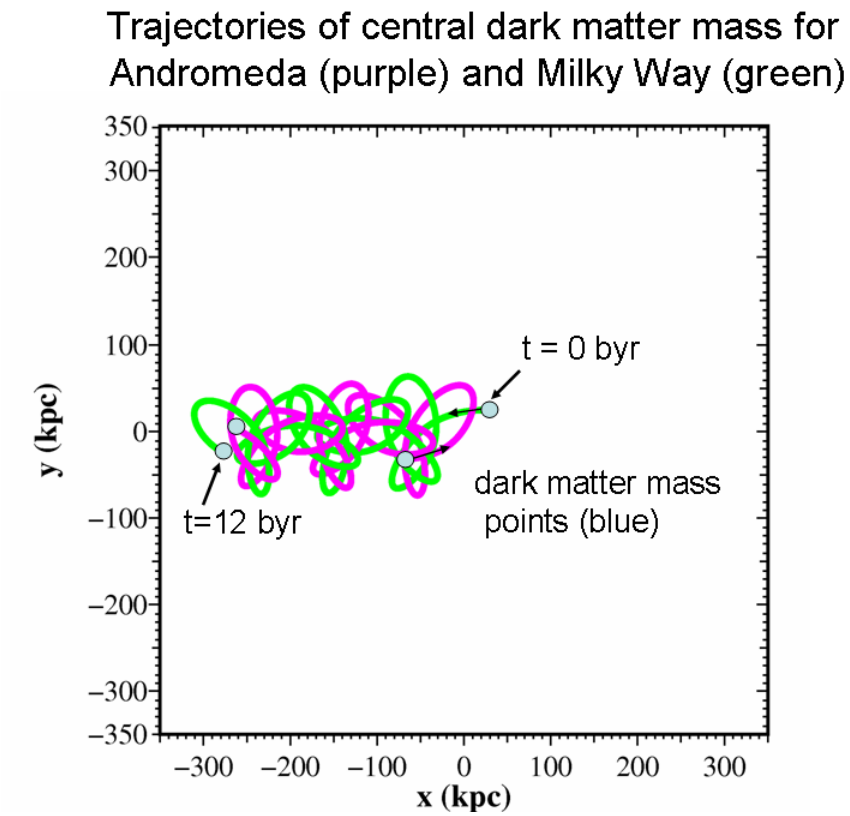


Figure 15: Shows simulation 1 trajectories, Andromeda and the Milky Way stay in a compact orbit and form an elliptical.

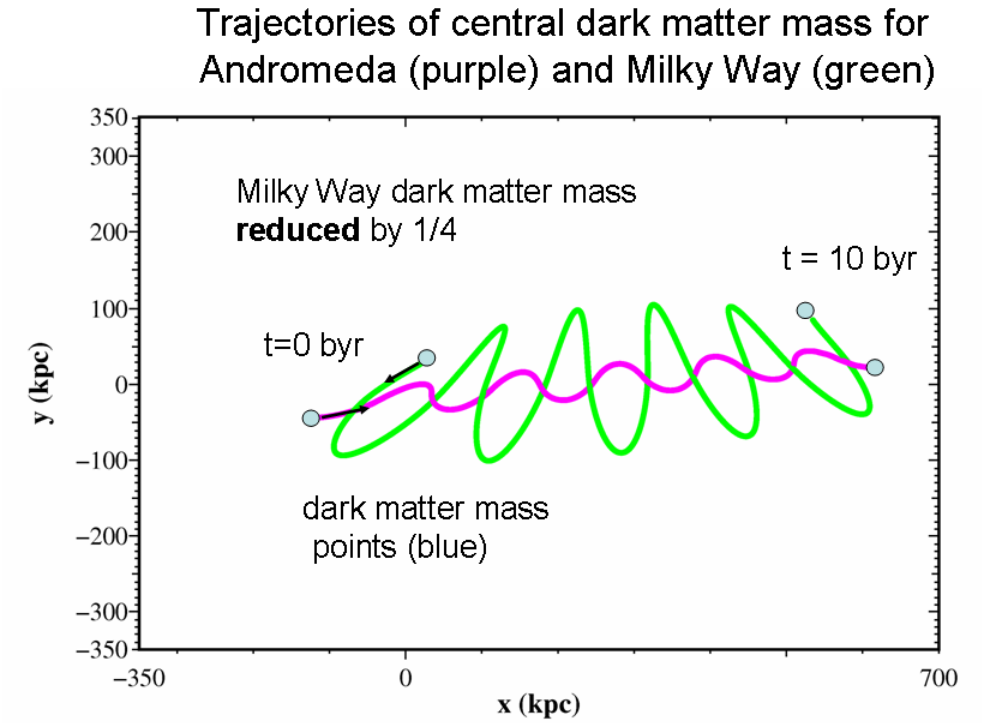


Figure 16: Shows trajectories for simulation 2, Andromeda is not really affected but the Milky Way gets slung around Andromeda.

Simulation 3

In simulation 3, the galaxies have the realistic mass values (same as simulation 1) and are rotating in the same direction. Having different or same rotation does not change the end result, however it does change how the galaxy collides. Meaning that whether or not they are rotating in the same direction they will eventually form an elliptical galaxy, they will just collide different. Figure 17 below shows the two galaxies rotating in the same direction (both are going counter-clockwise).

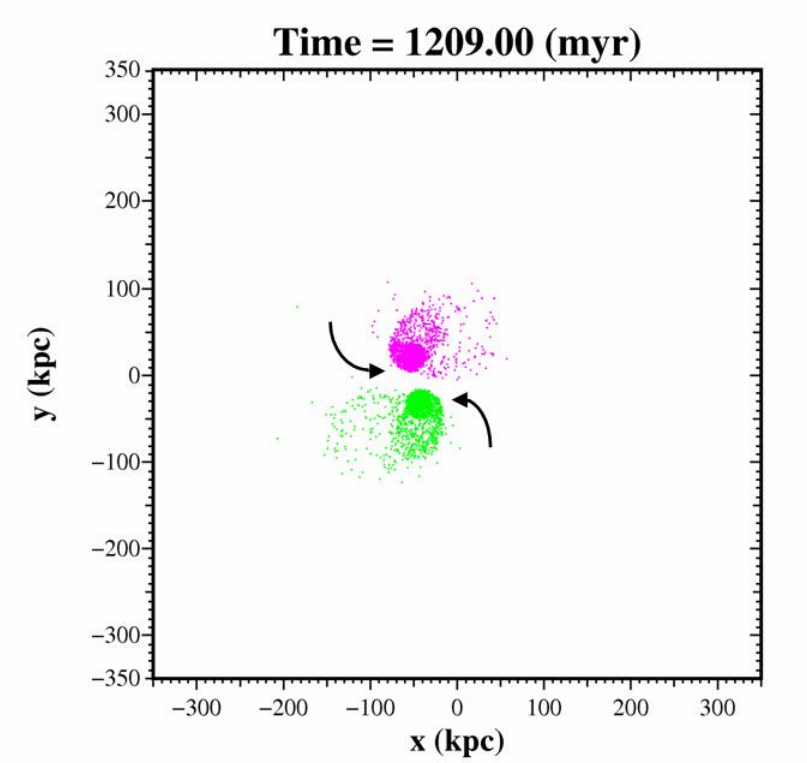


Figure 17: Both galaxies are rotating counter-clockwise and are even in mass.

MPI

MPI (Message-Passing-Interface) is a parallel programming language that is mostly used on supercomputers. It uses the machines multiple cores to work on one program, making it run much faster rather than a program that only utilizes one core. An MPI version of my galaxy model was written to parallelize my code and see how fast multiple cores would compare to a single core. After running the MPI version and comparing it to the normal C code, there was about a 50% speed up once the calculation reached about 16-32 thousand stars. Since this ran much faster, this would allow my model to have much more stars, which would also make it more accurate. Figure 18 (below) shows the amount of time it takes to do a single time step (with a 200 year dt) with different amounts of stars. The green line shows a single cpu, the red line shows two cpus using the MPI version of the code.

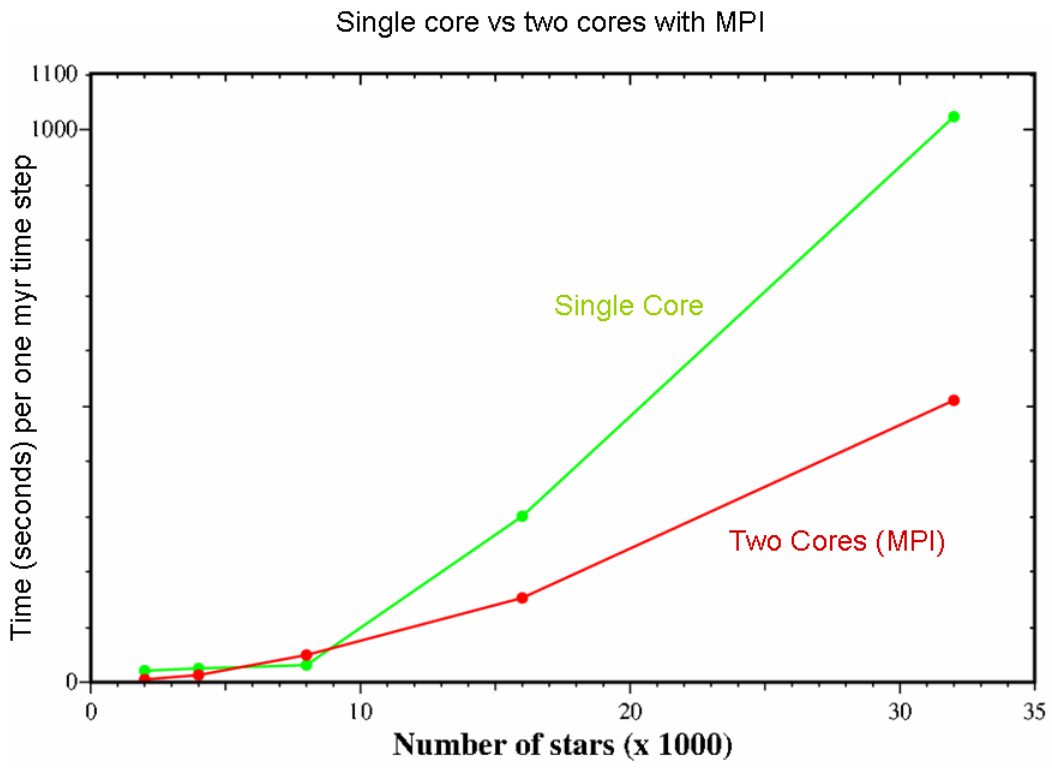


Figure 18: Shows the speed up between a MPI version and a normal code. MPI (red) takes about half the time it takes a single core from 16 – 32 thousand stars

Conclusion

I was able to successfully create a computer simulation of galaxy collisions using C. My 2D simulation results are consistent with many observed galaxy collisions and are also similar to 3D professional models. My simulation shows that dark matter has large effects on spiral galaxy collisions: (1) If the two galaxies dark matter halos are similar in mass, then both galaxies are distorted and eventually merge to form a larger elliptical galaxy, and (2) If one of the two galaxies dark matter halo is large, then the smaller galaxy is largely distorted, it orbits the larger one, and it is eventually absorbed. The larger galaxy remains mostly intact. The third simulation proves that if galaxies rotate in the same direction, in the end it will give the same result. In conclusion, this project can be used to help understand how dark matter effects galaxy collisions and what it really is.

Future Work

Future work includes comparing my nearest neighbor model to a full N-body calculation, replacing my static halo model with interacting/moving dark matter particles, going to three dimensions, and including gas in my galaxy models.

References

1) Navarro J. F., Frenk C.S, and White S.D. “A Universal Density Profile from Hierarchical Clustering.” The Astrophysical Journal **490** (1997): 493-508.
2) Begeman, K.G., “HI Rotation Curves of Spiral Galaxies. I. NGC 3198”, Astronomy and Astrophysics **223** (1989), 47-60.Print.
3) Zwicky, F., “Die Rotverschiebung von extragalaktischen Nebeln”, Helvetica Physica Acta **6** (1933): 110–127.
4) Milgrom, M., "A modification of the Newtonian dynamics as a possible alternative to the hidden mass hypothesis". Astrophysical Journal **270** (1983): 365–370.
5) Bennett, J.O., Donahue, M., Schneider, N., and Voit, M., “The Essential Cosmic Perspective”, Pearson; Addison Wesley, (3rd edition) 2005.Print.
6) Widrow, L.M. and Dubinski, J., “Equilibrium Disk-Bulge-Halo Models for the Milky Way and Andromeda Galaxies”, The Astrophysical Journal **631** (2005): 838-855.
7) Barns, J.E. and Hernquist, L, “Dynamics of Interacting Galaxies”, Annual Reviews of Astronomy and Astrophysics, **30** (1992): 705-742.
8) Withagen, J.C.J.G., “On the Collision between the Milky Way and the Andromeda Galaxy”, Masters Thesis, Universiteit van Amsterdam, 2008.
9) Dubinski, J., “The Great Milky Way - Andromeda Collision”, in Sky and Telescope, October 2006, pg. 31.

Optimizing Community Detection

New Mexico
Supercomputing Challenge
Final Report
April 3, 2012

Team 56
La Cueva High School

Team Members

Alexandra Porter
Stephanie Djidjev
Lauren Li

Teacher

Samuel Smith

Table of Contents

1.0 Executive Summary

Earth in itself is a large-scale network. It is a system connecting and overlapping the individual networks of daily life from communication systems to biological systems. In science, these networks are used to express connections between different nodes through arcs and paths. To understand and apply these structures, decomposition is of utmost importance. Decomposition allows for the breakdown of complex problems into smaller, more manageable ones, which adds an element of simplicity to the problem. With networks, this process results in the formation of communities which signifies the connectivity between distinct groups of nodes.

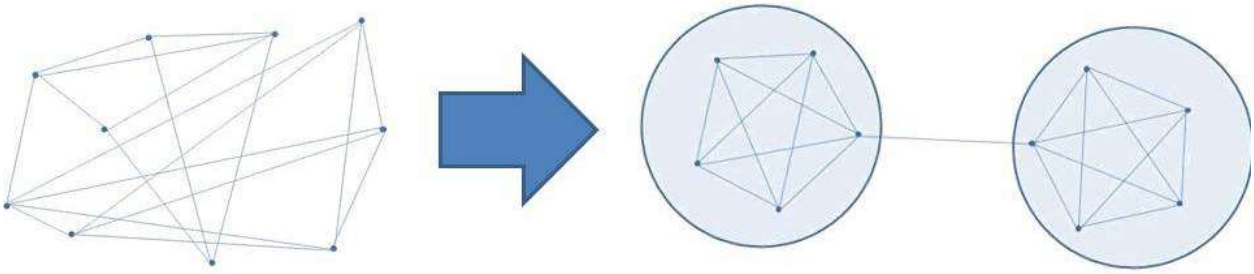
This project optimizes decomposition by finding the most efficient optimization method for partitioning of networks with maximum modularity. Java was used to implement and compare two main techniques: Girvan-Newman Algorithm and Ant Colony Optimization (ACO). Both algorithms were also tested with a variety of modifications. A Hierarchical Girvan-Newman Algorithm was implemented. Combinations between the ACO and Probability Summation as well as between the ACO and Power Iteration was used.

In discovering the most efficient optimization method, this project could be applied to most networks which naturally divide into communities or modules. Examples of such systems include transportation, manufacturing, communication, biological, and citation networks.

2.0 Problem Statement

What is the most efficient optimization method to maximize the modularity of a partitioned network?

The purpose of this project is to find an efficient and accurate method of maximizing the modularity of a partitioned network. Finding the maximum modularity of a network is extremely useful for any real-life situation the network may represent. Identifying groups makes it possible for someone to lay efficient connections or place the nodes in optimal locations.



3.0 Background Information

Networks are systems where sets of nodes are connected through links or edges. These systems appear in everyday life in the form of power grids, biological networks and pathways, metabolic networks, the Internet, social networks, and much more. Through the partitioning of networks in this project, systems are rearranged into communities allowing for further practical application of networks. For example, partitions within social networks can be used to represent social groupings based on common traits or interests. Citation network communities reveal related papers on certain topics, while the communities formed in the World Wide Web represent related sites. In economic networks, partitioning can reveal elements of vertical disintegration and groups of similar businesses. In a general sense, these partitions expose information which would be impossible to see in the big picture. With better understanding of networks through partitioning, correlations between network topology and function are revealed, and more efficient applications of networks can be discovered.

In the real world, the structure of networks is not random. Instead, various recurring patterns persist through networks. Sometimes patterns are obvious due to different contributing factors. In some cases, density of connections vary to form clusters or modules. Number of connections can also vary resulting in different costs of connection between nodes. Link capacity and structural patterns may also fluctuate. All of these factors involve the structural property of modularity, a significant characteristic of network topology.

Modularity is the qualitative measurement of the possible partitions within a network. A network with high modularity has dense connections between some nodes forming communities (modules) which are usually interconnected with other communities through sparse connections. Each module can be a dynamic community where there are more connections within modules

and fewer between. Examples of modularity exist in many fields. For example, in engineering, modules form physical components with one-to-one functions. Modules can also represent platforms and individual content in products. Within social networks, high modularity is exhibited in cliques, groups (like in social networking sites), and association mutuality. In nature, biology involves modules in cell clusters, classification, and molecular reactions. Ecological modules include niches, pathways, and elements of food chains. High modularity is beneficial in that it allows for resistance to outside attacks, and individual problems are reduced. Substitutions can also occur with modules leading to flexibility and variety in networks. Finally, modularity also allows for parallelization. In this project, modularity is used as a measurement of the accuracy of a network partitioning. Modularity is the number of in-community edges minus the number of expected in-community edges in a random graph.

$$Q = \frac{1}{4m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) s_i s_j = \frac{1}{4m} \mathbf{s}^T \mathbf{B} \mathbf{s}$$

4.0 Procedural Overview

4.1 Network Setup

In order to measure the algorithms used in this project, a customizable network creation system was made. This system allows the user to specify details about a network including the number of nodes, the number of communities in the network, and the probability of connections within or between communities. The network is then randomly generated based on these factors. This method creates networks with predetermined communities so that the accuracy of the optimization methods can be measured as well as the efficiency.

Within the network, each node and connection has a weighted value. This is important in using the network to represent a real situation. For example, in transportation methods such as roads, airplanes, and ships, each have a unique cost. Furthermore, physical distance is an important factor in many applications.

4.2 Visualization

The program developed in the project (shown in Figure 1) displays the network as a number of circles, each representing a node, and lines connecting them. Positions on the screen are initially random. A force-based layout algorithm then moves the nodes based on their proximity to other nodes and their connections.

In the force-based algorithm, each node is treated as an electron and each connection is treated as a spring. Nodes then repel each other while connections pull nodes together. The force on each node is calculated using Hooke's law, which relates to the springs, or connections, and using Coulomb's law, which relates to the electron charges. The weight of each node and

connection are also factors in determining the forces in the network. A node with high weight, for instance, repels other nodes with greater force than one with a low weight.

Program users can customize the criteria used in the visualization. The values of the constants in Hooke's law and Coulomb's law can be adjusted in order to alter the layout. There are also two options for coloring the network's connections. Connection colors can be based on connection weights or the betweenness of the connection. Nodes are colored based on their weights. Finally, nodes, connections, and their names can be hidden for convenience in viewing the network. Users can generate random networks or create custom networks, as shown in Figure 2.

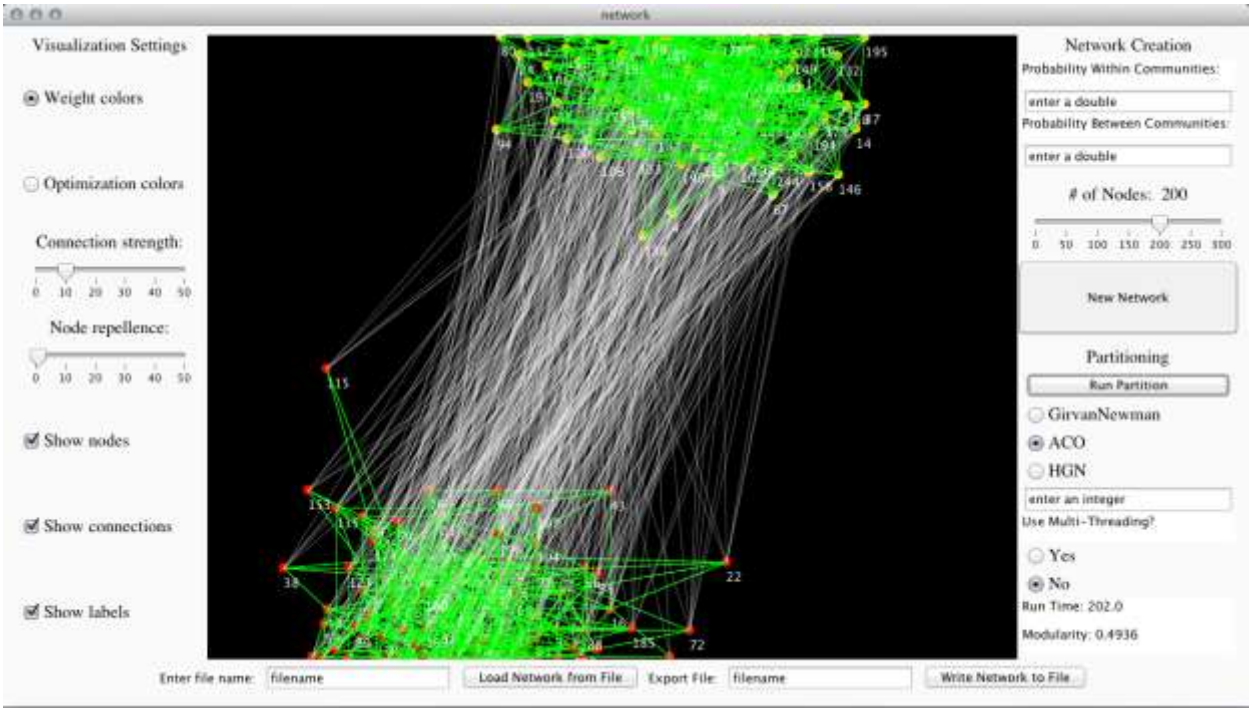


Figure 1

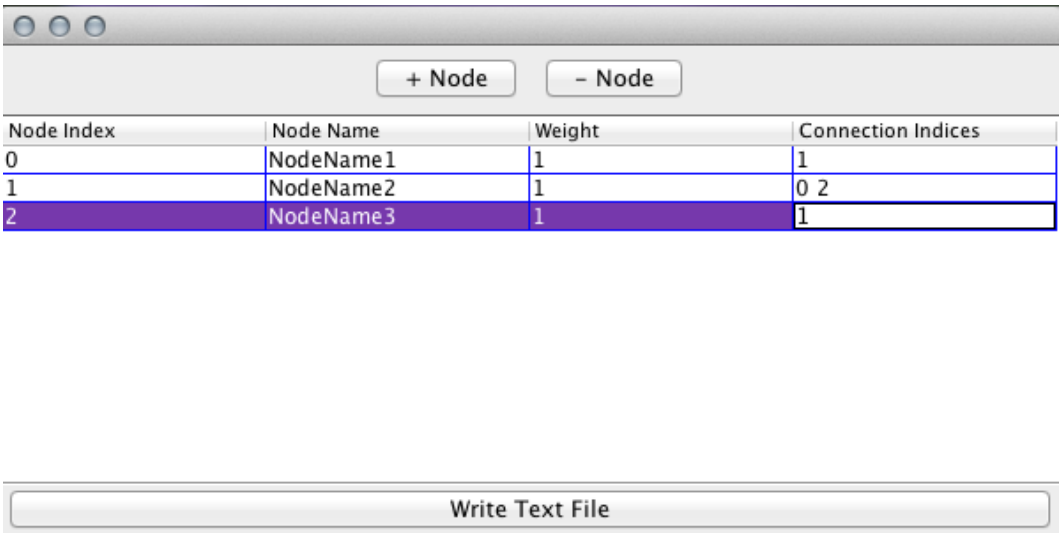


Figure 2

4.3 Algorithm Setup

All the algorithms used in this project have a similar feature; they calculate weights of the edges in the network in order to calculate edge betweenness. Edges with a high betweenness have a high probability to lie within the shortest path that goes from one randomly selected node to another. All edges are equally weighted at the initiation of each algorithm implementation. As the algorithm runs, weights are deposited on the edges. Depending on the algorithm, an edge with either a large or small weight will have high betweenness. Edges with low betweenness are edges that are within communities, while edges with high betweenness are edges that link different communities.

The Floyd-Warshall Algorithm calculates shortest path by first assuming that a connection cost is infinite. It then iterates through possible intermediate connections in order to find the path with the lowest cost. When all connection weights are equal to one, the shortest path is the fewest number of connections. This algorithm was implemented in some of the other algorithms in order to detect communities.

The first algorithm used is the Girvan-Newman Algorithm. It first calculates the shortest path between all possible pair of nodes, using the Floyd-Warshall Algorithm. The number of times a connection is used as part of a shortest path is summed and the connections most used have the highest probability of existing between communities.

The second algorithm used is the Hierarchical Girvan-Newman Algorithm. Using the Floyd-Warshall Algorithm, it calculates the shortest path between all possible pairs of nodes. Then, it creates a hierarchy outlining all the edges each pair of nodes needs to go through in the shortest path. The weight of each edge is compounded into all of the components that comprise its shortest path. The edges that have the highest weight have the highest probability of existing between communities.

The final algorithm used is a novel combination between an Ant Colony Optimization (ACO) and Power Iteration, called the "Power Iteration Ant Colony" algorithm. ACOs are algorithms based on ant foraging behavior. Ants leave "pheromone" behind each of the nodes they travel on. The network is initialized with random values of pheromone on every node. Ants are placed randomly on the network and move from one node to another via the edges. On each node, it calculates an updated pheromone of that node based on the nodes that are directly connected to it using the Power iteration algorithm that given a network N, the algorithm will produce a number b_k and a pheromone level p, such that $Np = b_k p$. In the case of two communities, this algorithm results in either negative or positive pheromone levels. Nodes with negative pheromone values are in one community, while nodes with positive pheromone values are in another community. The Power Iteration Ant Colony algorithm can be recursively implemented to find community structures of more than only two modules, by treating each newly formed community of nodes as a new network, and dividing that into even more

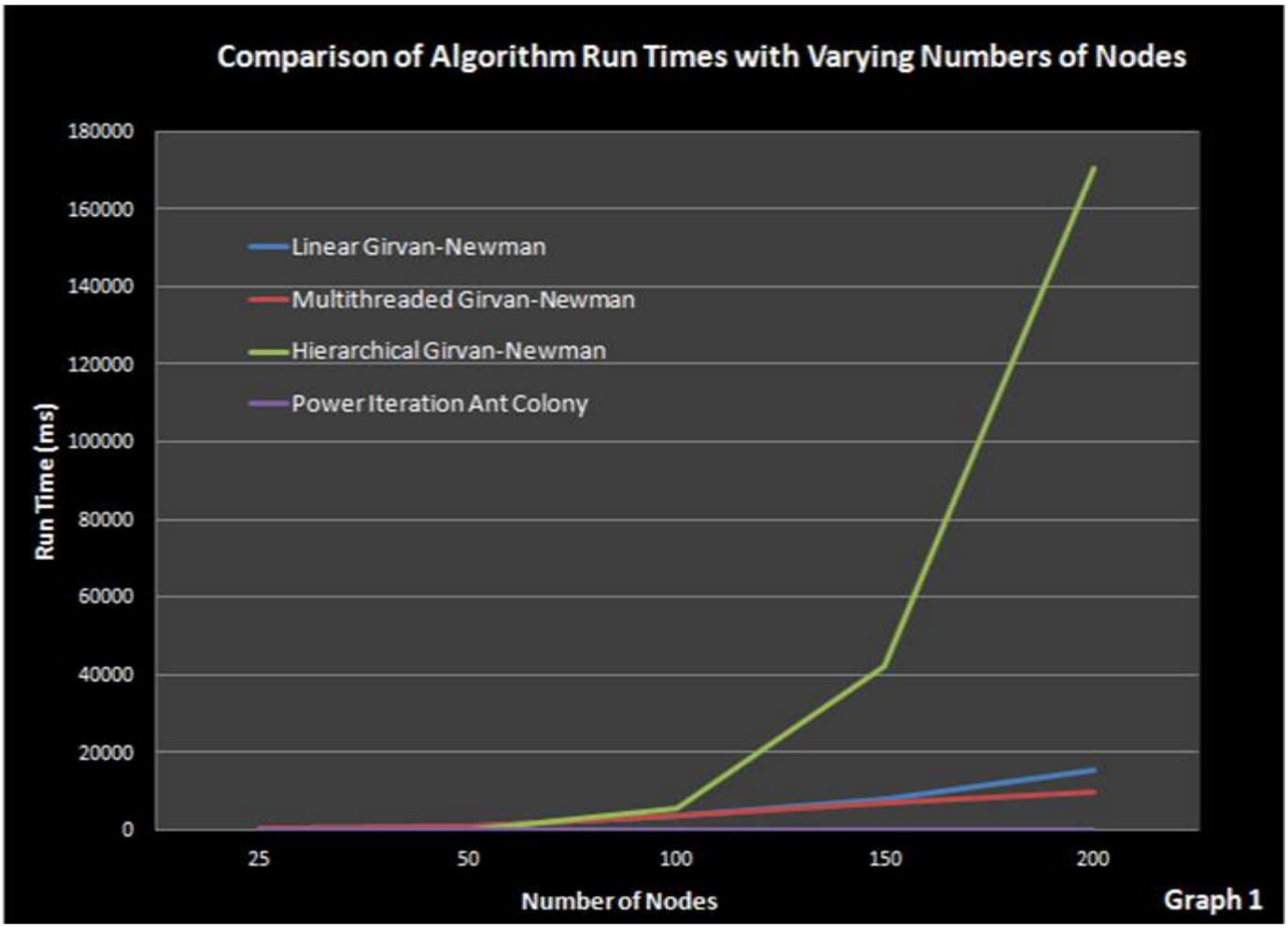
communities. The algorithm stops partitioning further when the new communities internally have a modularity of zero or less.

5.0 Results

5.1 Efficiency

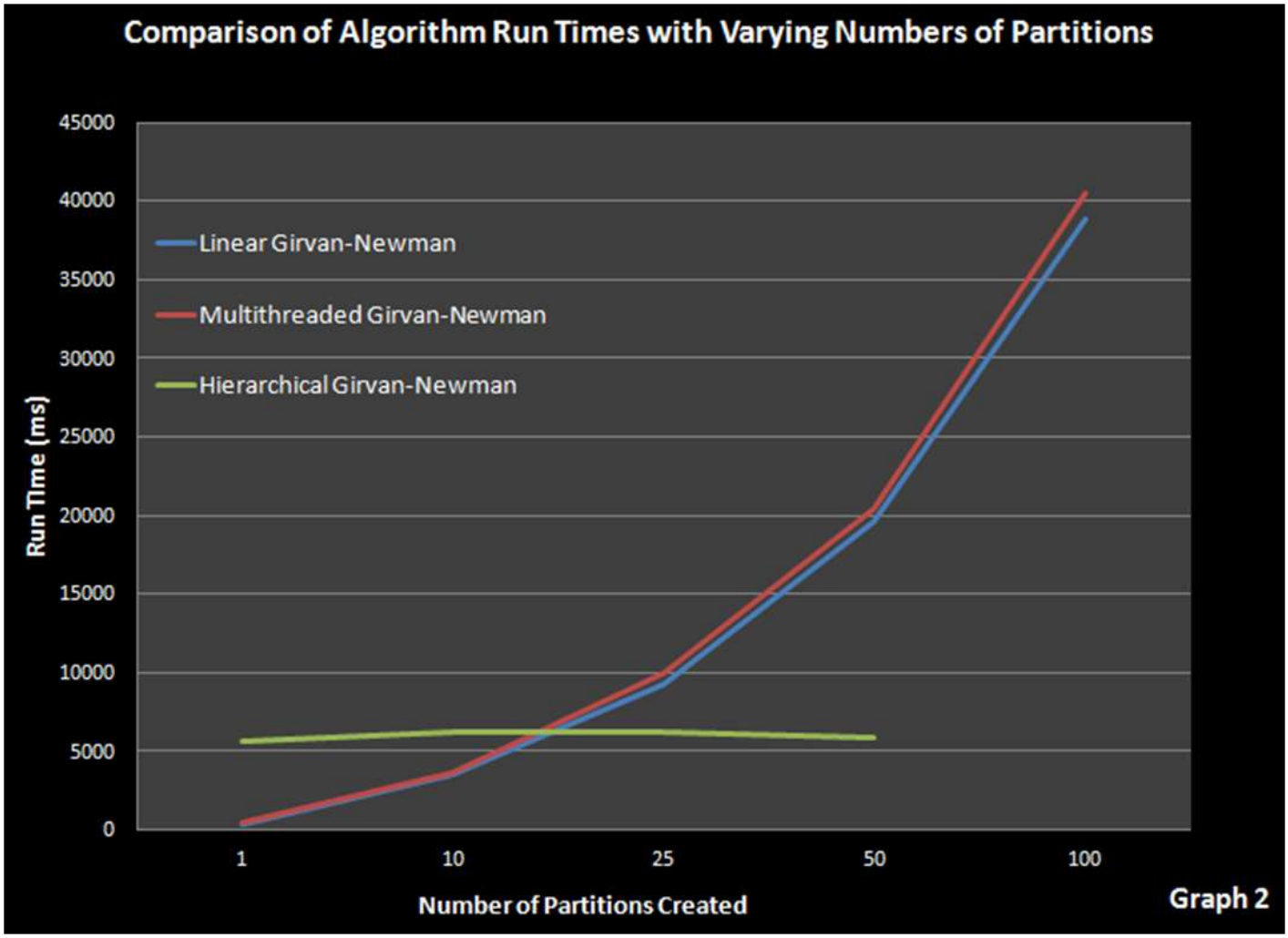
Run Times Varying Number of Nodes

Run times were recorded for partitioning a random network 10 times for increasing numbers of nodes. The probabilities of connection existence were 0.5 and 0.05 for all networks. All run times increase as the number of nodes increases. Due to the nature of the multithreading of the Girvan Newman Algorithm, the multithreaded version is more efficient after the number of nodes has passed a certain threshold, shown to be approximately 125 nodes for 10 partitions. The Hierarchical Girvan Newman is significantly slower and run times increase by a larger factor than for the Girvan Newman as the number of nodes increases. The Power Iteration Ant Colony Optimization was significantly faster than either of the other algorithms and run times did not visibly increase as the number of nodes increased.


Comparison of Algorithm Run Times with Varying Numbers of Nodes
180000
160000
140000
120000
100000
80000
60000
40000
20000
0
Run Time (ms)
Linear Girvan-Newman
Multithreaded Girvan-Newman
Hierarchical Girvan-Newman
Power Iteration Ant Colony
25
50
100
150
200
Number of Nodes
Graph 1

Run Times Varying Number of Partitions

The run times were recorded for partitioning random networks of size 100 nodes with the same connection probabilities as in Graph 1. In this comparison the number of partitions was increased. Because 100 nodes is less than the threshold for multithreading to be more efficient than the linear Girvan Newman Algorithm, the multithreaded times are greater. This study shows that the run times diverge slightly, with the multithreaded run times becoming larger in relation to the linear times as the number of partitions increased. The Hierarchical Girvan Newman run times stayed approximately constant, even decreasing slightly. This is due to the fact the the bulk of the HGN run time occurs regardless of how many connections in the resulting hierarchy are actually removed. Based on these results, the HGN becomes more efficient for networks demanding a large number of partitions to maximize modularity.


Comparison of Algorithm Run Times with Varying Numbers of Partitions
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
Run Time (ms)
Linear Girvan-Newman
Multithreaded Girvan-Newman
Hierarchical Girvan-Newman
1
10
25
50
100
Number of Partitions Created
Graph 2

5.2 Accuracy

Girvan-Newman Algorithm

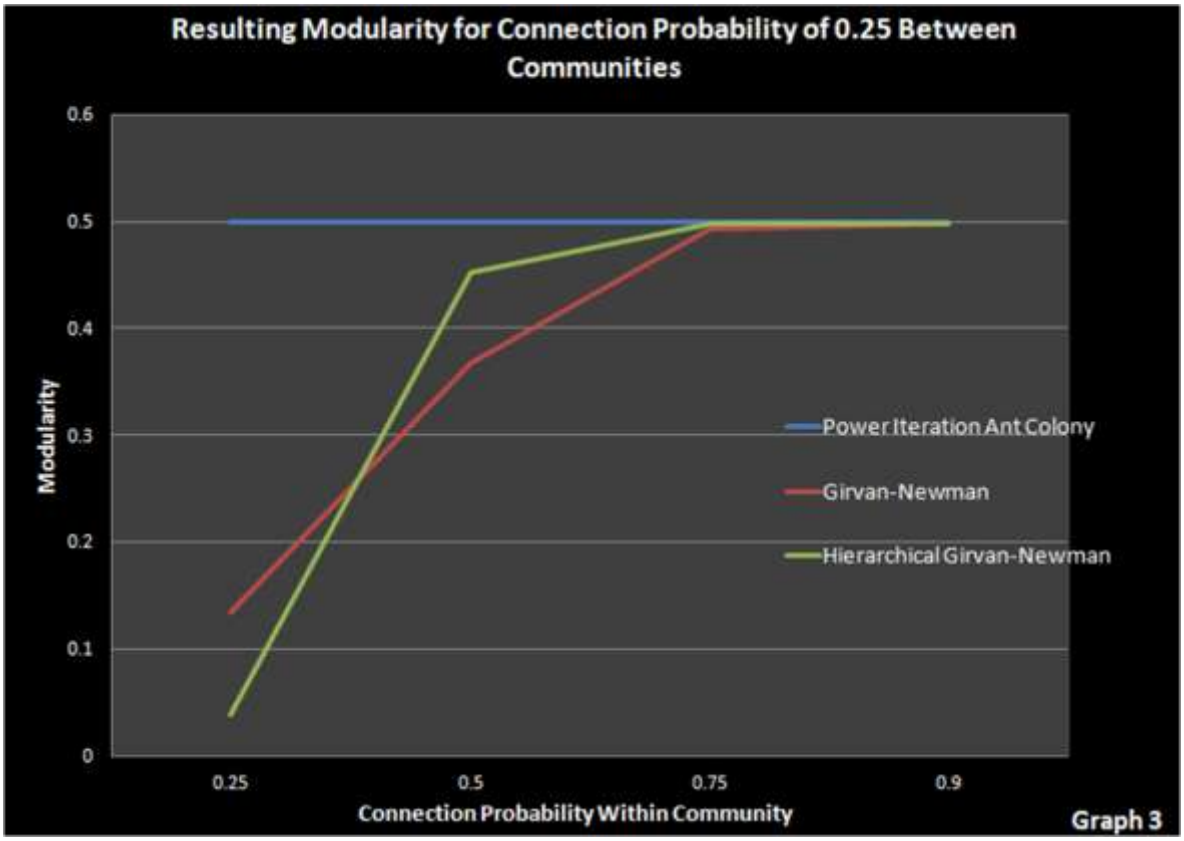
The Girvan-Newman Algorithm was able to reach maximum modularity for a network with 2 communities on almost all of the networks tested. As shown in Graphs 3 and 4, the modularity reached does drop off as communities become less distinct. The modularity reached is not as high because the algorithm removes connections within communities as well as between communities.

Hierarchical Girvan-Newman

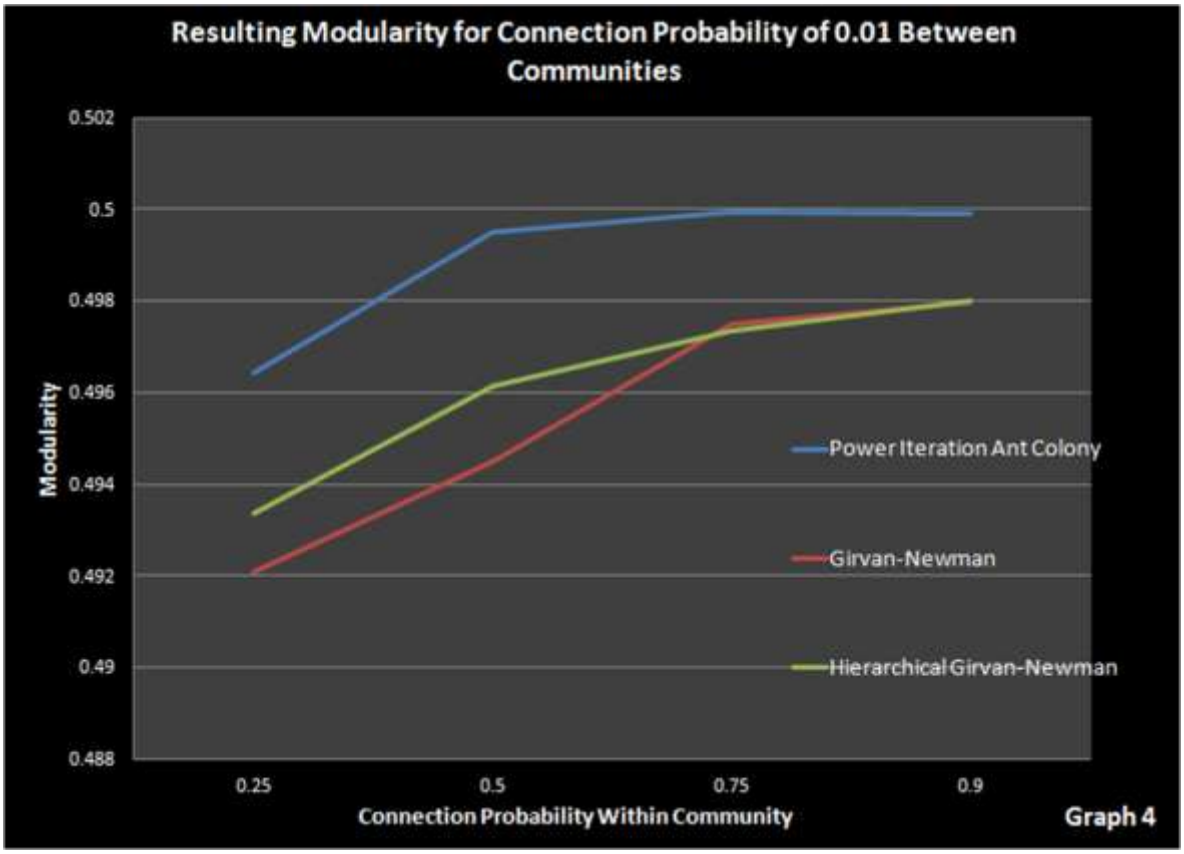
The Hierarchical Girvan-Newman reached modularities similarly to the Girvan-Newman Algorithm but showed higher accuracy in the more convoluted networks. Between inside community probabilities of 0.25 and 0.75, especially, the HGN reached higher modularity than the Girvan-Newman.

Power Iteration Ant Colony Optimization

The Power Iteration ACO was extremely accurate and consistent. It also showed (on Graph 3) the unique capability of finding a partition with high modularity on a network that has uniform likelihood (0.25 probability) of a connection and therefore no predetermined communities. In Graph 3, the Power Iteration ACO reaches almost exactly a modularity of 0.5 (the modularity of an ideal network of 2 densely connected communities) for all probabilities tested.



Graph 3



Graph 4

Real-World Applications

The program was also tested on two real-world networks. The first is a social network, created with randomly selected "friends," where connections represent friendships. The network was partitioned using the Power Iteration ACO. The communities identified were found to be closely related to factors including grade level and participation in band and orchestra. Node 'M', for instance is the only freshman included in the network. 'A', 'B', 'C', 'H', and 'O' are all seniors who are members of the band.

The second network used for testing was a food web. The Power Iteration ACO was used initially, but too many connections were removed and the result was not useful. The Hierarchical Girvan-Newman, however, effectively identified two communities. The first, represented by red nodes, includes all of the birds and mammals. The orange community is comprised of snakes, spiders, and all of the insects included in the web.

Social Network Partitioned using HGN

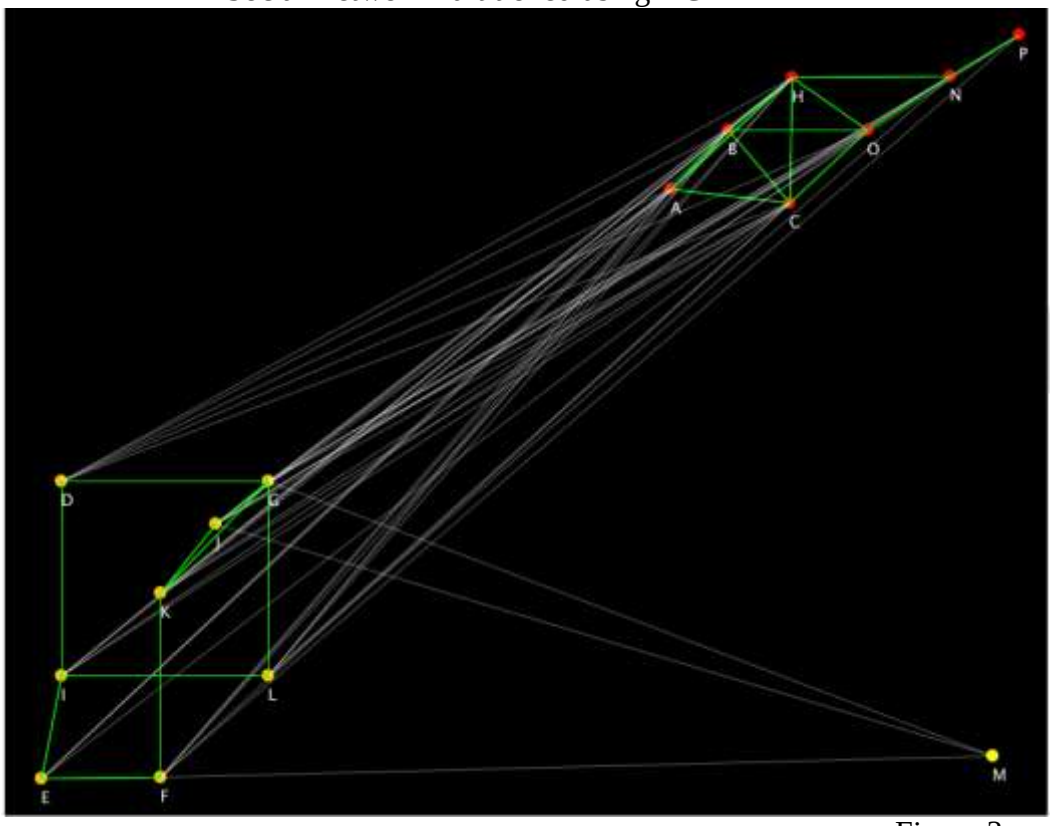


Figure 3

Food Web Partitioned with Power Iteration ACO

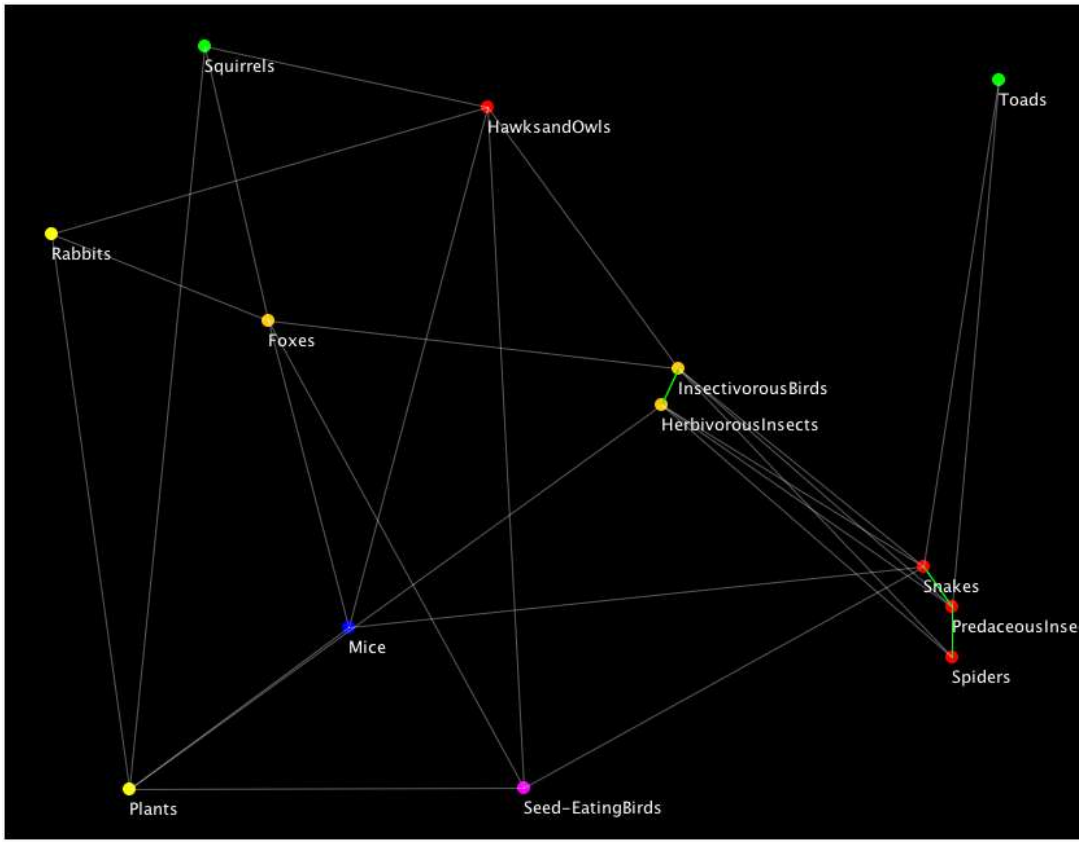


Figure 4

Food Web Partitioned with Hierarchical Girvan-Newman

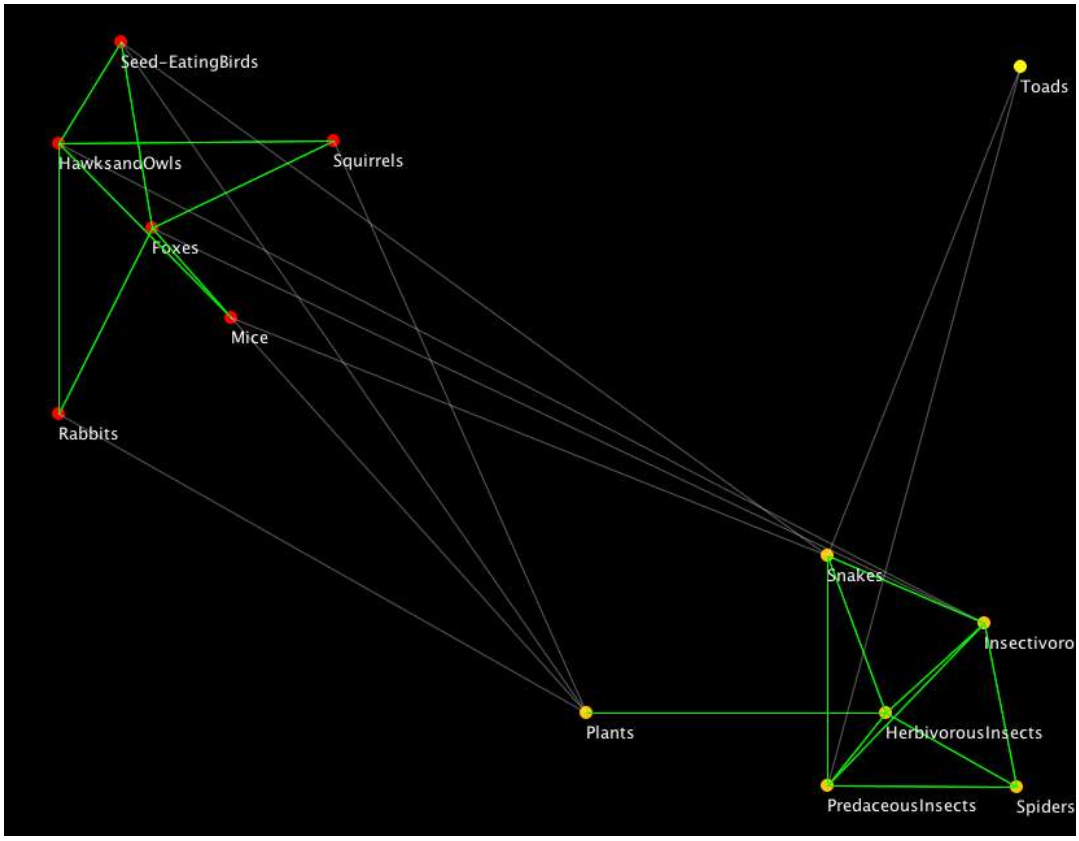


Figure 5

6.0 Conclusions

The multithreaded Girvan-Newman Algorithm was the fastest Girvan-Newman, as expected. However, the Hierarchical Girvan-Newman was slightly more accurate. The Power Iteration Ant Colony Optimization was the most accurate when applied to large, convoluted networks. It was also the fastest algorithm, so it is the best algorithm for partitioning larger networks. On smaller networks of approximately 15 nodes the HGN is actually the best algorithm because it reaches the highest modularity and in some instances the Power Iteration ACO partitions too many connections, therefore failing to effectively identify communities.

7.0 Significant Original Achievement

The main accomplishment of this project has been the development of the novel Power Iteration Ant Colony Optimization, a unique combination of two algorithms. The implementation of a Hierarchical Girvan-Newman algorithm is also original. In addition, a comprehensive program has been developed to allow users to input network data of any kind and identify a solution, using these algorithms.

8.0 Future Work

Next we plan to extend the Power Iteration Ant Colony algorithm to effectively partition more than two communities, which the Girvan-Newman and Hierarchical Girvan-Newman already do. To make this extension, the Power Iteration ACO will use a recursive method to partition each community into sub-communities. This also allows for parallelization, which we plan to add. In addition, we plan to add an algorithm to identify the preferable algorithm (HGN or Power Iteration ACO) for each individual situation.

9.0 Appendix

9.1 Acknowledgements

We would like to thank the organizers of the New Mexico Supercomputing Challenge for all of their assistance for this project. We would also like to thank David Rogers, Miguel Leyba, Janeen Anderson, Klaus Heinemann, and Bilal Shebaro for their helpful evaluations.

9.2 Works Cited

Capper, John, and Henrik Nilsson. "Static Balance Checking for First-Class Modular Systems of Equations." University of Nottingham, United Kingdom, 19 May 2010. Web. 1 Mar. 2012. <http://www.cs.ou.edu/tfp2010/files/09slides.pdf>.

C.L. Magee, *ESD 342 Class 14 Decomposition*, Spring 2010. (Massachusetts Institute of Technology: MIT OpenCouseWare), http://ocw.mit.edu (Accessed March 1, 2012). License: Creative Commons BY-NC-SA

Daniel E. Whitney, ESD 342 Network Representations of Complex Engineering Systems, Spring 2010. (Massachusetts Institute of Technology: MIT OpenCouseWare), http://ocw.mit.edu (Accessed March 1, 2012). License: Creative Commons BY-NC-SA

Girvan, M., and M. E. J. Newman. "Community Structure in Social and Biological Networks." *PNAS* 99.12 (2002): 7821-826. Web. 1 Mar. 2012. <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC122977/>.

The Impact of Forest Fires on Water Resources

New Mexico

Supercomputing Challenge

Final Report

April 4, 2012

Team 2

Desert Academy, Academy for Technology and the Classics, Santa Fe High School

Sara Hartse

Hugo Rivera

Nico Cruz

Teacher: Jocelyne Comstock

Table of Contents

1.0 Executive Summary

We intend to investigate the impacts that forest fires can have on the water resources of forest environments that human populations are dependent on. Apart from the general environmental disruption caused by forest fires, human water sources can be contaminated or otherwise compromised. For example, forest fires dramatically change the land cover of an area, lending to concerns about flash flooding, erosion and water quality. Based on factors such as tree density, moisture, land slope and weather, different forest types are more susceptible to damaging forest fires than others. The model is in Netlogo.

The model was model was created with a cellular automata, empirical method and incorporates several factors such as wind, elevation, and fuel type to in an attempt to accurately portray how a forest fire spreads. The model makes basic assumptions about the likelihood of given patches igniting based on established, empirical observations (for example; a fire burns longer with more fuel, it is suppressed by moisture, it is biased by air currents factors like wind and elevation). The model also incorporates GIS elevation data of various landscapes, including the region where the Pacheco Canyon fire burned in the summer of 2011. A final aspect of the project examined the optimization of firefighting tactics within our model.

Throughout the course of this project we collected field data and data from our model. The fieldwork was done in the regions burned by the Pacheco fire: water samples were analyzed and the landscape was observed. Many experiments were conducted within the model, including tests on the variables of windspeed, elevation, moisture and fuel density. Data was collected about the area burned and the debris levels of bodies of water.

The model was tested against the behavior of the Pacheco fire, and will not precise it did follow similar patterns. The data from this model illustrated a strong impact of windspeeds and elevation profiles upon debris levels.

2.0 Background

Dry weather and drought convert green vegetation into dry, flammable fuel; strong winds spread fire quickly over land; warm temperatures encourage combustion. When fuel, oxygen, and heat are all present, all that is needed is a spark to ignite a blaze that could last for weeks and possibly destroy acres of woodlands.

On average, more than 100,000 wildfires, also called wildland fires or forest fires, clear 4 million to 5 million acres of land in the U.S. every year. In recent years, wildfires have burned up to 9 million acres of land per year. A wildfire moves at speeds of up to 14 miles an hour, consuming everything in its path.

2.1 Impact of a Fire on an ecosystem

A forest fire is a dynamic, predictable, yet still uncertain, natural process. A forest fire can have numerous positive and negative effects on an ecosystem. It is important to balance these effects properly in order to ensure a sustainable ecosystem. Forest fires have the potential to have a tremendous impact on the water resources of an ecosystem. The impacts are various interrelated mechanisms including effects on soil chemistry, soil physical properties, soil biological response, the hydrological cycle and water quality. As illustrated in figure 1 (below), all of these systems are highly complex, tying together the physical, biological and chemical aspects of an ecosystem as well as the properties of its upper and lower ground layers.

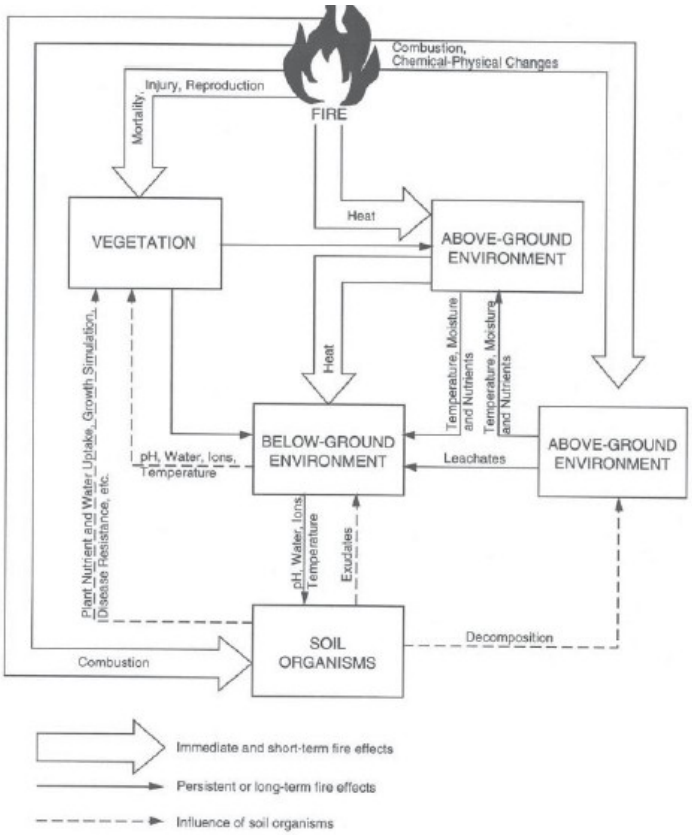


figure 1: Immediate and long-term responses to forest fires

2.2 Describing and Quantifying Fires

To begin quantifying and modeling these impacts, it is necessary to start defining different types of fire and their potential impacts on an ecosystem. A broad way of defining the role of a fire within an environment is through defining a particular *fire regime*. This is useful for examining the role of fires within certain vegetation types, especially the likelihood of land types to have certain fire types. The fire regime types are as follows:

- *Understory fire regime*: A regime type that is most likely to have a fire type that is considered non-lethal and in which about 80% of the vegetation survives the fire and the vegetation type remains fairly unchanged. It applies to vegetation types which include many fire resistant wood types.
- *Stand replacement fire regime*: This regime type has the potential to have a fire which is lethal to the majority of the vegetation where about 80% dies off as a result of the fire. It applies to fire susceptible forests, woodlands, shrublands and grasslands.
- *Mixed fire regime*: A regime which supports fires whose severity varies between lethal stand replacement and non-lethal understory. This is the most common type as temporal and spatial variation in parts of an ecosystem often results in a wide disparity of potential impact.
- *Non-fire regime*: A landscape type in which a fire is not at all likely to occur.

Another important aspect to be addressed when trying to quantify the impact of a fire is a measure of the magnitude of the fire, specifically the intensity or severity of it. Intensity and severity are defined as two different aspects of fire measurement with different values for assessing impacts of fires.

Intensity refers to a measurement of the rate of heat released from a fire. A quantifiable measure of intensity comes from an equation known as Byram's definition of fireline intensity. Byram's equation is as follows;

$$\mathbf{I=Hwr}$$

Where *I* is the intensity of the fireline (the front of the fire and measured in kW/m/s), *H* is the heat yield value of the fuel source (in kW/kg), *w* is the mass of the fuel consumed (in kg/m2) and *r* is the rate at which the fire is spreading (in m/s) (also known as the Fire Behavior).

The measurement of fire intensity is useful in predicting the extent and magnitude of a burn. For example, intensity has been shown to be directly proportional to flame length, and flame length is used for predicting possible damage a fire may inflict on buildings. The *depth of burn* of a fire is the measure of how deeply the fire burned or how deeply lethal levels of heat reached into the organic soil horizon (the layer of soil that includes the majority of plant and other biological matter). Depth of burn is the main factor of importance when determining the impact on soil and water resources. It relates to the amount of bare mineral soil exposed to erosion, the depth at which chemical changes may occur and the microbial populations which may be affected. It is important for examining how erodible land becomes and how hydrological recovery process is changed. Another types of fire measurement is *Severity,* which differs from intensity in that it is concerned with above and below ground heat pulses. It is the concept which relates intensity and depth of burn. Severity incorporates a two-dimensional quantification of fire magnitude, accounting for both above- and below-ground heat level.

2.3 Impact on Water Resources

Forest fires can impact water resources in several different ways. Fires can shower water sources with debris. This extra sediment can slow or damage water filtration processes and increase the resources needed to produce clean drinking water (Meixner and Wohlgemuth, 24). The debris can also change the chemical makeup of the water. For example, large amounts of ash and burned material have been shown to raise nutrient levels, especially phosphorus and nitrogen (Meixner and Wohlgemuth, 25).

It is also an observed phenomena that forest fires are often be responsible for increased watershed response, namely flooding and mudslides. These natural occurrences are threatening to human safety, and can disrupt water resources by, for example, wiping out a dam. This relationship between fire severity and erosion potential is demonstrated in Illustration 2, which shows the relationship between fire severity and the magnitude of

watershed response. Essentially the graph shows that as fires become more severe (that is to say, they become more intense and have a higher depth of burn) and time passes, the potential for a large scale watershed response becomes greater. Studies in the Santa Monica mountains have found that erosion rates in a burned watershed can be up to 50-100 % greater than a vegetated one. Forest fires can also have the impact of creating a larger amount of so-called 'hydrophobic soils', a soil type that becomes impermeable to water, preventing it from soaking in and causing the vital resource to simply slide off the surface (Ainsworth and Doss).

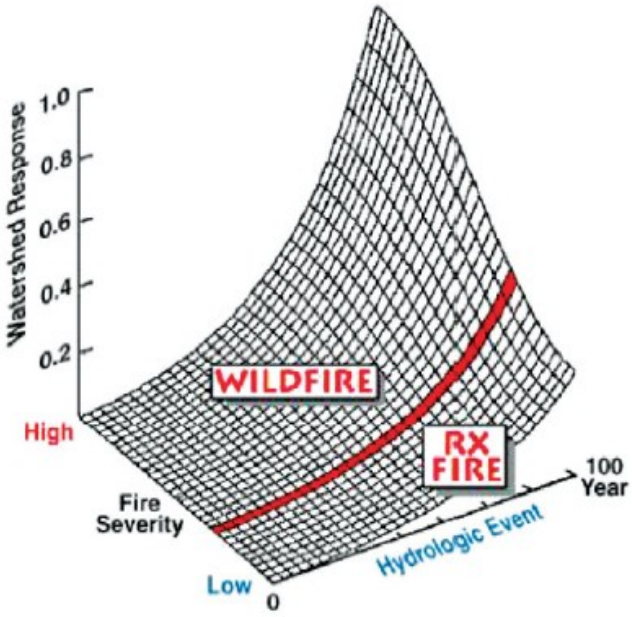


Illustration 1: An approximation of the magnitude and the timing of watershed response to fires of varying severity, including wildfire and the lower section corresponding to prescribed burns (Neary, Ryan and DeBano, 16).

2.4 Firefighting Techniques

When a forest fire is spotted, teams are immediately dispatched to fight the flames. The first thing they do, when available, is to establish a fire line. They do this by removing all vegetation (fuel) that could assist the fire. This is an attempt to allow the fire to die out slowly before it reaches any other fuel sources. After cutting off the fuel, firefighters must use tools such as shovels, picks, chainsaws, and hoses to prevent the fire from spreading further.

3.0 Fieldwork

By looking at real world data, we can incorporate more realistic parameters into our model for better results. We are examining one major fire that happened recently: the Pacheco Canyon Fire. We determined phosphate levels in the surrounding water sources.

Picture 1: A picture taken showing the Nambe River drainage at a region burned in the Pacheco Canyon Fire.

3.1 Pacheco Canyon Fire

On June 18th, 2011, a wildfire started in the Santa Fe National Forest, about 9 miles north of the city of Santa Fe. The fire burned an estimated 10,250 acres in about 10 days, primarily in the Nambe River drainage, as seen in *Map 1*. As part of the background research for the modeling and analysis of our subject, we decided to collect field data about this particular fire. The primary analysis shows how the fire affected the water quality in the Nambe River and other small rivers approximately four months later.

The water samples (see *Picture 2*) were obtained on October 9th and 10th 2011, 3.5 months after the fire. They were collected from four separate locations. Two samples were collected from two upper sections of the Nambe River at points upstream of the burned regions (Samples A and B), and two samples were collected from water sources downstream of the burned region, one from the main Nambe River (Sample D) and other from the Rio Capulin (Sample C).

There were immediately clear distinctions between the water samples collected upstream of the fire as opposed to those which would have flowed through burned regions. Judging from the appearance, the water from upstream was very clear and had little to no visible particulate matter. Samples C and D were very cloudy and dark with comparatively large quantities of particulate matter.

Chemical analysis was performed on the water samples in the form of a test for phosphate levels. Phosphate is often observed to appear in elevated levels in water after forest fires, and is attributed to ash and other debris being washed into the water due to erosion. Elevated phosphate levels can cause growth explosions in aquatic plants, which subsequently leads to reduced oxygen levels, making water uninhabitable for aquatic organisms.

3.2 Determination of Phosphates in Water Samples

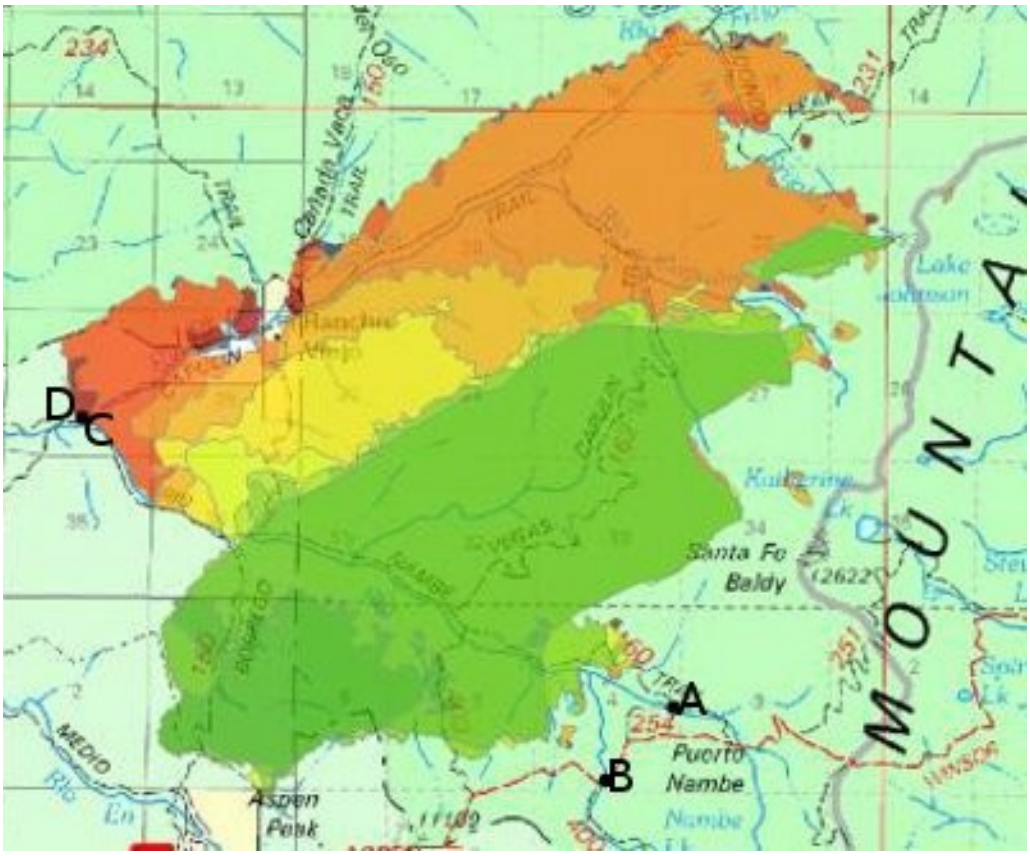
The basis of this experiment is to add a reagent to the water samples; it reacts with any phosphates present causing them to be visible as blue precipitates. A standard is created where this reagent is mixed with a water with a known quantity of phosphate. This standard makes it possible to estimate the concentrations in the samples being tested (Schumann).

The reagent is composed of the following:

- 50mL H_2SO_4 solution
- 5mL potassium antimonyl tartrate ($K(SbO)C_4H_4O_6 \cdot 0.5H_2O$) solution
- 15ml ammonium paramolybdate ($(NH_4)_6Mo_7O_{24} \cdot 4H_2O$) solution
- 30mL ascorbic acid solution

The standard solution is composed of distilled water and potassium phosphate dibasic (KH_2PO_4), they are adequately diluted to make solutions with phosphorous concentrations of

Map 1: The burned region of the Pacheco Canyon fire. The shading represents progression over time. Sample locations A, B, C, and D are also labeled.



10, 5, 2.5, 1, 0.5, 0.2, 0.1 and 0 ppm. 25 drops of each solution are placed in a well culture plate and then 4 drops of the reagent are added to each well. As expected, there was a clear trend of darker blue in the solutions of the highest concentrations, those with little phosphorus were almost clear and the solution with no phosphorus was completely clear. Next four samples of 25 drops each are taken from each water sample (A, B, C and D) and placed in a 16 well culture plate (taking four samples from each sample is designed to minimize random error) and four drops of the reagent is added to each well.

After the reagent was added, samples A and B remained quite clear and Samples D and C turned a faint blue. D and C were not as dark as the standards of 10, 5, 2.5 or 1 ppm, but appeared to be fall somewhere between 0.5 and 0.2 ppm. These results of this test indicate that the samples taken from rivers that flowed through burned areas acquired measurably higher levels of phosphorus than those that were not exposed to fire debris.

There are certain sources of error associated with this experiment, an important one being that this test was not performed until several weeks after the water samples were collected. It is possible that the prolonged storage in containers could have changed the water chemistry. However, all the samples were stored identically, so any change would have affected them all equally. Another source of error was that the comparison of the water colors was purely observational. This was somewhat accounted for by having two different people unfamiliar with the experiment examine the samples (their observation correlated with those expressed previously). However, this subjectivity could further be improved upon by taking pictures of the standard and samples and using software to more accurately compare the colors.

Picture 2The four water samples taken from various locations upstream and downstream of the Pacheco Canyon fire. Starting at upper left and moving clockwise, they came from the Rio Capulin (C), the upper Nambe River (A, from Puerto Nambe), the upper Nambe (B, from Nambe Lake), and the lower Nambe (D).

However, taking all the errors into account, it can still be reasonably asserted that there was certainly a difference in the phosphorus levels between the samples, with those that were taken downstream of the burned region having measurably higher amounts. This correlates with our predictions. These results are of further interest because these samples were not taken until 3.5 months after the fire and the fact that the effects of the fire are both visible (in the cloudiness of the water) and measurable (in the phosphorus levels) is indicative of the long-term environmental impacts of forest fires.

3.3 Observation of the Landscape

There was further observation performed at the locations the samples were taken from. *Picture 3* shows the location that Sample A was taken from. The water is very clear and there does not appear to be evidence of any serious erosion. However, the snow cover does make it difficult to determine the erosion levels. *Picture 4* was shot at the location Sample D was taken from. It is clear from the image that the water is much cloudier than that of the Upper Nambe. Another major difference is the apparent erosion: the banks of the river appear washed out and the the accumulation of broken branches and other debris indicate that flooding has occurred.

Picture 3: The upper Nambe River, the location where Sample A was taken.

Picture 4: The lower Nambe River, the location where Sample D was obtained.

4.0 Model

The model was constructed in Netlogo. Various elements of the code are described in the following.

4.1 Fire

The fire is based on a cellular automata/empirical modeling method. A cellular automata model is characterized by a series of local interactions between cells, based on a global set of transition rules, on a latticed, 2 dimensional grid (Bodroži and Stipani). This model makes use of Moore's neighborhood (the eight neighbors that touch a 'cell' or a 'patch) and simulates the spread of fire from one patch to another every time step. The model is more empirical or statistical than physical. This means that the rules governing the spread of fire are based on acknowledged probable fire behavior and a simple energy balance rather than a theoretical model which would be based on mathematically established physical principles. A physical model would have the advantage of being able to process a larger range of input variables and might be considered to be more accurate. An empirical model, for our purposes, has the advantage of flexibility and a simpler design process that is based on qualitative research.

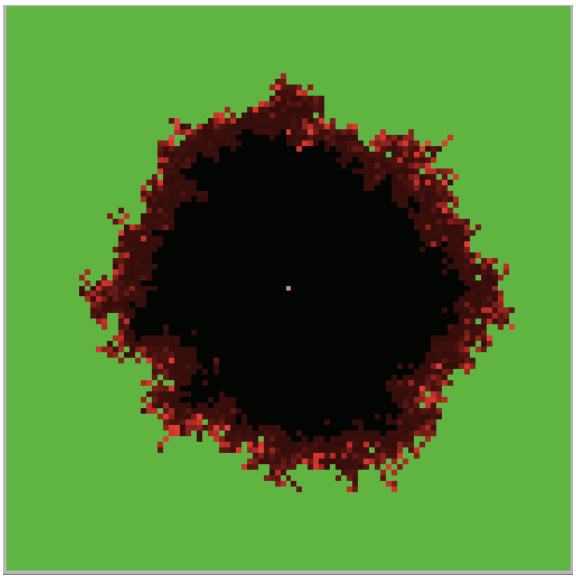
The fire spreading model is based on five primary variables; fuel, moisture, elevation, wind-speed and wind-direction. Every patch in the program has a value for fire, fuel, moisture and elevation. All of these values fall upon a sliding scale. Fuel is necessary for a fire value to exist in a patch, the higher the fuel value, the larger the fire value becomes, but at the beginning of each timestep the fuel is depleted based on the fire value. A value for moisture also is subtracted from the fire value, representing the ease with which dryer materials are burned. Moisture and fuel do not determine which patch the fire spreads to, that is done based on wind-direction, wind-speed and elevation. Global variables determines the wind-direction and wind-speed. At the beginning of each timestep, every patch with a fire value larger than zero examines its eight neighbors. It selects the one with lowest value for the number:

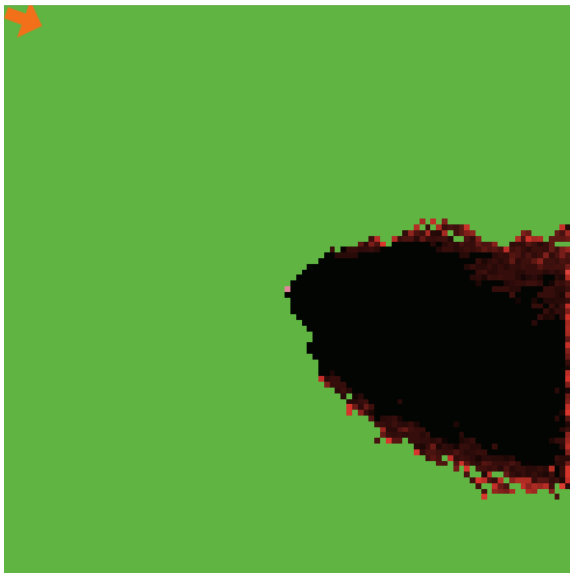
random((abs(pdirection - wind-direction) wind-speed) - elevation)*

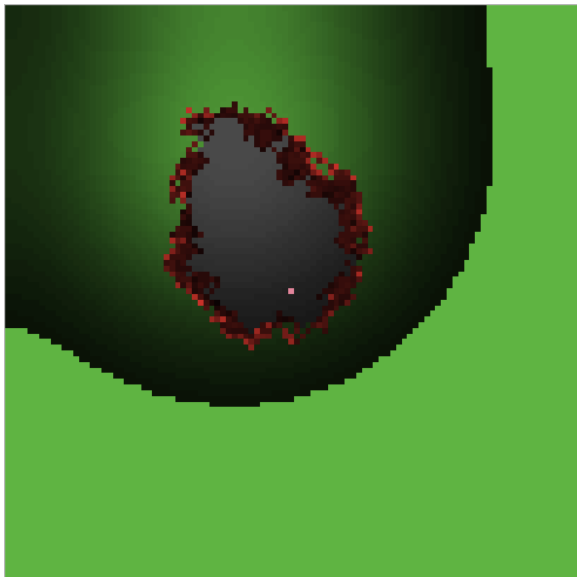
Essentially, the patch most likely to be chosen is the one that has the heading (relative to the original patch) that is closest to that of the wind and that has the highest elevation. The

selected patch ignites. The weight of the wind-direction in this equation is dictated by the value for wind-speed. This means that as the wind-speed increases, the fire is more likely to spread in the direction of the wind and when speed is very low, the fire is much more influenced by the elevation, burning uphill.

Here are examples of the fire model at its simplest, the pink square at (0,0) represents the starting point of the fire and the orange arrow points in the direction of the wind. Green patches still have fuel, black have neither fuel nor fire and red have some level of fire, brighter being higher. Moving clockwise, the images have: fuel only, wind and fuel, elevation and fuel.







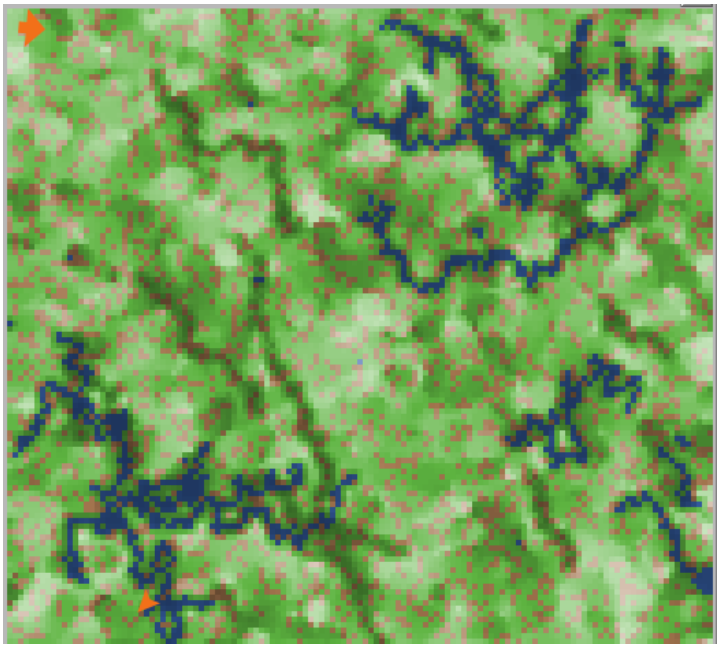
In addition to fire spread, the model also keeps track of the fire's intensity, based on the intensity equation:

I = Hwr

H, heat yield is calculated as the ratio of fuel to moisture in every patch. *w*, weight is determined by keeping track of the total number of patches burned. *r,* is found by dividing the total area burned by the time since the fire started.

4.2 Landscape Generation and GIS

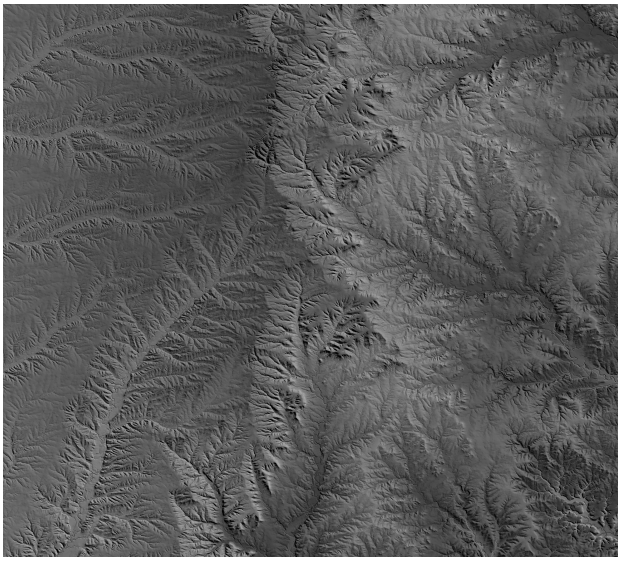
We used a randomized, procedural generation algorithm to create the test landscapes. The basic rules for the model are described here. First, all patches are set to either maximum or minimum elevation. Then, patch elevation is diffused evenly. Following this, 'origin patches' bring their neighbor's elevation level closer to their own height. Origin patches are picked randomly according to the 'elevation-roughness' variable; it ranges from 0% to 100% of the patches. In this model, 70% to 99% are used as default values for elevation-roughness. After this, grass is added according to biome type, patch fuel amount, and soil health. Lakes are then added to all patches below 1200 if altitude is set to lower or mountainous, or all patches below 5500 if upper are transformed into water. Trees are added according to tree density, if it is 95%, then 95% of dry patches are assigned a tree. Finally, rivers are made. Agents 'carve' rivers by making one or three patches in their path deeper and full of water, then smooth the surrounding dry land.

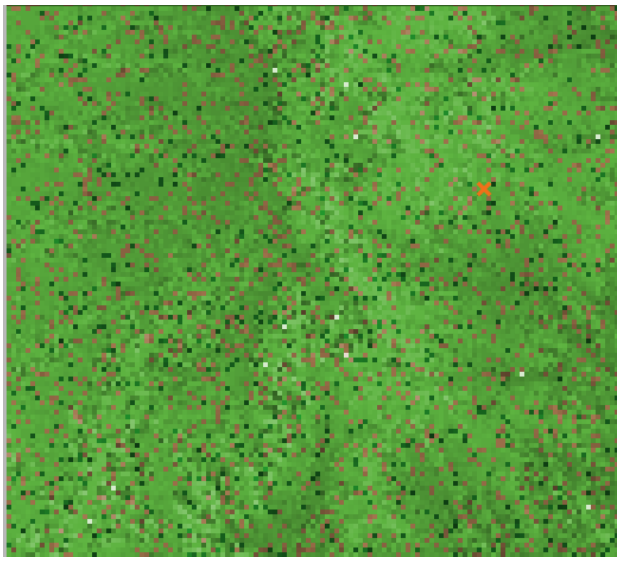


The biome types (Savanna, which is seen to the left, Temperate, Rainforest and Swamp) do not directly correspond to data for these land types. Instead, they represent relative moisture and fuel levels. For example, the Savanna land type is the driest of

the four, followed by Temperate, Rainforest and Swamp. The Savanna land type also has the lowest fuel levels of the four, designed to represent a primarily grass based biome. The Rainforest and Temperate have high fuel levels, representing a high density of trees.

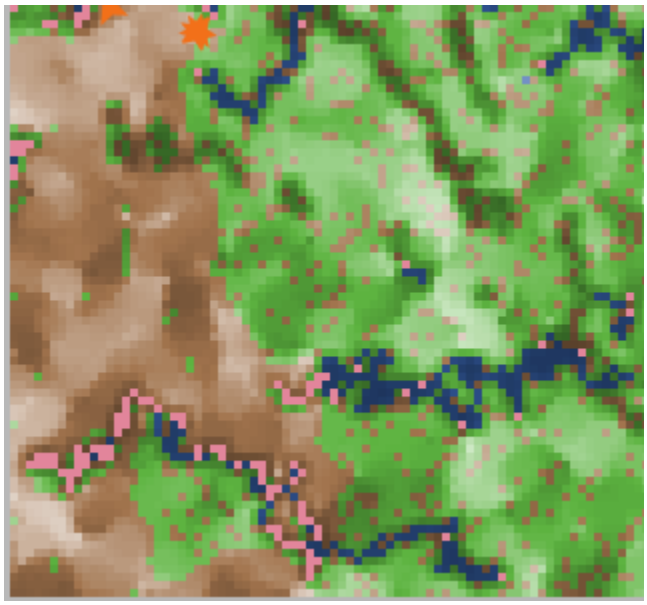
In the Netlogo model, GIS data can be used to create a landscape. This data must come in the form of a grayscale heightmap. A heightmap represents elevation data in an image file by assigning low elevation values to a certain range of colors, and high elevations to another range of colors. Our model uses dark grays to represent low altitudes, and light grays to represent high altitudes. Any grayscale image is acceptable input. After the elevation data is parsed by the patches, grass and trees are created. Patches below a user-defined elevation are turned into water. The images below are examples of a GIS heightmap and the corresponding Netlogo Model





4.3 Watershed Response

The watershed response in this model is represented by modeling the “debris levels” of the bodies of water in the given landscape. This is done by making any patch that no longer has fuel generate an agent. The agent travels as far downhill from its location as possible. If it happens to end up in a river, the debris-level of that patch is raised by one. The total pollution value for the model is calculated by summing the debris levels of all the river patches and dividing by the total number of river patches. This gives the value which for average debris-level per patch. In the model, the debris is represented by pink patches which appear after the fire



has ended. The aspect of the model is designed to replicate the effects of a rainstorm on a burned landscape by calculating how much much debris could be expected to be washed into a given body of water after a fire.

4.5 Scale

The scale of the dynamically generated landscape model's scale is roughly based on '1 patch = 1 square meter' scale. This means that the default grid (128 x 128 patches) is very close to being 4 acres in total. The timestep for measurement purposes is 1 tick = 1 minute. This is not an ideal timestep, but it is a reasonable approximation for our purposes. The timestep would be more important if the data being collected pertained to burn time, but the primary variables are area burned and debris levels, both unaffected by time. This scale does not remain the same for GIS landscapes

4.6 Firefighters

On of the goals of this project was to examine the ways fire damage might be mitigated with the protection of water resources in mind. One way fire damage is controlled is through firefighting. While not currently implemented in the main model, we developed another Netlogo model that is essentially a genetic algorithm designed to evolve the most successful firefighting techniques. The way this works is that the model begins with an initial population of agents with random values assigned for certain variables. The variables control things like the distance from the fire that an agent begins to 'dig' a fireline (this is accomplished by removing fuel from the patch), the length of the fireline and the direction the agent moves when it becomes close in proximity to the fire. These variables are designed to do two things; primarily to protect the agents from being burned by the fire and secondly to minimize the area burned by the fire. These priorities are reflected by the way further generations or agents are selected.

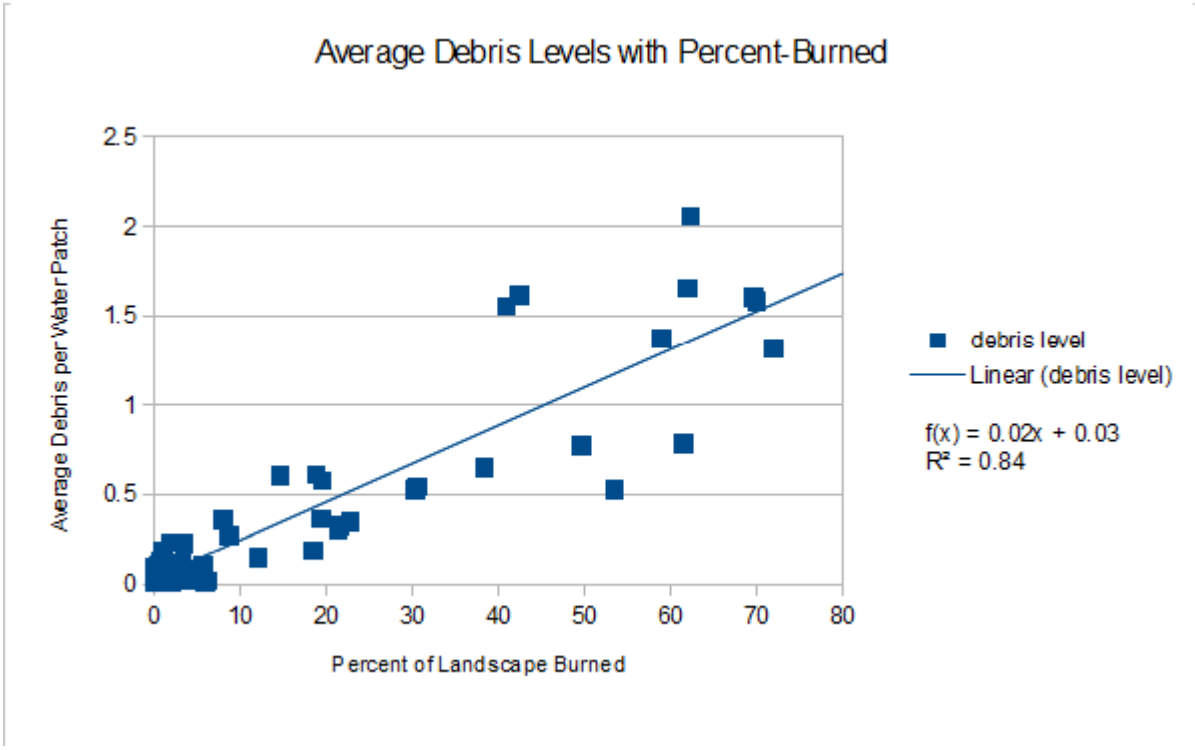
The process is as follows; when the entire fire has burned out, a new landscape is generated. The agents are then evaluated and ranked in 'fitness' based on how successful they were in avoiding the fire. The agents who have above average fitness then reproduce, passing their variable values down to their children and then dying. The children, ideally, represent the most successful parents and also have small amounts of random mutation, designed to promote diversity.

5.0 Data and Results

The primary measurements taken in these tests were the percent of the landscape burned and the average debris-level per patch. The graphs illustrate the correlation between the two as well as the impact of different variables on them. The experiments were conducted within Netlogo, using the Behaviorspace feature. The data processing was done in LibreOffice Calc.

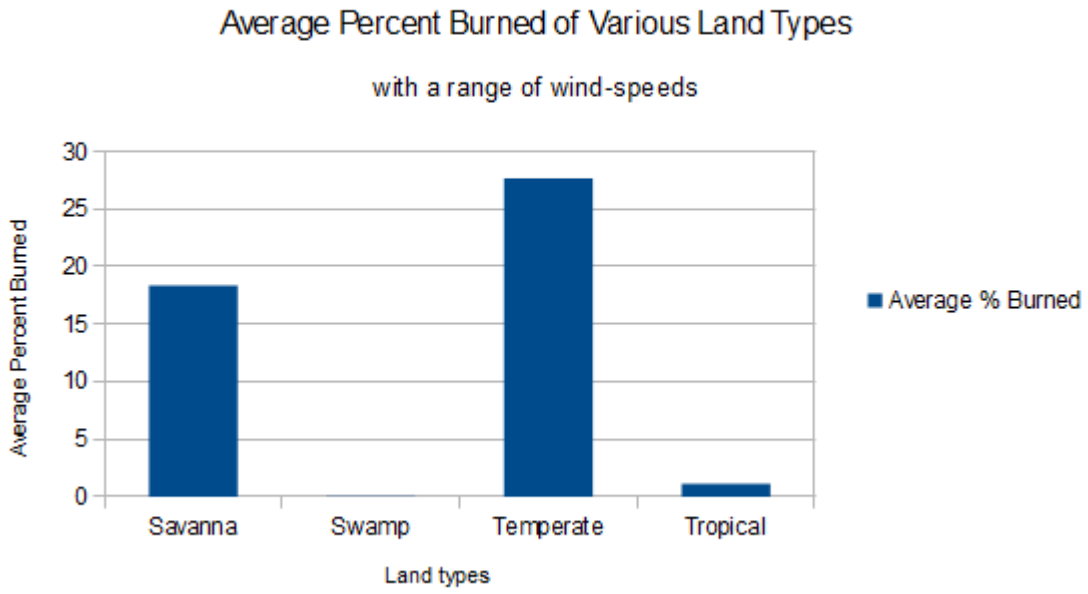
5.1 Netlogo Experiments

Correlation Between Area Burned and Average Debris Levels

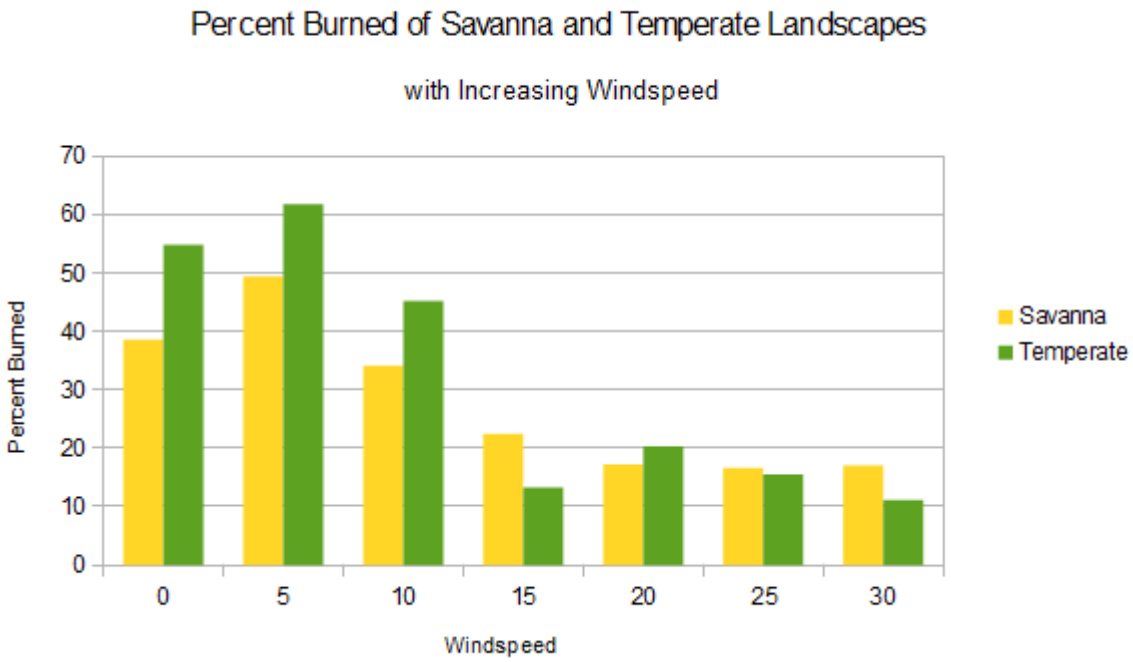


This graph has data taken from numerous trials of different land types and at different wind speeds (every land type, three repetitions of the windspeeds 0, 5, 10, 15, 20, 25 and 30). The reason that the majority of the data points are clustered near 0,0 is due to the fact the Swamp and Rainforest land types were included, and fire had very little effect on them. Overall, this graph demonstrates a positive linear correlation between debris-level and area burned. This is supported by the relatively high R^2 value of 0.84. The indications of these results are

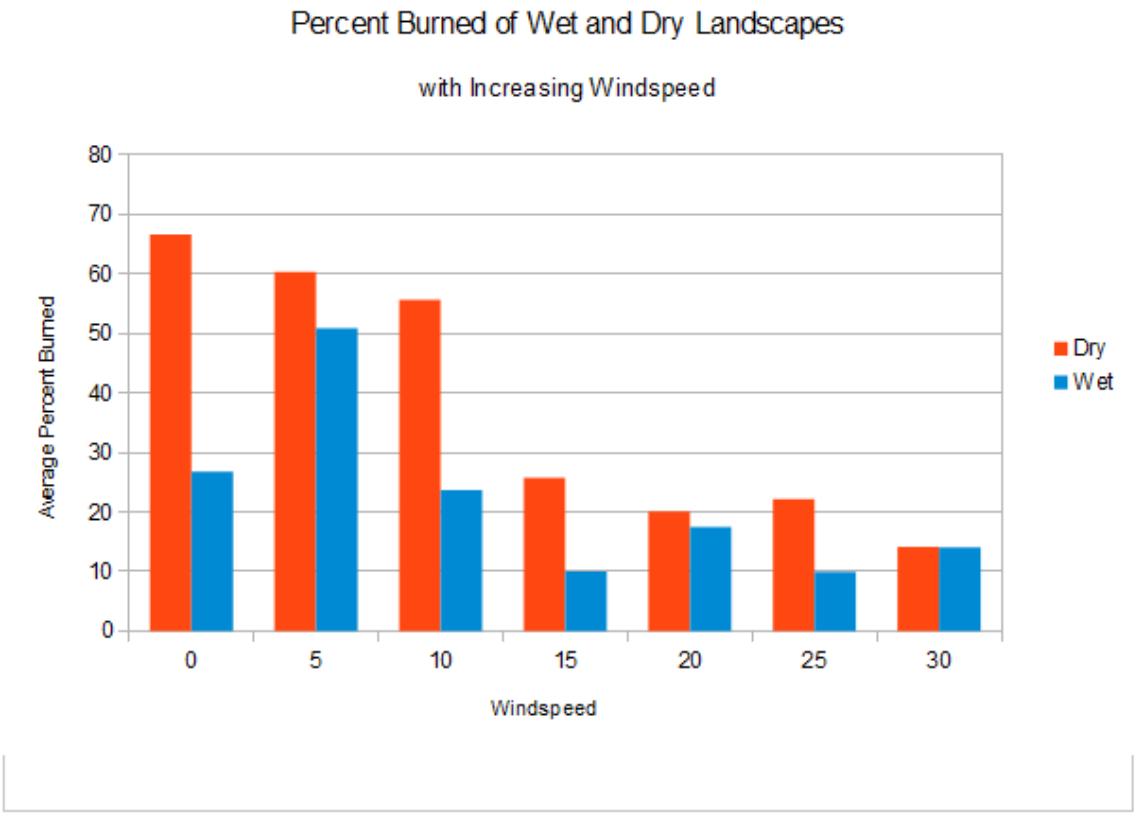
logical: the more land that is burned, the higher the average debris-levels will be in the rivers. The assumption of correlation is maintained throughout the rest of the analysis.



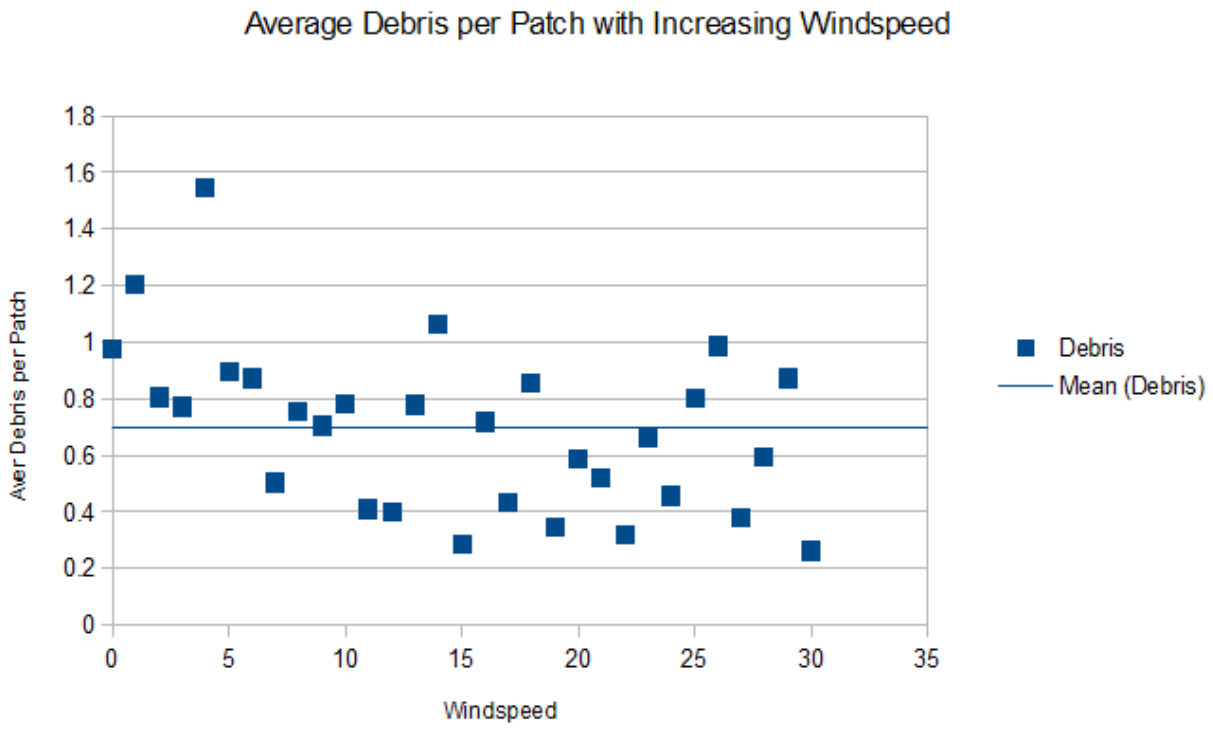
This graph is based on the same trials as the previous graph. It consists of the average percentage burned of the four landscape types, across a variety of windspeeds. It is clear from this figure that the impact of burns in the Swamp and Tropical landscapes are fairly negligible compared to Temperate and Savanna. Because of this, no more tests were performed on the Swamp and Tropical land types. Instead the tests focused on the land types which had large burn areas (and subsequently larger average debris-levels.)



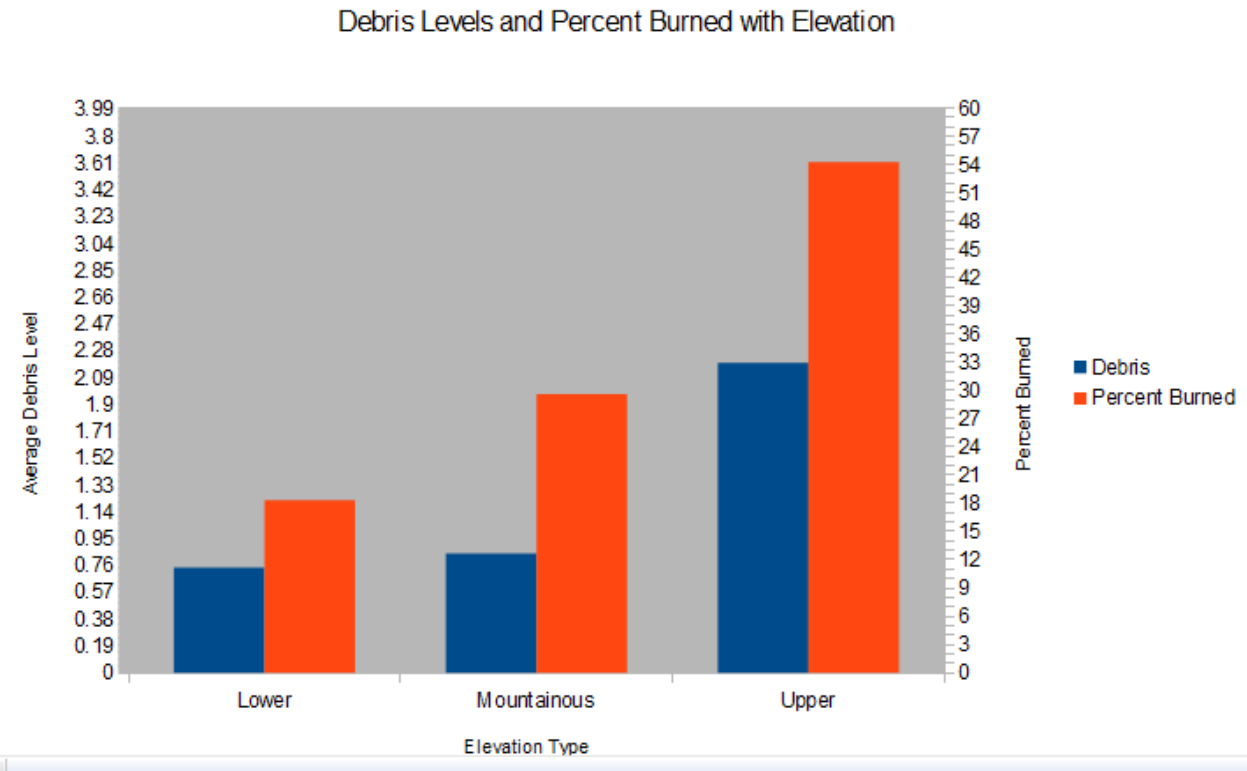
This graph is based on the same dataset as the previous two. It illustrates the differences in burn area between Savanna and Temperate land types. The downward trend of the area burned with increasing windspeed is immediately clear. This is a consistent trend throughout the experiments and will be discussed in detail at a later point. This figure also shows that when windspeeds are 10 or below, the area of Temperate forest burned is substantially higher than Savanna. However, Savanna passes Temperate at a windspeed of 15 and remains quite close from there on. This could be accounted for by the fact that the Temperate land type has more fuel, but is wetter than the Savanna and has more rivers. The higher fuel levels mean that the fires are more sustained and thus have more potential to speed. The increased larger number of rivers means that in the Temperate land type, higher windspeeds have a higher probability of putting out the fire along a river.



This graph shows the values for area burned for trials (from the same dataset as the previous graphs) separated by Wet and Dry. “Wet” landscapes are simply the landscape type with its average soil-moisture increased by 20% and the average number of rivers increased by 30%. The trend of decreasing area with increasing wind remains and it is also evident that, for the most part, dry landscapes have larger burn areas. This is what we would expect to see in a real-world situation. Fire will spread more quickly and persistently when it can more easily ignite fuel. When a fuel is very moist, or when fuel has been replaced by a body of the water, it makes sense that a fire would have trouble burning large areas. One notable exception to this trend are the values when the windspeed is 30. At this point there is no notable difference between Wet and Dry. This could be an anomaly, or it could speak to the unilateral impact of very high windspeeds.

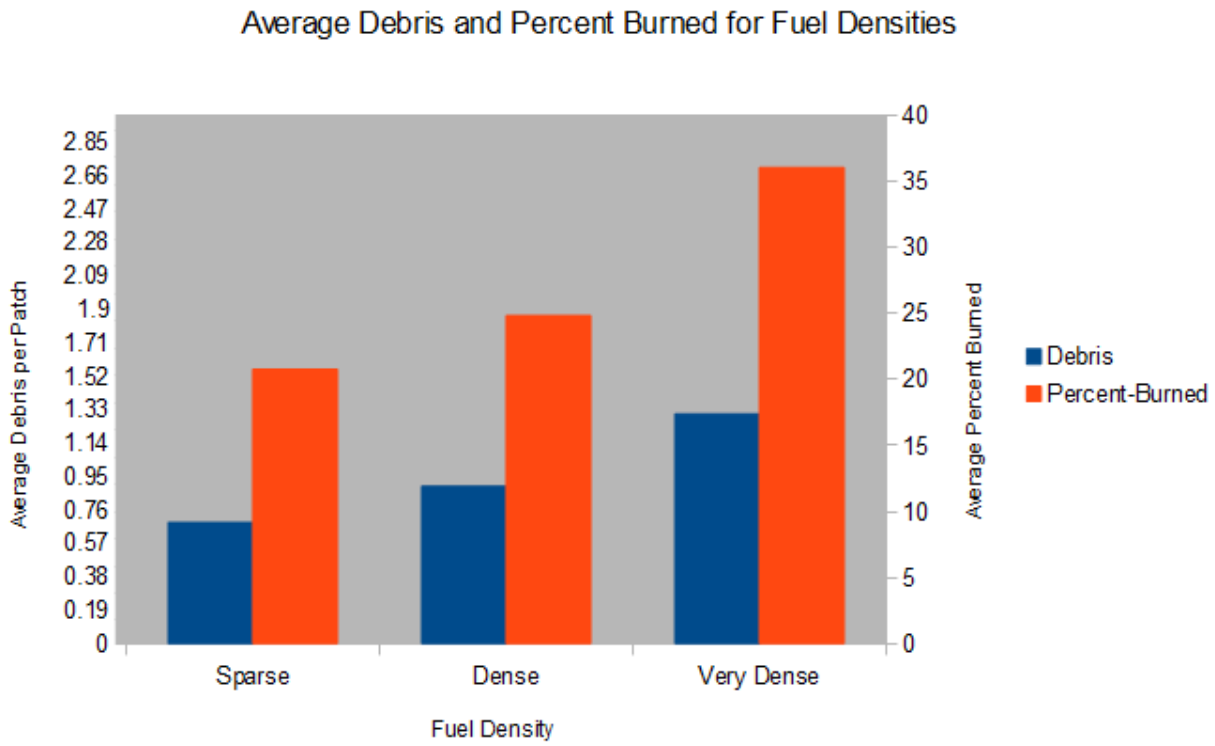


The data is this graph was obtained from a series of trials which tested Savanna and Temperate land types over windspeeds 0 to 30. The graph depicts the average values for debris levels per patch for each of the windspeeds. The trend, while not completely linear, does seem to trend towards lower debris levels with higher windspeeds (related, presumably, to the fact that previous graphs showed lower burned areas with higher windspeeds). The trend is evidenced by the fact that the majority of the points before 15 fall above the Mean and the majority of those after fall below. This trend is interesting and says a lot about the nature of our model. Basically, the higher windspeeds produced much more focused (though more powerful) fires. Very low windspeeds allow for more homogeneous fire spread, especially along the sides of watersheds as fire unbiased by wind has a tendency to burn uphill. This leads to larger debris concentrations for lower windspeeds.

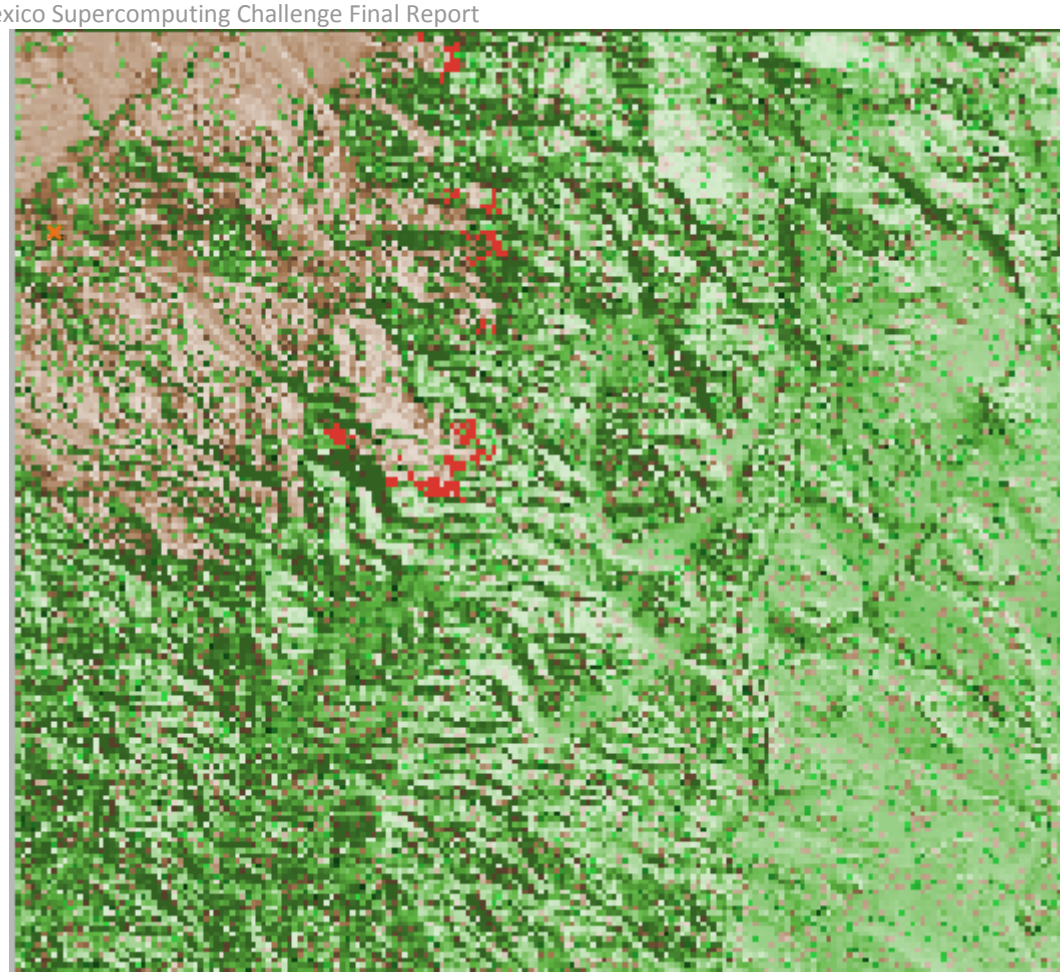


This graph is comprised of averaged data based on an experiment of repetitions of Temperate land types with different elevation categories at a constant windspeed of 15. The elevation profiles of Lower, Mountainous and Upper are assigned based both on the average elevations of the terrain types and on the variation between areas within the landscape. This means that Upper landscapes have higher elevation levels than Lower and they also have more variation, with lower corresponding to a flatter, prairie like landscape and Upper to high mountain ranges. Mountainous is an average of the two.

The results again show the correlation between area burned and debris levels. They also show a clear correlation between the elevation types and the area burned and a slightly less dramatic correlation between the elevation types and the debris levels. The Lower elevation type had less burn area and lower average debris. The next highest was mountainous, although the debris levels were close to that of Lower. The highest values were found in the Upper elevation type and they were substantially higher than the others.

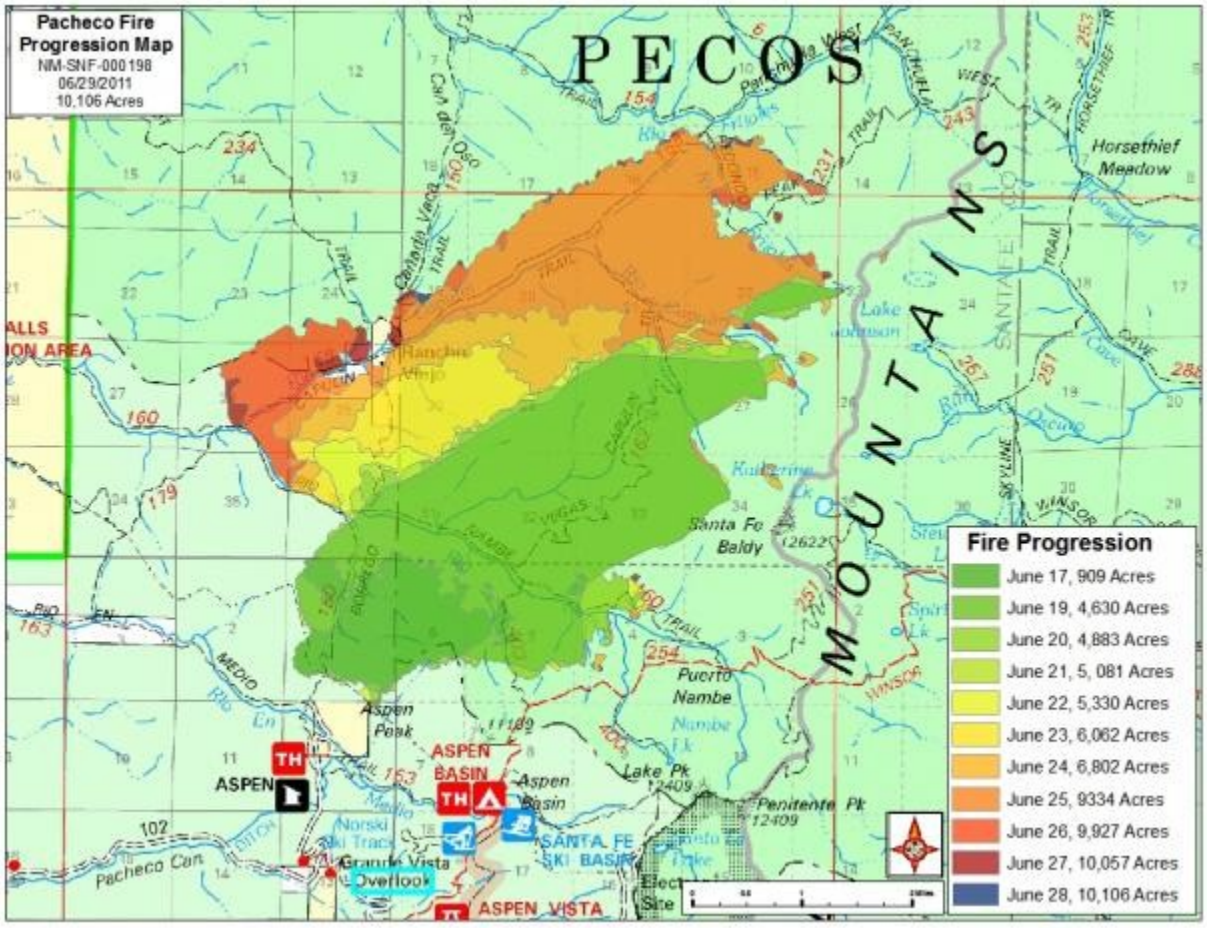


This graph consists of data collected from numerous repetition of Temperate land type at a windspeed of 15 over various 'Tree Density' levels. The trend again shows the correlation between debris level and area burned. Additionally, it is clear that there is a relationship between the tree density and the debris levels and area burned. The varying density levels essentially refer to the evenness with which trees are spread across the landscape. The value assigned to Tree Density reflects the probability of a given patch receiving a full load associated with a tree (much larger than the value for grass). These results make sense because it is expected that a fire with access to more fuel would burn more area, and as previously demonstrated, more burned area means that there will be high debris levels.



This image is the Netlogo processed elevation data for the Santa Fe National Forest, focusing on the area where the Pacheco Canyon fire burned. The burned region represents a test exploring the correlation of our model to an actual fire. The starting point (signified by an orange 'x') is close to the starting point of the fire. The progression map from the actual fire is shown below. The characteristics of the Pacheco fire were that it started in the Nambe river watershed, initially spread east, uphill towards Santa Fe Baldy and then North, uphill along the side of the Nambe drainage. Our model by no means produced a perfect replication od the Pacheco fire. The fire in our model grew larger and did not stay within exactly the same drainages as the real on. However, considering that we did not account for wind conditions and the fuel approximation was very rough, our model demonstrated striking similarities. The fire was ignited and began climbing the sides of the watershed, at the same time burning quickly uphill towards Santa Fe Baldy. The fire was stopped at points of sharp downhill slope and burned quickly uphill, slowing as it approached the divide. This model did not include bodies of

water, but it is evident based on where the fire burned that large amounts of debris would be expected in the Nambe and Capulin rivers, the rivers our water samples were taken from.



6.0 Conclusions

The results of these experiments point to several important trends, which both comment upon the validity and usefulness of the model and provide a new perspective on the subject.

6.1 Improvements

As an empirical model attempting to represent a complex environment, there are naturally many simplifications. We attempted to isolate variables important to our question, the primary variables explored were: wind, slope, fuel and moisture. Improvement along the lines of these variables include the following. The use of GIS data for fuel and moisture profiles could help standard the fuel assumptions made in the model and help bring them into the real world. At this stage, the model has a direct correspondence to elevations, but fuel and moisture are estimated and standardized only relative to each other. Also, method of programming in or otherwise accounting for various weather patterns could help with both wind variables and moisture. Additionally, if we wish to establish a more predictive model, it will be necessary to standardize the scale and timestep.

6.2 In Summary

Despite the simplifications of of this model, we were able to gather interesting and useful information.

For example, some of the most interesting results have to do with the impact that high winds have on forest fires and the areas that are burned. Based on trials and observations from our model, we came to the conclusion that higher wind speeds often result in a smaller area burned, due to focused fire, heavily influenced by the wind. However, the high windspeeds did create fires that were a lot more persistent, for example they could pass through very low fuel areas without going out, where fires with low windspeeds would be stopped. This points to the fact that a constant windspeed could be a very useful factor in predicting the the direction and speed of the fire. Wind has the power to dramatically changing the direction of the fire. In some cases, the high winds forced the fire against natural fire lines, such as rivers or canyons and

helped to extinguish the fire relatively quickly. Although this would be very negative if, for example, the fire was being forced towards a town. In our model, the wind direction remained fairly constant, but in a setting where the this was not the case, this could cause to a highly unpredictable and dangerous fire.

Another important conclusion of this project is the impact that various elevation profiles can have on both fire spread and post fire debris. The experiments demonstrated that fires burning in mountainous regions had great potential to affect water resources. This is something that has been observed in landscape studies in New Mexico (Veenhuis). The results pointing to this in our model were partially due to the fact that fires do best when they are burning up steeper slopes, and therefore are likely to burn more area. In the real world the impact of this fires on water resources in areas of uniform elevation is magnified due to increased erosion potential of steeper slopes (which our model did not account for).

The experiments conducted within the GIS terrains, particularly where the Pacheco Canyon fire burned, showed that would model has some amount of value as predictive tool, at least as far as behavior with respect to slopes and general fuel profiles.

Currently the firefighter portion of the project has not been completed. There are some results which have illustrated the value of fire lines as a firefighting technique. More than anything, the current firefighter program illustrates the principle that single, unconnected, uninformed firefighters are highly disadvantaged. It has demonstrated the importance not only of global communication between the firefighters, but of a global appreciation of the behavior of the entire fire.

The results from these experiments are potentially important to firefighters as well as humans who live near or depend on water resources that pass through fire zones. This type of information allows for predictions about the movement and severity of forest fires as well as the expected post-fire watershed response.

7.0 Appendix

Acknowledgments

We would to thank our parents for their support, our teachers Jocelyne Comstock and Jeff Mathis, and Stephen Guerin for his advice pertaining to fire models and GIS data.

Bibliography

Ainsworth, Jack, and Troy A. Doss. "Natural History of Fire & Flood Cycles." *Natural History of Fire and Flood Cycles*. University of California, Santa Barbara, 18 Aug. 1995. Web. 10 Mar. 2012. <http://www.coastal.ca.gov/fire/ucsbfire.html>.

Bodroži, Ljiljana, and Darko Stipani. "Forest Fires Spread Modeling Using Cellular Automata Approach." Web. <http://marjan.fesb.hr/~ljiljana/radovi/1.6.03.Forest%20fires%20spread%20modeling%20using%20cellular%20automata.pdf>.

Schumann, Barbara. "Student Activity: Determination of Phosphates in Natural Waters;." *The Woodrow Wilson National Fellowship Foundation*. Web. Jan. 2012. <http://www.woodrow.org/teachers/chemistry/1989/43phosphates.html>.

"Fire Behavior." *Fire Fundamentals*. Tropical Savannas CRC & Bushfire CRC, 2012. Web. Jan. 2012. <http://learnline.cdu.edu.au/units/sbi263/fundamentals/behaviour.html>.

Meixner, Tom, and Pete Wohlgemuth. "Wildfire Impacts on Water Quality." *Southwest Hydrology* (2004): 24-25. Web. 10 Oct. 2011. <http://www.swhydro.arizona.edu/archive/V3_N5/feature7.pdf>.

Neary, Daniel G., Kevin C. Ryan, and Leonard F. DeBano. *Wildland Fire in Ecosystems: Effects of Fire on Soil and Water*. 2008. Web. Oct. 2011.

"Pacheco Fire." *InciWeb: Incident Information System*. 23 July 2011. Web. 20 Nov. 2011. <http://www.inciweb.org/incident/2344/>.

Veenhuis, Jack E., and Philip R. Bowman. *Effects of Wildfire on the Hydrology of Frijoles and Capulin Canyons in and near Bandelier National Monument, New Mexico*. USGS, Nov. 2002. Web. 24 Mar. 2012.

“Wildfires: Dry, Hot, and Windy.” National Geographic. Web. 29 Mar. 2012 <http://environment.nationalgeographic.com/environment/natural-disasters/wildfires/>.

Language Acquisition in Computers

Supercomputing Challenge Final Report
April 2012

Team #36
Desert Academy

Team Members
Megan Belzner
Sean Colin-Ellerin

Teachers
Jocelyne Comstock
Jeff Mathis

Mentor
Jorge Roman

Executive Summary

This project explores the nature of language acquisition in computers, guided by techniques similar to those used in children. While existing natural language processing methods are limited in scope and understanding, our system aims to gain an understanding of language from first principles and hence minimal initial input.

The first portion of our system is focused on understanding the morphology, or word structure, of language using bigrams which are two-letter sequences within words. The program was developed first in C++, and then translated into Java to take advantage of the ability to use non-standard characters. We use frequency distributions and differences between them to define and distinguish languages. English and French texts were analyzed to determine a difference threshold of 55 before the texts are considered to be in different languages. The program was also tested with Spanish texts and found to work with the same threshold, and the frequency distributions for the individual languages were analyzed.

The second portion of our system focuses on gaining an understanding of syntax, or sentence structure, of a language using a recursive method. The program uses one of two possible methods to analyze given sentences based on either sentence patterns or surrounding words. Both methods have been implemented in C++. Using a minimum of initial input, the program is able to understand the structure of simple sentences and learn new words.

In addition, we have provided some suggestions regarding future work and potential extensions of the existing program. We have described how the program might currently analyze certain sentence constructs along with how the program could be edited to provide better understanding. We have also made suggestions regarding how to implement a computationally intuitive understanding of semantics, or the meanings of words in a language.

Contents

1 Introduction

Natural language processing is a wide and varied subfield of artificial intelligence. The question of how best to give a computer an intuitive understanding of language is one with many possible answers, which nonetheless has not yet been answered satisfactorily. Most existing programs work only within a limited scope, and in most cases it cannot realistically be said that the computer actually understands the language in question.

This project seeks to give a computer a truly intuitive understanding of a given language, by developing methods which allow the computer to learn the language with a minimum of outside input, on its own terms. We have developed methods to teach the computer both the morphology and the syntax of a language, and have provided some suggestions regarding language acquisition at the semantic level.

1.1 History

The idea of natural language processing originates from Alan Turing, a British computer scientist, who formulated a hypothetical test, known as the "Turing Test". The "Turing Test" proposes that the question "Can machines think?" can be answered if a computer is indistinguishable from a human in all facets of thought, such as conversation; object identification based on given properties; and so forth [1]. After Turing's proposition, many attempts have been made to create natural language processing software, particularly using sound recognition, which is currently used in cell-phones, most proficiently in the iPhone 4S Siri system. However, most of these programs do not have high-level semantic abilities, rather they have a very limited set of operations, for which keywords are assigned. For example, the Siri system can send an email or text message. When told to send an email or text message, the software uses these keywords to open a blank e-mail or text message and when told what is to be written in the e-mail or text message, there is no semantic

understanding of the message, simply a transcription of the words using voice recognition [2]. Similarly, there is a lot of software, such as LingPipe, that is able to determine the origin and basic 'significance' of a term or sentence by searching the term(s) on a database for other uses of the term(s). These programs do not, however, gain a semantic understanding of the term(s), rather they simply collect patterns of information with association to the term(s) [3].

1.1.1 Statistical Parsers

There have also been some more technical, less commercially efficacious attempts at natural language processing, such as statistical language parsers, which have been intricately developed by many educational institutions. Parsers are a set of algorithms that determine the parts of speech of the words in a given sentence. Current parsers use a set of human-parsed sentences that creates a probability distribution, which is then used as a statistical model for parsing other sentences. Stanford University and the University of California Berkeley use probabilistic context-free grammars (PCFG) statistical parsers, which are the most accurate statistical parsers currently used, with 86.36% and 88% accuracy, respectively [4] [5]. The different parts of speech are separated as in Figure 1.

In Figure 1, NN = noun, NP = noun phrase, S = subject, VP = verb phrase, and the other symbols represent more specific parts of speech. One can see that the parser splits the sentence into three parts: the subject noun phrase, the verb phrase, and the noun phrase. Each of these parts is then split into more parts and those parts into parts, finally arriving at individual qualifications for each word. The assignment of a given part of speech for a word is determined by tentatively allocating the part of speech that is most probable for that word, which is then tested within its phrase (i.e. subject noun phrase, or verb phrase, etc.), and if the probability remains high then that part of speech is set for the word. These

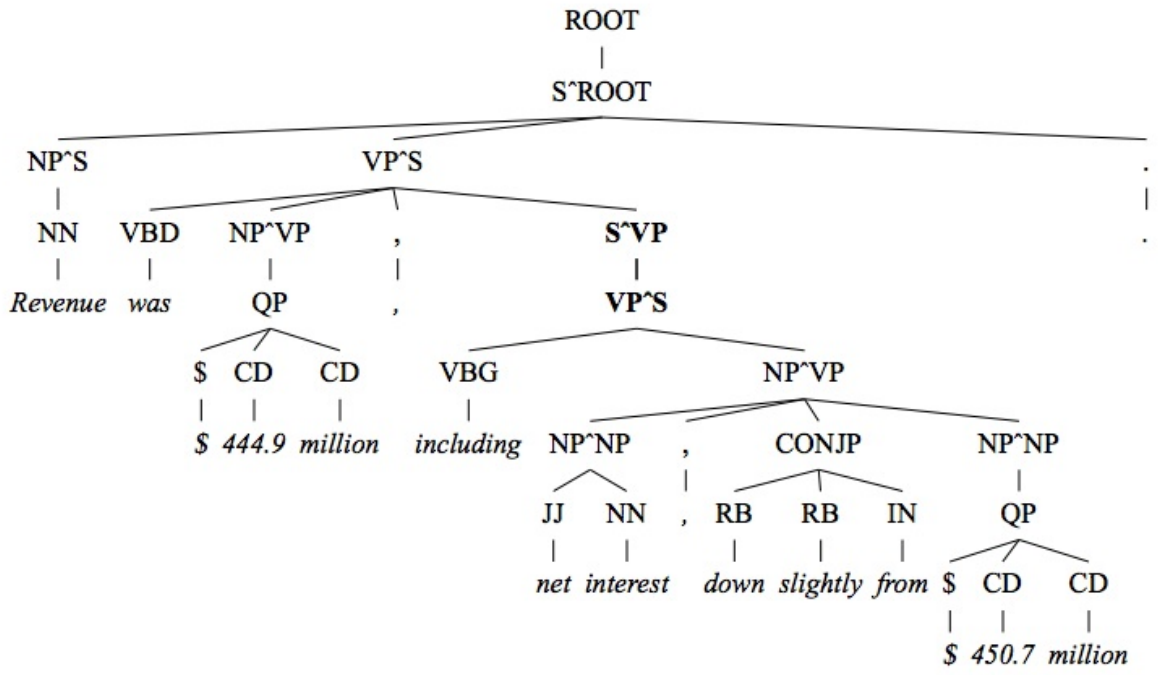


Figure 1: Statistical parser tree [4]

parsers are called context-free because the parse of a word is not affected by the other words in the sentence other than those in its phrase, while the less accurate parsers obtain an overall probability for a sentence and adjust their parsing accordingly [6].

1.1.2 Other NLP Programs

In addition to statistical parsers, which only determine the syntax of a sentence, some elementary programs have been written for evaluating the semantics of a given body of text. There is a system called FRUMP that organises news stories by finding key words in an article that match a set of scripts and then assigns the article to a certain category, in

which it is grouped with other articles of similar content. SCISOR summarizes a given news article by analyzing the events, utilizing three different sets of knowledge: semantic knowledge, abstract knowledge, and event knowledge. As the program sifts through the body of text, certain words trigger different pieces of knowledge, which are compiled to gain the best understanding of that word or sequence of words. The resultant meanings can then be organized and rewritten using similar meanings, equally balanced among the three sets of knowledge as the original piece of information. Similarly, TOPIC summarizes a given text by distinguishing the nouns and noun phrases and then analyzing their meaning through a "thesaurus-like ontological knowledge base", after which the program uses these equivalent meanings to rewrite the text [7].

1.2 Common Language Properties

Certain elements of language are commonly used both in natural language processing and the general analysis of language. These properties include bigrams and recursion, two properties which play a significant role in this project.

1.2.1 Bigrams

An n-gram is a sequence of n letters. The most commonly used forms of n-grams are bigrams and trigrams because these offer a specific indication for a set of information, without signifying extremely rare and complex aspects of the subject. A good example of this is the use of bigrams in cryptography. A common method of decoding a message that has been encoded using a keyword, like the Vigenere Cipher encryption, is to calculate the distance in letters between two of the same bigrams in order to determine the length of the keyword, and then the keyword itself [8]. If any n-grams are used, where n is greater than or equal to 4 or even in some cases if n is equal to 3, then the number of same n-grams for a given

message would be very rare and make determining the length of the keyword increasingly difficult.

Similarly, in natural language processing, bigrams are used for word discrimination, which is the understanding of an unknown word based upon bigram correspondence with a reference list of known words. In addition, word bigrams are used in some lexical parsers, the Markov Model for bag generation, and several text categorization tools [9].

1.2.2 Recursion

The principle of recursion is an essential aspect of human language, and is considered one of the primary ways in which children learn a language. For a sentence with a particular pattern, a word with a specific part of speech can be exchanged for another word of the same part of speech, indicating that the two words have the same part of speech. For example, given the sentence: "The boy wears a hat", it can be determined that "the" and "a" are the same part of speech by reconstructing the sentence as "A boy wears the hat". In addition, the word "boy" can be exchanged for the word "girl", indicating that these are also the same type of word, thereby expanding the lexicon of the child.

In addition, the words of a sentence can remain unchanged, while the pattern changes, thereby introducing a new part of speech. If we have the sentence "The boy wears a hat" or "A B C A B" if represented as a pattern of parts of speech, we can add a "D" part of speech by creating a new sentence, "The boy wears a big hat" (A B C A D B). The child ascertains that "big" must be a new part of speech because no words have previously been placed between an "A" and a "B". This method can be repeated for any new part of speech, as well as embedded clauses such as "The boy, who is very funny, wears a big hat."

Finally, recursion can be used to indicate the grammatical structures of a language. Let us examine the following poem:

When tweetle beetles fight,
it's called
a tweetle beetle battle.

And when they
battle in a puddle,
it's a tweetle
beetle puddle battle.

AND when tweetle beetles
battle with paddles in a puddle,
they call it a tweetle
beetle puddle paddle battle.

- Excerpt from Dr. Seuss' *Fox in Socks* [10]

The author uses the recursive principle to indicate that "tweetle beetle" can be both a noun and an adjective, and then repeats this demonstration with "puddle" and "paddle". Further, the correct placement of the new noun acting as an adjective is shown to be between the old string of nouns acting as adjectives and the object noun. Conversely, the poem illustrates that a noun acting as an adjective can be rewritten as a preposition and an added clause, e.g "a tweetle beetle puddle battle" can be rephrased as "a tweetle beetle battle in a puddle" [11]. Thus, the principle of recursion can allow a child to acquire new vocabulary, new types of parts of speech, and new forms of grammar.

1.3 Linguistic Interpretation

There is a basic three-link chain in the structure of language. Phonetics is the most basic structure, which is formed into meaning by units, known as words. Units are then arranged syntactically to form sentences, which in turn forms a more extensive meaning, formally called semantics [12].

It is fundamental in learning a language that a computer understand the connections in

the phonetics-syntax-semantics chain, and the structure and computations of each, with the exception of phonetics. Phonetics and their formulation can be disregarded because these refer more greatly to the connections of the brain to external sounds, than the core structure of language. In fact, the entire external world can be ignored, as there need not be an inference between external objects in their human sensory perception, and their representation as language, formally known as pragmatics or in Chomskyan linguistics as E-language (External Language). Instead, the relations between words and meaning, intricately augmented by the semantics created by their syntactically accurate formed sentences, is the only form of language necessary, denominated contrastingly as I-language (Internal Language) [12]. Noam Chomsky argues that I-Language is the only form of language that can be studied scientifically because it represents the innate structure of language [7]. Language similar to this form can be found in humans who are literate in a language, yet can not speak it. Therefore, due to the increase in structure, the lack of external representation in natural language processing may be more advantageous than one might think.

1.4 Project Definition

This project examines the nature of language acquisition in computers by implementing techniques similar to those used by children to acquire language. We have focused primarily on morphology and syntax, developing methods to allow a computer to gain knowledge of these aspects of language. We have developed programs in both C++ and Java.

Regarding morphology, the program is able to analyze the word structure of given languages and distinguish between languages in different samples of text using bigram frequencies, and we have examined the usefulness and limitations of this method in the context of existing methods. Using this technique we have developed computationally understandable definitions of English, French and Spanish morphologies. We have also described and par-

tially implemented a novel technique for understanding the syntax of a language using a minimum of initial input and recursive methods of learning both approximate meanings of words and valid sentence structures. Finally, we provide suggestions for future work regarding the further development of our methods for understanding syntax as well as potential methods for gaining a rudimentary understanding of semantics.

2 Morphology

To analyze the morphology of a given language, bigrams can be used to define and compare languages. Since the frequency distribution of a set of bigrams is unique to a given language (across a large enough sample text), this can be used as an accurate identifier of a language with minimal effort.

2.1 Algorithm

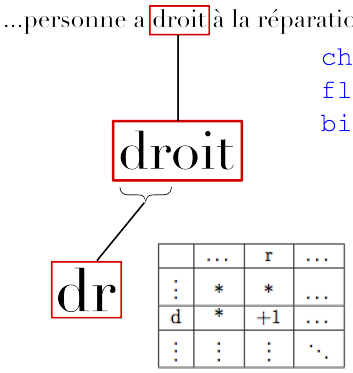
The program was initially developed in C++ then translated to Java to take advantage of non-standard characters, and is set up in two main portions. The first step involves generating a table of frequency values from a file, and the second step is to compare the two tables and determine the level of similarity.

To generate a frequency table, a two-dimensional numerical array is created from the set of valid characters such that the array includes a space for every possible bigram, with the initial value for each being set to zero. For each word from the input file, the program checks each pair of letters, adding one to the corresponding position in the bigram frequency table. Once the end of the file is reached, each frequency count is divided by the total number of bigrams found times 100 to give a percentage frequency. This process is shown in Figure 2.

This produces an array similar to Table 1, which can be analyzed separately to examine common and uncommon bigrams for a given language, or compared with another text's table to distinguish between languages as detailed below.

After the frequency tables are created for each file, the two must be compared to determine the level of similarity between the languages of the two files. This is done by finding the absolute values of the differences between corresponding frequencies for the two files, then finding the sum of these differences as seen in Figure 3.

This gives an approximate measure of how different the two files are in terms of the




```
char alpha[] = set of valid characters
float bigram[alpha.length][alpha.length]
bigram[i][j] = 0 for all (i, j)
    for all words in file
        for all two character subsets xy
            find position of x in alpha
            find position of y in alpha
            bigram[pos(x)][pos(y)]++
for all (i, j)
    bigram[i][j] = (bigram[i][j] / total_bigrams) * 100
```

Figure 2: Learning morphology with bigrams

	a	b	c	...
a	0	0.204	0.509	...
b	0.080	0	0	...
c	0.298	0	0.269	...
⋮	⋮	⋮	⋮	⋱

Table 1: Sample of frequency array

frequency of given bigrams. As each language tends to have a unique frequency distribution, a large net difference suggests a different language for each file while a smaller net difference suggests the same language. The threshold dividing a determination of 'same language' or 'different language' was experimentally determined to be approximately 55.

```
float differences = 0
for all (i, j)
    differences += abs(file1.bigrams[i][j] - file2.bigrams[i][j])
```

Figure 3: Calculating morphology differences between sample texts

2.2 Limitations

This method does have certain limitations, however. Since the program deals with bigrams (though it can be easily made to use n-grams for any n greater than 1), single-letter words are not taken into account. While this does not have a large overall effect, it produces some inaccuracies in the analysis of frequencies for a given language.

A more significant limitation is the requirement that all "legal" characters be defined before the program is run. Although it would be relatively straightforward to dynamically determine the character set based on the input files, this creates issues where the character sets for each file are not the same, making it difficult, if not impossible, to accurately compare the two files. Even ignoring this, varying the number of characters may produce variations in the threshold used to determine language similarity. The program is also effective only for files of considerable length to allow for a large enough sample size.

2.3 Results

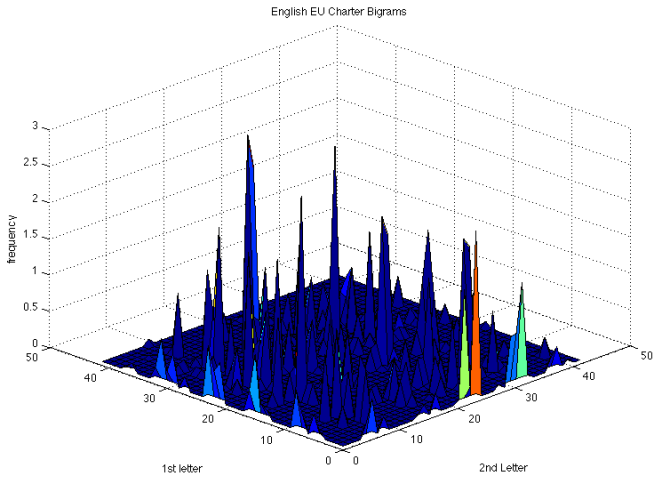


Figure 4: Distribution of English bigrams

The program was run using a series of files in English, French and Spanish. Initial frequency tables for analysis of individual languages were created using the EU Charter in each respective language [13], producing the frequency distributions shown in Figures 4, 5, and 6 for English, French, and Spanish, respectively.

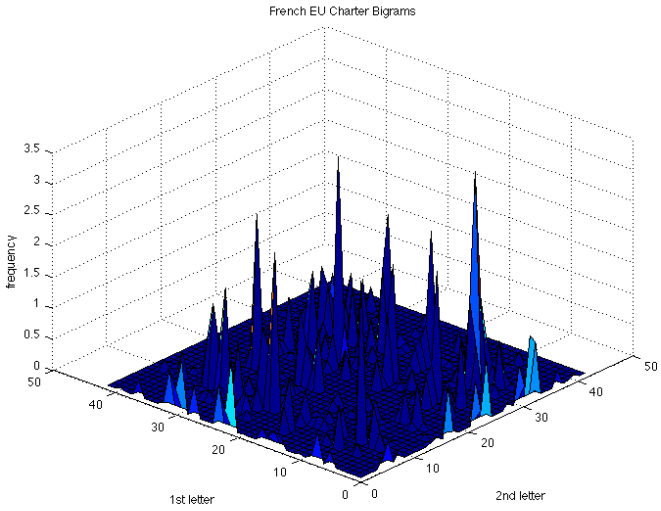


Figure 5: Distribution of French bigrams

These frequency graphs show that each language has a handful of extremely common bigrams (in addition to some which appear little or not at all). In English, this includes "th" and "he" with percentage frequencies of 2.9 and 2.8, respectively, along with "on" and "ti" also both above 2.5%. This data is slightly skewed by the text used, though "th" and "he" are indeed the most common bigrams in English. A study conducted using a sample of 40,000 words [14] gave the two frequencies of 1.5 and 1.3, respectively, though the next most common bigrams in the sample text are not as common in the English language as a whole as their frequencies here would suggest. This is largely due to an inherent bias in the text, as words such as "protection" or "responsibilities" appear frequently in the EU Charter.

French resulted in "es" and "on" as the most common bigrams, followed by "de" and

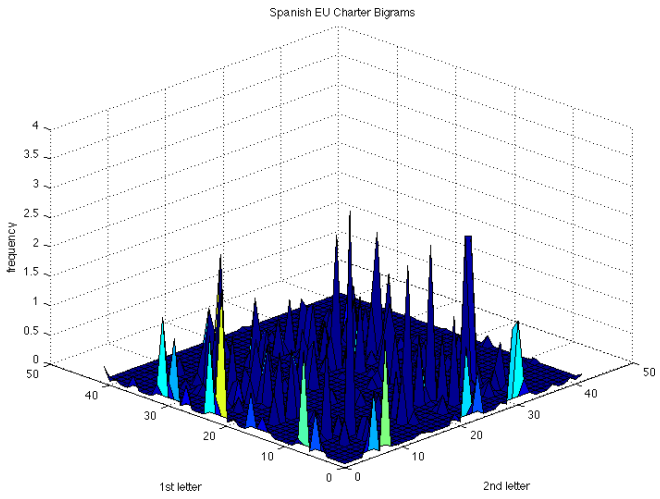


Figure 6: Distribution of Spanish bigrams

"le". In Spanish, the most common bigram by far was "de", followed by "cl", "en", "er", and "es". Again, however, these likely suffer from slight biases due to the nature of the text.

The comparison threshold was initially determined using a series of randomized Wikipedia articles of considerable length in English and French. The same threshold was also used to compare English and Spanish texts and French and Spanish texts with continued high accuracy for texts of considerable length, showing that this method does not vary notably with different languages. The outputs of these tests are shown in Table 2.

	philosophy	encyclopedia	france (fr)	capitalism (fr)	jazz (es)	nyc (es)
philosophy	...	...	...	...	...	...
encyclopedia	33.13	...	...	...	...	...
france (fr)	73.83	75.31	...	...	...	...
capitalism (fr)	73.94	79.62	30.15	...	...	...
jazz (es)	67.68	69.56	64.76	67.76	...	...
nyc (es)	71.76	73.42	66.41	70.46	28.95	...

Table 2: Sample of sum differences

Finally, this method was tested with files of varying lengths. For the set of two English texts which were tested with decreasing word counts, the point at which this method was no longer accurate was between 400 and 200 words. For other files this is likely to vary, and could be lessened with further fine-tuning of the threshold. At a certain point, however, the difference values begin overlapping due to variation and bias from the words used in the text, making accuracy impossible.

3 Syntax

To analyze the syntax of a language, a "recursive learning" method is implemented using C++. Since the program would ideally require an absolute minimum of initial information, this method takes a small initial set of words and builds on this by alternately using known words to learn new sentence structures and using known sentence structures to learn new words as seen below.

{cat, man, has} $\rightarrow$ "The man has a cat" $\rightarrow$ "The x y a z" $\rightarrow$ "The man wore a hat" $\rightarrow$ {cat, man, has, wore, hat}

3.1 Algorithm

There are two main elements to understanding the recursive learning system, namely understanding both how the information is represented and the methods used to analyze new information.

The information this program gathers can be split into two pieces, information on the words themselves and information on valid sentence structures. For words, the program keeps track of the word itself and the word's type. Only two specific types are defined in the program, "noun" and "verb", and the actual meaning of each is defined by context. Any word which does not fit these definitions is defined relative to these definitions. For sentences, the program keeps track of the number of words in a given sentence pattern and the type of the word in each position (in an array). These structures are shown in Figure 7.

The methods used by the program to analyze new information can also be split into two pieces. Both methods require that some information is already known about the sentence in question, but are used in slightly different ways. The first method analyzes new words based on the structure of the sentence, by selecting the most applicable of existing patterns

```
"word" datatype
    string word
    string type
"pattern" datatype
    int word_count
    string pattern[]
```

Figure 7: Word and sentence pattern datatypes

based on correspondence with known information. The program keeps track of how many "matches" there are between the sentence and the known pattern, taking the one with the most matches (if greater than half the word count) and using it to set unknown word types. This method is seen in Figure 8

```
int matches
for all known patterns
    matches = 0
    if sentence.word_count = pattern.word_count
        matches++
    for all sentence positions i
        if word[i].type = pattern[i].type
            matches++
    if matches > word_count/2
        for all sentence positions i
            if word[i] undefined
                word[i].type = pattern[i].type
```

Figure 8: Sentence structure method of learning

This method is particularly useful for learning new nouns and verbs, in situations where other words in the sentence are primarily known grammatical particles. The other method uses the surrounding words to define the type of any unknown words. The program notes the types of the words before and after the unknown word, and the type of the unknown word is then stated as "a<type of previous word> b<type of next word>". For example, a word with type "anoun bverb" would be one which tends to come after nouns and before verbs.

This type could also be analyzed and modified to give further insight into other types of words such as adjectives, as detailed in Section 5.1. The program can redefine known words if new information is found on their positioning, using the new definition if it is shorter, and thus more general, than the previous definition. This is seen in Figure 9.

This method is best used when most of the nouns and verbs in a sentence are known, but other words exist which are not known. The program would be able to dynamically select between the two methods based on what information is already known. As a rule, the program would default to the first method initially as this is more likely to provide accurate results.

```
for all sentence positions i
    if word[i] undefined
        word[i].type = ""
        if word not first in sentence
            word[i].type += ("a" + word[i-1].type)
        if word not last in sentence
            word[i].type += ("b" + word[i+1].type)
    else if word[i].type is not noun or verb
        temp_type = ""
        if word not first in sentence
            temp_type += ("a" + temp_type)
        if word not last in sentence
            temp_type += ("b" + temp_type)
        if temp_type.length < word[i].type.length
            word[i].type = temp_type
```

Figure 9: Word-based method of learning

3.2 Limitations

With proper development this method could be used to learn many types of sentences. However, it still has several limitations. Although it is sufficient for simpler grammatical constructs, complicated words and patterns could cause confusion. In particular, words which have multiple meanings with different parts of speech would confuse the program

tremendously, and at present it also has no way of connecting related words (such as plural or possessive forms of nouns).

In addition, this method requires carefully crafted training sentences. Although it can work with a minimum of initial information unlike most existing systems, it still has to learn the constructs somewhat sequentially and avoid having too many new concepts introduced at once. This could be partially remedied by implementing a method by which the program stores any sentences it does not yet have the tools to analyze to be recalled later, so that a more general text could be used as training material.

3.3 Results

As a proof of concept for these methods, a series of three training sentences were input and analyzed by the program to learn a handful of new words and concepts. Although the program begins with far less information than many existing programs, it nonetheless needs some initial input - in this case three words which will, together with "a" and "the", make up the initial sentence, as seen in Table 3.

Word	**Type**
has	verb
hat	noun
man	noun

Table 3: Initial program input

The first sentence input into the program is "the man has a hat," which is analyzed using the word-based second method. From here, two new words are learned - namely "a" and "the" which are defined based on the surrounding words. The sentence pattern is also catalogued, and the known information reads as in Table 4.

At present, although "the" must only appear before a noun, "a" is assumed to require a

Word	Type
has	verb
hat	noun
man	noun
the	bnoun
a	averb bnoun

5 bnoun—noun—verb—averb bnoun—noun

Table 4: Program knowledge after run one

preceding verb. To correctly define "a", the next sentence reads "a man has the hat." While using the same words, the two known grammatical particles are reversed. The program redefines the word "a" with a more general definition (i.e. simply requires a succeeding noun), however the definition of "the" remains the same as the program determines that replacing the definition would add more constraints, which is counterproductive. The sentence pattern is again catalogued, and the known information reads as in Table 5.

Word	Type
has	verb
hat	noun
man	noun
the	bnoun
a	bnoun

5 bnoun—noun—verb—averb bnoun—noun
5 bnoun—noun—verb—bnoun—noun

Table 5: Program knowledge after run two

Although the two existing sentence patterns are functionally identical, and the first should actually be redefined as the second, both are kept to demonstrate the first method based on sentence patterns. For this, the sentence "the dog ate a biscuit" is used, having the same structure as existing sentences but a different set of nouns and verbs. Although some matches exist with the first pattern, the redefinition of "a" results in only two matches. Instead, the program finds this to be the same as the second sentence pattern, as the number of words matches as do the types of the words "the" and "a". Hence, the program defines the unknown

words based on this pattern, resulting in the set of information shown in Table 6.

Word	**Type**
has	verb
hat	noun
man	noun
the	bnoun
a	bnoun
dog	noun
ate	verb
biscuit	noun

5 bnoun—noun—verb—averb bnoun—noun
5 bnoun—noun—verb—bnoun—noun

Table 6: Program knowledge after run three

4 Analysis

In this project we have developed methods for allowing a computer to understand and learn both the morphology and the syntax of a language. Using novel techniques or applications, we have designed and implemented these methods and tested their capabilities for learning language.

Although the use of bigrams in language analysis is not a new idea, we have implemented it in a novel way by working to develop it as a defining quality and learning mechanism for natural language processing. The method proves very useful for understanding the morphology of a language, though only to a point. It is extremely effective when using a large enough sample text, but with smaller sample texts it is no longer able to accurately compare languages. Hence, although it creates a computationally effective "definition" of a language, its actual ability as a learning mechanism is limited. Used in tandem with other methods, the bigram method could prove extremely powerful.

The recursive learning method implemented for gaining an understanding of syntax proves very useful and has great potential to be developed further. Both of its submethods work with high accuracy for simple sentences, and hence it is able to develop a growing model of sentence construction. Even more particularly, it is able to do this with a very small amount of initial input and with methods which could be applied to many types of languages.

4.1 Comparison to Existing Programs

The use of bigrams to understand and analyze different parts of a given language has been studied and implemented substantially. For example, there are programs that calculate bigram frequencies to evaluate a language's morphology. However, unlike our program, none to date have utilized the differences in bigram frequencies between two languages to distinguish one language from the next.

The system in the program that was used for determining the parts of speech in a sentence has rarely been attempted, and when it has been used, only partially and in conjunction with other methods. Most natural language processing programs have been designed to be as the efficient and effective as possible. As a result, many use large banks of initial data, which the program then analyzes and uses for subsequent input. As discussed previously, the most common and successful programs of this sort are statistical parsers. On the contrary, our program uses the recursive principle to acquire new vocabulary and forms of syntax for a given language, provided only a very small initial set of data. In practice, our model only required one sentence with verb and noun indicated to determine the parts of speech of all other words, although not denoting them in linguistic terms (article, preposition, etc.), as well as learn new words and, in theory, to learn new sentence patterns. Despite the extensive power of the recursive method, it has rarely been used in the history of natural language processing. The results of our program illustrate the potential abilities of the recursive method that have not been seen previously.

5 Extension

The program as it stands now can learn and understand a range of language elements, including morphology and simple sentence patterns. However, there is still significant room for further exploration both by developing the techniques for learning syntax to allow for a more complete range of sentence patterns, and by developing methods for a computational understanding of semantics.

5.1 Other Sentence Structures

In addition to existing sentence structures and constructs, the syntax program can learn other common constructs. Although some may be understandable at present, others may require some additions to the program to fully understand. Below are some examples of other simple sentence constructs and how the program would interpret them.

"The man has a blue hat."

Here, the only unknown word with present knowledge is "blue". The program at present would interpret it using the word-context method, resulting in a type of "abnoun bnoun". Continuing in this manner would quickly lead to complications, however, so the program could be extended to understand words based on the form of types which are not nouns or verbs. The word directly before blue has the type "bnoun", so "blue" would be interpreted in this manner as a noun, resulting in two nouns in a row. The program could additionally be edited to interpret two like words in a row as a "noun phrase" or "verb phrase", which would differentiate adjectives from nouns and adverbs from verbs in a way more intuitive to the computer.

"The hat is blue."

This sentence presents a different use of an adjective effectively in the place of a noun. Assuming "is" had been previously learned as a verb, this sentence would be more-or-less

readily understandable as "blue" is still interpreted as related to nouns. This introduces another common sentence construct as well, namely that some nouns can appear directly after verbs without a "bnoun" word in between. Here, it may become worthwhile to add some indicator to new nouns about whether they can appear directly after a verb or not.

"The man has one hat."

This sentence would introduce numerical words, which would be readily understandable as "bnoun" words similar to "a" or "the". This is sufficient and accurate for most cases, though as the program expands into semantics this construct may require more specific definition.

"The man has four hats."

Here, the concept of plurality is introduced. This sentence would simply be another example of the above sentence with regards to syntax alone—the word "hats" would just be considered a new noun. However, this would likely be the most problematic concept regarding semantics. Without an external concept of meaning or some other indication, plurality would have to be learned simply by similarity at the word level. In many cases this would be sufficient, such as "hats", but some words do not follow standard plurality rules such as "mice" versus "mouse". Similar issues apply to verb tenses, though here the rules are even less standardized. This issue could be at least partially remedied by creating an artificial semblance of external understanding, though this would likely prove difficult as well.

Many other sentence constructs are built from these sorts of basic patterns. For example, another common sentence construct involves prepositions such as the sentence "the man threw the hat in the trash". Assuming prior knowledge of the nouns present, the program could interpret "in" as verb-like, appearing between two noun phrases. This is, again, a sufficient interpretation in most cases. A marker indicating what would effectively be two full

constructs could be implemented as well. Embedded clauses would be interpreted similarly, where the program would find the pattern of the outside clause and then the pattern of the inside clause. An understanding of punctuation would make such sentences more easily interpretable.

Despite some issues, the program can, currently or with only minor modifications, understand many common sentence constructs using the two main forms of learning and still a fairly small amount of initial information. The program proves fairly versatile, though of course not at the full level of human understanding.

5.2 Semantics

Semantics is the meanings of words or sentences, not only determined by direct definition, but also by connotation and context. To gain a computational understanding of semantics in a given language, without external representation for words, we have devised an associative approach. For a noun, different adjectives and verbs can be used when employing the noun in a sentence. The particular set of adjectives and verbs for that noun are representative of the nature of the object. A second noun will have some shared verbs and adjectives, and the degree to which this is true will indicate the similarities of the two objects. Conversely, a verb would be understood based on the different objects associated with it, especially the order of the nouns, i.e the subjects and objects used with that noun. From a set of sentences, a threshold of similarity could be experimentally determined, which would result in the categorization of the different nouns and verbs. The sequence of sentences could be as follows:

1. The man wears the big hat.

2. The man throws the small hat.

("hat" is known to be something that can be "big", "small", "worn", and "thrown")

3. The man throws the big ball.

4. The man bounces the small ball.
   ("ball" is like "hat" in having the ability to be "big" or "small" and can be "thrown", however it has not yet be found that it can be "worn", and "hat" has not be shown to be able to be "bounced")

5. The man wears the big shoes.

6. The man throws the small shoes.
   ("shoes" is then understood as the same type of object as "hat")

7. The man uses the telephone.

8. The man answers the telephone.

Let us stop here because the fundamental logic of the sequence of sentences can now be seen. Using these characteristics of nouns, a complex associative web would be formed, where objects have meaning based on their relation to a set of other objects, which also have meaning in relations to another set objects that the first may not. This web may be represented visually by a venn diagram similar to the one in Figure 10.

For the verbs "wear", "throw", "bounce" etc., the program would interpret the above sentences as suggesting that only "man" can conduct these actions, but cannot be the recipient of these actions, while certain nouns can be the recipient of these actions. Further, it should be noted that the training sentences do not require a strict order because a characteristic of a particular noun would be stored until another noun was found to have the same characteristic, and the two nouns would then share a link of the web. Similarly, a verb would have tentative subjects and objects associated with it until more were found. Nevertheless,

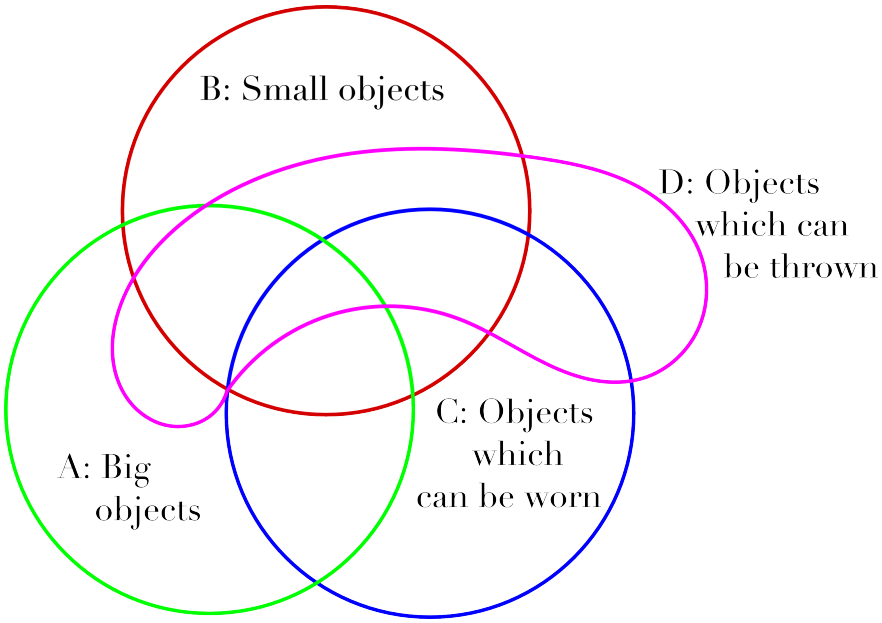


Figure 10: Sample semantic web

this method is limited by the requirement of large amounts of sentences in order to gain any significant understanding of a given word.

To understand a concept such as time, which is essential for semantics and acknowledging tenses, the program would require some initial specification, as well as using a time indicator in all sentences referring to the past or present. It would start with the conditional parameter that if a sentence has the word "yesterday", "ago", "tomorrow", "later", "past", or "future", then a time different from the current moment is being referenced, and the verb used is similar to another verb (the present form, which is the first form the program learns), but with a slightly different morphology. Yet, a problem arises because "last" and "next" applied in "last/next week" or "last/next month" cannot be used strictly as indicators of time, as they can be used in other contexts, such as "He ate the last cookie in the jar" or "He is next in line at the supermarket". Thus, in certain cases, time would present a difficulty for the semantic understanding of the program. The same applies for the use of negation, as in the

sentence "The man is not big", whereby "not", "never", "none", "no longer", and "no more" would have to be explained prior and would result in the noun or verb being categorized as different from other nouns and verbs with shared associative terms.

In addition, the associative method only allows for a definitional understanding of a given word, which can be substantially limited due to the possible changes in the meaning of a word as a result of its context. To allow a computer to have an understanding of context, a system could be implemented to keep track of previously learned information. Cognitive scientist Marvin Minsky suggests that a series of "frames" which represent generalized situations can be used to represent cognition computationally. These general frames could then be modified with situation-specific variables [15]. This idea could prove useful in natural language processing to give a computer an understanding of context. For example, if a series of sentences read "John is playing soccer. He kicked the ball," the program would be able to select a general frame which it could use to keep track of relevant variables, such as "John" being the subject of this action—hence linking this to the "he" in the next sentence.

Another issue that might arise is the issue of connotative meaning of words rather than merely denotative meaning. This is also related to the idea of symbolism, another element of language which can prove difficult for a computer to understand. Here, a method similar to the associative approach above could be implemented after the initial denotative associations were formed. Here, the training sentences would be ones using symbolism rather than literal ones as above. If a word is suddenly associated with a word which is not within the proper categorization, such as "a heart of stone," it could be interpreted by the computer as a connotative association. This would allow the computer to examine characteristics related only to one or the other, hence gaining an understanding of the symbolic associations of words.

6 Conclusion

Using certain principles of language, we have designed a novel method by which a computer can gain an intuitive understanding of language rather than simply an artificial understanding. We have developed techniques by which a computer can learn and analyze the morphology of any given language, and hence understand differences between two languages. We have also developed a recursive learning system for understanding sentence patterns and constructs, which uses a minimum of initial information. At present, the program can interpret many basic sentences, and we have also provided possibilities and suggestions for extending the capabilities of the program. This approach is unique compared to common natural language processing systems because of this lack of need for significant initial input and its recursive design, and could have great potential in the field of natural language processing.

7 Acknowledgements

We would like to thank our mentor, Jorge Roman, for his help in designing a practical method for understanding the morphology of language, as well as his coding suggestions. We would also like to thank our teachers, Jocelyne Comstock and Jeff Mathis, for their help. Finally, we would like to thank all those involved with the Supercomputing Challenge for their extensive work in organizing this program.

References

[1] E. Reingold. The Turing Test. University of Toronto Department of Psychology. Available at `http://www.psych.utoronto.ca/users/reingold/courses/ai/turing.html`.

[2] Learn More About Siri. Apple Inc. Available at `http://www.apple.com/iphone/features/siri-faq.html`.

[3] LingPipe. Alias-I. Available at `http://alias-i.com/lingpipe/`.

[4] D. Klein and C.D. Manning. Accurate Unlexicalized Parsing. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, Vol. 1 (2003), 423–430.

[5] M. Bansal and D. Klein. Simple, Accurate Parsing with an All-Fragments Grammar. In: *Proceedings of the 48th Annual Meeting on Association for Computational Linguistics* (2010).

[6] E. Charniak. Statistical Parsing with a Context-Free Grammar and Word Statistics. In: *Proceedings of the Fourteenth National Conference on Artificial Intelligence*. AAAI Press/MIT Press, Menlo Park, CA (1997), 598–603.

[7] C.M. Powell. From E-Language to I-Language: Foundations of a Pre-Processor for the Construction Integration Model. Oxford Brookes University (2005).

[8] R. Morelli. The Vigenere Cipher. Trinity College Department of Computer Science. Available at `http://www.cs.trincoll.edu/~crypto/historical/vigenere.html`.

[9] H.H. Chen and Y.S. Lee. Approximate N-Gram Markov Model Natural Language Generation. National Taiwan University Department of Computer Science and Information Engineering (1994).

[10] Dr. Seuss. *Fox in Socks*. Available at `http://ai.eecs.umich.edu/people/dreeves/Fox-In-Socks.txt`.

[11] M.H. Christiansen and N. Chater. Constituency and Recursion in Language. In: M.A. Arbib. *The Handbook of Brain Theory and Neural Networks*. 2nd ed. MIT Press, Cambridge, MA (2003), 267–271.

[12] B.C. Lust. *Child Language: Acquisition and Growth*. Cambridge University Press, Cambridge, UK (2006).

[13] The Charter of Fundamental Rights of the European Union. European Parliament. Available at `http://www.europarl.europa.eu/charter/default_en.htm`.

[14] Digraph Frequency. Cornell University Department of Mathematics. Available at `http://www.math.cornell.edu/~mec/2003-2004/cryptography/subs/digraphs.html`.

[15] M. Minsky. A Framework for Representing Knowledge. In: J. Haugeland, editor. *Mind Design*. The MIT Press, Cambridge, MA (1981), 95–128.

Duel of the Fuel

New Mexico Supercomputing Challenge

Final Report

_____

Team #41

Edgewood Elementary

Team Members

Ethan Hintergardt

Chase Podzemny

Emily Robinson

Keith Stevens

Pete Talamante

Teacher/Sponsor

Carol Thompson

Jennifer Wiggins

Team Mentors

Wayne Bitner

Joaquin Roibal

Table of Contents

Executive Summary

Our project was based on the Kirtland Air Force Base fuel spill and how microbial life forms could assist in the clean-up. Our hope was that microbes would consume the contaminate EDB (ethylene dibromide,) which is an anti-knocking additive that used to be in jet fuel. The EDB leaked into the ground in the 50's and 60's, before it was discovered to be a carcinogen. To test the clean-up we used StarLogo, an agent based program that allowed us to simulate different scenarios, with various circumstances, and produced realistic results. We focused our project only on EDB, at one test well. We did not consider the size of the plume, the flow rate, or any other chemical. Our models represented a charcoal filter and additives such as lactate or oxygen, but we also modeled using no additives at all. We researched lactate and oxygen, making models with these two additives. We also tested if the clean up could be done without any additives, which is called natural attenuation. We hypothesized that with the additive lactate, the EDB would break down the fastest. We thought the EDB would break down the slowest with no additives. For each model we timed how long it took for the EDB agents to disappear. To make our model realistic, we decided that every clock cycle should be considered a day. We used Microsoft Excel to make graphs of our results. We found that our hypothesis was correct and that the lactate was most efficient and natural attenuation was the least effective. To further develop our project after interims, based on the judges' suggestion to combine two of our models, we made a new model, pump and treat, so that we could learn something interesting.

Introduction

The purpose of this project is to see which method would clean up EDB (ethylene dibromide) at the Kirtland Air Force Base jet fuel spill most efficiently. The options we studied were natural attenuation, a charcoal filter, and additives such as lactate and oxygen.

We chose this subject because it is a local problem, which threatens Albuquerque's drinking water supply. We had local resources to help us. We used the KAFB website extensively for maps, data, and information. We were very fortunate to meet Mr. Wayne Bitner, Chief of Environmental Restoration at KAFB, and discuss the spill in person. He helped us to better understand the problems.

The problem's significance to us is that Albuquerque's water comes from the aquifer that could be contaminated by the jet fuel, and most importantly, EDB, a known carcinogen. Now, even though we don't live in Albuquerque, our friends and relatives who live there will be affected.

To prepare ourselves for this project, we researched microbes, aquifers, and fuel spills. Here are the results of this research:

EDB and the Fuel Spill

The research showed that there have been a lot of oil spills all over the world and in many cases contamination got in to the ground water. The one that leaked from Kirtland had EDB (ethylene dibromide), which has not been legal in the USA for a long time. EDB was used in the jet fuel as an anti-knocking substance. The jet fuel that spilt from Kirtland has not reached Albuquerque's ground water supply yet. The Air Force has placed monitoring wells across the city to monitor the spill.

The spill has leaked an estimated 8 million gallons of jet fuel in to the ground. It has been leaking since the late 1950's. Some people dislike what they are doing to try to clean it up. Some people say that the drilling that they do is loud. From websites that I have looked at it says that even a sniff of the EDB can be very harmful to your body. EDB can cause cancer and other side defects that will harm your body. Some of the side defects are breathing difficulties, skin itching and rashes, eye irritation and burning, and coughing and throat irritation.

The fuel has not reached the monitoring wells yet. They estimate that it is 500 feet below ground. Kirtland Air force base says that they are willing to spend whatever it takes to clean it up. From my research it says that the jet fuel is only 1.6 miles away from drinking water wells.

Aquifers

Do you ever wonder where your water comes from? They come from aquifers. We will tell you what an aquifer is!

An aquifer is an area underground where water is. Aquifers can occur at various depths. They can be 5 feet down, or 200 feet down. A lot of the time aquifers can have clean water. But sometimes they do not. There are two different types of aquifers with clean or unclean water.

There are saturated and unsaturated aquifers. Saturated aquifers have clean water usually and their atmospheric pressure is bigger than the water head. The definition of a water table (which is like an aquifer) is that the water head pressure is the same as the atmospheric pressure.

Unsaturated is when above the water table the gauge pressure is negative (absolute pressure can't be negative but gauge pressure can) and the water that doesn't fully go into the pores, goes into a suction (a suction is when water goes into a funnel.)

Many aquifers that are close to the surface get a lot of their water from the rain fall. The ones that are underground mainly get a lot of their water from lakes that go deep downward. Lots of desert areas such as New Mexico can have water resources near silicon and sandstone. The underground water can also be in underground rivers as in underground caves. This has a chance of happening near eroded silicon, which only makes a small percentage of the Earth. Usually this can be like a kitchen sponge; it sucks saturated water into holes, which can be used for water.

We depend on groundwater as much as we depend on air to live. Our fresh-water aquifers (the ones that aren't contaminated) can get water from salt-water. This could turn out to be a serious problem. We cannot drink salt-water so if it (or other contaminates) gets into our water supply, we can cause some serious damage. This can also be a big problem near the ocean, where pumping aquifer water is excessive.

The aquifers that are close to the ground can be used for watering crops. A lot of the aquifers for use of crops are near the surface and are fresh-water aquifers. Also, aquifer depletion is a big concern.

When fresh-water aquifers are near the ocean there is salt-water that leaks into the aquifer. However, salt-water is denser than fresh-water so the salt-water goes downward. So, it makes it easier to get the fresh-water out but you can still have the chance of pumping up salt-water. One of the largest underground aquifers is the Great Artisan Basin. This aquifer provides a lot of water for Queensland and parts of South Africa. This aquifer is 1.7 million KM in area.

Microbes

Have you ever wondered if there are living things inside your body other than cells or dust mites? Well there is. There are microbes too!

Microbes are tiny organisms that can only be seen with a microscope. Some microbes are even *submicroscopic* and can only be seen by special electron microscopes. Microbes have the ability to live in extremely harsh environments of intense heat or extreme cold. Some microbes are referred to as germs or bacteria and can make you sick while others are essential to all life. Surprisingly there are several billion more bacterial cells in the human body than there are human cells. The bacterial form of microbes is vulnerable to Clorox wipes, hot water, air freshener, and to most disinfectants.

Microbes are used for many things. Sometimes they are used at water treatment plants to remove contaminates that cannot be left in water. Another use for microbes is to help clean up fuel spills. The microbes will quite literally eat the fuel. Certain types of microbes can live longer than a human! When microbes die, carnivorous microorganisms will come and eat the dead ones. Other types of carnivorous microbes will eat living microbes.

Microbes can be separated into five different groups; Archae, Bacteria, Fungi, Protista, and Viruses. Archae are bacteria look-alikes that are living fossils and are evidence of the very first living things on earth. Bacteria are often dismissed as germs but some are very helpful. An

example of how Bacteria help us is they support the atmosphere or eat types of dangerous garbage. Fungi can be the size of a grain of yeast to a 3 and a half mile wide mushroom and is useful for decomposing waste. Protista is a plant-like algae that produces much of the oxygen that we breathe. Viruses are unable to do much of anything on their own, but they will attack host cells, wreak havoc, and cause diseases. From my research I have discovered that microbes can be both helpful and unhelpful as well as being both essential for life and deadly enough to cause disease or kill.

Description

Our project was chosen because it was a local problem that was threatening Albuquerque's water supply. We wanted to see which additive (lactate or oxygen) would do the best job assisting microbes to clean up the EDB in the Kirtland Air Force Base jet fuel spill. We researched KAFB for information about our project. We researched oil spills, microbes and aquifers. We modeled EDB (Ethylene Dibromide) in the water table and how we could use different ways to decrease the levels of EDB using several types of models like a charcoal filter, and additives such as lactate, oxygen, and natural attenuation. For every model, we used data from KAFB well number 106076. There are 360 µg/L EDB, and 60 µg/L oxygen at the site. We are modeling one liter of water in our models, with the exception of our pumping models. We focused our project only on EDB, at one test well. We did not consider the size of the plume, the flow rate, or any other chemical. The materials we used were the programs StarLogo, Excel, various search engines, poster boards, and Legos.

Natural attenuation means letting nature do the clean up. In this model we used 360 EDB agents, 100 microbe agents, and 60 oxygen agents. When the microbes collided with EDB, the EDB agents disappeared and the microbes gained five energy. When the microbes have over ten energy they hatch, creating more microbes. When the microbes collide with oxygen they gained ten energy. Microbes died when they ran out of energy.

The oxygen model increased the number of oxygen agents from 60 to 180. Like every model, it had 100 microbes. The oxygen assisted the microbes in cleaning up the EDB, because when the oxygen agents collided with the microbe agents, the microbes gained energy and hatched, or

reproduced, so that there were more agents to consume the EDB. When the microbes collided with the EDB, the EDB died. When the microbes ran out of energy, they died.

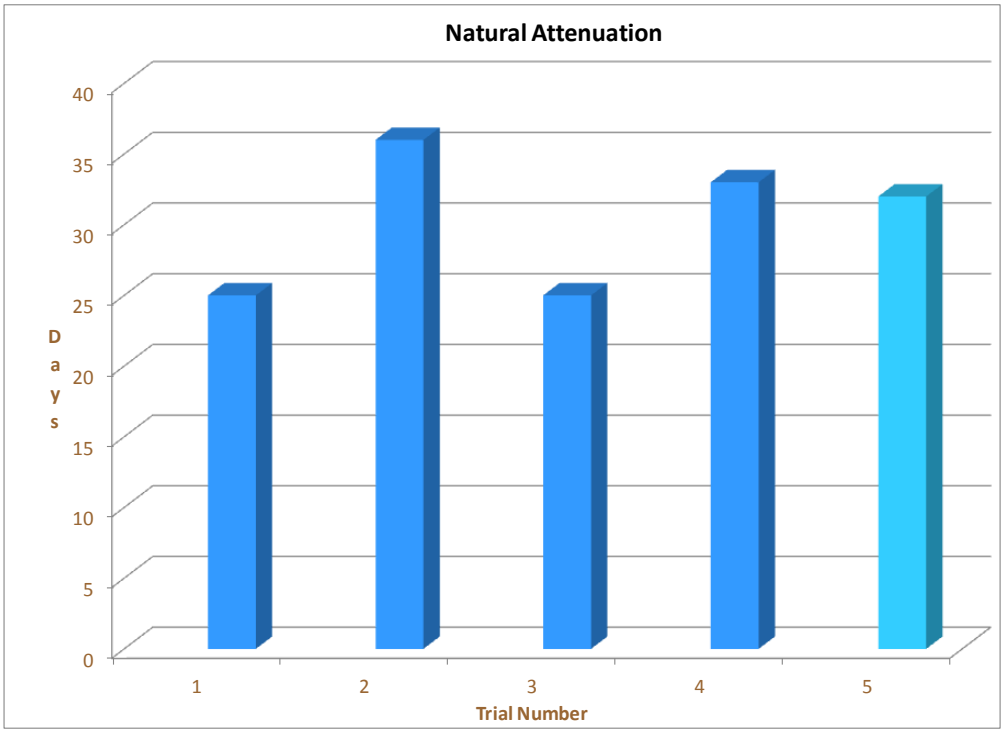
The lactate model had 250 lactate, 100 microbes, and no oxygen agents, because it was an anaerobic model. When the lactate collided with the microbes, the microbes gained energy, reproduced, and died.

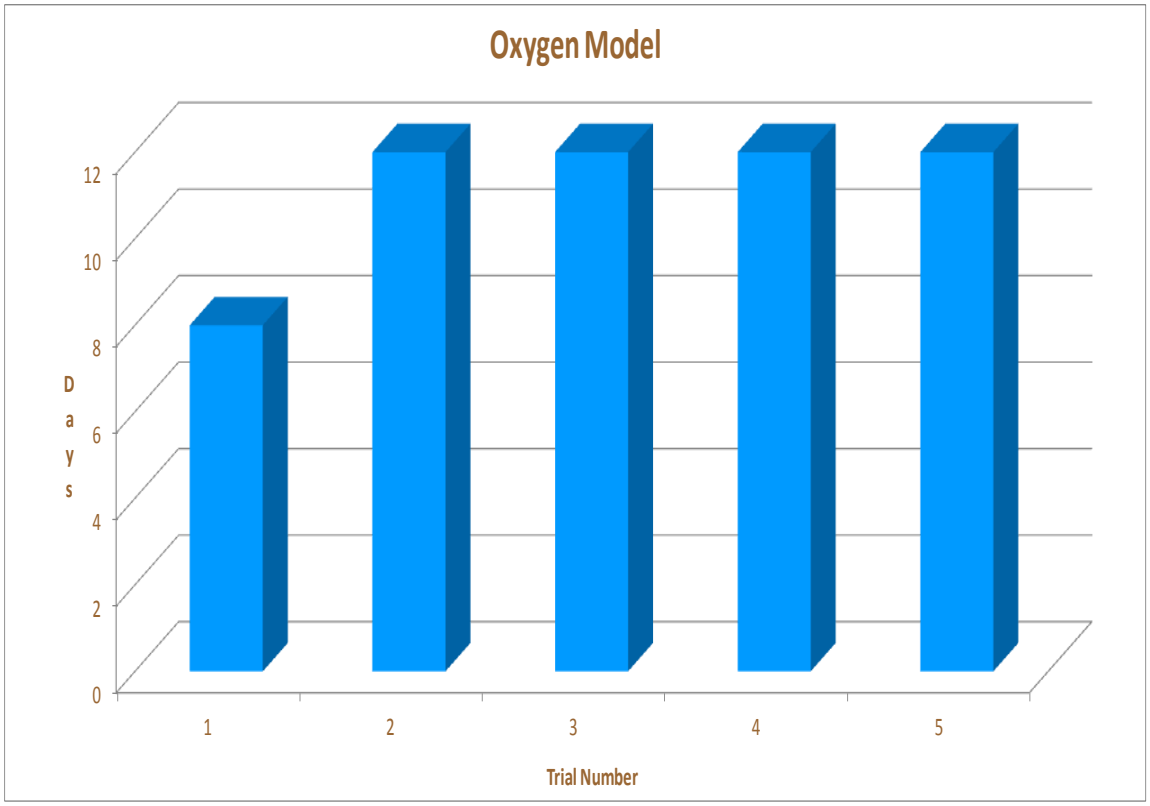
The charcoal filter model is a pump and treat model where EDB-contaminated water is pumped through a charcoal filter, cleaned, and then returned into the ground. In this model, there were 1370 agents total consisting of 1000 water agents, 360 EDB agents, and 10 microbe agents. When the microbes collided with an EDB agent they consumed it but otherwise the EDB moved with the water, by gravity, to the bottom of the pump and were pumped up. After that, the EDB agents disappeared and the water was pumped back into the ground. Then they lost energy. When the microbes collided with EDB, the EDB died.

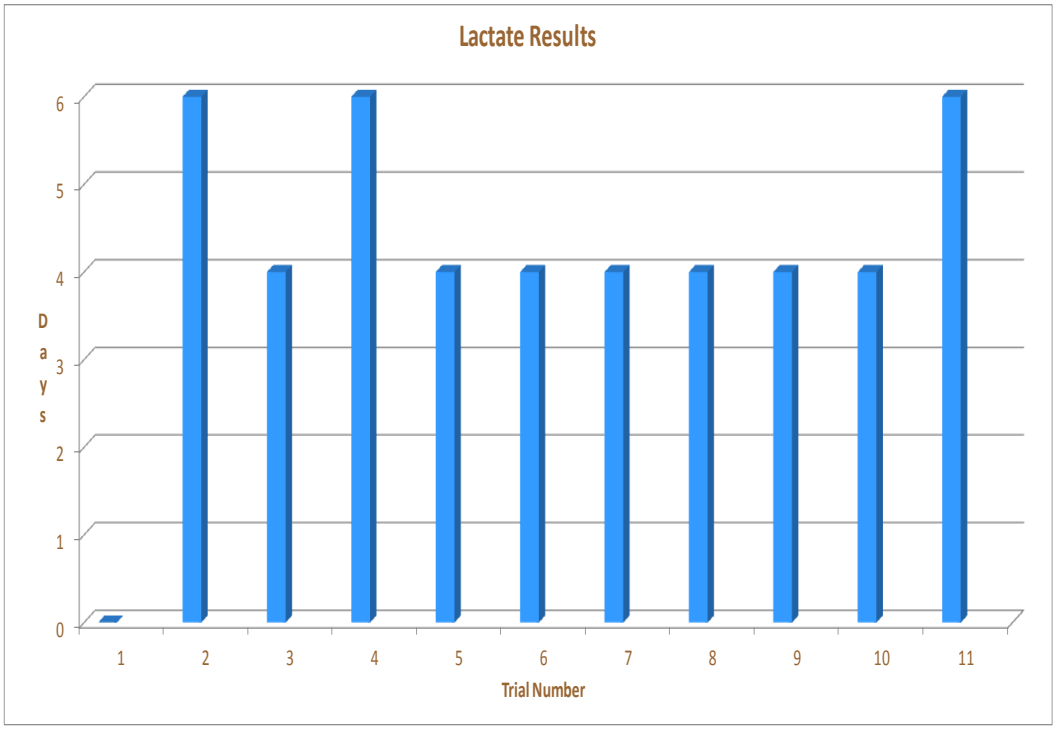
The pump and treat model has two wells that pump oxygen into the ground on either side of a charcoal filter that pumps the water. The microbes get an oxygen energy boost to help clean up the EDB. The pump and treat model used water, filters, oxygen, and microbes and they all work together to do the job. The pump and treat model has 3 sliders for different variables. The first slider is for how much oxygen we put into each trial, the second slider has microbe aggression, which is how vigorously microbes hunt EDB. Our purpose for this slider is to calibrate the model. The last slider is for how much energy microbes need to reproduce. When the microbes collide with EDB they gain one energy and when they collide with oxygen they gain a random five energy which means they gain anywhere from one to five energy. We manipulated the sliders to see which setting works best. While running this model, we found that EDB agents were getting trapped on one side or the other, so we added another injection well to improve the water flow.

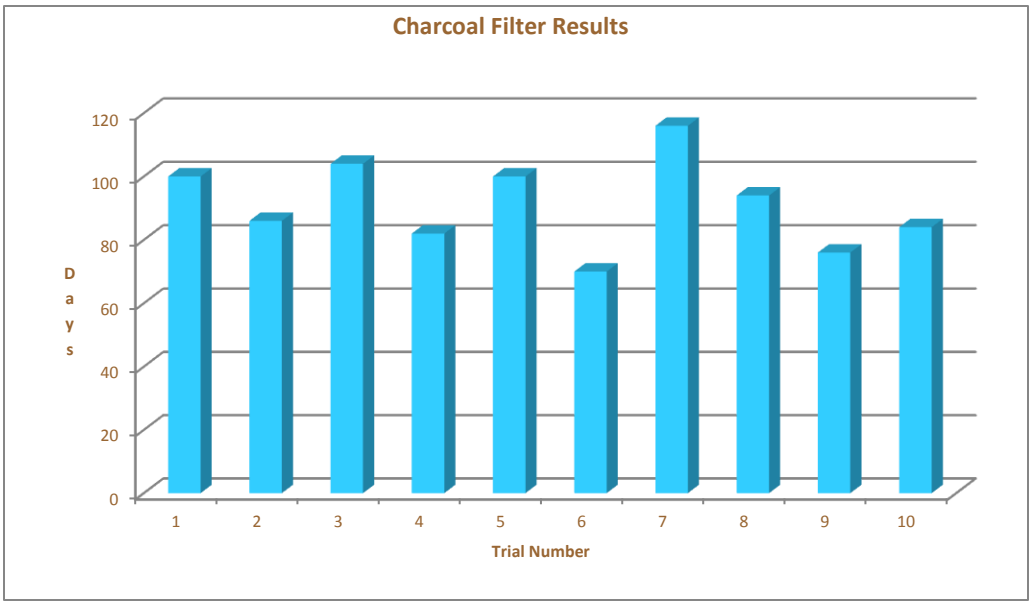
Results

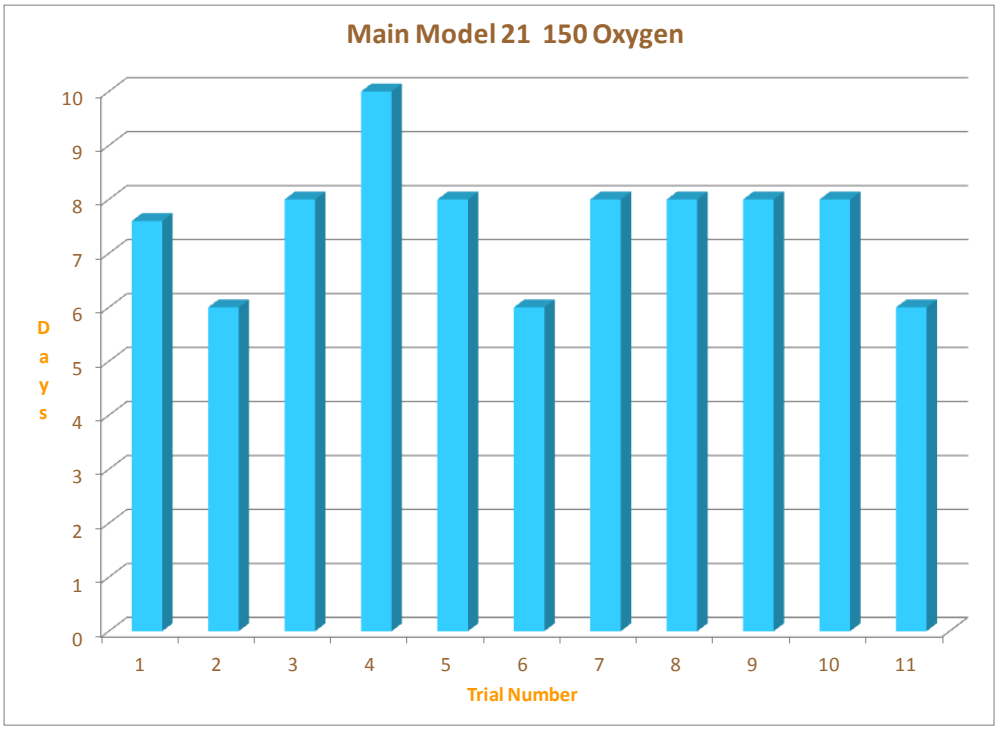
We ran each model 5-10 times. The results for the natural attenuation model were that the EDB was never completely removed from the groundwater; we found out from this model that natural attenuation was the slowest technique for this problem. On average, there were 30 EDB agents left after each trial. This is realistic because you will never remove all of the contaminants from the ground completely. The results for the lactate model were that the EDB disappeared in an average of five days. This sounds like a very short time, but we modeled the amount of EDB in only one liter of water. We learned that lactate is anaerobic, which means that it worked better without oxygen. The results for the oxygen model were that it took an average of 11 days for EDB to be removed from one liter of groundwater. The charcoal filter model took an average of 91 days for all of the EDB to be removed from the groundwater. The results for the pump and treat model were an average of seven days for all the EDB to be removed from the groundwater.

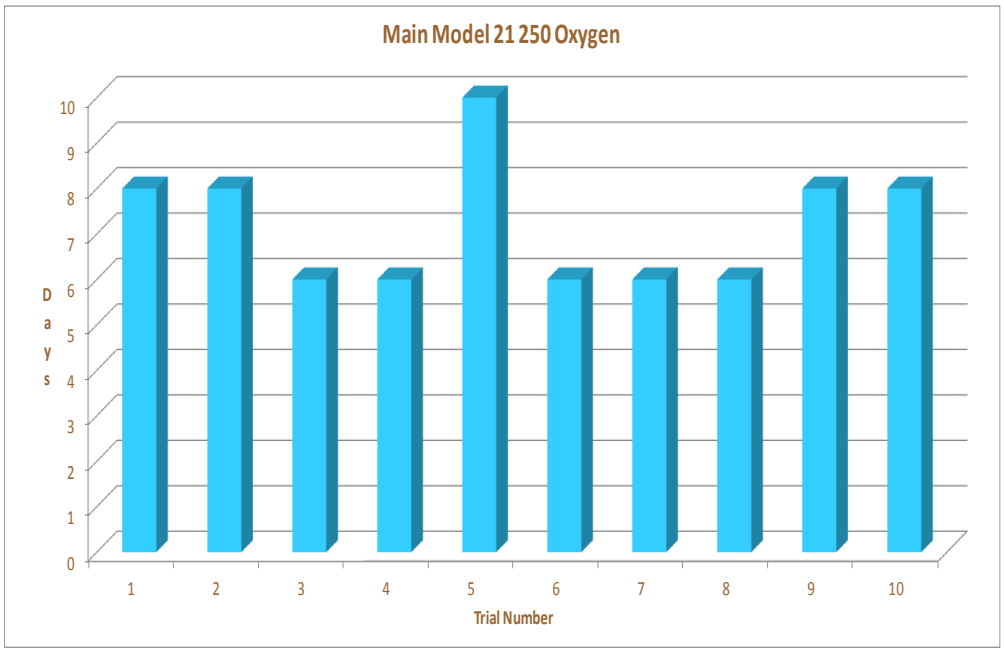


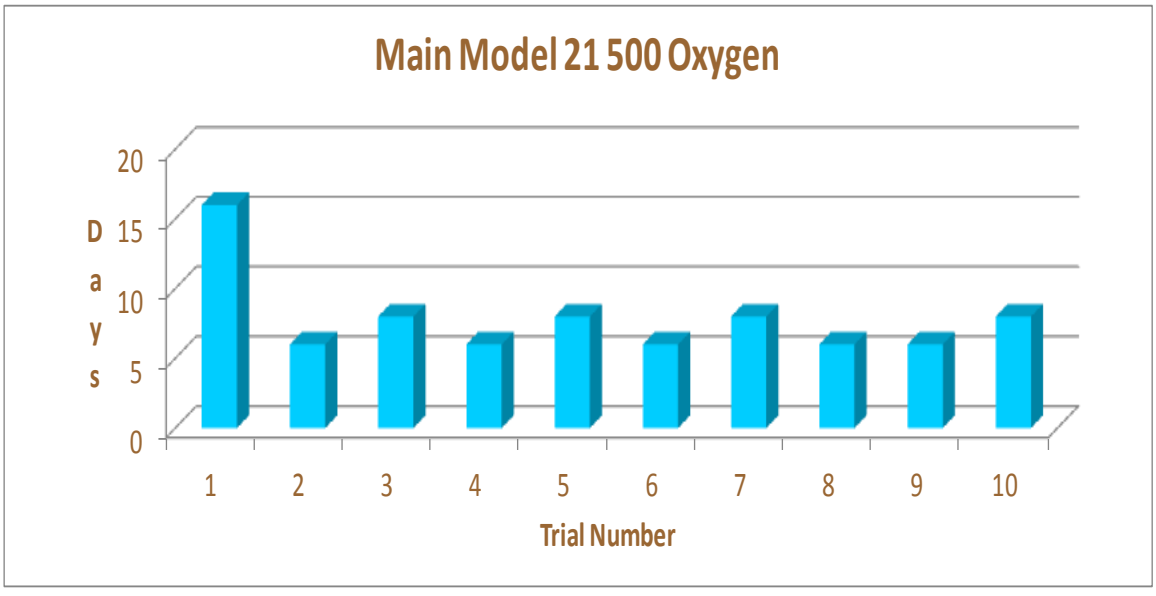

Oxygen Model
Days
0
2
4
6
8
10
12
1
2
3
4
5
Trial Number

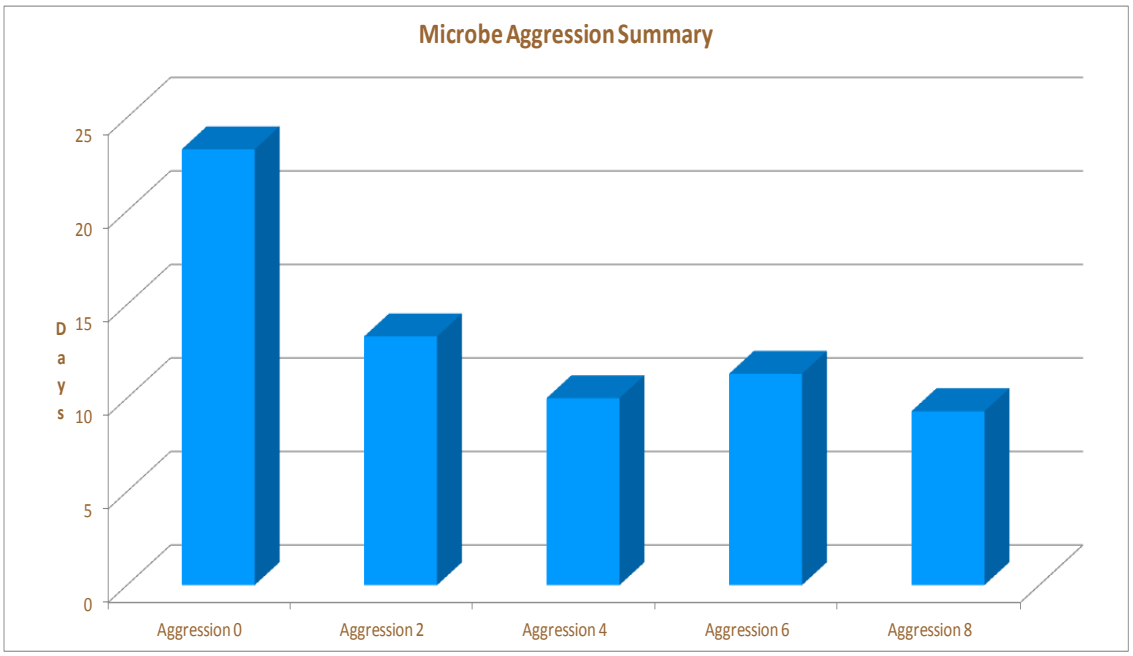

Lactate Results
Days
0
1
2
3
4
5
6
1
2
3
4
5
6
7
8
9
10
11
Trial Number

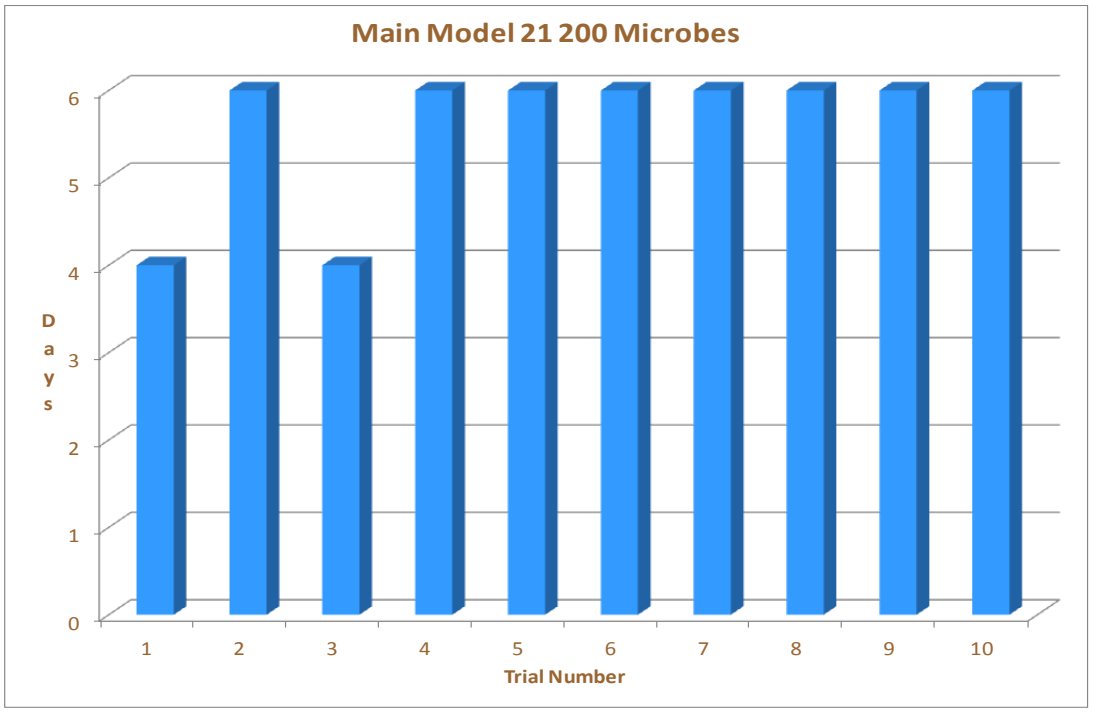

Charcoal Filter Results
120
100
80
60
40
20
0
Days
1
2
3
4
5
6
7
8
9
10
Trial Number


Main Model 21 150 Oxygen
10
9
8
7
6
5
4
3
2
1
0
Days
1
2
3
4
5
6
7
8
9
10
11
Trial Number


Main Model 21 250 Oxygen
Days
0
1
2
3
4
5
6
7
8
9
10
1
2
3
4
5
6
7
8
9
10
Trial Number


Main Model 21 500 Oxygen
Days
0
5
10
15
20
1
2
3
4
5
6
7
8
9
10
Trial Number


Microbe Aggression Summary
Days
25
20
15
10
5
0
Aggression 0
Aggression 2
Aggression 4
Aggression 6
Aggression 8


Main Model 21 200 Microbes
Days
6
5
4
3
2
1
0
1
2
3
4
5
6
7
8
9
10
Trial Number

Conclusion

The conclusion for this project is that the hypothesis was correct because the lactate took, on average, five days to be removed from the groundwater. The lactate idea would work the best in anaerobic conditions. Although the lactate worked the quickest, there was a problem. We learned from our mentor, Mr. Bitner, that lactate will only work in soil and not groundwater, making it ineffective for our purposes. After Interims, based on the judge's suggestions, we created a model that combined the charcoal filter model and the oxygen model to make a pump and treat model. This pump and treat model gave us results almost as good as the lactate model's results. We found that adding more than 250 oxygen agents did not significantly improve our results. These results were unexpected. We found that adding a hunt procedure to the microbes improved results, but increasing microbe aggression did not make a significant difference.

In all our research there is no hard evidence that microbes are able to break down EDB in ground water. We think that pumping and filtering is going to be an important part of the solution. Lactate is very encouraging in its ability to break down EDB in soil. More research needs to be done on how it can be effective in ground water. Our sources at Kirtland Air Force base report that the microbe species Dehalococcoides might degrade EDB over time.

Recommendations

As for recommendations, there are so many ways to go with this project it would be hard to list them all. We only looked at EDB, in one liter of water. We didn't look at any other chemical, any other amount of water, the way the water moved, etc. Our test well was initially the highest concentration of EDB, but during the third quarter, it shifted from 360 µg/L to 180, and a well 1/10 of a mile to the northeast changed to 370 µg/L EDB. If we were to continue working on this project we might want to look at how EDB concentrations change over periods of time. We could also investigate how the geology around the spill affects the fuel.

Unfortunately, after we had already gotten the results from our first models, we found out that lactate wouldn't work in groundwater. Although it wouldn't work where we needed it

to, it does work in soil, so if we took the project any further, we could study how to move the EDB into soil, where lactate could break it down. We would want to do this because the fastest way of cleaning up EDB was done with lactate.

Acknowledgements

- Wayne Bitner, Chief of Restoration at KAFB
- Joaquin Roibal, Student at NM Tech
- The people at UNM from our fieldtrip
- Stefan Bocchino, Public Relations KAFB
- JP Gonzales, Santa Fe Institute, GUTS
- The Supercomputing Challenge: David Kratzer, Celia Einhorn, and Betsy Frederick for organizing all of the great events we have attended this year.
- Our Parents: Bird Podzemny, Doug Podzemny, Pete Talamante, Laura Talamante, Dana Mitchell, Charles Mitchell, Riley Robison, Angela Robinson for transportation and supporting us.
- Our Teachers: Carol Thompson, Jennifer Wiggins, Samantha Eaton
- Our principal, Nicole Dray, for giving us time to attend activities away from school.

Bibliography

Actionbioscience. "Microbes: What They Do & How Antibiotics Change Them." *Actionbioscience*. Actionbioscience, Jan. 2001. Web. <http://www.actionbioscience.org/evolution/meade_callahan.html>.

Anonymous. "The Mariner Group -- Oil Spill History." *The Mariner Group*. Web. 15 Mar. 2012. <http://www.marinergroup.com/oil-spill-history.htm>.

"Aquifer." *Wikipedia*. Wikimedia Foundation, 03 Nov. 2012. Web. 2011. <http://en.wikipedia.org/wiki/Aquifer>.

ASTDR. "Toxic Substances Portal - Ethylene Dibromide." *Toxic Substances Portal*. ASTDR. Web. <http://www.atsdr.cdc.gov/MMG/MMG.asp?id=1143&tid=251>.

"Aviation Fuel." *Wikipedia*. Wikimedia Foundation. Web. <http://en.wikipedia.org/wiki/Aviation_fuel>.

"Bacteria." *Wikipedia*. Wikimedia Foundation, 03 Nov. 2012. Web. <http://en.wikipedia.org/wiki/Bacteria>.

"Burn Pits - Public Health." *Public Health Home*. Web. <http://www.publichealth.va.gov/exposures/burnpits/index.asp>.

"Decades-old Jet Fuel Spill Sparks Water Contamination Fears in South Bibb County." - *Local & State*. 15 Aug. 2010. Web. <http://www.macon.com/2010/08/15/1229614/south-bibb-residents-fear-contamination.html>.

"Decades-old Jet Fuel Spill Sparks Water Contamination Fears in South Bibb County." - *Local & State*. Web. <http://www.macon.com/2010/08/15/1229614/south-bibb-residents-fear-contamination.html>.

"Decades-old Jet Fuel Spill Sparks Water Contamination Fears in South Bibb County." - *Local & State*. Web. <http://www.macon.com/2010/08/15/1229614/south-bibb-residents-fear-contamination.html>.

"EGeo Services, Inc." *EGeo Services, Inc.* Web. 12 Mar. 2012. <http://www.egeoservices.com/Technology.html>.

Fitzgerald, Matt. "The Lactic Acid Myths." *Competitor.com*. Competitor, 26 Jan. 2010. Web. 7 Dec. 2011. <http://running.competitor.com/2010/01/training/the-lactic-acid-myths_7938>.

Garcia, Eddie. "It Could Take 2 Years to Clean up Jet Fuel Contamination." *Kob.com*. Kob, 19 Oct. 2011. Web. Oct. 2011. <http://www.kob.com/article/stories/s2336566.shtml>.

"Hot Topics:" *Kirtland Jet Fuel Spill Seeps Into Neighborhood*. Web. 2011. <http://www.koat.com/news/27306599/detail.html>.

"Jet Fuel." *Wikipedia*. Wikimedia Foundation. Web. <http://en.wikipedia.org/wiki/Jet_fuel>.

"Killing Us Softly with EDB." *Argyle Bartonville Communities Alliance*. Web. 2011. <http://abcalliance.org/?p=1581>.

Kirtland AFB. "Media Advisory: Kirtland Releases Quarterly Fuel Plume Data Results." *Kirtland AFB*. Kirtland AFB, 10 Mar. 2011. Web. <http://www.kirtland.af.mil/news/story.asp?id=123274506>.

"Kirtland Fuel Spill Ready for Cleanup." *KRQE TV*. Web. 6 Nov. 2010. <http://www.krqe.com/dpp/news/Kirtland-fuel-spill-ready-for-cleanup>.

"The Mariner Group -- Oil Spill History." *The Mariner Group*. Web. <http://www.marinergroup.com/oil-spill-history.htm>.

"Microbes." Web. <http://www.niaid.nih.gov/topics/microbes/pages/default.aspx>.
"The Origin, Evolution and Classification of Microbial Life." *Online Textbook of Bacteriology*. Web. 12 Mar. 2012. <http://textbookofbacteriology.net/themicrobialworld/origins.html>.
"Redirect Notice." *Wikipedia*. Wikipedia. Web. <http://www.google.com/url?q=http%3A%2F%2Fen.wikipedia.org%2Fwiki%2FMicrobial_corrosion>.
"Residents near Kirtland Believe Tap Water Is Contaminated." Web. 2011. <http://www.kob.com/article/stories/S2237756.shtml>.
"Soil & Groundwater Remediation Project." *Hanford Site*. Usa.gov, 8 July 2011. Web. 28 Feb. 2012. <http://www.hanford.gov/page.cfm/SoilGroundwater>.
"Supplemental Content." *National Center for Biotechnology Information*. U.S. National Library of Medicine, Jan. 2003. Web. <http://www.ncbi.nlm.nih.gov/pubmed/12502073>.
United States Environmental Protection Agency. "Ground Water Cleanup at Superfund Site." *EPA*. Environmental Protection Agency, Dec. 1996. Web. Oct. 2011. <http://www.epa.gov/superfund/health/conmedia/gwdocs/brochure.htm>.
USGS. "Groundwater Quality." *Water Science for Schools*. USGS. Web. <http://ga.water.usgs.gov/edu/earthgwquality.html>.
USGS. "Pesticides in Groundwater." *Pesticides in Groundwater*. USGS. Web. <http://ga.water.usgs.gov/edu/pesticidesgw.html>.
"Why Does Lactic Acid Build up in Muscles? And Why Does It Cause Soreness?: Scientific American." *Science News, Articles and Information*. Scientific American, 23 Jan. 2006. Web. 8 Dec. 2011. <http://www.scientificamerican.com/article.cfm?id=why-does-lactic-acid-buil>.

PDF Documents

http://www.kirtland.af.mil/shared/media/document/AFD-100510-060.pdf

http://www.epa.gov/tio/download/citizens/bioremediation.pdf

http://www.epa.gov/tio/download/citizens/mna.pdf

http://www.epa.gov/nrmrl/pubs/600r08107/600r08107.pdf

http://www.ecs.umass.edu/cee/reckhow/courses/697w/papers/Rob1.pdf

http://www.epa.gov/oust/cat/Section_8-Remediation_and_Treatment.pdf

http://www.kirtland.af.mil/shared/media/document/AFD-110930-094.pdf

http://www.epa.gov/ogwdw/pdfs/factsheets/soc/ethylene.pdf

http://lakeland.edu/AboutUs/MSDS/PDFs/543/Jet%20Fuel%20No%203%20%28All%20Brands%29.pdf

http://www.kirtland.af.mil/shared/media/document/AFD-080731-037.pdf

http://water.usgs.gov/wrri/00grants/NMaquifers.pdf

http://www.google.com/url?q=http%3A%2F%2Finfo.ngwa.org%2Fgwol%2Fpdf%2F872944335.PDF&sa=D&sntz=1&usg=AFQjCNEgkJqvtyyPEsiHHQhRaF8-SjqLjQ

http://www.kirtland.af.mil/shared/media/document/AFD-100510-060.pdf

http://www.epa.gov/nrmrl/pubs/540s02500/540S02500.pdf

Newspaper Articles

Albuquerque Journal, Tuesday October 4th, 2011. Pages A1 and A2.

Special thanks to abqjournal.com and Kirtland Air Force Base for the map of the oil spill that we used all throughout the project!

Warehouse Layout and Picking Simulation

New Mexico

Supercomputing Challenge

Final Report

April 4, 2012

Team 64

Los Alamos High School

Team Members:

Sudeep Dasari

David Murphy

Colin Redman

Teachers:

Lee Goodwin

Mentors:

Elizabeth Cooper

Executive Summary:

This project explores what factors contribute to increased order picking time in a specific warehouse layout. Warehouses are extremely complex and as a result an difficult portion of the supply chain to model. Of all the processes that occur inside a warehouse order picking is probably the most important as it accounts for 55% of all total costs. In order to determine the biggest contributors to inefficiencies in the picking process we created a simulation that can model the picking process in a warehouse. The simulation was an agent based model written in the Java programming language and built with the MadKit platform. Agents represented the various actors present in a warehouse, such as the pickers, conveyor segment, and the warehouse itself. warehouse layout program was also created in order to feed the simulation information related to the warehouses' layout, and a database was used to store item stock, incoming orders, and final results of the simulation. The simulation was subsequently used to determine some interesting relationships between the warehouse layout and the picking times for a pre-determined set of customer orders. The parameters changed included the number of orders allowed to be processed at one time, whether or not the picker was collecting multiple items on a picking route, how the items were distributed (sorted) on the shelves, and after our results were analyzed it was found that there were many parameters that had an influence on the picking time.

Problem Statement:

As e-commerce companies such as Amazon become the dominant way people buy goods, increasing strain has been put on getting an order to a customer as quickly, reliably, and efficiently as possible [2]. A critical part of this supply chain is the logistics of the warehouse that these online companies use to store products until they are purchased, picked, and shipped. Inarguably, the most important function in a warehouse is the picking process, which accounts for 55% of operational expenses [1]. Since under-performance in picking can lead to bad overall effectiveness and high operational costs, it is imperative that a company's warehouse is organized in such a way that it allows for the most effective order picking possible [3].

The basic principle behind order picking is that a worker, called a "picker", receives an order and picks its components off of storage shelves, after which he or she deposits the items in a tote which is placed on a conveyor belt. Figure 1 below demonstrates linear order picking across a shelf.

Figure 1- Depicts a picker moving along a pick line to pick items x and y.

There are two basic implementations of order picking, zone and discrete order picking. "In discrete order picking a single worker walks to pick all the items necessary to fulfil a single customer order" (Eisenstein, 2006). Discrete order picking is demonstrated in Figure 2. As you can see, a picker is moving throughout the warehouse

picking the various parts of an order before returning to a depot to return the fully picked order. The picker then gets the next order and repeats the process.

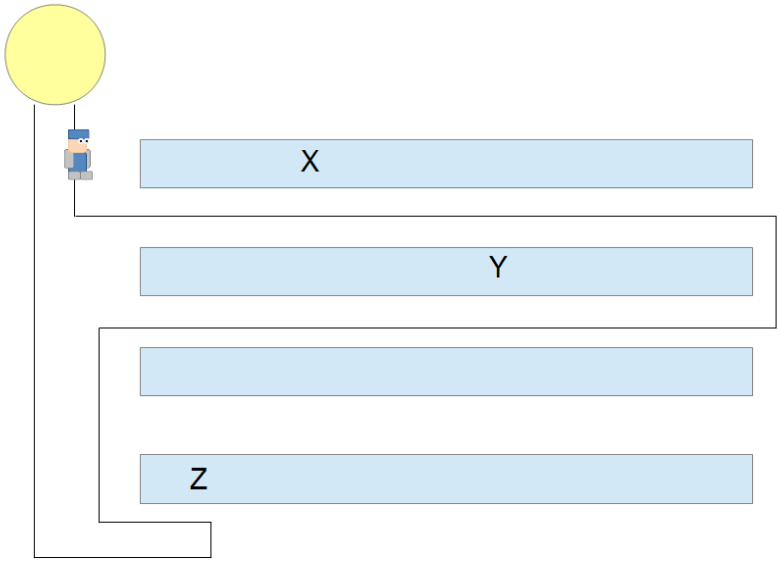


Figure 2- Depicts discrete order picking implemented in such a way that the picker starts at the depot, picks items x, y, and z, and then returns to the depot to get the next order.

In zone picking, each picker is assigned a zone and he or she picks all the items of a set of orders that occur in his or her zone. Once all the items in a zone are picked, they are put on a conveyor in units called totes, before being batched together into their separate orders and sent to a shipping area [1], [3].Figure 3 shows a picker picking all the various items in an order that fall in his row before depositing them on a conveyor.

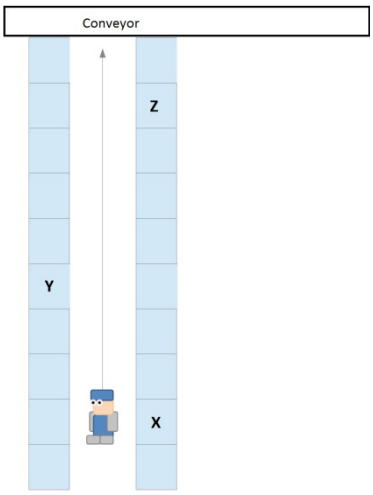


Figure 3- Depicts a picker moving in his zone, picking items x, y, and z, and then depositing them on a conveyor.

While discrete order picking has many advantages, mainly reliability and simplicity, zone picking is definitely the better organizational scheme in a warehouse due to significantly decreased travel times on smaller orders [3]. Simply put, zone picking offers faster pick times than discrete order picking, with the same reliability.

The most important thing to consider in order picking is the distance a picker walks during each order, as this directly correlates with the amount of worker hours a company must pay for [1],[3]. It is possible to cut the distance walked by configuring the layout of a warehouse, and by improving the organization of the warehouse. Configuring the layout of the facility is achieved by modifying the positions of the various conveyors, depots, and shelves. Carefully sorting the items and picker distribution based on item demand can help substantially increase the organization of the warehouse [3].

Problem Solution:

The goal of this project was to create a program that can accurately simulate the picking process in a warehouse. We wanted to find out which factors would have a greater impact on the picking times than others. We have identified several parameters within the capability of our simulation which can be adjusted in behavior space to impact the overall picking times for a set of orders in a specific warehouse layout. These parameters will be more fully explained later in this document.

To summarize, the problem solution was implemented in Java in two distinct applications. First, we developed a graphical layout program which we called the “Warehouse Layout Manager” using Java in the Eclipse Integrated Development Environment (IDE). This program was used to create the various warehouses for testing.

The simulation itself was also coded in Java and it utilized a framework called MadKit[9], a multi-agent platform based on the Agent-Group-Role (AGR) philosophy. We also up a SQL database (using HSQLDB) with tables of items and orders so that these could be kept constant from run-to-run. The output of the Warehouse Layout Manager was saved as a file that was read in by the simulator. The simulator can be run with any number of orders and a set of items stored in the warehouse. The other parameters that can be varied from run-to-run are the way the items are stored on the shelves, how they are sorted onto the shelves and how the items are picked for an order. Although we can simulate discrete picking we concentrated on zone picking for our results. However we could set whether or not the picker fetched more than one item at a time while in the shelves, or if he just picked up one item at a time and took it back to the tote. After the simulation was run we recorded the simulation steps for completing all the orders in the set and the other parameters that were set for the simulation run.

Warehouse Layout Manager:

The “Warehouse Layout Manager” is a program that is used to layout the various components of a warehouse. It was implemented using Swing, Java’s primary Graphical User Interface (GUI) toolkit, along with the Eclipse’s WindowBuilder plug-in. When starting a new warehouse layout it starts with a dialog box to input the warehouse dimensions in grid units (a

grid unit is considered to be a 2' x 2' area in the warehouse). Once the dimensions were entered the dialog box was replaced by a new window which contained a panel of three buttons as well as a grid space where various warehouse components are added. The components are the stop and start segments of a conveyor, conveyor segments (with a flow direction), item stops (special conveyor segments where a tote is ejected for picking), and shelf areas. The grids not filled with any of these objects were treated as open areas for worker movement. More specific instructions on using the Warehouse Layout Manager can be found in Appendix A. The save files utilized the unique T64 file extension which was named after our team number. The layouts were saved in zipped ArrayList objects utilizing the: java.io.FileOutputStream, java.io.ObjectOutputStream, and java.util.GZIPOutputStream. Saved files were read back into the layout manager for editing. Most importantly the saved files could also be parsed by the simulator to set up the simulation.

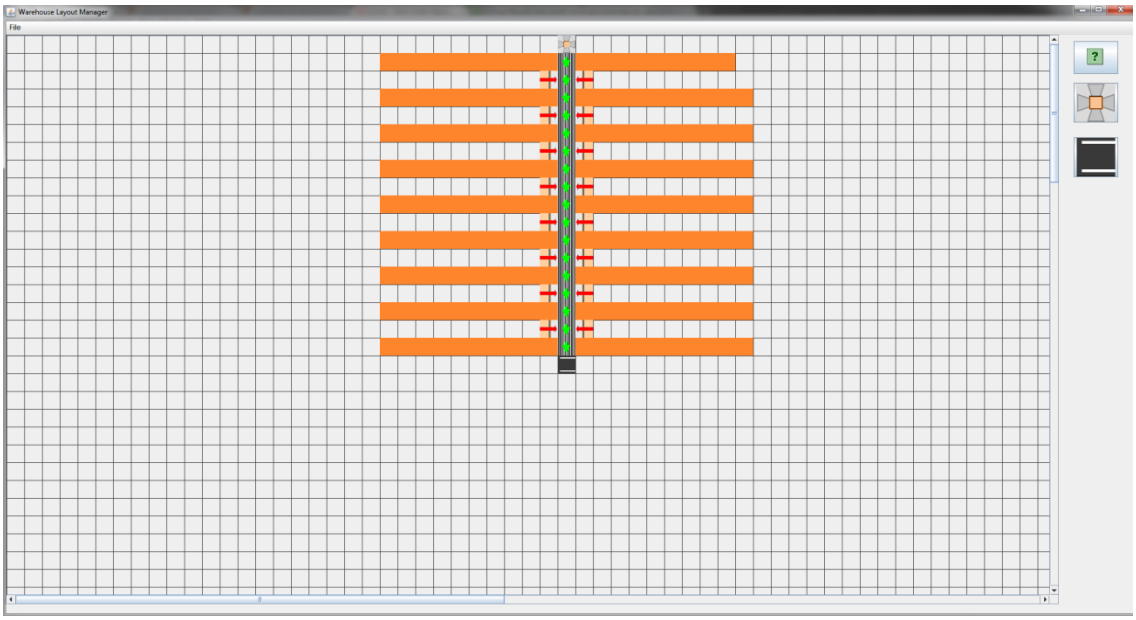


Figure 1 - This is the layout manager window displaying the layout of a demo warehouse. The orange areas are shelves and the graphics with the directional arrows are conveyor segments. Picking stations can be seen off the conveyor located near the shelves.

Database:

The SQL database had two purposes. The first was to contain pre-generated sets of items for the warehouse and orders with sets of items. The second purpose was to record results of the simulation runs.

The database has three table structures. The “item” table contains more than150,000 items, each with a unique id, and their generated popularity. A warehouse simulation used a subset of these items to stock the shelves. For example if a run was set up to use 100 items, the first 100 items were selected from the database. Likewise if a run was set up to use 500 items (for a larger warehouse), the first 500 items were selected to stock the shelves. The only other parameter stored in the items table (besides the item id) was the item popularity. Realistically in e-commerce the goal is to sell and ship items quickly so most of the items in the table were weighted to be more popular [2]. To calculate the popularity a random number was generated and weighted using the square root of the randomly generated number. This resulted in a curve shown below with the resulting popularity on the y-scale based on any randomly generated number (0-1) on the x-axis.

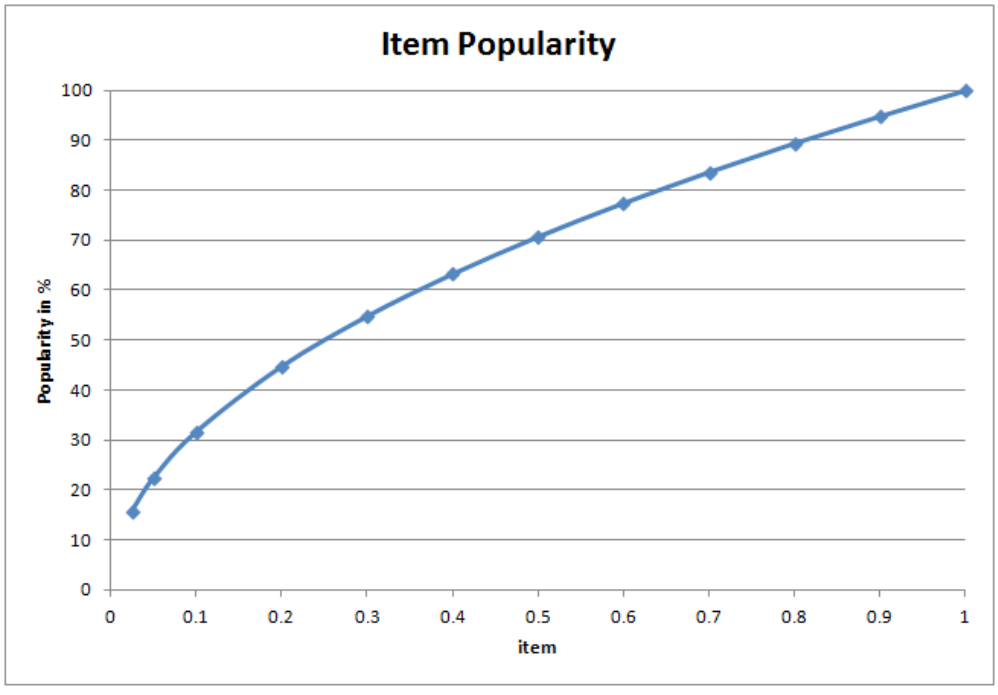


Figure 2 - The probability that an item is popular

The idea behind the popularity is that most orders will contain popular items, and the popularity can be used to stock the shelves in a more efficient manner so that the picking time is optimized. How the shelves were stocked with these items was based on simple or more complex algorithms in the simulator initialization. The most simple was to randomly assign bins to the items (assuming most were popular anyway) and the most complex was to carefully store items by popularity.

The “orders” table is actually several tables. The items in the orders had to be based on a pre-determined number of items in the warehouse so that the orders contained a subset of these items. In other words a warehouse with 100 items had orders that contained items 0-100, whereas a warehouse with 1000 items had more variety in the orders (could contain any items with ids 0-1000). So orders were pre-generated using a random number of items (0-12, with 6 being the most popular based on the average number of items in an on-line shopping cart [7].) However, we did account for outliers in our generated order, and as a result the range of orders in our database was between one and twelve. This is demonstrated in the graph below. The orders table had two columns: a unique order id and a list of items in the order. Orders were always selected from the database in the same way (same lookup) to provide consistency in the simulation runs. Any number of orders (up to 10,000) could be selected from any of the order tables. Order tables were set up for 100, 200, 300, 400, 500, 1000, 1200, 2000, 4000, and 10000 items in the warehouse.

The last table is the “results” table. Following a simulation run, a variety of information about each run is collected and placed in the database under a unique run id. A unique ID is also generated for a set of runs that are created from one behavior space run so all the results of this experiment can be compared with greater ease by simply finding all the runs that share this ID.

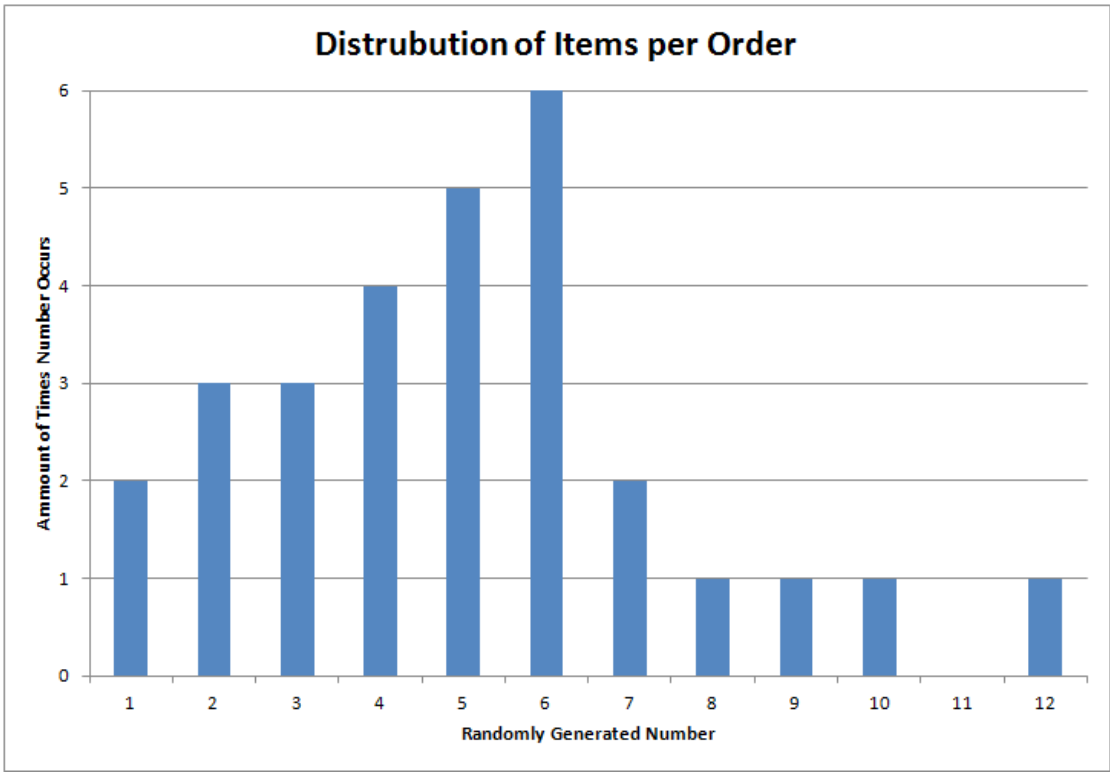


Figure 3 - This graph shows a range of up to 12 items, but most orders have around six.

Simulation:

The first step to set up a simulation is the creation of a warehouse in “Warehouse Layout Manager”. In this program the output will look similar to what is pictured below.

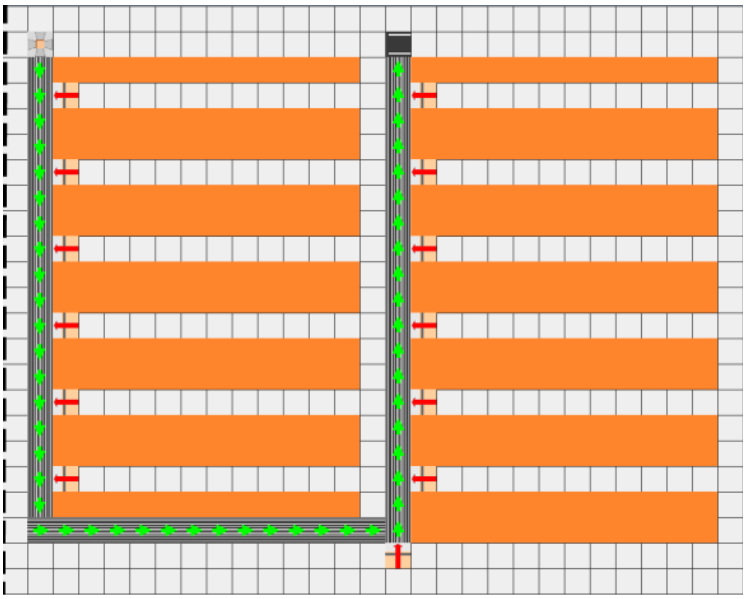


Figure 4 – How the warehouse is built in the “Warehouse Layout Manager”

This warehouse is saved as a binary file with the ".t64" extension. Once a file is created in this manner, it can be selected to be loaded into a simulation.

The simulator can be run with or without graphics, and also with command line parameters (useful for exploring behavior space) or with a panel to set specific simulation parameters. The screen shot below shows what a parameter panel looks like. The parameters that can be set include which warehouse layout to use, how many items should be stocked, which warehouse should be loaded, how many orders should be processed, and what the maximum number of orders to be processed should be in addition to several other important parameters. Once this dialog is closed the program queries the database for a list of orders the length of which corresponds to how many orders are supposed to be processed.

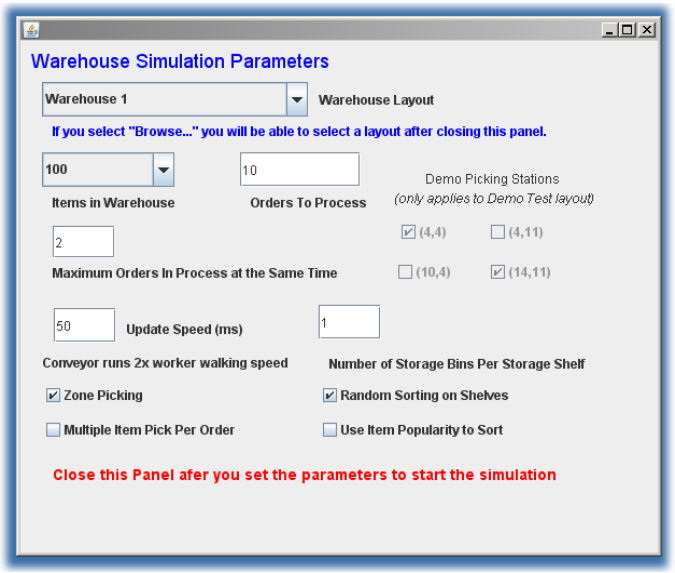


Figure 5 - The parameter panel for a single simulation run

It is also possible to run the program without these by providing the parameters directly to the simulator for the purpose of consecutive runs for an experiment, and this is the method that was used to find our results. A helper program we created called “Behavior Space” let us make many simulation runs by varying one parameter at a time in this manner. The orders table of the database is important because while we could generate random orders for each individual simulation, a database allows us to make a controlled experiment; no random parameters affect our results so we are sure they are consistent to the input we provided. Our simulator does not contain icons but is still a

direct representation of the file as it was created. Picking zones are calculated for the simulation, and that accounts for the re-coloration of the shelves and the addition of circles next to the picking stations in this screen shot (colors are randomly generated since we do not know ahead of time how many zones will be in the warehouse). A picker is represented by the circle (he moves around on the grid as he works) and the shelves that are part of the picker's “zone” or “domain” are in the same color as the picker.

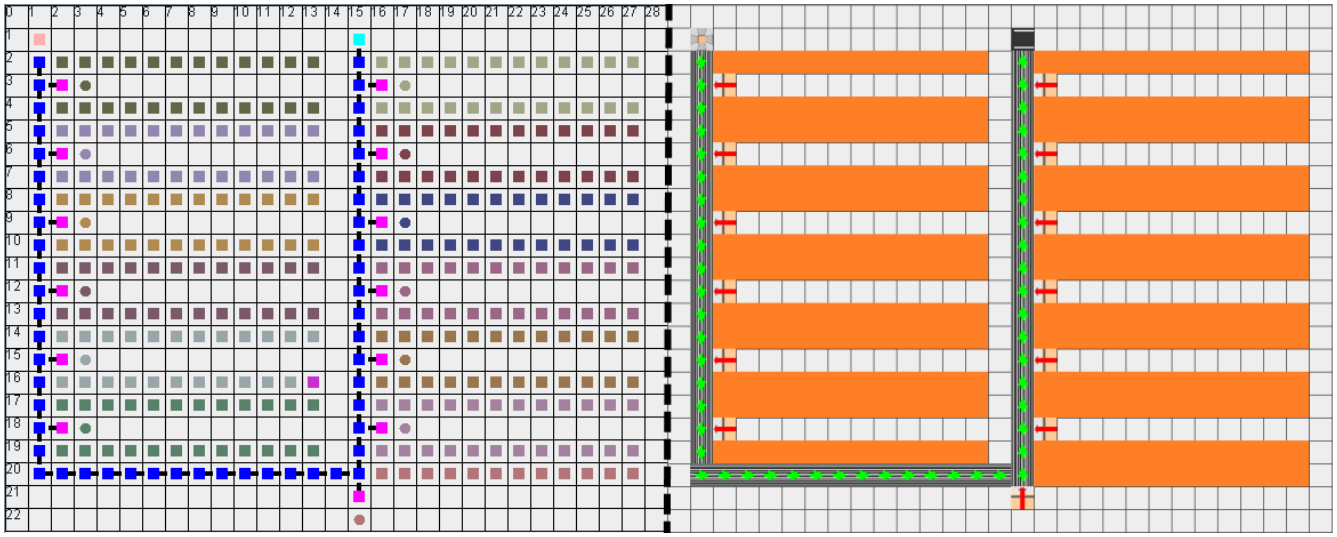


Figure 6 – The warehouse layout and resulting simulation screen

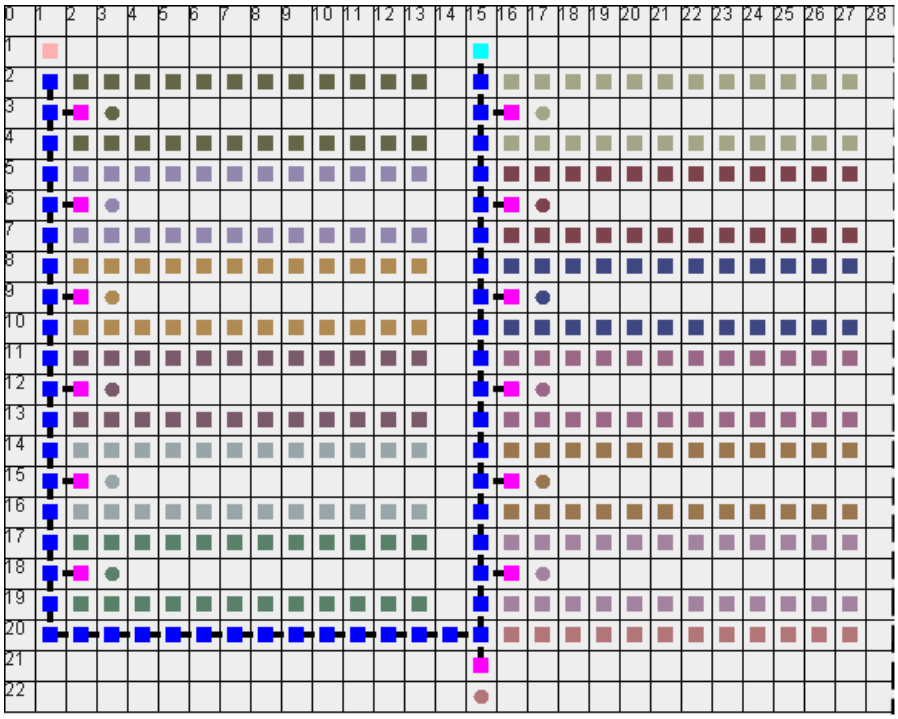
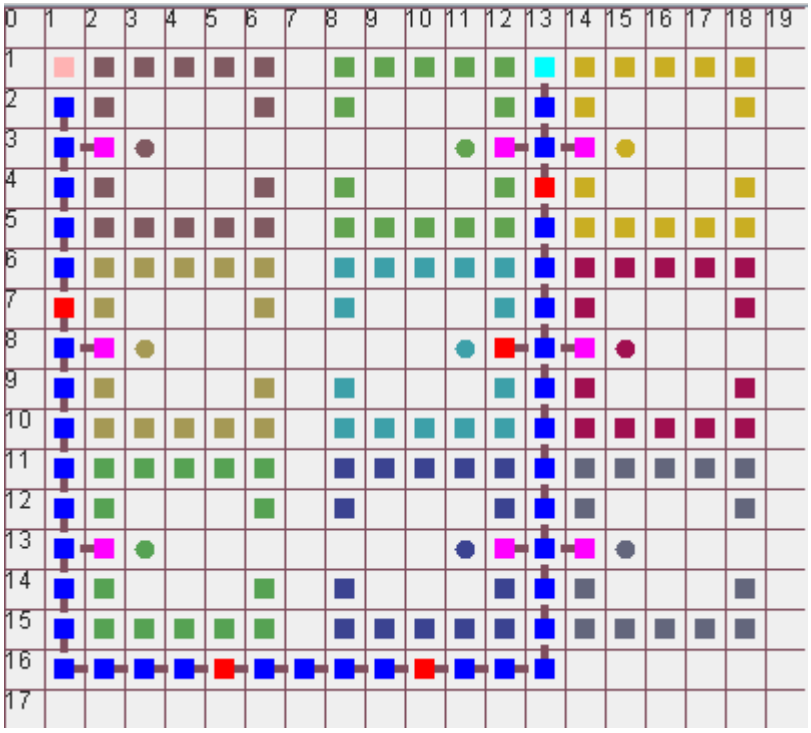


Figure 7 – Aisle Layout

Directly above is a screen shot of an example simulation setup called the “Aisle” warehouse. The conveyor that snakes through the facility is blue with a thick black line pointing in the direction it is facing and is broken into conveyor segments. The picking stations are pink with a thick black line pointing to the conveyor they are connected to. Next to each picking station a picker is generated and assigned to that station so that he will stay in that zone. Each station has the shelves with the closest walking distance added to its zone, so that when a tote that needs an item in those shelves is passing by it will be redirected so the picker can collect that item. Zones in this example are mostly aisles, so it is apparent what is closest to the station. The shelves in a zone and the picker that is assigned to that zone are colored in the same way so it is easier to see where they are. Zone colors are randomly generated. At the start of the conveyor there is always an origin, colored a pink salmon color, for the totes where they are placed onto the conveyor and at the end of the conveyor there is a place where the totes are removed from the system (the order is ready to be shipped), which is cyan. A tote on the conveyor paints that segment red.

Below is another layout we used called “Boxy” Layout.



The A* (pronounced "A star") path finding algorithm was core to our project in order to get the best paths for the purpose of the movement of pickers, calculation of walking distance for picking station zones, etc. The form of the algorithm we used was adapted to java by memoization.com (http://memoization.com/2008/11/30/a-star-algorithm-in-java/) [8] from the Maze example in *AI Application Programming* by M. Tim Jones, and was heavily modified to fit our needs. The general idea of an A* search algorithm is that it uses a heuristic to find the best next point from each point as it works out a path, where once each best point is selected it is added to the list of best path points. Once this path reaches the goal this list is saved as the path to be used. Specifically, our program generates a complete second overlay grid to be used for the path finding, where the points on the grid that are not accessible (conveyors, storage, etc.) are removed from this second grid because they cannot be part of the worker's path. Starting from the beginning point where the path needs to come from, the program looks at all adjacent points to find the one that is closest to where it needs to be, and weights it additionally based on how far the path thus far is from the start point. It then adds the point to be the "parent" of the point it just came from, and repeats the process. Once it reaches the end point the program follows the chain of points back to where it originated and forms a list of all the points, which is the result. In our program a router class is available to be

instantiated for use by any other portion of the program, and it is inside these routers that the path finding is performed. The path finding part of the simulation was by far the most time consuming part of the development. Despite the weeks spent finding the cause of of stack overflows, we are very satisfied with the resulting code and its functionality.

The Agent Model

At the heart of our simulation is the agent framework MadKit [9]. Part of the learning curve for our simulation was understanding how MadKit agents interact and communicate with each other as needed. Madkit is based on messaging between agents in the same community and groups. For our simulation all agents were in the "warehouse" community. The Warehouse agent itself belonged to several other groups so that it could be the central relay for other kinds of agents to interact. There was one Warehouse agent in the model, but there could be any number of conveyor segment agents (which communicate with each other to transfer totes) and any number of picker agents.

Item Stops (where the pickers put items into totes) were another type of agent. The role of the Item Stop was to tell a picker when it has a tote to fill and to put the tote back onto the conveyor system when it was filled with the items from its zone. Zones (called a domain in the code) are calculated based on the item bins closest to itself by walking distance (using A*). Therefore zones are actually a collection of item bins that are closest to the picking station.

End conveyor segments and Start segments (known as ToteSpawns) are subclasses of the normal conveyor segments and have different roles in the "conveyor segment" group. The ToteSpawn is responsible for initiating the order in the system. It puts a new tote on the conveyor system with an order number.

The simulation agents themselves are prompted to act by schedulers that operate once each tick (the unit of time in the simulation). Schedulers call specific methods in each of their agents each time tick. Some Agents act independently of timers, such as the central Warehouse agent. All agents can receive and send messages to either specific agents or to all agents of a certain group and role (as a

broadcast message). For example in our model the Item Stop can send a message only to its own picker to tell it to work on an order.

In the initialization stage of our simulation, after the layout is read in, all the agents are started and launched by the Warehouse. One Picker is launched for each ItemStop (and is assigned to the stop at this time). When agents are launched in the MadKit platform they call their “activate” method which sets the agent's community, group, and role. After this the agent begins to listen for messages and handles them in its “receiveMessage” method. Messages in our simulation are all subclasses of the generic MadKit Message class. The reason we made so many subclasses is that it is easy to determine what kind of message it is when received by the receiveMessage method. Schedulers are special agents that call methods in the Agent classes on a timer. The Timer scheduler is the main scheduler which calls the “Tic” method in the Warehouse to check the status of the order process.

The basic process of the simulation begins when the Warehouse sends the first “NewOrder” message to the ToteSpawn segment of the conveyor system. It responds by placing a tote with the order number on the next conveyor segment. The next conveyor segment for the ToteSpawn (and all conveyor segments) is set up during the initialization of the simulation (when the layout file is read).

If the system is set up to handle multiple orders at a time, more totes with orders are launched, up to the number allowed by the maximum allowed orders in process at one time (a simulation parameter).

The Tote continues along the conveyor until it reaches an "Item Stop" (picking station). If the order in the tote has an item in the Domain of this ItemStop then it is ejected onto the ItemStop segment. When the ItemStop receives the tote it examines the order and sends the Picker off to collect one or more items (depending on if the picker is operating in single-pick or multi-pick mode). The ItemStop monitors the tote (one of the methods called every time tick of the simulation) to see if it is full yet. If it is full it places it back on the conveyor system to head toward the End Segment (or toward another Item Stop for more items).

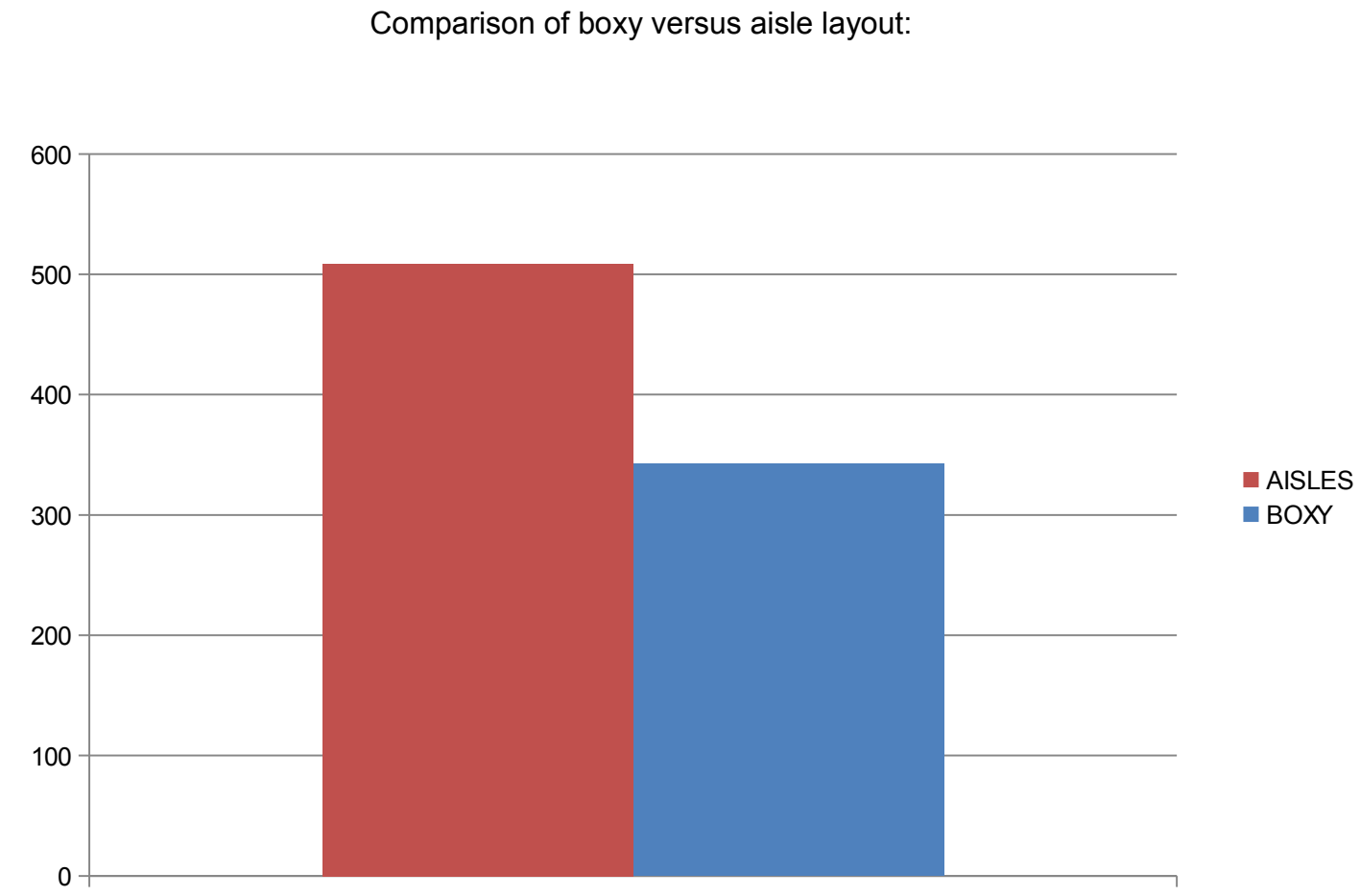
Sometimes a tote that needs to get off at an Item Stop cannot because the ItemStop already has a tote. So this Tote will go around the conveyor and become a partially filled

order. Partially filled order totes are placed back on the conveyor system by the ToteSpawn if the number of orders in process is less than the maximum allowed at one time.

The tote finally reaches the end of the conveyor (End segment). If the order is complete this order process is finished and the order is removed from the system (shipped). If the tote's order is not yet filled it is put back on the conveyor to go past the item stops again.

When the orders are all complete the database record is updated with the number of steps for the simulation and other simulation parameters.

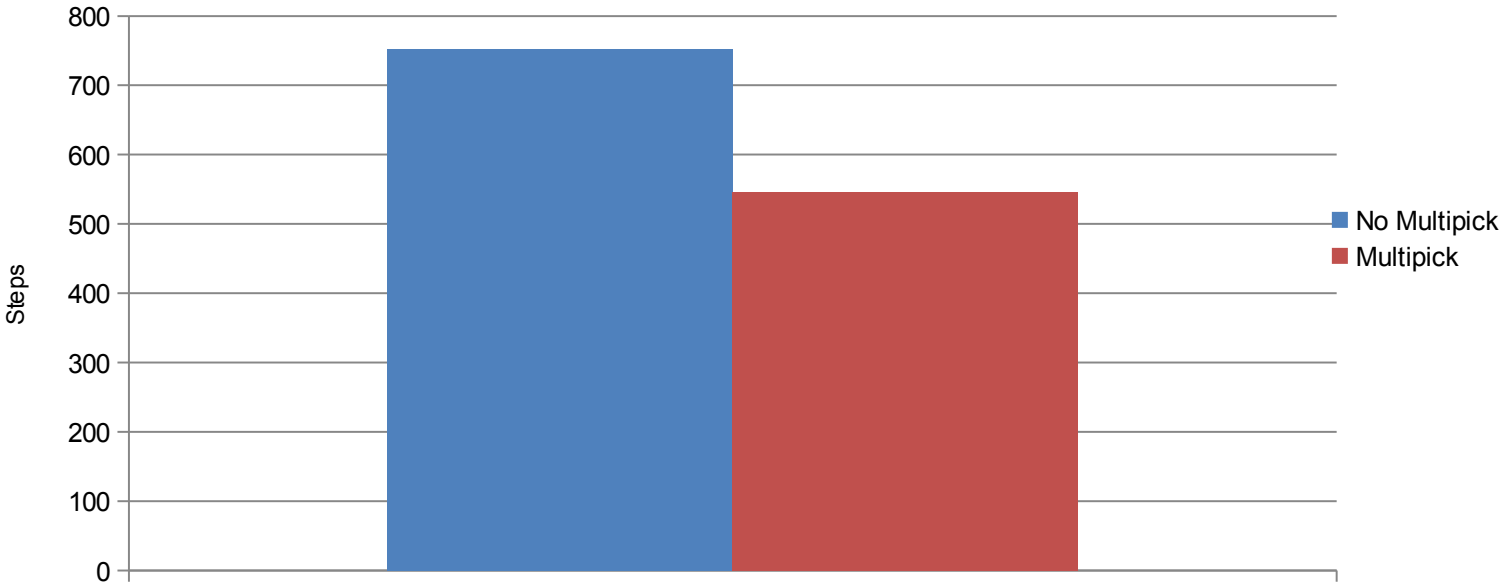
Results and Analysis:



Our simulations of the boxy warehouse ran faster than the aisle picking warehouse.

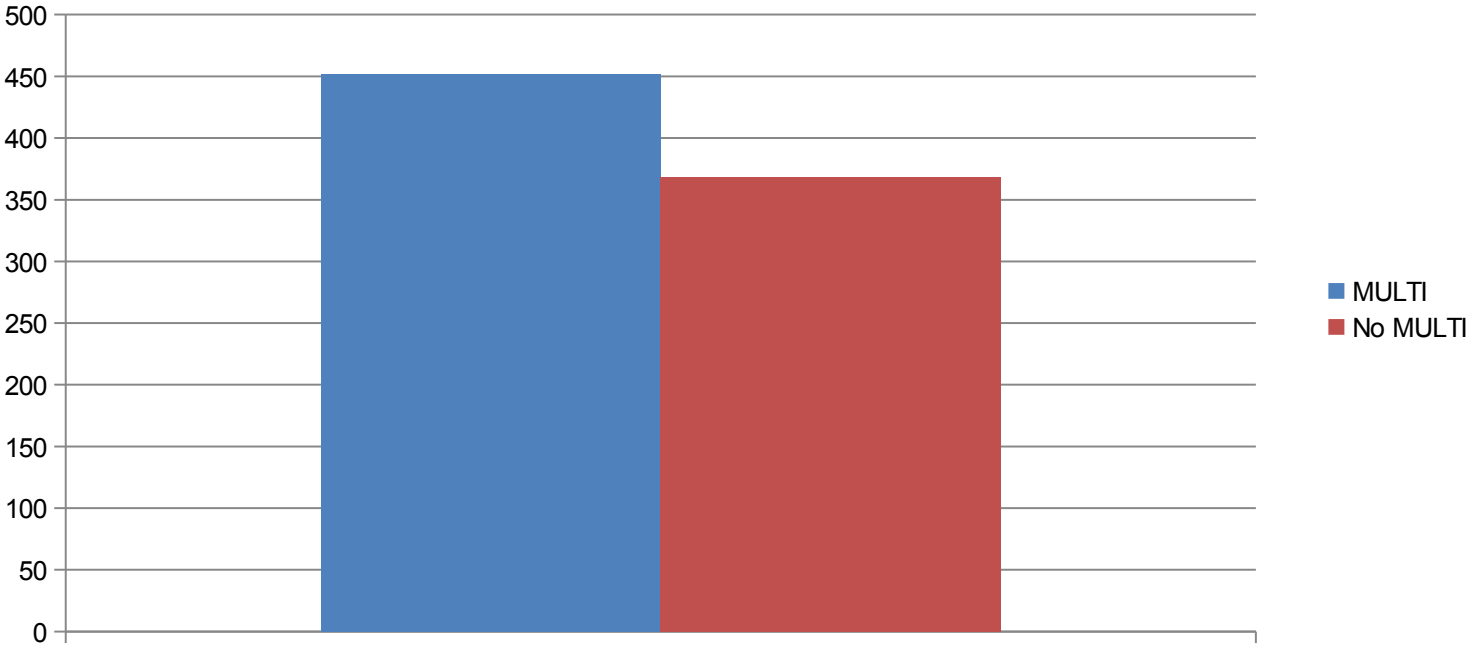
Comparisons of picking multiple items at a time versus picking one item at a time:

Aisles:



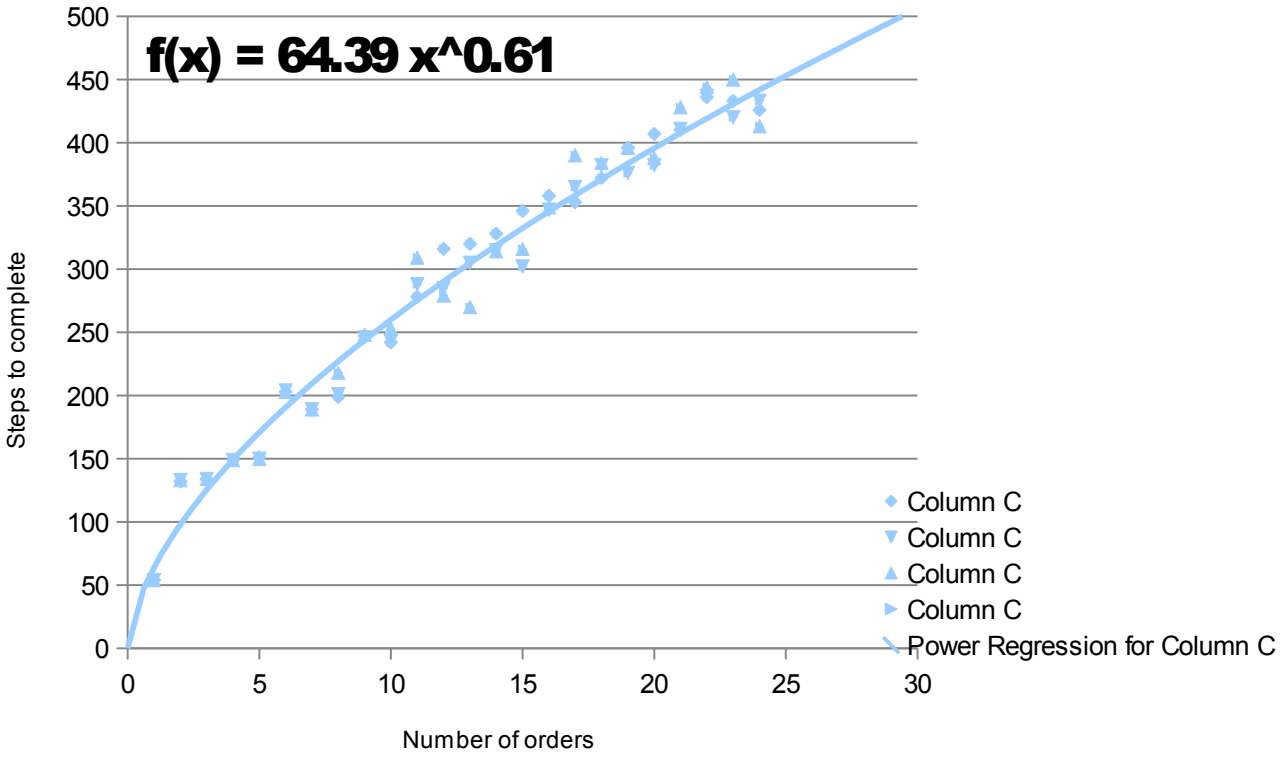
On the facility with pickers in aisle shaped zones there was a clear time saving in multipicking.

Boxy Layout:



There were fewer items in this run and so fewer total steps. There is a slightly smaller saving.

Number of orders:



This was an interesting set of runs comparing the number of orders to the amount of time that the run takes to complete. This graph was created based off of the data, and one can tell that the pick time does not increase linearly with the number of orders.

Conclusions:

From the data generated by our program we can draw some interesting conclusions. The boxy warehouse appears to be more efficient, although it is likely that this is because we reduced the number of items to 100 for both layouts and so the aisles were not full, leading to a loss of efficiency. Multi- picking is always faster than single picking, and this result makes sense because it cuts out countless unnecessary trips for the picker returning to his station in between picking up items. However, single-picking is less inefficient on a boxy layout. Again, this makes sense because the picker is never as far from the station as it is inside an aisle

where it must walk all the way back down the aisle between picks. Finally, we retrieved an interesting result when increasing the number of orders that are run through the database. The time it takes to pick for an increasing number of orders increases, of course, but the increase was not linear. At lower order quantities the increase vastly increased the time required for a complete pick, but as the order quantities increased each individual order came to make less of an impact.

The most important result of our project was the simulator itself. These results are only changing several parameters, but with slight modifications we could measure much more.

Significant Original Accomplishments:

Creating a working simulation of the picking process was a major task. It was made simpler by using the MadKit framework and implementing the Warehouse Layout Manger to generate warehouse layouts. MadKit worked great but took some planning and some trial and error to set up the agents and messages. Overall we highly recommend that any Java programmer building a simulation that involves agent interactions use MadKit because of its flexibility and adaptability to large agent-based projects.

The Warehouse Layout Manager was a significant accomplishment that helped us (and others interested in using our simulation) lay out a warehouse for testing. We think that adding this application was an excellent idea so that the warehouse configurations did not have to be hand coded for the simulation (although we used a simple “Demo Test” configuration for testing code that is hand coded and is part of the WarehouseLayout code).

Adapting the A* Maze route algorithm to path finding in the Warehouse was a straightforward idea, but took a lot of work to make it function correctly in our environment. This Route code was use extensively in the simulation.

Once all these parts were put together and the simulation was running it was beautiful to watch the pickers go to the shelves and grab items, add them to the order, watch the totes leave the picking stations and travel around the conveyors until the orders were all filled.

Acknowledgements:

First, our team would like to thank the New Mexico Supercomputing Challenge for making the project possible. Next, we would like to thank Elizabeth Cooper who was our mentor this year. She helped us with creating the simulation and proofreading the final report. Our team would also like to thank our teacher Lee Goodwin, who provided encouragement throughout the year. Many experts advised us on the development of our project and we would like to recognize their valuable input and assistance as well. John Snider answered questions we had related to the warehouse business, which he currently works in. His input was useful in cementing our grasp of the internal workings in a warehouse. We would also like to thank Jeff Grantham for giving us feedback on the interim report and those who evaluated our project at the Northern New Mexico Community College. We would like to thank Ms.Parkinson, the teacher that runs our school's writing lab, who helped grammar check our final report. Finally we thank Fabien Michel, a primary MadKit developer, for being an active supporter of our project.

Background on our Team and Experience:

Our team is composed of three freshman students from Los Alamos High School. Colin Redman has been participating in the supercomputing challenge since 2005. This is Sudeep Dasari's 5th year participating, and David Murphey's 2nd year. Experience from prior years was helpful in planning and implementing this project. Last year's project on sorting packages in a shipping facility was similar but implemented in Greenfoot (also a Java based agent system). One major difference between that project and this one was that there were less variable parameters that could be adjusted so the simulation was not as interesting.

This is the second MadKit project that Colin, our chief programmer, has produced for the New Mexico Supercomputing Challenge. The first one was three years ago, simulating a closed environment in a long voyage spacecraft. This used a previous version of MadKit (version 4.2) and although a lot of thought went into the simulation it was not very complete (too many parameters to consider). This year MadKit made a lot more sense and we used an newer updated version (version 5.0) which was easier to integrate with the simulation code since it was not designed as an IDE, but as a library.

Sudeep worked with this team last year and this year was the secondary programmer, but chief developer of the Warehouse Layout Manager. David was also with the team last year (his first year) and contributed by drawing graphics for both programs and for the illustrations in the final report and evaluation presentation. Everyone contributed to the final report but Sudeep and David did most of the work while Colin was finishing the simulation code and producing results.

Citations:

1. Eisenstein, Donald D. *ANALYSIS AND OPTIMAL DESIGN OF DISCRETE ORDER PICKING TECHNOLOGIES ALONG A LINE*. Rep. Graduate School of Business, The University of Chicago, 27 Dec. 2006. Web. 7 Mar. 2012. <http://faculty.chicagobooth.edu/donald.eisenstein/research/pick12.pdf>.

2. Ricker, Fred R., and Ravi Kalakota. *Order Fulfillment: The Hidden Key to E-Commerce Success*. Rep. Logistech, 1999. Web. 3 Mar. 2012. <http://www.logistech.us/lsi/resources/SCM9911ecomm.pdf>.

3. Koster, Rene De, The Le-Duc, and Kees Jan Roodbergen. *Design and Control of Warehouse Order Picking: A Literature Review*. Rep. Rotterdam, The Netherlands: Erasmus University, 2006. *Design and Control of Warehouse Order Picking: A Literature Review*. ScienceDirect, 25 Oct. 2006. Web. 26 Mar. 2012.

4. Vrysagotis, Vassilios, and Patapios Alexios Kontis. *Warehouse Layout Problems : Types of Problems and Solution Algorithms*. Rep. International Scientific Press, 31 Aug. 2011. Web. 13 Mar. 2012.

5. Bartholdi, John J., and Steven T. Hackman. *WAREHOUSE & DISTRIBUTION SCIENCE Release 0.95*. Rep. The Supply Chain and Logistics Insitute, 21 Aug. 2011. Web. 26 Mar. 2012.

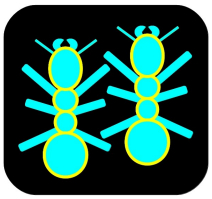
6. "Maximize Pick Productivity." *The Avery Way. Warehouse Consultant Gets Results Fast!* Avery Way. Web. 14 Mar. 2012. <http://www.elogistics101.com/Article/MaxPickProd.htm>.

7. "Good News! The Average On-line Order Value Was up More than 10 Percent from Last Year." *Good News! The Average On-line Order Value Was up More than 10*

Percent from Last Year. Pinnacle Cart. Web. 19 Mar. 2012. <http://blog.pinnaclecart.com/2010/11/22/good-news-the-average-on-line-order-value-was-up-more-than-10-percent-from-last-year/>.

8. Admin. "A-Star Algorithm in Java." *A-Star Algorithm in Java*. Memoization, 30 Nov. 2008. Web. 28 Mar. 2012. <http://memoization.com/2008/11/30/a-star-algorithm-in-java/>.

9. MadKit http://www.madkit.org/. A Multi-Agent Development Kit by Olivier Gutknecht, Jacques Ferber, and Fabien Michel, LIRMM Montpellier France



ExcellAnts

New Mexico
Supercomputing Challenge
Final Report
April 4, 2012

Team 66
Los Alamos High School

Team Members:
Peter Ahrens
Dustin Tauxe

Teacher:
Lee Goodwin

Mentors:
James Ahrens
Christine Ahrens

Contents

1 Executive Summary

The aim of this project is to create an efficient parallel implementation of Ant Colony Optimization **(ACO)** applied to Traveling Salesman Problem **(TSP)**. It should also be portable and easy to understand or modify. ACOs are algorithms based on ant foraging behavior. The TSP is a problem in which cities in an undirected graph must be connected by the shortest tour possible. A tour is a path that visits each city once and only once. ACOs have applications in problems including vehicle routing, networking, communications, and scheduling. **Data Parallelism** is a style of parallelism that usually consists of running the same fine grained operation for each piece of data in a very long vector.[6] The very large size of data that must be processed in a parallel ACO makes a data parallel implementation attractive. Due to the small size and large number of computations that must be performed at the same time, data parallelism particularly lends itself to computation on the Graphics Processing Unit **(GPU)**. GPUs must do many tasks via "threads" in parallel to display pixels on the screen. They are convenient to use as general purpose processors for hardware acceleration of programs, as they are readily available on most computers. We used **Thrust**, a data-parallel C++ template library modeled off of the C++ Standard Library's[13] vector operations and implemented in a data parallel fashion, to implement our data-parallel ACO. Our implementation proved to be very effective. We achieved a speedup of about 100 (it varies with problem size) over a serial implementation. Our program has a computational complexity of O($nlog(n)$) while the serial implementation has a complexity of O(n^3). Excellants has made original contributions. Firstly, our implementation is portable to targets other than GPU. Also, we describe a tree algorithm that is important. Our code is also available and open source, both in this report and online. Additionally, our code is written in an easy to understand and modify C++ template library called Thrust. These advantages are important to anyone looking to use our work in a practical application or to extend it in the research world.

2 Problem Statement

Ants can forage for food quite efficiently. When an ant finds food, it leaves a pheromone trail back to the ant hill, which compels other ants to follow the same path. However, as the wind blows and the sun shines on this trail, the pheromones start to evaporate. Only the most traveled trails can continue to exist. Thus shorter, more popular paths are generated. Ant Colony Optimization **(ACO)** is a technique inspired by this ant foraging behavior, and can be used to generate good solutions to combinatorial optimization problems very quickly.[8]

Although Ant Colony Optimization seems more suited to foraging, it has proved itself a powerful metaheuristic that can be applied to problems ranging from routing to machine learning. However, ACOs are most conceptually suited to and commonly applied to the Traveling Salesman Problem **(TSP)**. This is a very well-documented combinatorial optimization problem. In Symmetric TSP (referred to as TSP in this paper), **n** nodes in an undirected graph must be connected in the shortest tour possible. A **tour** is a path that visits each node once and only once. Each node is defined as a **city**, and a path connecting two cities is called a **route**. The TSP represents the dilemma of one unlucky salesman who has several cities to visit, but limited gas money and time in which he may do so. Our salesman would like to travel the shortest tour possible. Unfortunately, the TSP is a very difficult problem. A brute force approach to a TSP of **n** cities would have a computational complexity of

$$\frac{(n-1)!}{2} \tag{1}$$

Solving a 200-city TSP using brute force would take approximately $2.062x10^{360}$ years on Computer A [1], yet an ACO can get to within 2% of the optimum in 200 seconds. For the purposes of this project, the Travelling Salesman Problem will be used as a standard problem to solve, but is important to note that the TSP is not the focus of this project. Many excellent TSP solvers already exist. The focus is on ACO, which can be applied to many problems ranging from networking to protein folding, and TSP will be used as an example problem for ACO to tackle.

In some cases, the problem may change during the time an ACO is generating a solution. Let us use, as an example, the case of a truck driver delivering packages. He must deliver several hundred packages a day, and finding an optimal tour could take a while. To make matters worse, unan-

ticipated packages may arrive in the early morning. A fast ACO would save him time (less time spent waiting for his solution to be generated). It would also save money (if the optimization runs faster, using the time he has more efficiently, he would get a better tour and thus have to spend less money on gas). Thus, there are two advantages to having a faster ACO. Our driver's day is not over, however. Suppose there was an eleven-car pileup on a major road. The solution his program had generated was operating under the assumption that this road would be a convenient route, but now the driver needs a new tour. He could wait for the long simulation to run again, but if he had an ACO that could quickly generate a new solution based on previous calculations, he could get home to his family much faster.

This TSP with cities that change as the problem is being solved is called the Dynamic Travelling Salesman Problem **(DTSP)**. The DTSP can be thought of in two ways. It could be that the problem changes after a solution has been generated, and the ACO simply resumes working with the previously calculated pheromones, or it could be thought that the problem changes as the ACO is solving it, and it must cope with the changes. The former situation being more suited toward a more stable real world application like our truck driver, and the latter being suited to something more volatile like a network routing problem. These two perspectives may arise out of different situations, but they are fundamentally identical in solution, in that the simulation must simply alter the data it has calculated before the change to fit the new conditions.

Clearly, an ACO applied to a DTSP would have to be fast, and could thus benefit from a parallel implementation. Implementing ACO in parallel is difficult, however, due to the random memory access patterns and the coordination of large parallel tasks.[8] Even though an ACO will not actually be applied to DTSP in this paper, it provides an excellent reason for an ACO to be sped up, and any advances in ACO techniques for a TSP could easily be applied to DTSP.

Due to the large amount of data that must be processed in an ACO and the relative simplicity of the computations that must be performed, a parallel implementation of ACO would be desirable. Also, because the TSP is simply a sample problem, the code of such an implementation would have to be simple enough to be modified for other problems. The aim of this project is to create an efficient parallel implementation of ACO on the GPU applied to TSP. It should also be portable and easily understood or modified.

3 Background

In order to understand the methods used to create an efficient parallel implementation of ACO, one must first understand the traditional implementation of an ACO applied to TSP. The mechanism of action for an ACO can be described as follows.

All ACOs have the same approximate structure. To initialize, they calculate all the distances between cities, make pheromone and probability matricies (a way to store the values of all the pheromones on all the trails), and create ants. Once the data is initialized, then the program enters the main loop in which the ants construct solutions and then lay pheromone based on the quality of these solutions.

In a TSP, ants start their tour construction at a random city. They then use probabilistic rules to decide where to move next until they have visited all the cities. Two factors influence these decisions. The first factor, τ_{ij} is the pheromone on a route from city i to city j. The second factor, η_{ij} is the inverse of the distance. The probability p_{ij} that ant k at city i will move to city j is given by:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta} \ , \ if j \in N_i^k \tag{2}$$

where N_i^k is a collection of all the cities the ant has not yet visited and α and β are parameters. Pheromone update is achieved in many different ways for different algorithms. In all cases, evaporation occurs on the routes first. The new amount of pheromone on a route τ'_{ij} is given by:

$$\tau'_{ij} = (1 - \rho)\tau_{ij} \tag{3}$$

where ρ is a parameter (from 0-1). Then, the ants deposit pheromone on their tours. Usually, the base unit of pheromone an ant lays down, $\Delta\tau_{ij}$, is given by:

$$\Delta\tau_{ij} = \frac{1}{C_{ij}} \tag{4}$$

where C_{ij} is the ant's tour length. After the ants deposit pheromone in some configuration, Pheromone update occurs in many ways, so the above equations are to help the reader understand the basic ways the pheromone update works.

3.1 Previous Work

For last year's Supercomputing Challenge, we created the most common implementations of ACO in Python. These implementations ran in parallel on the CPU using Python's multiprocessing module. We had great success with the performance enhancements that came from the parallelization of the algorithms. However, last year's program was not optimized for speed.[5] Designing last year's code inspired us to build a much faster, more optimized, more efficient version this year. All code from last year had to be scrapped as we were using a faster language and more powerful tools, including an entirely new approach to ACO parallelization.

4 Methods

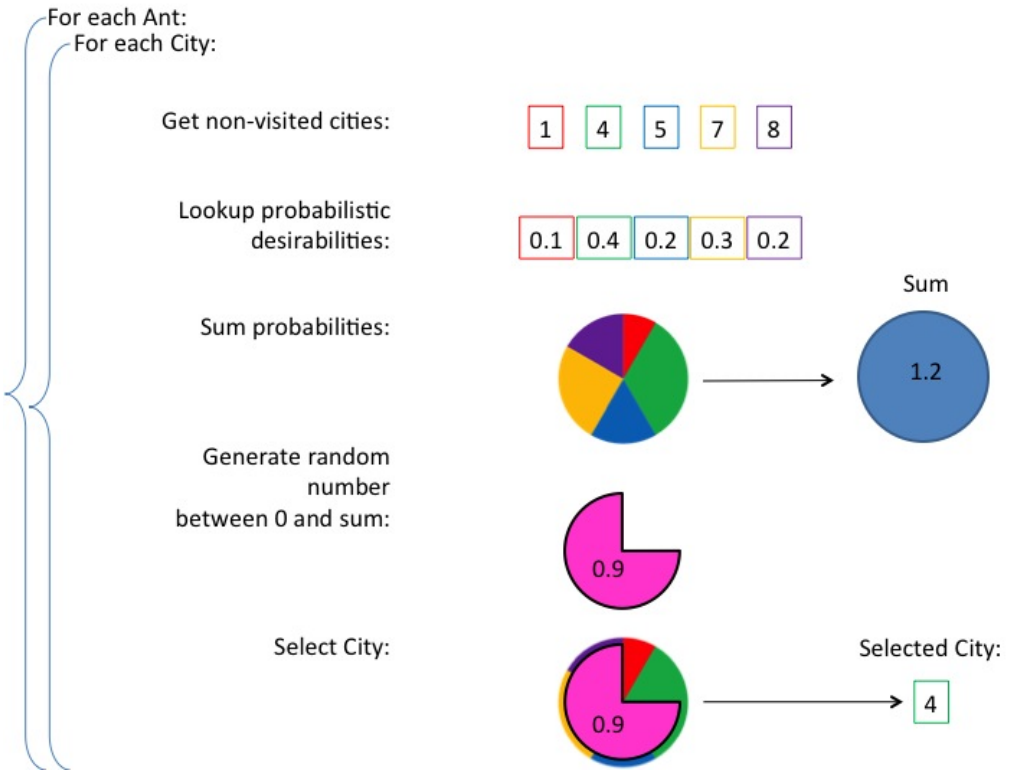
4.1 Tour Construction

Tour construction is the step in an ACO when all ants must construct paths that visit every city only once. Because tour construction takes up most of the time in an ACO[8], this is the aspect on which we focused most. Many implementations were tried and tested to find a suitable parallel tour construction method.

4.1.1 Serial Implementation

The probability of an ant at city i going into city j is described in Equation 2. In a traditional tsp, this probabilistic selection is accomplished through method analogous to a roulette wheel. The various probabilities that an ant may visit are gathered into a list. A random number is generated between 0 and the sum of these probabilities. The ant then iterates over each probability and selects its next city to visit. (One can imagine the roulette ball starting at the top and travels counter clockwise and ends up landing in a pie piece corresponding to a particular city) This process is repeated until an entire tour is created. See Figure 1. This will be very computationally costly as it is done for every ant for every city.

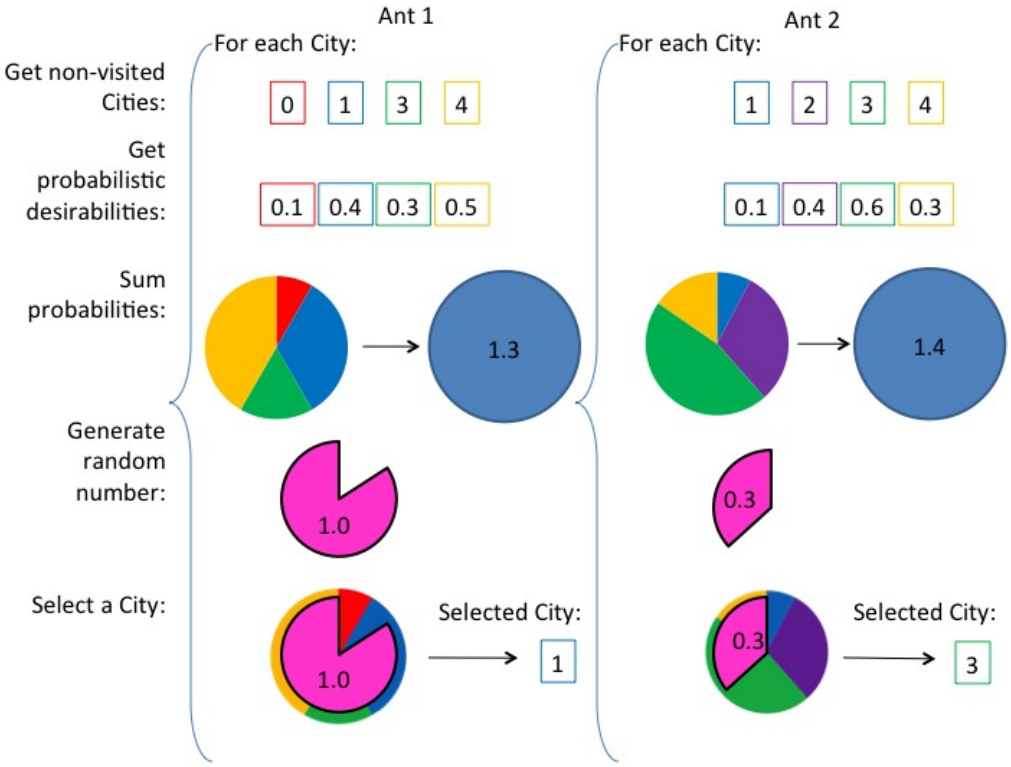
Figure 1: A typical serial implementation.



4.1.2 Task Parallel Implementation

Task Parallelism is the typical style of parallelism in which the parallelism is focused on doing different tasks on different pieces of data. Task parallelism usually consists of running processes to compute the results of more coarse grained tasks. In hopes of speeding up the traditional ACO, it would be tempting to simply run the ants in parallel. It does not scale efficiently, as each ant must still look at all of the next cities at every step of its tour sequentially ($O(n^2)$, where n is the number of cities. It also creates excessive overhead. All of the ants have random access patterns, the tasks may take different times to complete, and the size and number of tasks is also neither suited to a GPU nor CPU. Each processor must make an entire tour for one ant. The overhead significantly reduces the power of such an approach.[8] See Figure 2.

Figure 2: A typical task parallel implementation.



4.1.3 Turning to Data Parallelism

Data Parallelism is a style of parallelism focused on performing the same tasks on the same pieces of data. It usually consists of running the same fine-grained operation for each piece of data in a very long vector. [6] The very large size of data that must be processed in parallel makes a data parallel implementation attractive.[6] Due to the small size and large number of computations that must be performed at the same time, data parallelism particularly lends itself to computation on the Graphics Processing Unit **(GPU)**. GPUs must do many tasks via "threads" in parallel to display pixels on the screen. They are convenient to use as general purpose processors for hardware acceleration of programs, as they are readily available on most computers. We chose to use Thrust [11], a data-parallel C++ template library modeled off of the C++ Standard Library's[13] vector operations and implemented in a data parallel fashion described by Guy Blelloch[6] in his

thesis, which has sometimes been described as a data-parallel bible. **Thrust** is very easy to use and modify for data-parallel operations. The most common Thrust target is CUDA, but it can also target OpenMP, OpenCL, or Thread Building Blocks. Thrust essentially translates the data parallel operations to primitive functions in CUDA, OpenMP, etc. It also provides host and device vector types, that store data on either the host or computation device. [11] Using Thrust eliminated the need for us to create a very specific and optimized data parallel functions and allowed us to create a more general ACO within the time allotted for this project. Some of these Thrust functions are used extensively in our code and assume a very important role in what we do. These functions are described by example in Table 1.

Table 1: Common primitive data-parallel functions supplied by Thrust.

Inputs		
A	0 5 1 8 7 3	
B	2 2 2 2 2 2	
C	1 2 5	
D	0 1 1 2 3 3 4 5	
Function	Output	Description
gather(C in A)	5 1 3	Index A by indicies in C
permutation_ iterator(C in A)	5 1 3	Gather with kernel fusion
inclusive_ scan(A)	0 5 6 14 21 24	Cumulative sum of A
transform(A and B with +)	2 7 3 10 9 5	Add A to B
reduce(A with +)	24	Sum A
sort(A)	0 1 3 5 7 8	Sort A
upper_ bound(C in D)	3 4 7	Find last index of D where C could be inserted without violating ordering

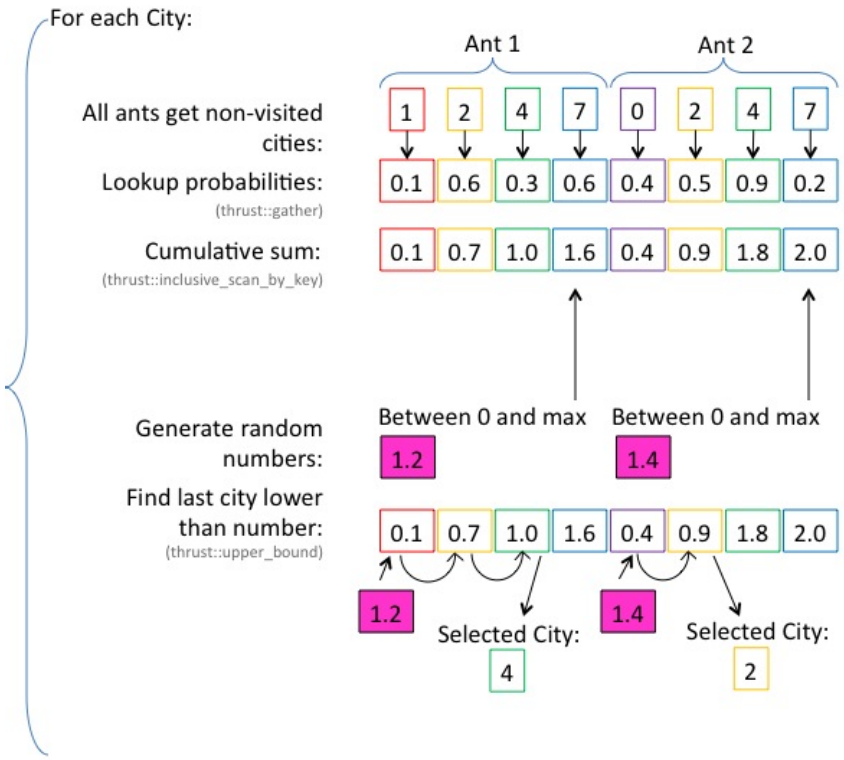
If a programmer can use Thrust functions as frequently and correctly as possible, their code will be both modular, portable, and efficient. For example, when performing multiple memory bound operations on a vector of data, it is better to use kernel fusion, or condense each of the operations to be performed into the same kernel, or chunk of code that will be executed on the device. Another example is the usage of Thrust functions such as the zip_ iterator to create virtual arrays that can be processed without having to actually move or reorder large amounts of data.

4.1.4 Data Parallel Implementations

The first data parallel method we tested was analogous to stacking the previously described roulette wheels and selecting cities for every ant at the same time. Each ant gathered data for all the cities it was going to visit. This is very quick on a GPU because all of the lookups can be performed in parallel and there are many processors with which to do this. Then, a prefix sum(cumulative sum) was performed on a list of all the probabilities. This has the effect of evaluating each piece of the roulette wheel previously described in the traditional ACO at the same time. Then, the random numbers are generated, and the list of probabilities is iterated over in parallel by every ant.

This implementation suffers, however, due to the number of operations that must be performed in series. At every step of the tour construction, the cities the ant visited had to be updated, the probability gathered, the probabilities prefix summed, random number bounds selected, random numbers generated, and searches performed. While this implementation was fairly straightforward to code and understand it needed to be improved upon. See Figure 3.

Figure 3: Our initial data parallel implementation.



A new method had to be implemented that was completely different from all the others. More of the operations had to be grouped and performed at the same time. To accomplish this a tree-based algorithm was implemented. All the probabilities are gathered as previously described for each ant. Then, each probability is assigned to a thread, along with the city it is associated with, and a random number. At each step in the tree, two cities are reduced to one. The random number is used to probabilistically select a city based on the probabilities given. Then, the probabilities are summed and the chosen city is given to the next level of the tree. The unused random number is given to the next level of the tree. This mathematically can select from a large list of cities a single city randomly with a bias toward the probabilities in the same way that the previous algorithms have described. The reason the probabilities are summed has to do with multiplication of probabilities. The probability of going from city i to city j should be equivalent to the desirability metric of that city divided by the sum of the probabilities of all

the other cities that ant can visit. The probability of an example city 1 being selected out of five cities in the tree algorithm is shown below in Equation 5. The probability of city 1 being selected is equivalent to the probability of city 1 being selected at every level of the tree.

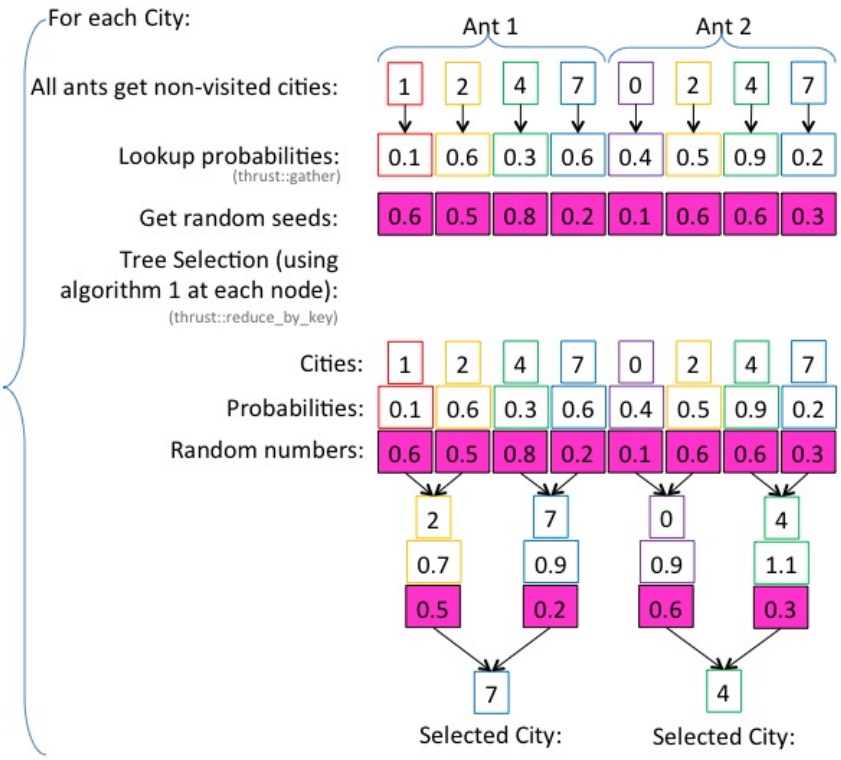
$$p^k_{5\ 1} = \frac{p_1}{p_1 + p_2} * \frac{p_1 + p_2}{p_1 + p_2 + p_3 + p_4} = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N^k_i} [\tau_{il}]^\alpha [\eta_{il}]^\beta} \tag{5}$$

As one can see, the probabilities simplify to Equation 2 as desired. This implementation performed very well and can also scale efficiently. The reduction can perform city selection for all of the ants and all of the cities in parallel for every level of the tree, making the reduction O($log(n)$). The reduction must be performed for every city, so the whole tour construction step is O($nlog(n)$). The implementation is described in the diagram below, and pseudocode is given for the decision function at each node in the tree. See Algorithm 1, Figure 4.

Algorithm 1 The algorithm used to reduce two cities in the tree selection method.

```
Inputs: (City A, Probability A, Random A),
        (City B, Probability B, Random B)

if (Random A * (Probability A + Probability B) < Probability A)
{
    return (City A,
        (Probability A + Probability B), // Probabilities added
        Random B) // Unused random returned
} else {
    return (City B,
        (Probability A + Probability B), // Probabilities added
        Random B) // Unused random returned
}
```

Figure 4: Our best data parallel implementation.



4.2 Pheromone Update

Even though the tour construction is the most time consuming, it was simple enough to implement pheromone updates in parallel, and important to do as it must not become a bottleneck. It is also advantageous to do this on a GPU as data transfer between the GPU and the CPU is expensive in terms of computation time.

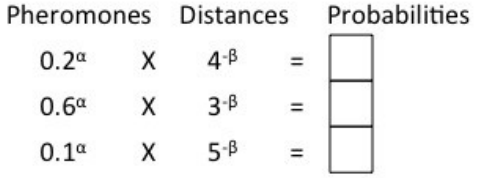
The pheromone update is straightforward to implement but can be different for every ACO variant. As a general rule, it makes sense to add the previously calculated amount of pheromone to each route in the necessary tours. For our project we settled on a Rank Based Ant System as it is simple to implement. In a Rank Based Ant System, ants deposit pheromone according to their rank. Their rank is assigned based on the quality of their tour. Most of the time the number of ranks is usually around six so only six ants will deposit pheromone. Thankfully, Thrust has a very well implemented

and efficient sort, so this sort was used to assign the six ranks.

4.3 Probability Calculation

Calculating probabilties is also simple to do in parallel. Each probability for a particular route is calculated by a separate thread, using the distance of that route and the pheromone on that route. Probability calculations for each route are easily parallelized. Each probability is calculated in parallel. The distance and pheromone for that particular route can be looked up by a separate thread, making this a very simple and quick implementation. thrust::transform() was used to calculate Equation 2 ($probability = pheromone^{\alpha} * 1/distance^{\beta}$) for all of the cities in parallel. See Figure 5

Figure 5: Probability calculation with thrust::transform().



5 Results

It was initially decided that Marco Dorigo's code would be used as a benchmark test against which to compare our code.[8] This, however, led to issues because his code did not allow for large numbers of ants (i.e. greater than 100). For this reason a new ACO was selected. We settled on libaco[10] because it is a very standard ACO, and can accommodate the large number of ants that we require. It is as accurate as Dorigo's, but may be slower. We cannot measure because Dorigo's code does not allow for large numbers of ants. The exact speed may not be as important of a metric as how the algorithm scales. This is because the two algorithms are being run on different devices. For this reason, both experiments measuring speedup and experiments validating theoretical scaling have been run. There is, however, a limit to scaling, as the number of threads and amount of local memory on a GPU may reach an upper limit. Another metric used to assess the

quality of both algorithms is a comparison of the quality of the tour after a certain amount of time. This is a similar metric to a simple speedup calculation, but is looking at a fixed time limit instead of a fixed quality (same number of mathematically identical iterations) as shown above. These two metrics address both sides of the time/quality trade-off described in detail in the problem statement. All statistical calculations were done with R, a statistical language.[9]

5.1 Validation

5.1.1 Experiment 1

The first challenge was to validate our ACO. Even though the two implementations were created to produce equal results, we felt that evidence of this should be given. To do this, we compare the tour lengths after 20 iterations from 20 trials of both implementations on Computer A[1] on dj38.tsp from National TSP [3]. This is a simple test problem of the largest population centers in Djbouti. A two sample t-test of these 20 trials yields a p-value of .638. Thus, assuming the programs produce identical results, the probability that the discrepancies in the results we obtained were due to chance is 63.8%. This is likely enough to assume that the two implementations produce identical results. Knowing that the implementation is valid, tests can be run to determine scaling capability.

5.2 Scaling and Speedup

An ACO could scale with respect to a few metrics. For clarity, the number of cities will be referred to by the parameter n. The number of ants will be referred to by the parameter m.

5.2.1 Experiment 2

The first test of scaling capability was done with respect to the number of cities. The number of ants was held at a constant 128. The city sets for this test were the first n cities of usa13509.tsp[12], a set of cities in the United States. This and all following scaling tests were performed on Computer B[2]. The results are shown for both implementations in Table 3. Note that a maximum speedup of 100 was achieved.

Table 2: The time to complete one iteration with respect to the number of cities.

Number of Cities	libaco Time (s)	ExcellAnts Time (s)	Speedup
32	0.0675	0.0138	4.89
64	0.2881	0.0254	11.34
128	1.2141	0.0630	19.27
256	5.2334	0.1344	39.35
512	22.3871	0.3707	60.39
1024	95.1977	0.9484	100.38

The expected scaling of a serial implementation with respect to cities should be O(n^2), as each city must be examined at every city in the tour. With this assumption, a fitted linear regression was performed on the data. The correlation constant was 1.000, and the residuals were scattered. r^2 was 0.9998, meaning that 99.98% of the variation in the data is explained by the theoretical scaling. This means that we can safely say the above model is correct.

The expected scaling of our data-parallel implementation with respect to cities should be O($nlog(n)$), as all the cities are reduced (in log(n) time) at every city in the tour. With this assumption, a fitted linear regression was performed on the data. The correlation constant was 0.9973, and the residuals were scattered. r^2 was 0.9947, meaning that 99.47% of the variation in the data is explained by the theoretical scaling. This means that we can safely say the above model is correct.

5.2.2 Experiment 3

The second test of scaling capability used different numbers of ants with the same number of cities. The city set for this problem was held at the constant first 128 cities of usa13509.tsp[12]. The results are shown for both implementations in Table 3. Note that we achieved a maximum speedup of 81.

Table 3: The time to complete one iteration with respect to the number of ants.

Number of Ants	libaco Time (s)	ExcellAnts Time (s)	Speedup
32	0.3032	0.0484	6.26
64	0.6077	0.0532	11.42
128	1.2124	0.0628	19.31
256	2.4238	0.0642	37.75
512	4.842	0.0912	53.09
1024	9.6713	0.1184	81.68

The expected scaling of a serial implementation with respect to ants should be O(m), as the tour construction is performed for every ant. With this assumption, a fitted linear regression was performed on the data. The correlation constant was 1.000. The residuals were patterned, but they were too small to acknowledge. r^2 was 0.9999, meaning that 99.99% of the variation in the data is explained by the theoretical scaling. This means that we can safely say the above model is correct.

The expected scaling of data-parallel implementation with respect to ants should be O($log(m)$), as all the ants' possible cities are reduced (in log(n) time) for a constant amount of cities. With this assumption, a fitted linear regression was performed on the data. The correlation constant was 0.9310. r^2 was 0.8665, meaning that 86.65% of the variation in the data is explained by the theoretical scaling. Although this is a relatively low r^2 value, the residuals are very scattered, and show absolutely no clear pattern. This means that we can safely say the above model is correct.

5.2.3 Experiment 4

The third test of scaling capability considered both ants and cities. The number of ants was set to the recommended amount, the number of cities[8]. The city sets for this test were the first n cities of usa13509.tsp[12]. The results are shown for both implementations graphically in figures 6 and 7. Note that the vertical axes have different values. Note that a maximum speedup of 340 was achieved.

Table 4: The time to complete one iteration with respect to the number of cities and ants.

Number of Cities and Ants (n = m)	libaco Time (s)	ExcellAnts Time (s)	Speedup
32	0.0169	0.0099	1.71
64	0.1429	0.0281	5.09
128	1.2123	0.0631	19.21
256	10.4507	0.1861	56.16
512	89.6375	0.6784	132.13
1024	756.8478	2.2204	340.86

Figure 6: The time for libaco to complete one iteration with respect to the number of cities and ants.

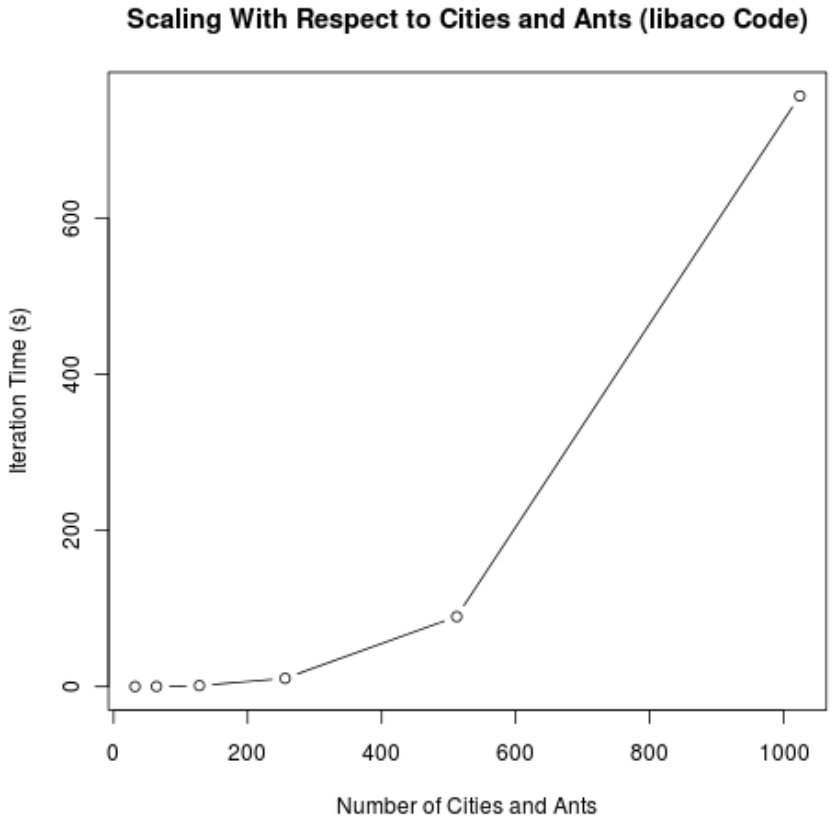
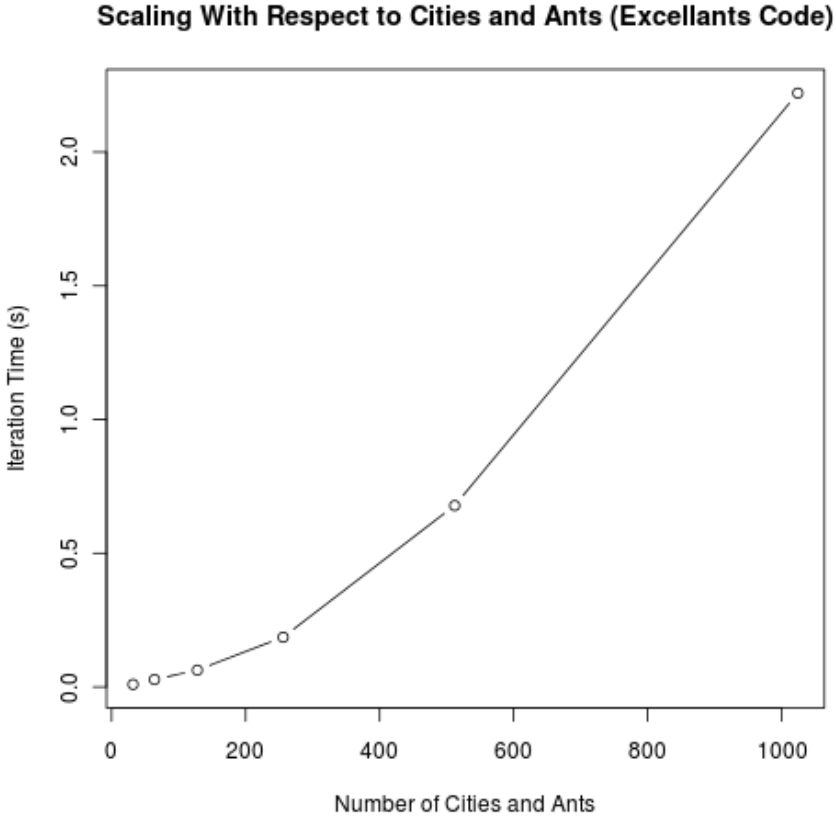


Figure 7: The time for ExcellAnts to complete one iteration with respect to the number of cities and ants.



The expected scaling of a serial implementation with respect to cities and ants should be $O(n^2m)$ or $O(n^3)$ $(n = m)$, as each city must be examined at every city in each ant's tour. With this assumption, a fitted linear regression was performed on the data. The correlation constant was 1.000, and the residuals were scattered. r^2 was 0.9999, meaning that 99.99% of the variation in the data is explained by the theoretical scaling. This means that we can safely say the above model is correct. The expected scaling of our data-parallel implementation with respect to cities and ants should be $O(nlog(n))$ as all the cities for all the ants are reduced (in log(n) time) for every city in the tour. This is the same scaling equation that we saw in Experiment 2 for the serial code. The reason for this is that although the cities must be

evaluated for every ant, they are all reduced at the same time, producing $O(nlog(nm))$, or $O(nlog(n^2))$ $(n = m)$, which reduces to $O(nlog(n))$. With this assumption, a fitted linear regression was performed on the data. The correlation constant was 0.9982, and the residuals were scattered. r^2 was 0.9765, meaning that 97.65% of the variation in the data is explained by the theoretical scaling. This means that we can safely say the above model is correct.

5.3 Quality

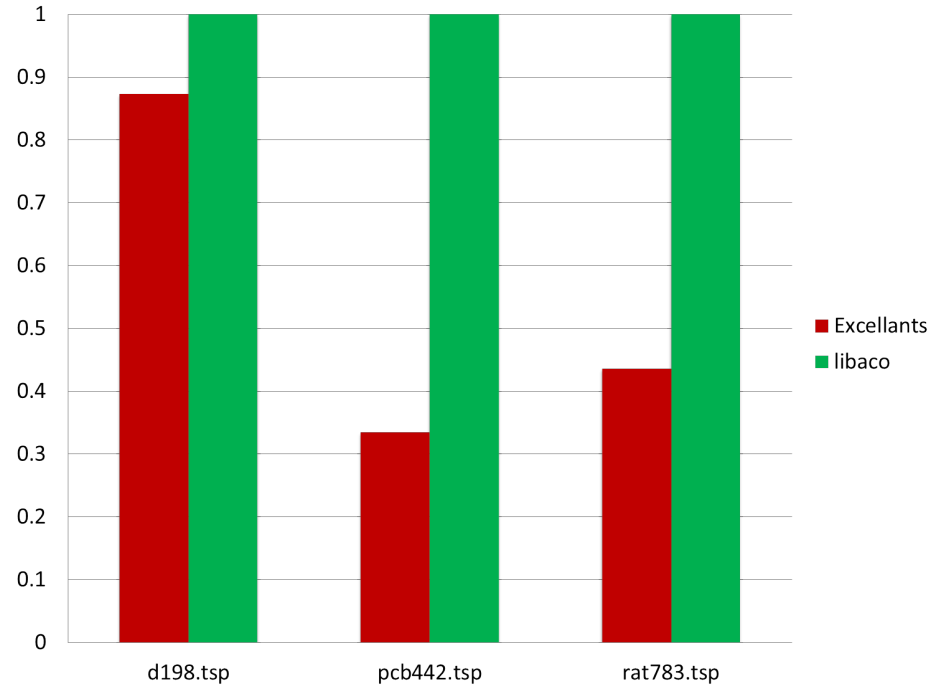
5.3.1 Experiment 5

One could think of performance as getting a faster solution or getting a better solution. In this experiment, the two implementations were compared by their end tour quality. The following tests were run on Computer A.[1] Both libaco and ExcellAnts tour qualities were measured at the end of 100 seconds on different datasets from TSPLIB. [12] The results are presented in Table 5 and the proportional improvement can be visually compared in Figure 8. The Excellants quality was up to three times better.

Table 5: The best tour lengths generated by each ACO in 100 seconds.

TSP	d198.tsp	pcb442.tsp	rat783.tsp
ExcellAnts	16536	64727.6	3165.93
libaco	18941	193819	7263.16
libaco/ExcellAnts	0.87	0.33	0.44

Figure 8: The proportional difference in best tour lengths generated by each ACO in 100 seconds.



6 Conclusions

A data-parallel ACO on the GPU that is easy to program and is portable to multiple targets has its advantages. First and foremost, it's faster. The parallel code outperformed the serial code in all cases.(see Tables 3, 2, 4) We achieved a speedup over the libaco code of about 100 with 512 ants and 512 cities. The speed can also lead to better quality. In some cases, ExcellAnts achieved three times the quality of a serial implementation. Also, it scales much more efficiently. If one ran their simulations at the recommended number of ants, libaco would scale at $O(n^3)$ while ExcellAnts would scale at $O(nlog(n))$. This project successfully created a working data-parallel implementation of ACO that significantly outperformed the serial version, with vast improvements in scaling capability. Even a simple task-parallel approach would only reach a scaling of $O(n^2)$ with respect to cities and ants, as all cities must still be evaluated at each city in the tour. Our program was

also portable to different targets. We were able to run our program with OpenMP. Our code also made use of an off the shelf data parallel library, for ease of programming.

7 Significant Original Achievement

ExcellAnts has made some important contributions. Firstly, we wrote an efficient, high-quality data-parallel implementation of ACO (via Thrust [11]) that is portable to multiple targets (CUDA, OpenCL, OpenMP, or Thread Building Blocks) without having to change the code. Secondly, the methods used to construct the tree-based selection process are described in detail. This is important to anyone attempting their own implementation using our methods. Thirdly, our code is also available and open source, both in this report and online.[4] Fourth, the time to build this efficient, portable implementation was less than what it would have taken to create a hand-tuned implementation on each target, since Thrust provides a library that is easy to use. These advantages are important to anyone looking to use our work in a practical application or to extend it in the research world.

8 Related Work

It came to our attention as this paper was being written that a paper on a hand-tuned GPU implementation of ACO was available online in January 2012.[7] However, all the original ideas presented here were conceived independently from this work. We did not read it until we had done our implementation and started writing our paper. We have had several original achievements, and actually some improvements over this work. The first one of these is the tree-based selection process. This sped up our code considerably and also allows it to scale to larger data sets more efficiently. Ours is described in detail, their's is not. The other methods initially attempted were also of our own design. Our code is also written in Thrust, while theirs is written in CUDA and highly optimized to a GPU, making it difficult to understand and non-portable. While their code is not currently available, The code for ExcellAnts is available in an open source format online.[4]

9 Future Work

This report does not mark the end of our project. We aim to continue our work on our Google Project page.[4] We have several plans which we expect to apply to the code in the future. One of the most prominent of these is that we plan to use our ACO on a dynamic traveling salesman problem (DTSP) and optimize it for use with DSTPs. Another plan is to overlap CPU and GPU computation by running suitable operations on both processors in order to maximize the use of the computational power of the system on which it is running. Yet another goal for us to work toward in the future is to add more implementations of ACO to give the user more flexibility.

10 Work Products

10.1 Code

```
/**************************************
 * Setup.cpp                          *
 * Peter Ahrens                       *
 * Sets up the ACO                    *
 **************************************/

#include "RankBasedAntSystem.h"
#include "TSPReader.h"
#include "Comm.h"
#include "Writer.h"
#include <iostream>
#include <unistd.h>
#include <string>
#include <cctype>
#include <ctime>
using namespace std;

//Setup: The main control loop to the whole program.
int main(int argc, char* argv[]){
  //declare variables
  cout << "|SETUP|\n";
  string antHillType = "RBAS";
  int m = -1;
  int ranks = 6;
  int maxTime = 0;
```

Simulation of Multi-Agent Based Scheduling Algorithms for Waiting-line Queuing Problems

New Mexico Supercomputing Challenge

Final Report

April 4th, 2012

Team 73

Los Alamos Middle School

Team Members

Steven Chen

Andrew Tang

Teacher

Pauline Stephens

Project Mentor

Hsing-bung (HB) Chen

Table of contents

Executive Summary

In this project, we designed and implemented a multi-agent computer simulation program. We used this simulation software to model a real-life waiting line or queuing problem in variety of business and industrial situations such as supermarket's checkout lines and bank's teller service windows. Through our experiments we addressed the following issues: (1) How to model an independent task scheduling problem (single waiting queue (multiple servers with Multiple service queues) using NetLogo multi-agent simulation system, (2) How to provide an interactive approach to control run-time simulation activities, (3) How to collect performance data and justify implemented scheduling methods, and (4) Is it possible to create an useful Multi-agent education tool to teach scheduling problem. To solve the problems presented above, we would like to apply efficient scheduling solutions. Scheduling is a key concept in computer multitasking, the multiprocessing operating system and real-time operating system designs. Scheduling refers to the way processes assigned to run on available CPUs. This assignment is normally carried out by software known as a task scheduler or a job dispatcher. The "NetLogo" is an agent based modeling software tool that we can use to create and investigate various models for application problems. In reality, there is no universal scheduling algorithm to solve real-life waiting-line or queuing problems. However, using heuristic approaches is the most reasonable way to obtain acceptable solutions. We implemented five scheduling algorithms—round-robin, random selection, early start time first, less workload first, and a mixed selection heuristic that combines the four previously listed methods. The rich property of the random number generator in "NetLogo" is an excellent tool to generate random task behaviors such as task size and task arriving time. We conducted testing cases to cover various task patterns on our NetLogo simulation program. We defined and collected various performance matrices such as waiting time, turnaround time, and queue length. We found the Early Starting Time First algorithm to be the best heuristic in most of the testing cases. For example, it can obtain a shorter waiting time and average queue length and a faster turnaround time. We also demonstrated that our interactive multi-agent simulation program is a good tool to teach multi-processing task scheduling problem.

1. Problem statement

Waiting line queuing problems are commonly seen in everyday life. Some typical examples are:

1. Supermarkets must decide how many cash registers should be opened to reduce customers' waiting time.
2. Gasoline stations must decide how many pumps should be opened and how many attendants should be on duty.
3. Manufacturing plants must determine the optimal number of mechanics to have on duty in each shift to repair machines that break down.
4. Banks must decide how many teller windows to keep open to serve customers during the various hours of the day.
5. Peer-to-Peer, Grid, and Cloud computing need to effectively and quickly manage and schedule distributed resources for solving large-scale problems in science, engineering, and commerce.
6. Modern large scale HPC cluster machines need to schedule millions of processes or threads so it can provide fast turnaround time and better machine utilization.

Whether it is waiting in line at a grocery store to buy deli items (by taking a number), checking out at the cash registers (finding the quickest line), waiting in line at the bank for a teller, or submitting a batch job to available computers, we spend a lot of time waiting. The time you spend waiting in a line depends on a number of factors including the number of people (or in general tasks) served before you, the number of servers working, and the amount of time it takes to serve each individual customer/task.

To deal with the problems mentioned above, we must provide effective and reasonable solutions in order to reach goals of minimizing wait time in a queue, minimizing turnaround time, balancing workload among service points, and increasing server utilization. To simplify our project's problem description, we would like to formalize a waiting line and queuing problem as a task scheduling problem. We can treat all objects (clients, customers, mechanics, tellers, messages, jobs, processes, threads) waiting in a service line by putting them into a queue as tasks. The task scheduling problem is very challenging and interesting. This problem is a class of hard problems that cannot be optimally solved in a reasonable amount of computation time.

For this reason, researchers have spent the past several decades developing work heuristic (rule of thumb) methods to try and find a near-optimal solution.

The main goal of our project is to design and implement a multi-agent simulation model for task scheduling problems and provide an interactive software tool to learn distributed task scheduling problems. Now, why are we using simulation? Simulation appears to be the only feasible way to analyze algorithms on large-scale distributed systems using various resources. Unlike using the real system in real time, simulation works well, without making the analysis mechanism unnecessary complex, by avoiding the overhead of co-ordination of real resources. Simulation is also effective in working with very large hypothetical problems that would otherwise require involvement of a large number of active users and resources, which is very hard to coordinate and build at large-scale research environment for an investigation purpose.

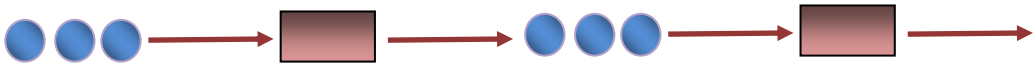
2 Multi-agent task scheduling simulation design and implementation

2.1 Simulation queue model

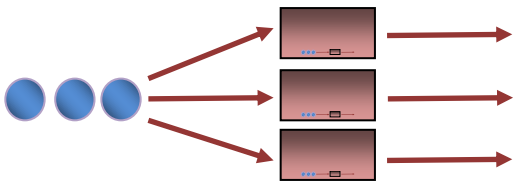
There are several Waiting line 's Queue models:  Task,  processing point

- Single-server, single-phase

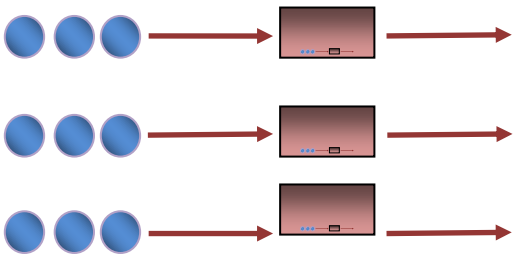
- Single-server , multiphase



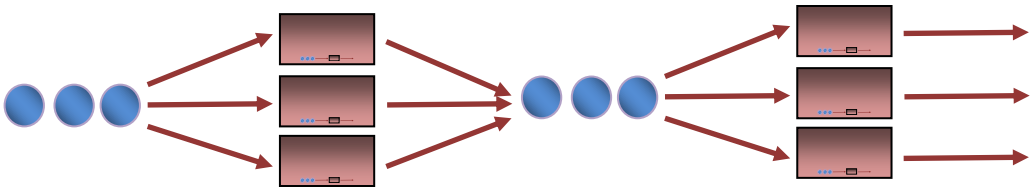
- Multi-server, single-line single-phase: centralized scheduler



- Multi-server, multiline, single-phase



- Multi-server, multiphase



In this project, we implement the multi-server, single-line, and single-phase queue model. This model typically and simply represents an independent task scheduling model on a distributed computing system such as Cluster, GRID or Cloud computing environment. A centralized task scheduler handles many randomly arrived tasks and finds available processing points to execute tasks.

2.2 Multi-agent task scheduling simulation System

Our multi-agent based models are composed of three different types of agents: the schedule agent, machine agent, and task agent.

The agents in a multi-agent system have several important characteristics:

- **Autonomy**: the agents are at least partially autonomous
- **Local views**: no agent has a full global view of the system, or the system is too complex for an agent to make practical use of such knowledge
- **Centralization and decentralization**: there is a designated and centralized scheduling agent and there are number of decentralized task agents randomly ask the scheduler to schedule a created "task" on a selective machine agent.

Figure 1 shows the system diagram of our simulation agents.

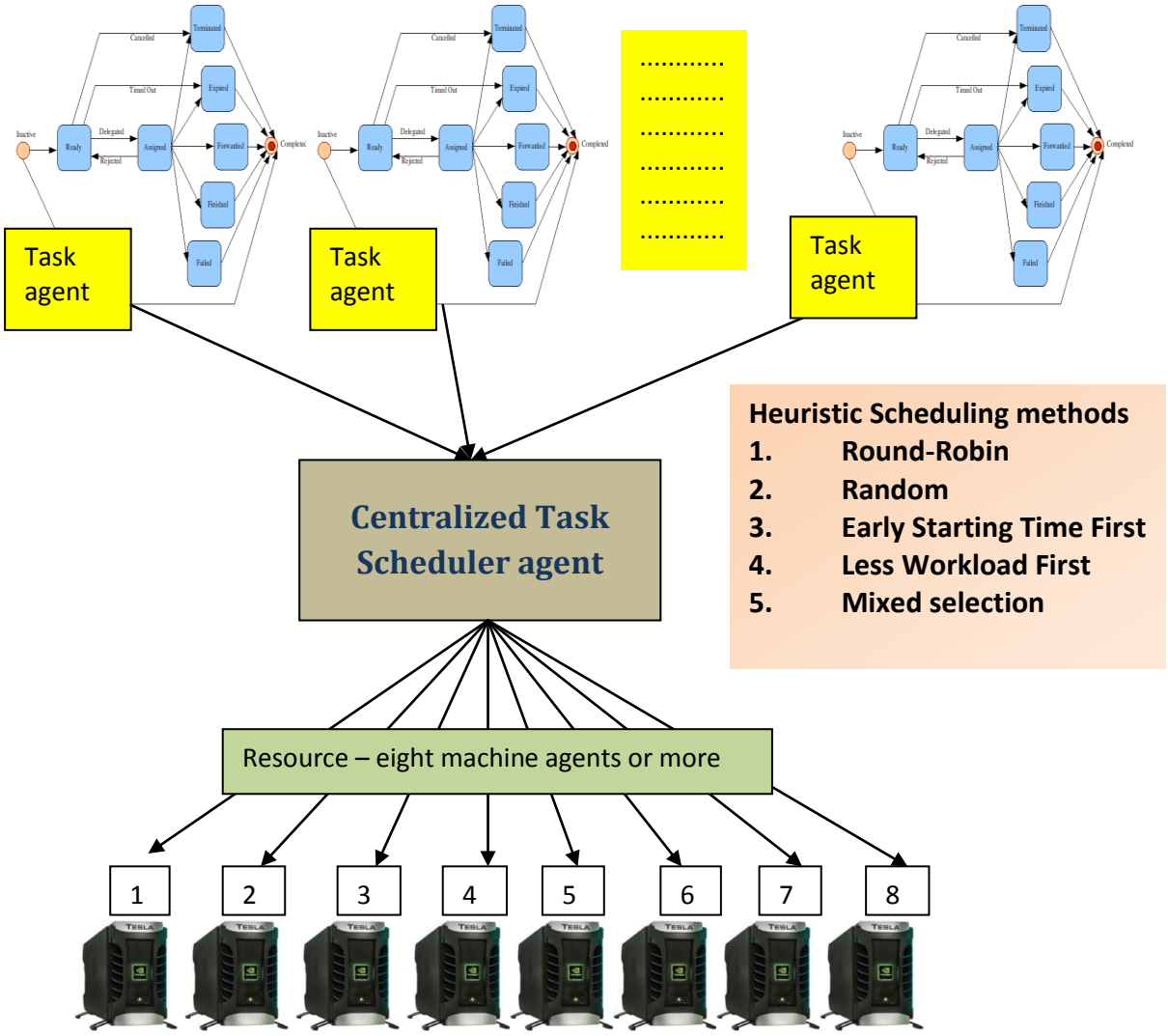


Figure 1: Multi-agent task scheduling simulation

2.3 Agent designs

Scheduler agent:

Only one centralized scheduler agent is defined here. We use the scheduler agent to receive scheduling requests from task agents and find available machine agents to run tasks. The selection of an available machine is based on the selected scheduling heuristic.

Machine agent:

A Machine agent is the main system resource to schedule a task. We have used eight and sixteen machine agents in this simulation. Each machine agent is used to receive a task assignment request from the scheduler agent and update its avail-time, accumulate-task-time, and idle time. We can simulate more machine agents but we have considered to provide and interactive approach and construct an education tool. Enploying eight to sixteen machine agents in our simulation is within a reasonalbe range to view -real-time task schedule activities on a monotoring screen.

Task agent:

We have used up to 99,999 task agents in our similation. Each task agent comes with a different arriving time and task execution length. We used various number of task agents to represent different run-time environments such as light workload (hundred tasks), moderate workload (thousand tasks), heavy workload (multiple ten thousand tasks), lots of small tasks (small execution time), lots of large tasks (very long execution time), or mixed small and large task sizes etc..

Task information

Property of a Task:

- Each scheduling task is an independent task. There is no dependency relation or related execution order between tasks
- Each task has been assign a random execution time, i.e. the length of a task
- Each task has been assigned a random arriving time
- Task arrival times are not known a priori. Every task has the attributes arrival time, worst case computation time, and deadline. The ready time of a task is equal to its arrival time. Task's arriving time is generated by a selected random number generator.
- Tasks are non-preemptive; each of them is independent.

We used two different random number generator provided by ~~the~~ NetLogo— random and Poisson-random. We used a random number generator to generate a task's arriving time and a task's execution time.

Task selection discipline :

The selection of a task is based on the First Come First Serve FCFS) order. The NetLogo system decides the task order in a queue when there are multiple tasks arrive at the same time tick.

2.4 Heuristic scheduling methods

We implemented five different decision-making heuristics. Various heuristic scheduling methods represent the intelligence and capabilities of each method. The heuristic is how we select a machine to execute an arriving task.

2.4.1 Round-robin method

The scheduler agent uses a round-robin order to select each machine agent and assigns an arriving task to it. Each machine agent takes an equal share of responsibility to run a task in turn.

2.4.2 Random Selection

The scheduler agent randomly selects an available machine agent and assigns ~~an~~ arriving tasks to it. It randomly selects a machine to run an incoming task.

2.4.3 Less Workload First

The scheduler agent selects an available machine agent with the smallest accumulated task workload and assigns an arriving task to it.

2.4.4 Early Starting Time First

The scheduler agent selects an available machine agent with the early task starting time to run a task and assigns an arriving task to it.

2.4.5 A Mixed Selection of the Above Four Heuristic Methods

A mixed selection of the above scheduling methods can be called a heuristic of heuristics. For each arriving task, the scheduler agent randomly picks one of the ~~above~~ four heuristic methods mentioned above and applies this selected method to find an available machine agent, and then assigns an arriving task to it.

2.5 Time step simulation and Task interactive sequence

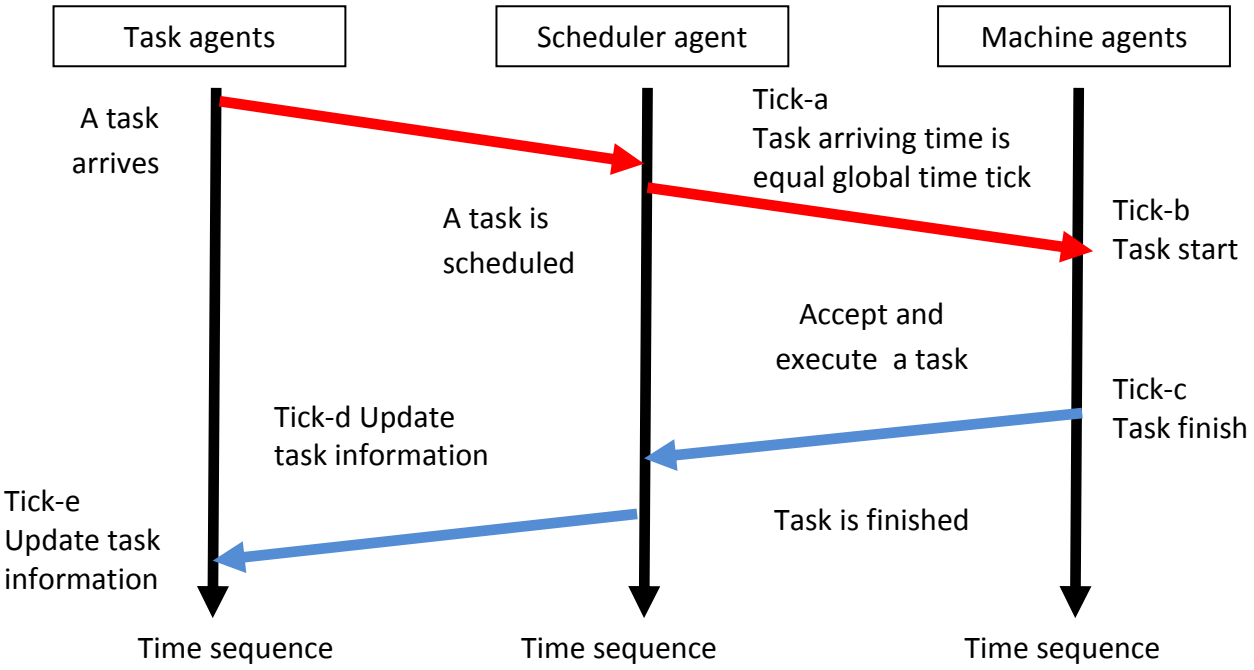


Figure 2: Interaction between agents based on time sequence

A global time tick is used in the simulation. This global time tick is advanced by "N" time ticks. “N” can be any number. We advanced one time tick each time. The global time tick is used as the wall clock and we used it to check the task arriving time. We also use it to monitor task activities such as task waiting, task scheduling, task execution, and task finishing. We used the global time tick to collect performance data. When a task's arriving time is equal to the current global time tick, this task agent will ask the scheduler agent to scheduler agent to find an available machine agent to execute it. Figure 2 shows the interaction between agents based on the time advanced sequence.

In Figure 3, we show the simulation NetLogo program architecture. "Ask" is the keyword used to query each agent about its status and expect activities. "Ask" also represents the required interactive activities between agents. A task agent checks its task-arriving time and the global time tick and sees if its task is ready to be scheduled. The scheduler agent use a selected heuristic method to find an available machine "X" and then ask the machine agent "X" to accept the arriving task and execute it. These interactive activities among agents are continuing until all tasks are scheduled and finishing execution. We collected performance data during the whole simulation process.

One of our goals for this simulation project is to provide education tolls for task scheduling problems. We adapted a visualized and interactive approach to build this simulation. We let users to define the run-time environment while we provided run-time animations of task scheduling activities and in-time performance data display during the whole simulation process.

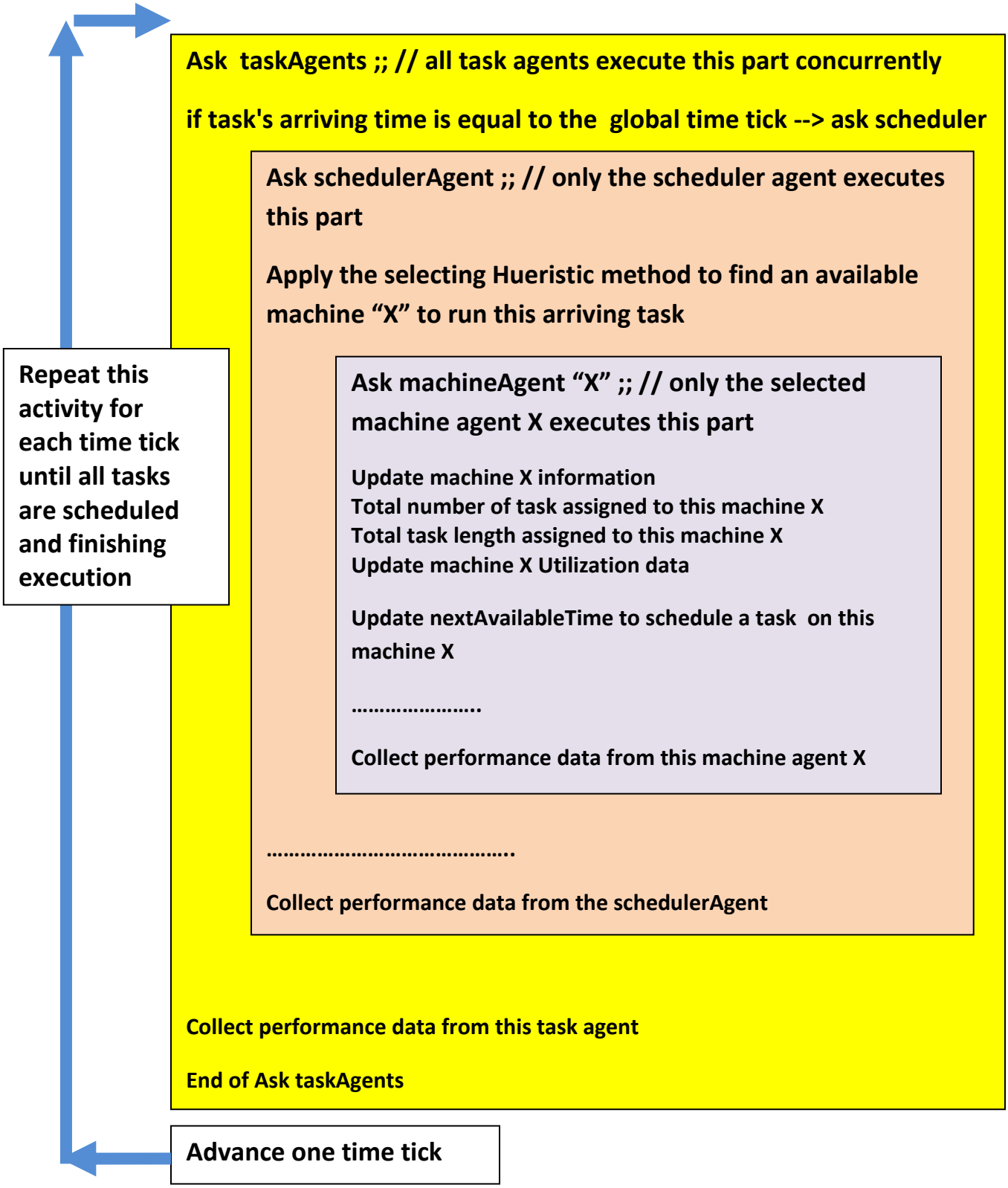


Figure 3: NetLogo simulation program -interaction between agents

2.6 Main Screen Design and Implementation

The Global tick count is shown in Figure 4-1.

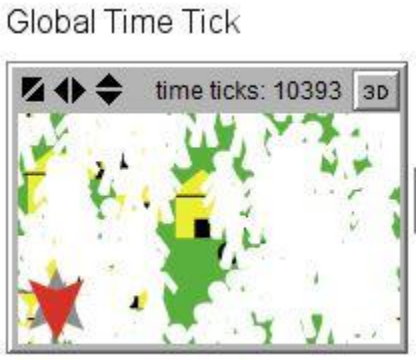


Figure 4-1: Global time tick counter

Users define parameters for simulation (Figure 4-2)

- the number of task agents used in each simulation,
- the range of task arriving time distribution,
- the range of task execution time distribution,
- the number of machine used in simulation

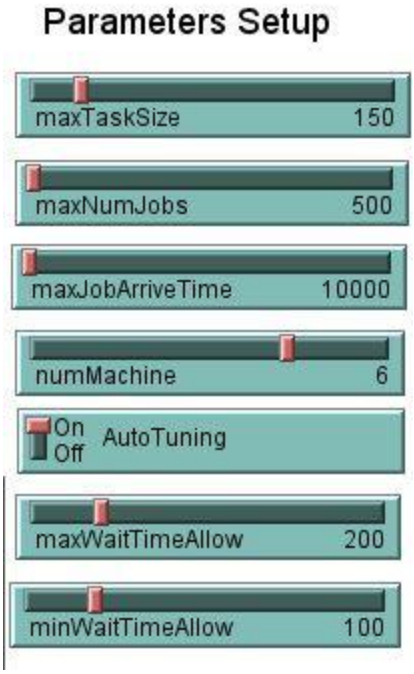


Figure 4-2: Setup testing parameters

Users select a random number generator used in simulation (Figure 4-3).

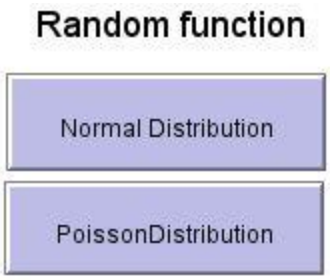


Figure 4-3: Select random number generator

Users select a scheduleing method used in simulation (Figure 4-4).

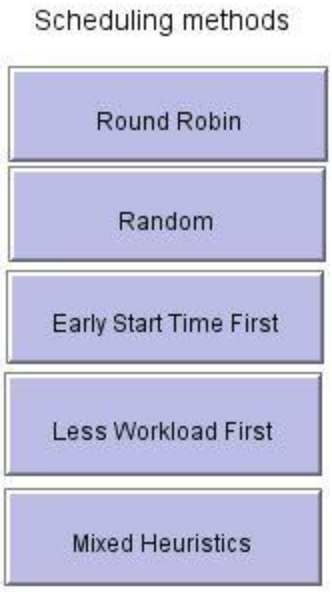


Figure 4-4: Select schedule method

Users interactively control simulation action (Figure 4-5).

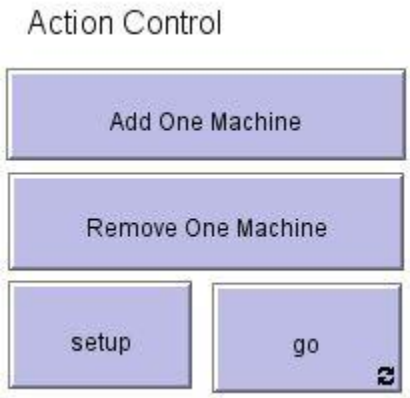


Figure 4-5: Run time control bottom

Monitoring run time performance data update and display (Figure 4-6)

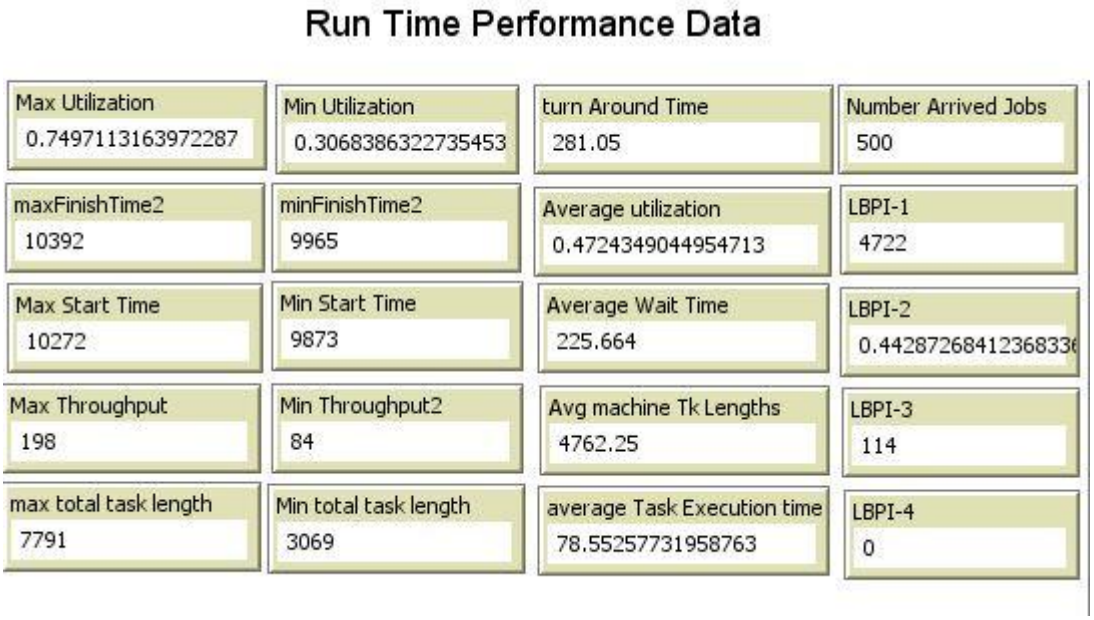


Figure 4-6: Rum Time Performance data update and Display

Messages area for displaying Testing Setup information and Run time activities is shown in (Figure 4-7)

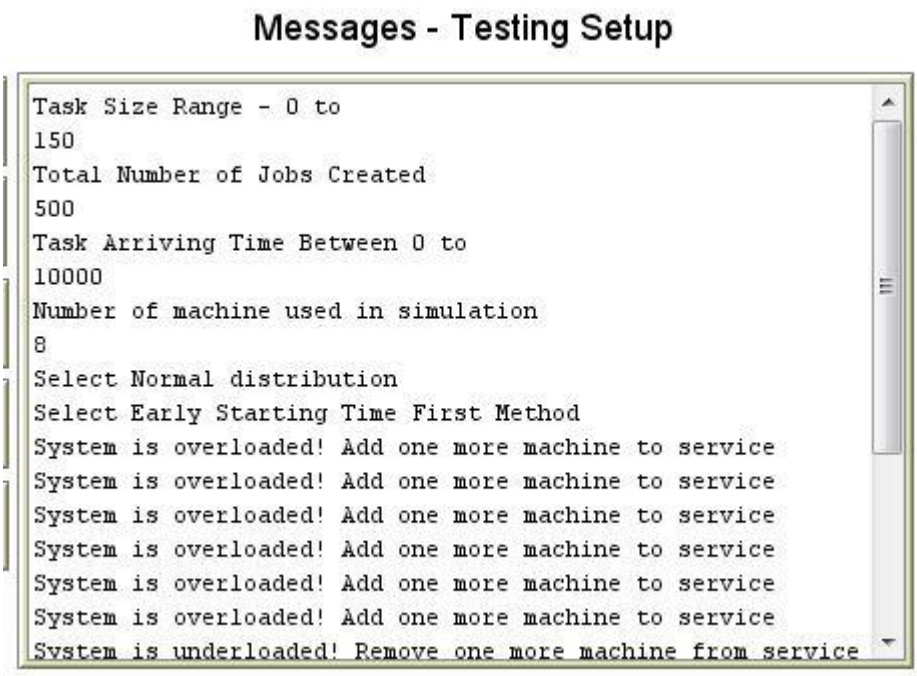


Figure 4-7: Message area for Testing setup and activities

Run Time visualization display for task scheduling activities is shown in Figure 4-8.

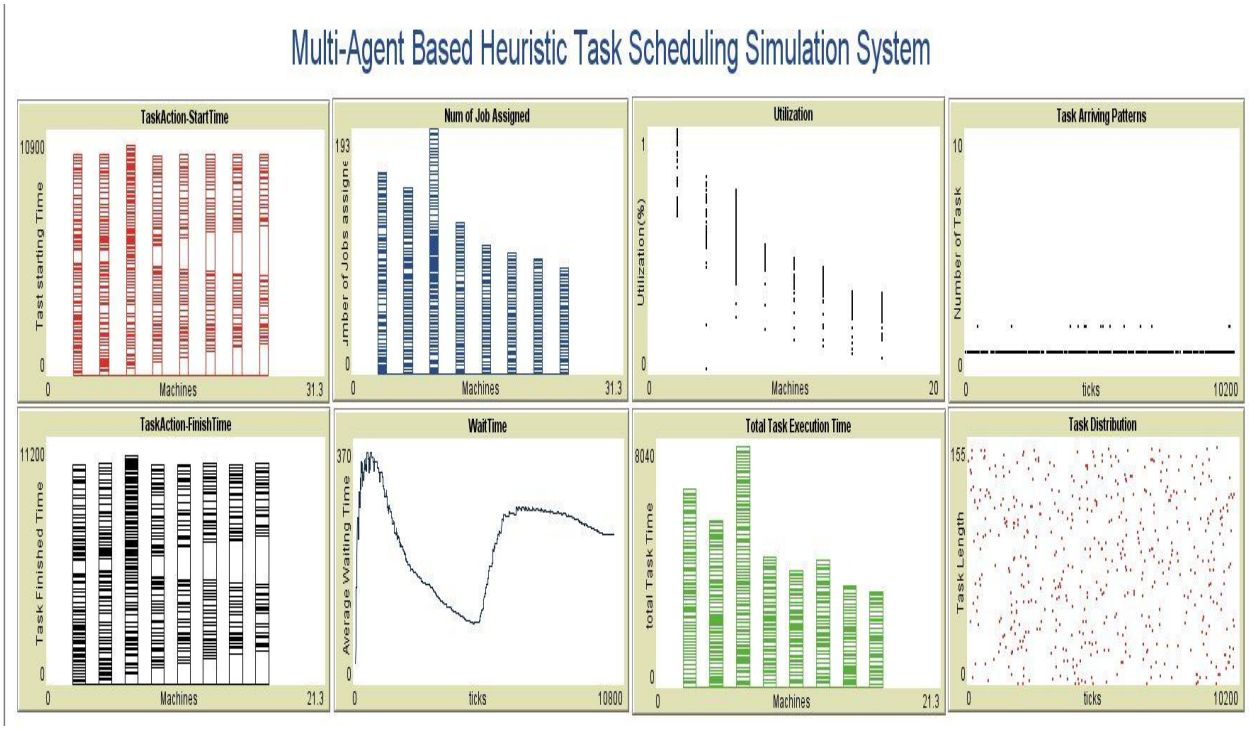


Figure 4-8: Run time visualization area

Figure 4-9 is the main screen for our multi-agent task scheduling simulation system.

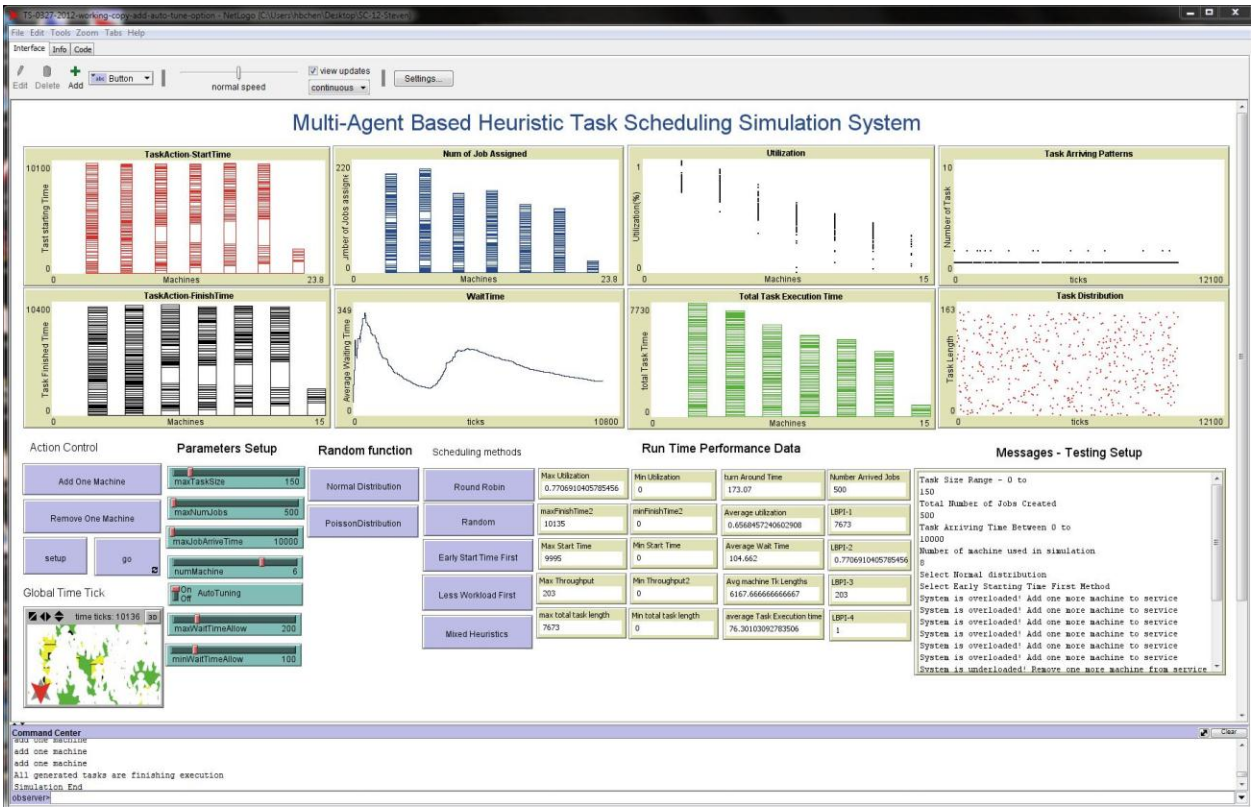


Figure 4-9: Main screen design

3 Testing and Performance data

We focused on the following performance Index:

a) The average number of tasks waiting in line on a machine agent - The number of tasks waiting in line can be interpreted in several ways. Short waiting lines can result from relatively constant task arrivals (no major surges in demand) or by the organization having excess capacity (too many machines open). On the other hand, long waiting lines can result from poor server efficiency, inadequate system capacity, and/or significant surges in demand.

b) The average time tasks spend on waiting in a queue,

c) The average time a task spends in the system - turnaround time

d) The system utilization rate - Measuring capacity utilization shows the percentage of time the machines are busy. Management's goal is to have enough machines to

assure that waiting is within allowable limits but not too many machines as to be cost inefficient.

3.1 Performance index definitions

We defined the following parameters in our simulation program and then collected them as performance data.

Global information - can be viewed and accessed by all agents

ticks : the global time tick as the reference wall clock

Number of Task: N, $Task_i$, 1 = 1 to N

Number of machine: M, $Machine_j$, j= 1 to M

Num of Scheduler: 1 , Scheduler

Task agent information:

Number of Task agent created : N, $Task_i$, 1 = 1 to N , i is referenced as the task ID

A Task i: $Task_i$

Task i execution time : $TaskLength_i$

A random number generator is used to create a task's execution time

Task i arriving time : $TaskArrive_i$

A random number generator is used to create a task's arriving time

Task i start executing time: $TaskStart_i$

Task i finish execution time : $TaskFinish_i$

$TaskFinish_i = TaskStart_i + TaskLength_i$

Task i Waiting time in queue: $TaskWait_i$

is the time between task's arriving time and the actual task's start execution time

$TaskWait_i = TaskStart_i - TaskArrive_i$

TotalTaskWaitingTime(Sum of $TaskWait_i$, i= 1 to N)

AverageWaitingTime: Average task waiting time

AverageWaitingTime = TotalTaskWaitingTime / Number of Task arrived

Task i Turnaround time: $TaskTRtime_i$, the amount of time spend on waiting and execution

$taskTRtime_i = TaskFinish_i - TaskArrive_i$

TotalTaskTurnaroundTime(Sum of $TaskTRTime_i$, i= 1 to N)

AverageTurnaroundTime: Average task turnaround time

AverageWaitingTime = TotalTaskTurnaroundTime / Number of Task Finished

Machine agent information

Number of machine: M, $Machine_j$, j= 1 to M, j is referenced as the machine ID

We create two version of simulation. One is using eight machine agents and the other is using sixteen machine agents

A machine j : $Machine_j$

$MachTotalTaskTime_j$: Total task execution time on a machine j

$MachAvailableTime_j$: the current available time to add a new task on a machine j

MachIdleTimej : Machine j Idle time upto the current golbal time ticks

$MachIdleTimej = ticks - MachTotalTaskTime_j$

MinimunTotalTaskTime ($MachTotatlTaskTime_{j,}$ j = 1 to M)

The smallest total task time on a machine agent

MaximunTotalTaskTime($MachTotalTaskTime_{j,}$ j = 1 to M)

The biggest total task time on a machine agent

$MachUtilization_j$: A machine j current utilization

$MachUtilization_j = MachTotalTaskTime_j / ticks$

The less utilized machine : MiumumUtil($MachUtilization_j$, j = 1 to M)

The most utilized machine : MaxmumUtil($MachUtilization_j$, j = 1 to M)

The average machine utilization : Average($MachUtilization_j$, j = 1 to M)

EarlyStartTime(Minumum($MachAvailableTime_j$, j = 1 to M)): the early start time for a task from all machine agents

$MachFinishTask_j$: Number of tasks finished on a machine agent j at current global time ticks

This is the throughput on a machine agent j

MinimunThroughput(MachFinishTask$_{j,}$ j = 1 to M)

MaximunThroughput(MachFinishTask$_j$, j= 1 to M)

TotalThroughput(MachFinishTask$_j$, j= 1 to M)

TaskWaitingEueueLength$_j$: waiting queue lenght on a machine agent j

MinimunQueuLength(TaskWaitingEueueLength$_j$, j=1 to M)

The minimum queue length

MaximunQueuLength(TaskWaitingEueueLength$_j$, j=1 to M)

The maxinum queue length

Load Balance Performance Index :

Load balance Performance Index1 (LBPI1) =

MaximumTotalTaskTime - MinimumTotaltaskTime

Check the total task execution time assigned on a machine agent

Calculate the biggest gap among machine agents

Load balance Performance Index2 (LBPI2) = MaximumUtil - MinimumUtil

Check the machine utilization on a machine agent

Calculate the biggest gap of utilization among machine agents

Load balance Performance Index3 (LBPI3) = MaximumThroughout - MinimumThroughout

Check the total task execution time assigned on a machine agent

Calculate the biggest gap of throughout among machine agents

Load balance Performance Index4 (LBPI4) =

MaximumQueueLength - MinimumQueueLength

Check the total task execution time assigned on a machine agent

Calculate the biggest gap of waiting queue length among machine agents

We defined these four Load Balance Performance Index to measure the capability of providing a local balance run-time environment for each task scheduling method. The less

variation of the LBPI value indicates a better load balancing result that is there is not a large difference between the "min" and "max". For example, we use the LBPI2 to check whether a scheduling method can assign even workload to each machine.

3.2 Testing cases

We used the task pattern

- Task size distribution: range 0 to 720 time ticks
- Number of Task agents : 1000 task agents created
- Task arriving time distribution: range from 0 to 100000 time ticks
- Using the default normal random number generator

We measured both strong scaling and weak scaling performance.

3.2.1 Strong scaling testing cases

In this strong scaling test case, the problem size stays fixed but the number of machines used is increased.

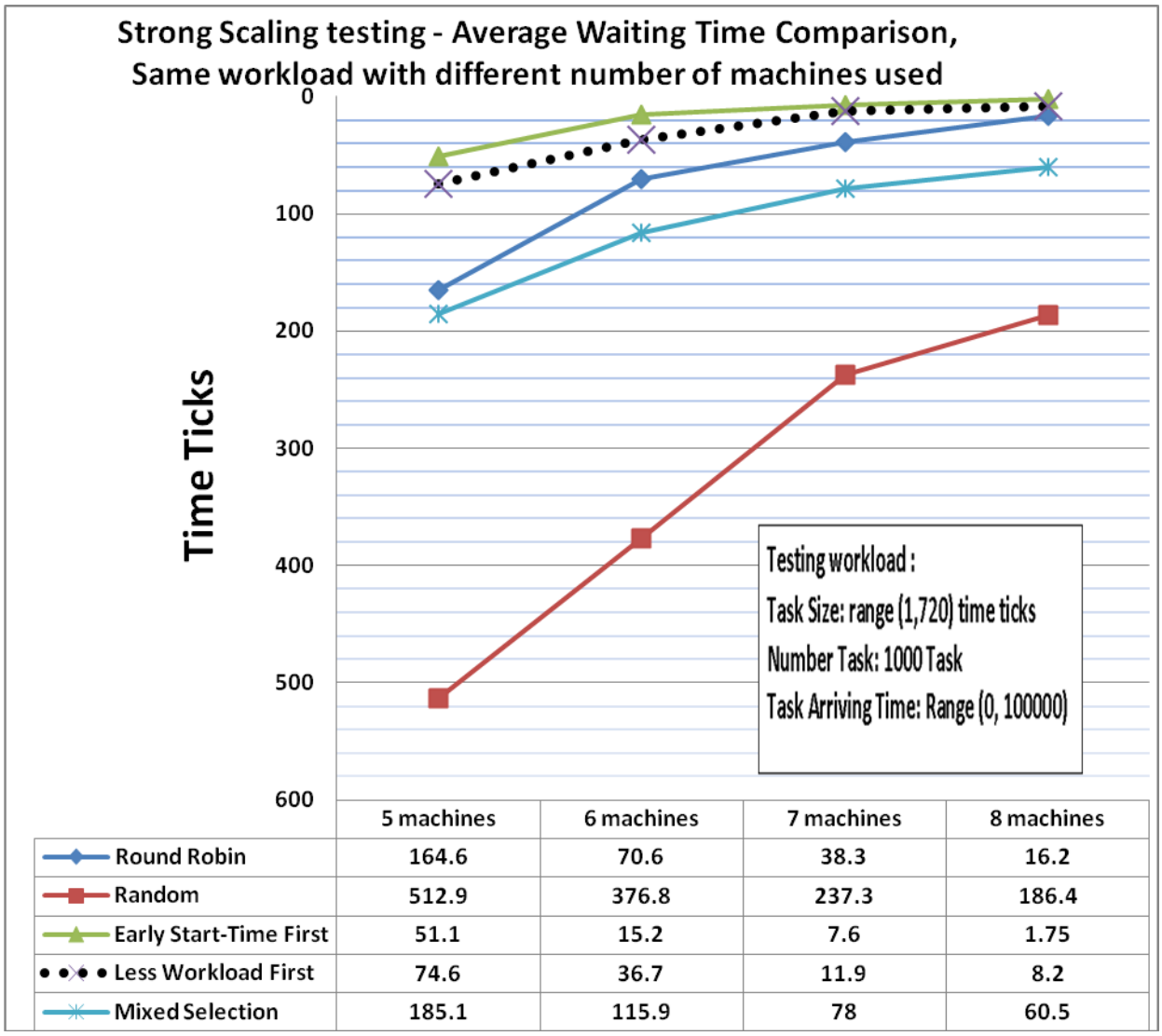
In strong scaling testing, we apple the above task pattern and start with using five machine in testing. For each round of testing, we applied the same workload and then increased one machine for each round of testing until we used up to eight machines.

Figure 5-1 shows the result of average waiting time comparison. The result is shown that every scheduling can reduce the waiting time when more machines are added to the simulation. We use a normalized ScalingIndex to compare the average waiting time. Table-1 shows the formula we used to calculate the ScalingIndex. The higher ScalingIndex value indicates a better scaling result.

	Average Waiting time	Normalized ScalingIndex-Average waiting time
Using 5 machines	**A**	**A/A = 1**
Using 6 machines	**B**	**A/B**
Using 7 machines	**C**	**A/C**
Using 8 machines	**D**	**A/D**

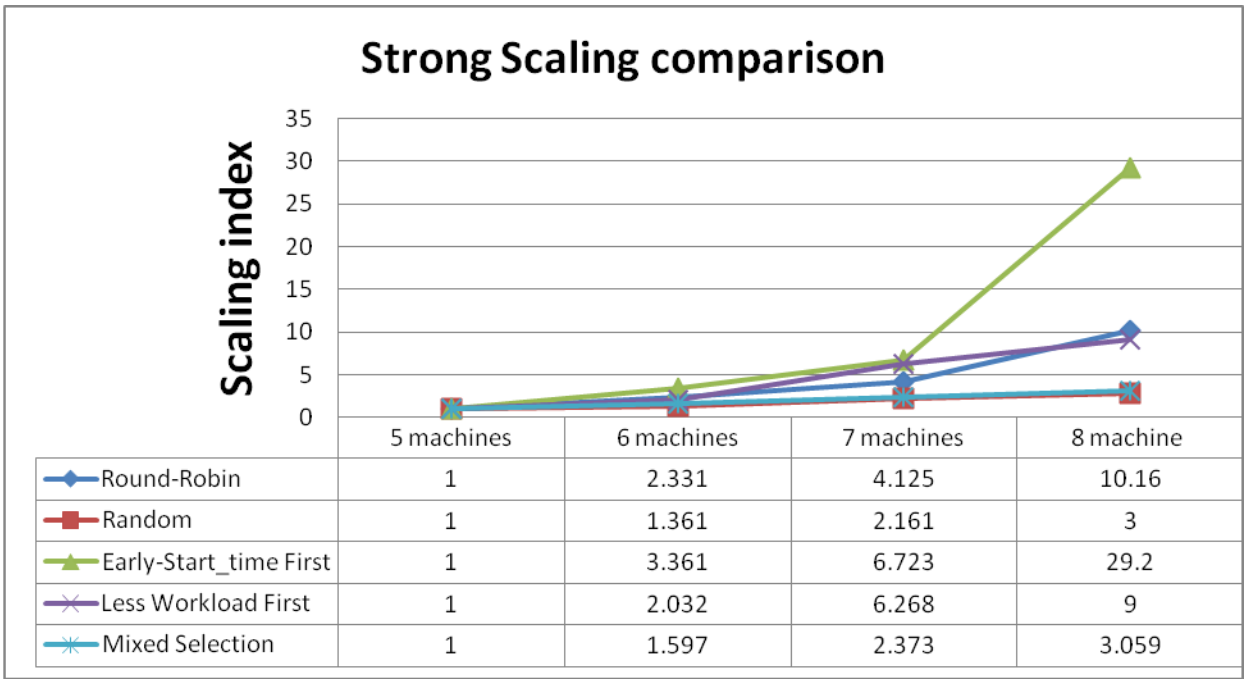
Table-1: Normalized ScalingIndex for Average Waiting Time

Figure 5-2 shows the ScalingIndex: Average Waiting Time comparison. The early Start-Time First method demonstrates as a better scheduling method in terms of strong scaling comparison.



	5 machines	6 machines	7 machines	8 machines
Round Robin	164.6	70.6	38.3	16.2
Random	512.9	376.8	237.3	186.4
Early Start-Time First	51.1	15.2	7.6	1.75
Less Workload First	74.6	36.7	11.9	8.2
Mixed Selection	185.1	115.9	78	60.5

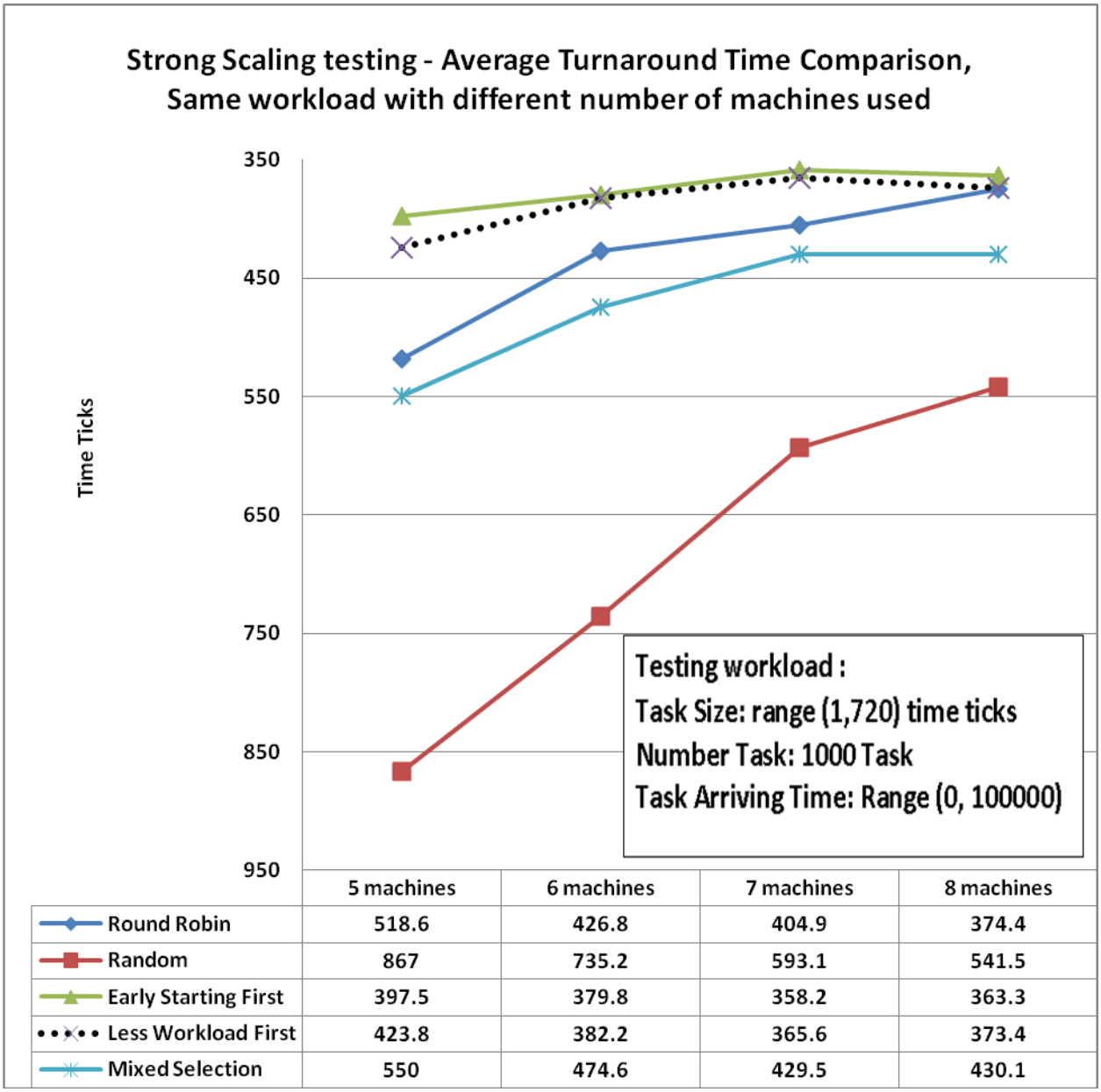
Figure 5-1: Strong scaling testing and average waiting time comparison, same workload with different number of machines



	5 machines	6 machines	7 machines	8 machine
Round-Robin	1	2.331	4.125	10.16
Random	1	1.361	2.161	3
Early-Start_time First	1	3.361	6.723	29.2
Less Workload First	1	2.032	6.268	9
Mixed Selection	1	1.597	2.373	3.059

Figure 5-2: Normalized ScalingIndex for Average waiting Time comparison

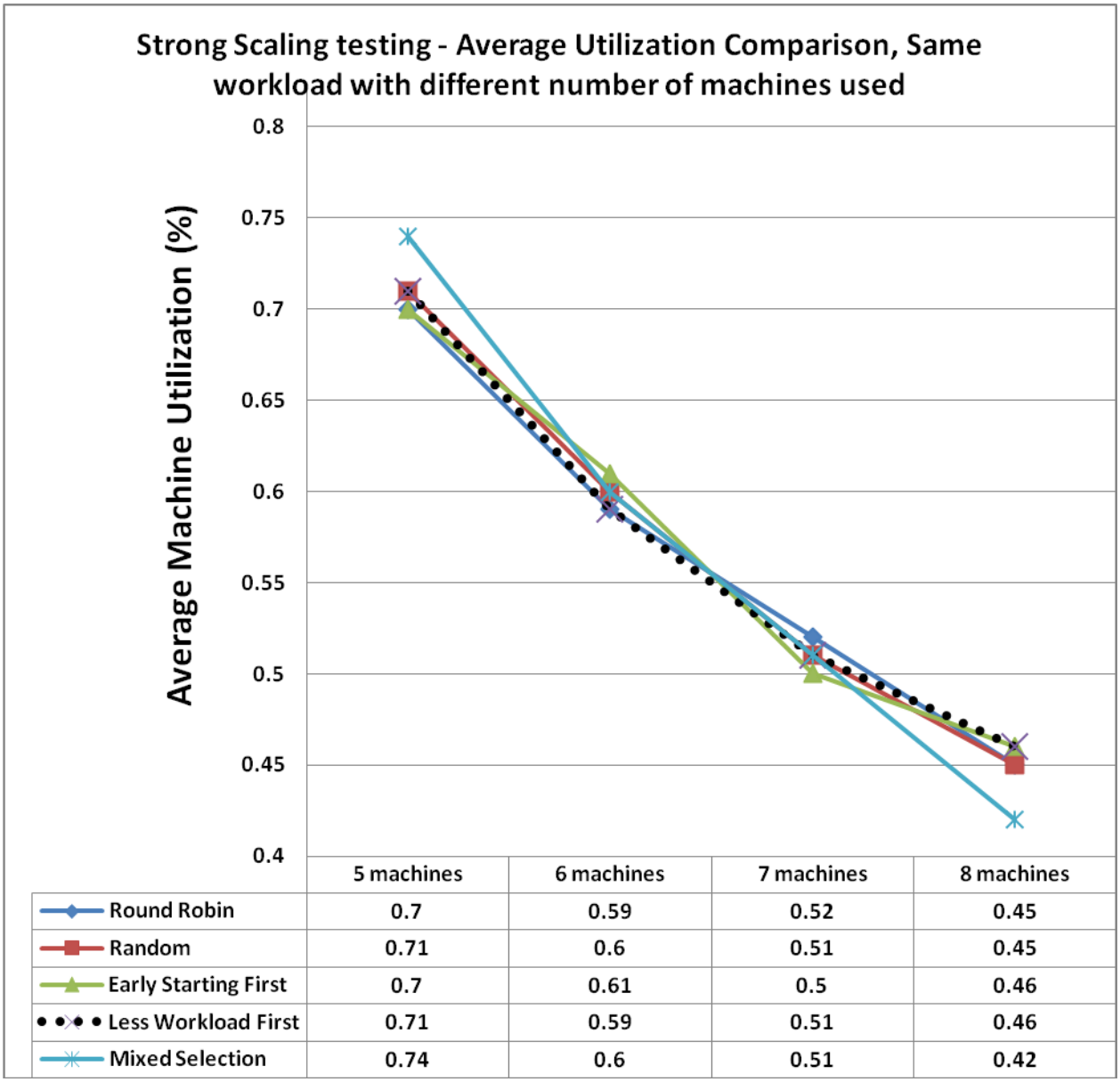
Figure 5-3 shows the result of average turnaround time comparison. This result shows that every scheduling can reduce the average turnaround time when more machines are added to the simulation. The Early Start-Time First heuristic can obtain the lower average turnaround time.



	5 machines	6 machines	7 machines	8 machines
Round Robin	518.6	426.8	404.9	374.4
Random	867	735.2	593.1	541.5
Early Starting First	397.5	379.8	358.2	363.3
Less Workload First	423.8	382.2	365.6	373.4
Mixed Selection	550	474.6	429.5	430.1

Figure 5-3: Strong scaling testing and average turnaround time comparison, same workload with different number of machines

Figure-5-4 shows the average machine utilization in string scaling testing case. We observed that there is not a large variation among all five scheduling methods. It seems that the strong scaling testing cannot clearly help us pinpoint the pros and cons of each scheduling heuristic.

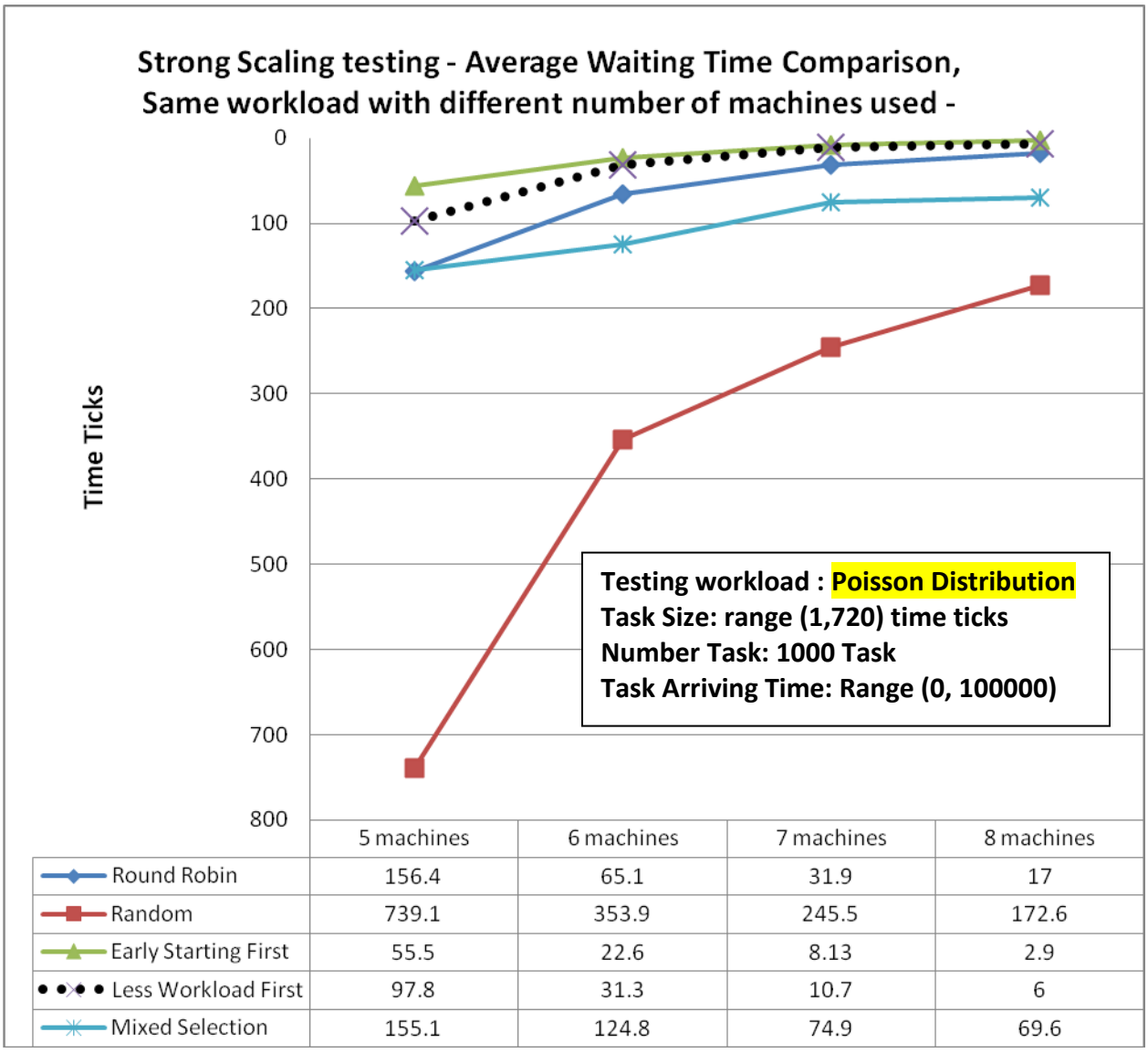


	5 machines	6 machines	7 machines	8 machines
Round Robin	0.7	0.59	0.52	0.45
Random	0.71	0.6	0.51	0.45
Early Starting First	0.7	0.61	0.5	0.46
Less Workload First	0.71	0.59	0.51	0.46
Mixed Selection	0.74	0.6	0.51	0.42

Figure 5-4: Strong scaling testing and average utilization comparison, same workload with different number of machines

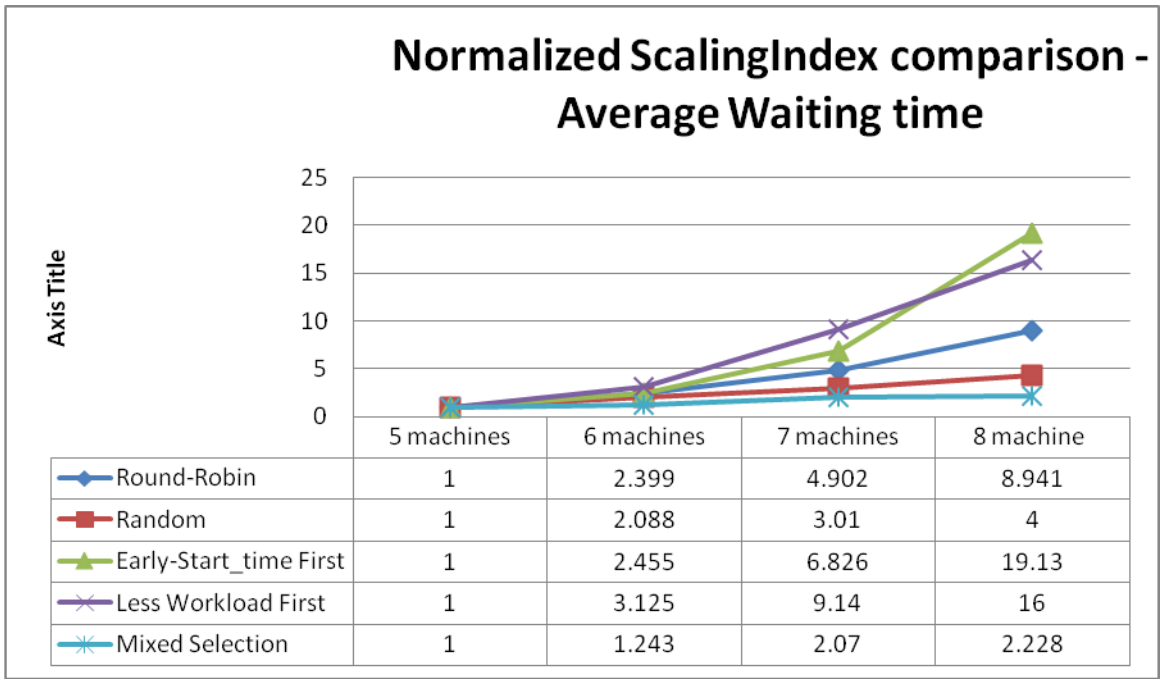
We also used the Poisson-random-number generator and repeated the strong scaling testing above. We present results in Figure 6-1, Figure 6-2, Figure 6-3, and Figure 6-4. Our results has show that there is no much difference of using the Normal random number generator and the Poisson random number generator provide in the NetLogo simulation system. The only

big difference is that the Less Workload First has a better normalized ScalingIndex when using eight machines.



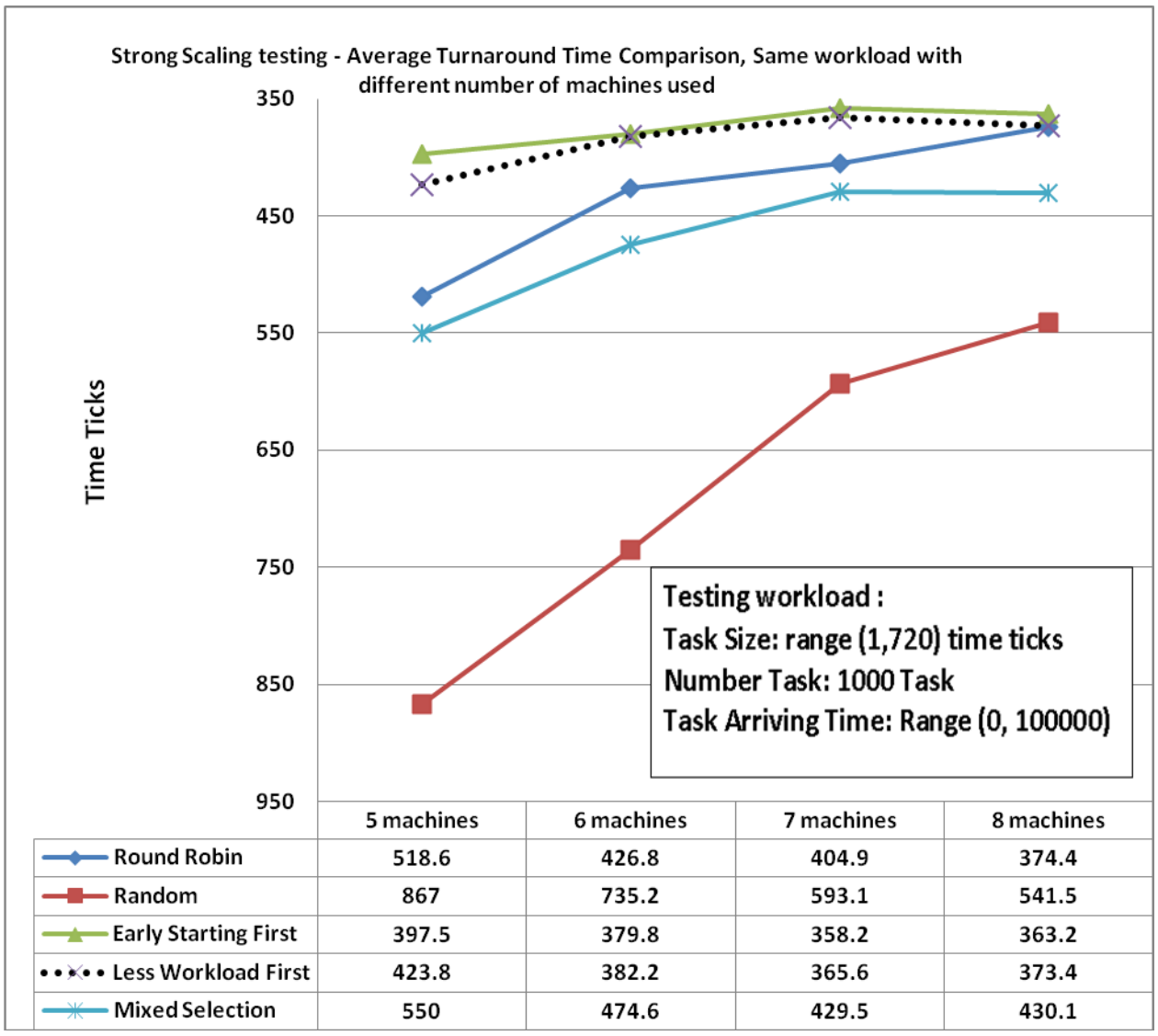
	5 machines	6 machines	7 machines	8 machines
Round Robin	156.4	65.1	31.9	17
Random	739.1	353.9	245.5	172.6
Early Starting First	55.5	22.6	8.13	2.9
Less Workload First	97.8	31.3	10.7	6
Mixed Selection	155.1	124.8	74.9	69.6

Figure 6-1: Strong scaling testing and average waiting time comparison, same workload with different number of machines



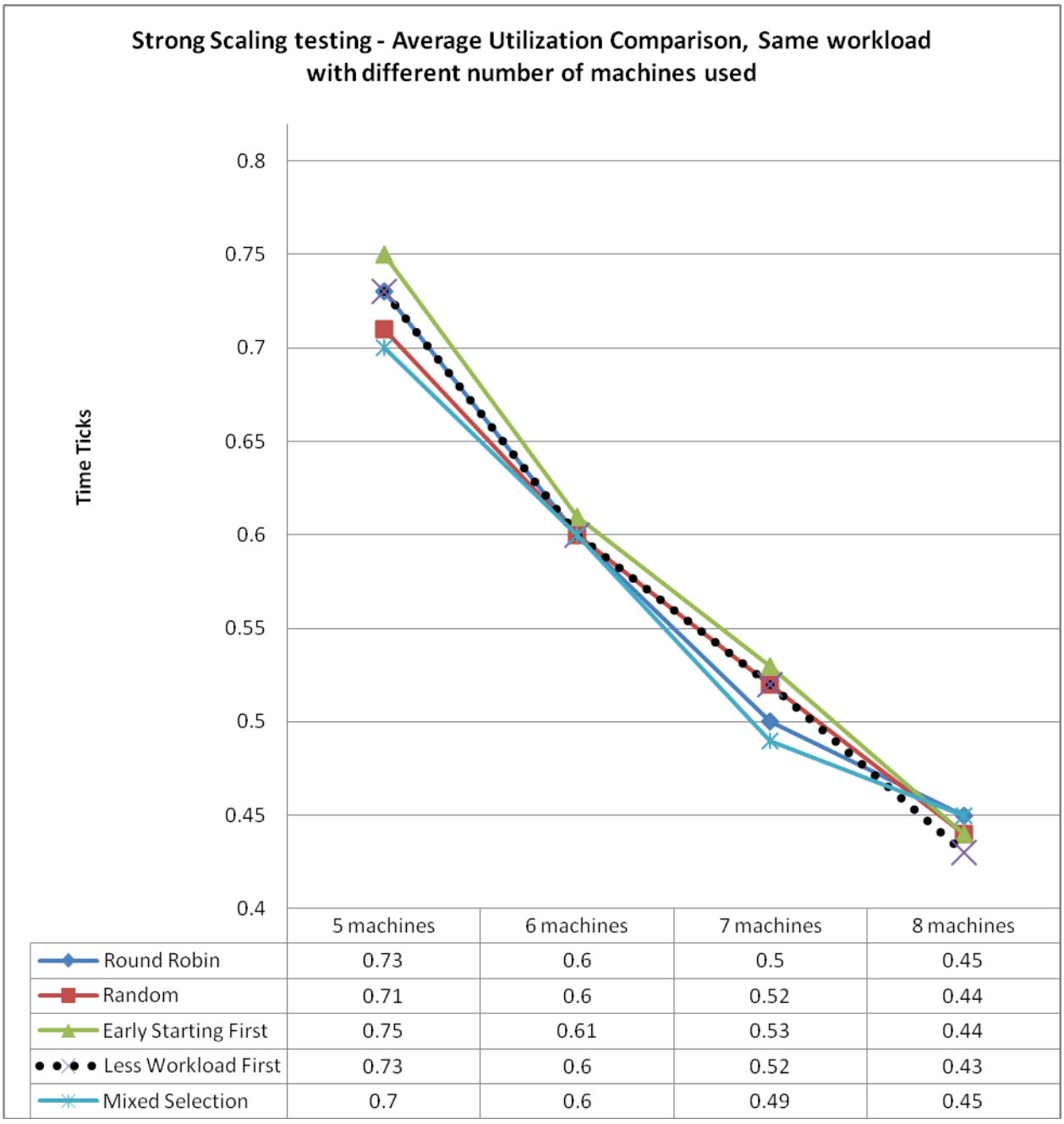
	5 machines	6 machines	7 machines	8 machine
Round-Robin	1	2.399	4.902	8.941
Random	1	2.088	3.01	4
Early-Start_time First	1	2.455	6.826	19.13
Less Workload First	1	3.125	9.14	16
Mixed Selection	1	1.243	2.07	2.228

Figure 6-2: Normalized ScalingIndex comparison - average waiting time



	5 machines	6 machines	7 machines	8 machines
Round Robin	518.6	426.8	404.9	374.4
Random	867	735.2	593.1	541.5
Early Starting First	397.5	379.8	358.2	363.2
Less Workload First	423.8	382.2	365.6	373.4
Mixed Selection	550	474.6	429.5	430.1

Figure 6-3: Strong scaling testing and average turnaround time comparison, same workload with different number of machines



	5 machines	6 machines	7 machines	8 machines
Round Robin	0.73	0.6	0.5	0.45
Random	0.71	0.6	0.52	0.44
Early Starting First	0.75	0.61	0.53	0.44
Less Workload First	0.73	0.6	0.52	0.43
Mixed Selection	0.7	0.6	0.49	0.45

Figure 6-4: Strong scaling testing and average utilization comparison, same workload with different number of machines

3.2.2 Weak scaling testing cases

We also conducted weak scaling testing cases. In a weak scaling testing case the problem size (workload) assigned to each machine stays constant. Table-2 shows the workload distribution when using different number of machines.

Number of machine used	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Workload(#task agent used)	**400**	**800**	**1200**	**1600**	**2000**	**2400**	**2800**	**3200**

Table-2: Workload distribution for weak scaling testing

Figure 7-1 shows that the Early Start-time First and the Less Workload First methods have better stable/static decreasing average waiting time when increasing machines in simulation. The random scheduling method shows an up/down or unstable average waiting time when increasing more machine in simulation.

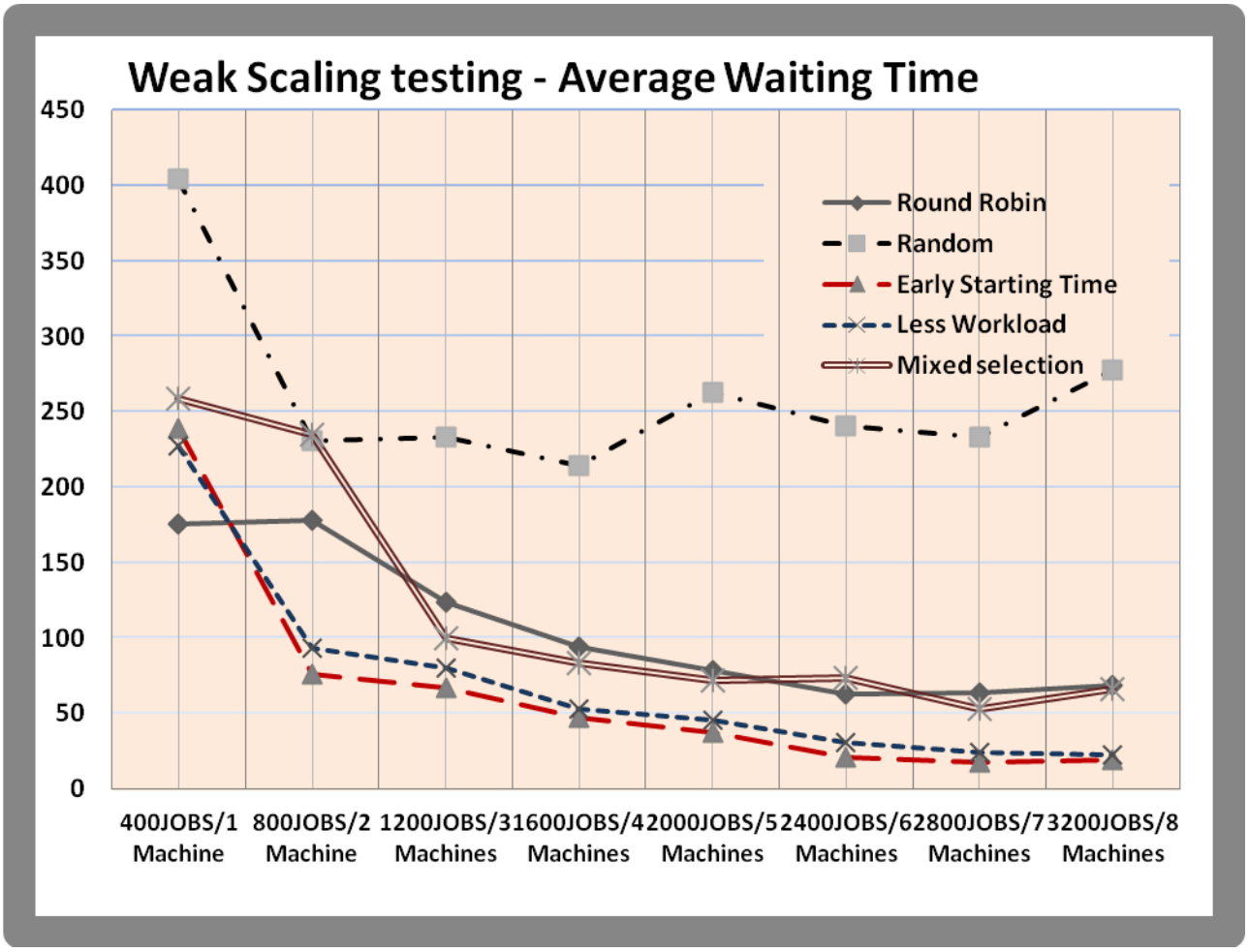


Figure 7-1: Weak scaling testing - Average waiting time comparison

Figure 7-2 shows the result of weak scaling testing cases in terms of average turnaround time comparison. We show that the Early Start-time First and the Less Workload First methods can decrease average waiting time when increasing machines in simulation. The random scheduling method shows an unpredictable.

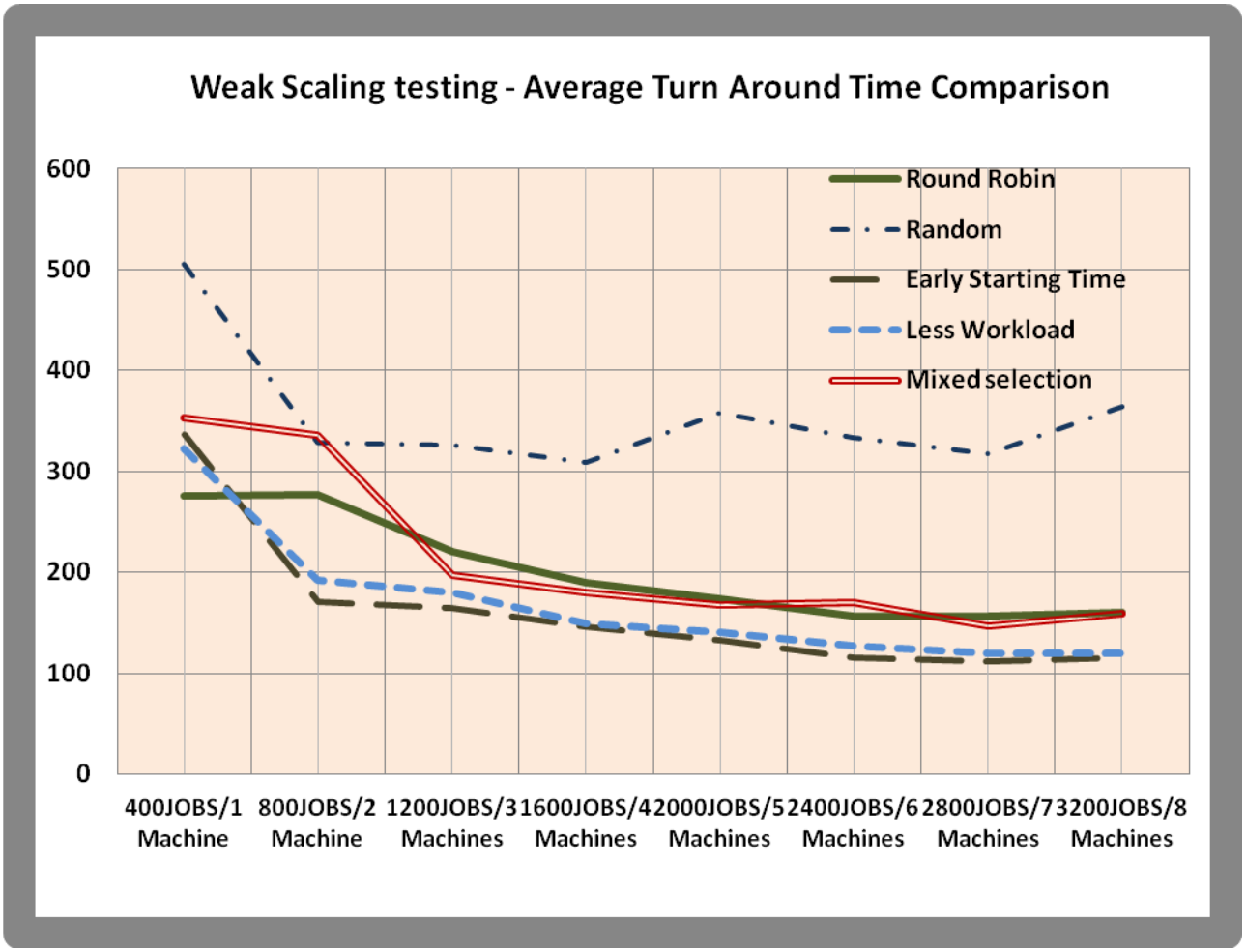


Figure 7-2: Weak Scaling testing - Average turnaround time comparison

3.2.3 Auto tuning feature

We also implemented an "Auto Tuning" feature. We used the "switch" to turn on an off the "Auto Tuning feature." We used the "slider" to set two parameters to control the "auto tuning" feature. They are:

- maxWaitTimeAllow - If the average waiting time is greater than this value, we then add one more machine to the service.
- minWaitTimeAllow - If the average waiting time is below this value, we then remove one machine from the service.

We used this "Auto Tuning" feature to dynamically control the run-time wait-line situation. A sample run-time screen-shot is shown in Figure 8-1, Figure 8-2, and Figure 8-3.

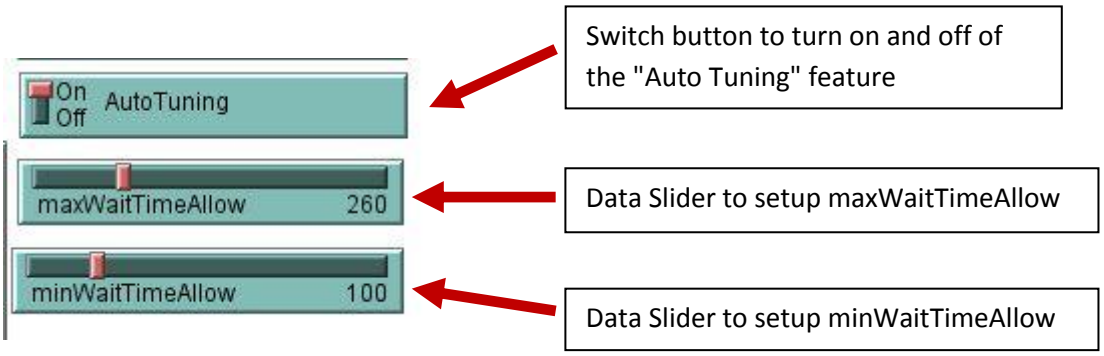


Figure 8-1: Setup: Auto Tunning"

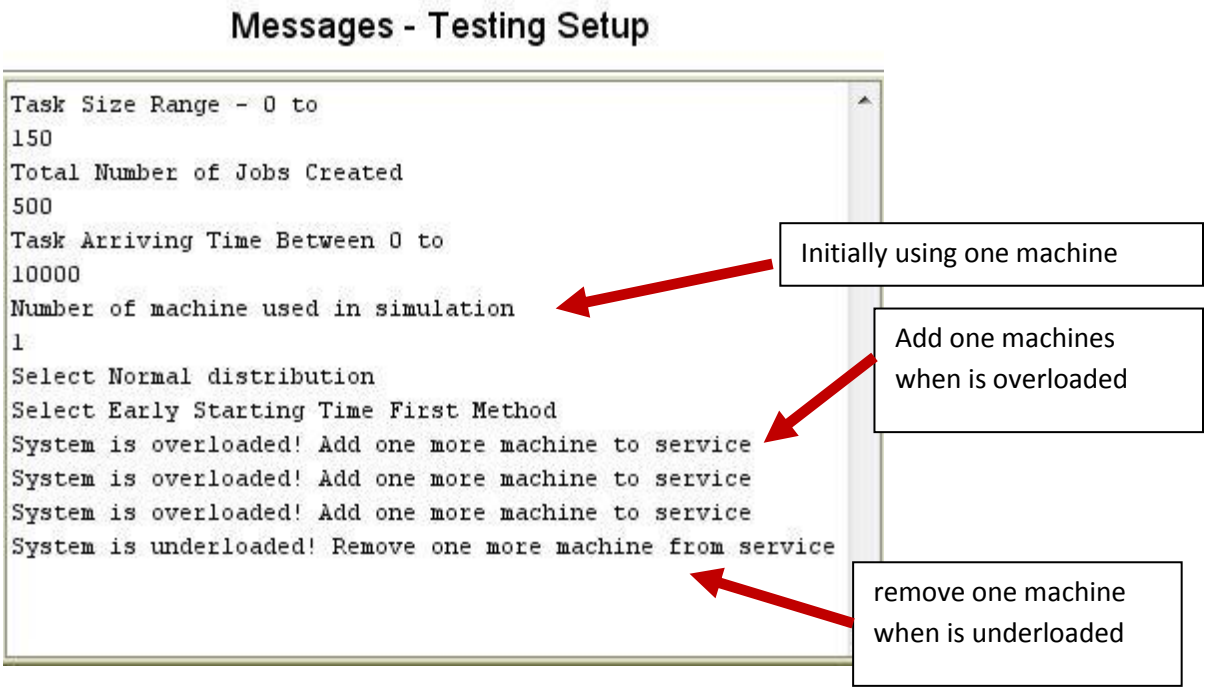


Figure 8-2: Run time messages show activities of the "Auto Tuning"

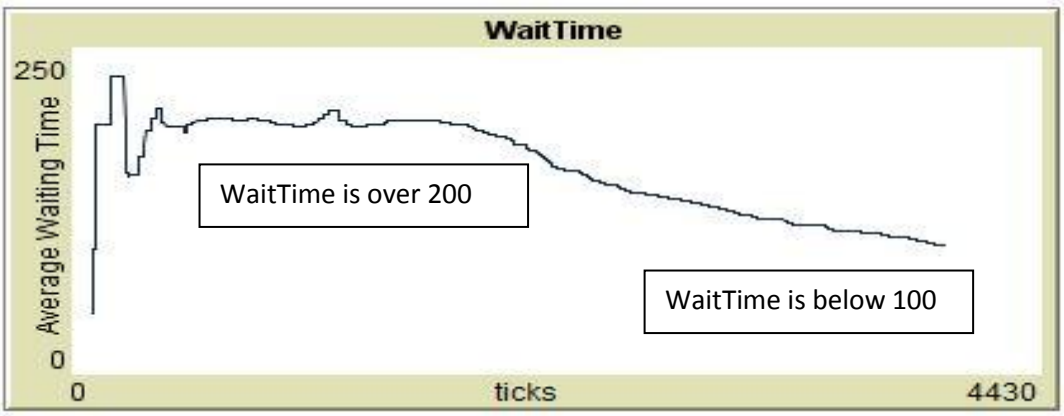


Figure 8-3: AverageWaitTime visualization data over the global tick counter

4 Conclusion

We successfully created a multi-agent task scheduling simulation system using NetLogo programming language and NetLogo run-time environment. We provided an interactive user interface and rich visualization of run-time information. We collected performance data and discuss the pro and con of each heuristic task scheduling methods. We showed that some simple heuristic could help to get better results. The interactive and visualized approach used in the simulation system proves to be useful and interesting. It helps to learn various task patterns such as light workload and heavy workload. It helps to understand performance matrices such as the strong scaling and weak scaling. It demonstrates how difference workload can have different impacts on the waiting time and machine utilization. Using our simulation system, we can conduct testing cases of various real-lives waiting line and queuing problems.

The major contributions of the Team 73 are: (1) the first ever to implement a multi-agent task scheduling simulation program, (2) comparing five heuristic scheduling methods, and (3) implementing a useful educational simulation program that can assist those who wish to study real-life waiting line and queuing problems.

Through this project, we have learned the following things: (1) how to use the NetLogo programming system, (2) what is the waiting line and queuing problem, (3) what is heuristic scheduling method, (4) how to convert a real-life problem to a multi-agent model, (5) how to design interactive user-interface, and (6) how to work together as a team towards a common goal and eventually finish this project on time.

5 Future works

We would like to extend our current simulation program to cover more real-life problems involving waiting line and queuing. There are several interesting areas that we plan to add to or modify our existing simulation program including Real-time task scheduling problem, Dependent task scheduling problem, Work Flow simulation, Multi-server and multi-line task scheduling, Add multiple phases to the existing simulation, and more interactive run-time visualization display such as Gantt chart etc. The initial implementation of the "Auto Tuning" feature is demonstrating our future enhancement plan to provide an intelligent scheduling environment for waiting line queue problems.

Acknowledgements

First, we would like to thank the Super-computing Challenging program and the committee members. The kickoff program at New Mexico Tech was especially helpful. Through this training, we gained knowledge, useful information, and a variety of idea to implement in our simulation program. Furthermore, we would like to thank the comments and suggestions from our intern reviewing judges. We would also like thank our Supercomputing challenge project teacher Mrs. Pauline Stephens for sponsoring our project and her constant encouragement throughout these in the past five months. In addition, we also would like to thank Ariel Chen for reviewing our report and HB Chen for mentoring our team.

Finally we would like to thank our parents for helping us prepare posters, editing the final report, and setting up different testing environments.

Bibliography and References

NetLogo from North Western University - http://ccl.northwestern.edu/netlogo/

NetLogo 5.0 User manual - North Western University

Seth Tisue and Uri Wilensky , "NetLogo: A Simple Environment for Modeling Complexity," ICCS 2004

Frederic Haziza, "Scheduling Algorithms," Department of Computer Science, Uppsala University

Terence C. Ahern, "Bridging the Gap: Cognitive Scaffolding to Improve Computer Programming for Middle School Teachers," 39th ASEE/IEEE Frontier in Education Conference, October 18-21, 2009, San Antonio, TX

E.O. Oyetunji, "Some Common Performance Measures in Scheduling Problems: Review Article," Research Journal of Applied Sciences, Engineering and Technology 1(2): 6-9, 2009

"Process Scheduling", class note, Department of Computer Science, University of Texas at Austin, Professor Lorenzo Alvisi

Sartaj K. Sahni, "Algorithms for Scheduling Independent Tasks,", Journal of ACM, Vol 23, Issues 1, Jan., 1976

Yiqiu Fang, Fei Wang and Junwei Ge, "A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing," Lecture Notes in Computer Science, 2010, Volume 6318/2010, 271-277

Yingzi Li, Xiaodong Zhang, and Shou Zhang, "Multi-agent Simulation System Study on Product Development Process,", Applied Mathematics and Information Science, 2011

Sebastien Paquest, Nicolas Bernier, and Brahim Chaib-Dras, "Multiagent System Viewed as Distributed Scheduling Systems: Methodology and Experiments," Proceedings of the 18th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2005, Victoria, Canada, may 2005

Waiting line and Queuing problem - Queuing theory:
http://businessmanagementcourses.org/Lesson21QueuingTheory.pdf

Ant Colony Conundrum

New Mexico

Supercomputing Challenge

Team 118

Sat. Science Math Academy

Team Members

Daniel Washington

Rachel Washington

Muhammad Musleh

Teacher

Debra Johns

Project Mentor

Wayne Witzel

Table of Contents

Executive Summary

Our project examines the effectiveness of decentralized systems, such as an ant colony, for solving a search problem. The problem that out project is attempting to solve is if a decentralized system such as an ant colony is capable of more effective search patterns than a centralized counterpart. In addition, we are looking at whether a system of numerous independent agents is able to adapt to environmental changes as a cohesive unit without specific communication between agents.

We will be examining the success of two primary behavior sets on a static and dynamic world and comparing the results. We will then attempt to create an ideal fusion of the two behaviors to maximize the efficiency of these ants. Finally, we will be exploring the usefulness of ant-based search patterns in real world applications such as search and rescue and hostile environments where robustness, reliability, and speed are key factors.

Colony Set up

The simulation was programmed with the intention of comparing two independent colonies side by side. The creation of the world is facilitated by a loop that duplicates a patch setup on the negative and positive sides of the X-axis. Every food pile is generated by a system that places food in a location within a radius measured to have a maximum food density of 256 seeds/pile.

```
if pile_true = 1 [
  set pile_count (pile_count + 1) ;increments pile counter
  ask patches [
    if (distancexy xm ym) <= pile_radius [ ;;creates a circle of food with area equivalent to 256 seeds
      if food_density <= 1 / (random 20 + .000001) [
        set pcolor yellow
        set mfood 1
        ask patch-at 200 0 [
          set pcolor yellow
          set mfood 1
        ]
      ]
    ]
  ]
]
```

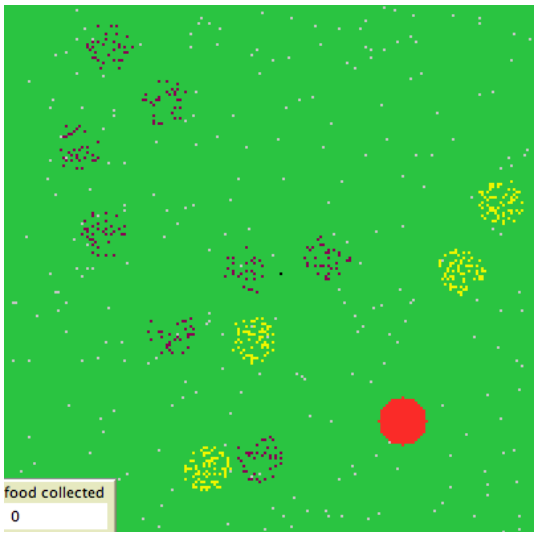
The food is distributed into four possible densities. 100% density is red food. 25% density is represented by yellow food. 12.5% density represents purple food and randomly distributed food is shown in gray.

The exponential distribution is built to create equal food quantities of variable densities.

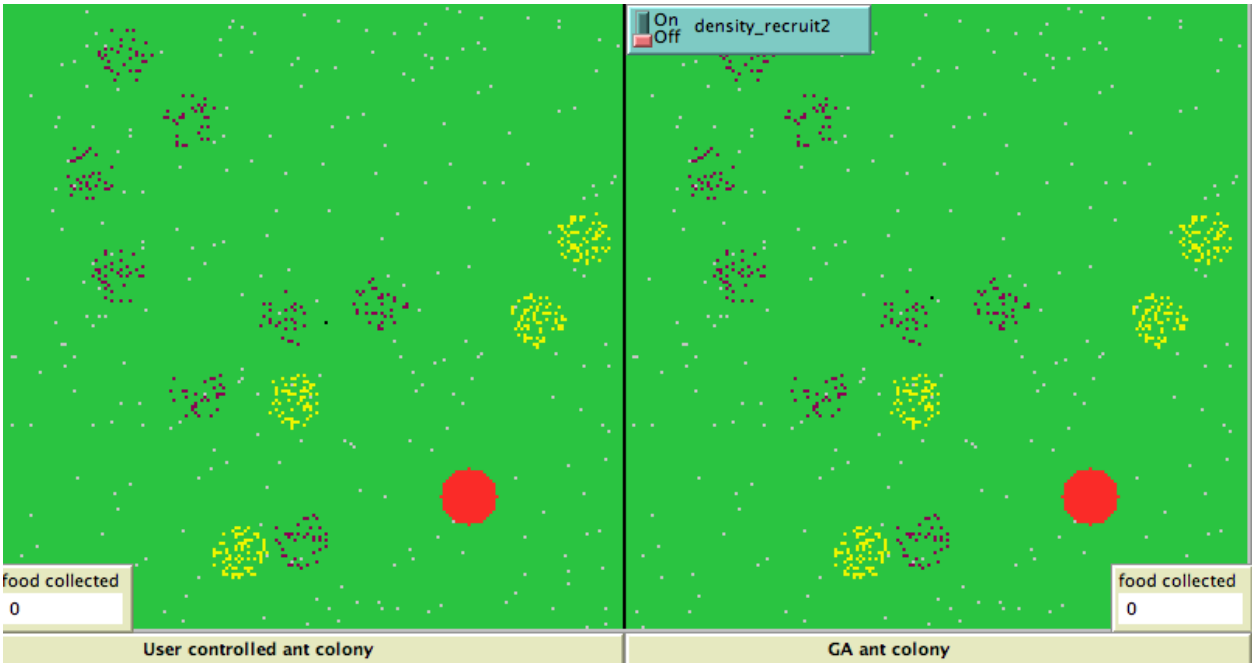
Red: 256 seeds * 1 pile = 256

Yellow: 64 seeds * 4 piles = 256

Purple: 32 seeds * 8 piles = 256



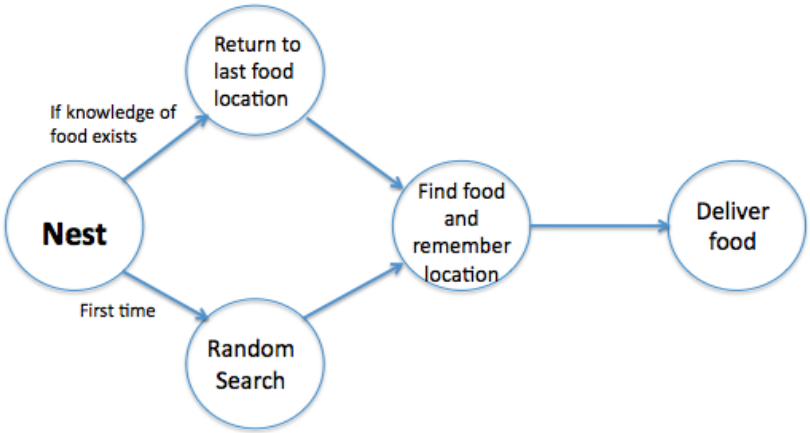
The simulation also creates two identical ant colonies, which it then compares in real time. These colonies are mirror images of each other with one difference: Their behavior parameters. For the purpose of accurate comparison, each colony is generated with identical food layout and colony size.



Behaviors and Parameters

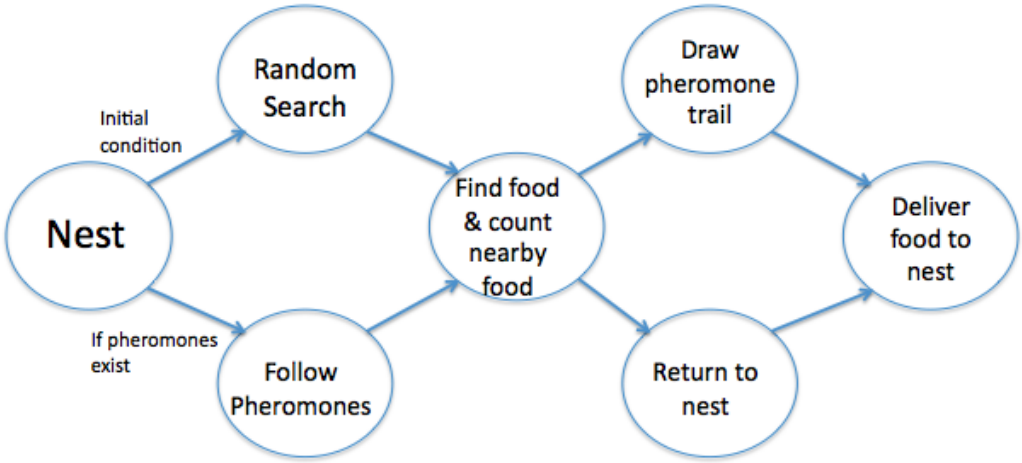
Each ant colony uses 2 major behavior sets: Site Fidelity and Density Recruitment.

Site Fidelity: Ants leave the nest using a random walk. They choose a random direction biased away from the nest to turn to and move forward a set amount. Upon finding food, they store the location internally. They then bring the food to the nest and return to the location of recently found food.



Density Recruitment: Ants leave the nest in a random direction and use the same random walk seen in Site Fidelity. Upon finding food, they make an assessment of food density in the neighboring area (food count **c**) (random 100/100 = P) (recruitment parameter = r).

P <= c + r . If this equation is true then ants will lay a trail. Ants at the nest will follow pheromones to a food source.



Pheromones

In nature, some ants have been proven to use a method of recruitment known as pheromones. Pheromones were discovered by placing absorbent paper between a nest and a food location. As the food source becomes more reliable, the strength of the chemical trail increases because more ants were returning to the location and laying trails. This effect is also seen in our simulation. The strength of the trails increases as the density of the food pile increases. These pheromone trails have also been proven to be volatile. All of the trails have an evaporation rate determined by a genetically optimized parameter.

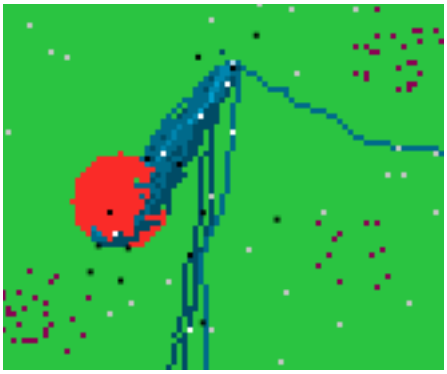
Trail evaporation equation

P = P initial * (1-E) if P initial > .001

P = 0 if P initial < .001

P = pheromone strength

E = evaporation constant (0<E<1)



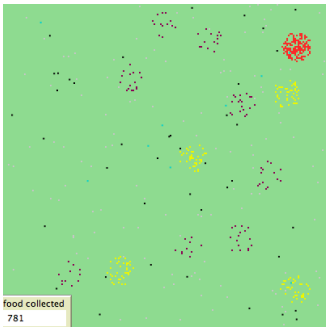
Dynamic Environment

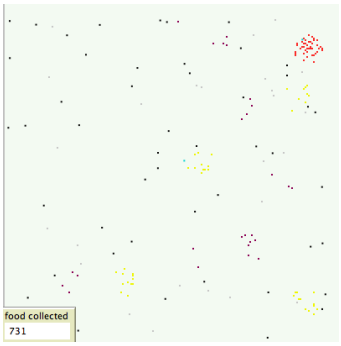
We are attempting to find a method of food collection that excels in both stationary and changing environments. For this reason, we programmed a simulation with a dynamic environment. This world begins with the same number of available food seen in the stationary model, but as time goes by, food amounts will decrease and increase with seasons.

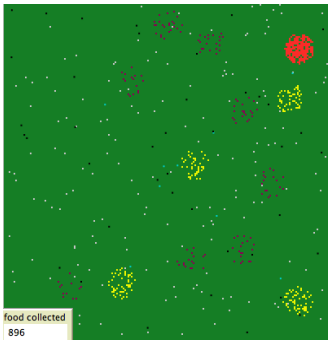
```
if (ticks / 6) = round (ticks / 6)[
  ask patches [
    if pcolor = red or pcolor = 123 or pcolor = 7 or pcolor = yellow [
      if random 250 = 1 [
        set pcolor background

      ]
```

This code block allows all the food to decrease at a rate of 0.067 % of the remaining total per tick over a 2500 tick interval. Resulting in approximately 50% fluctuation in food totals over all.

In addition to creating a fluctuation in available food, we created a visible environmental change in the way of seasons. The simulation will shift from dark green to white to a light green to symbolize a seasonal progression.



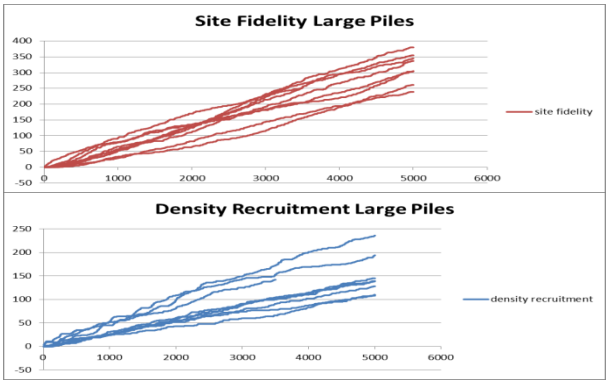




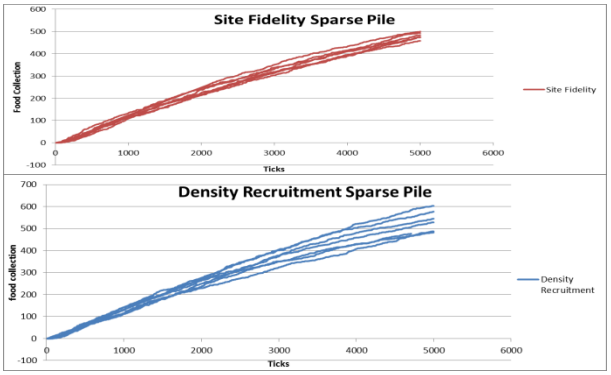
Graphs

To test the adaptability of different ant colonies, we tested many different scenarios of food distributions and trail evaporation. We tested our ant program several times to get the most dependable, reliable, and consistent data.

Random Food - Both graphs show the results of Site Fidelity and Density Recruitment over multiple trial runs. The food is distributed randomly. In this case, the ants perform almost the same not really varying much in behavior or food collection.

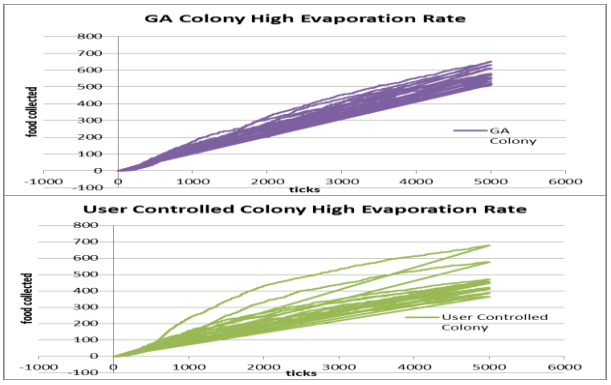


Large Piles/ Dense Food- The graphs are testing all dense piles of food. In this case Site Fidelity performs better than Density Recruitment. From collecting data and graphing we found that this was because in Density Recruitment, many of the ants never find food therefore they would never return to the nest, causing many of the ants never to pick up a pheromone trail.

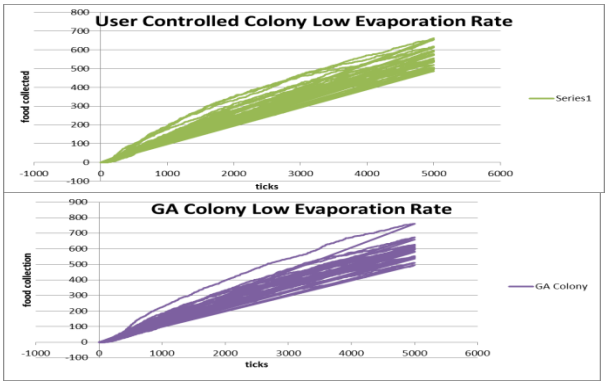


Sparse Piles- these graphs test sparse piles, which are the least dense pile of food in our program. Density recruitment performs better in this case. This is because Density Recruitment has a better chance of finding food and returning to the nest unlike the case with only a few dense food piles.

High Evaporation Rate- the higher the evaporation rate the faster the pheromone trails will disappear. The two colonies are in an environment with equal amounts of different piles of food. The GA colony performs better than the user-controlled colony with a high evaporation rate. It took the user controlled colony time to adapt to these extreme changes of their environment.



Low Evaporation Rate- In this case the ants had to adapt to the trails evaporation slower. The two colonies in this environment performed better than if the trails would evaporate faster and did not have to adapt to any sudden changes. Both ant colonies performed better as a result, with density recruitment depending on pheromone trails.



Conclusion

Ants are the world's greatest scavengers. Their behavior can teach us many things about how to find and collect resources through a distributed system such as a colony. However, few colonies live in an environment where their food availability is subject to extreme change. We are able to create these environments though the use of computer simulation. By running numerous tests, we have been able to pinpoint how to optimize the ant colonies in a system of extreme drought or extreme moisture. We have discovered how to optimize a colony for very dense and very sparse food sources, and we have learned how to adapt an ant colony to a changing environment through our knowledge of existing ant behavior and how parameters can affect it.

These new ant-based algorithms have numerous applications in the real world. A system of Ant-based robots would be extremely successful at both search-and-rescue style operations and in dangerous areas such as minefields. Because these ants are independent and highly efficient, a system of autonomous robots would be both cheap and reliable. The possibilities of robotic and digital ants are nearly endless.

Works Cited

1. T.P. Flanagan, K. Letendre, W.R. Burnside, G.M. Fricke and M.E. Moses. (2011). "How Ants Turn Information into Food." Proceedings of the 2011 IEEE Conference on Artificial Life:178-185

2. Netlogo website: http://ccl.northwestern.edu/netlogo/. (2011)

3. Supercomputing Challenge Webite http://challenge.nm.org (2012)

4. Marco Dorigo, Tomas Stuzle (2012) http://www.scholarpedia.org/article/Ant_colony_optimization

5. Nikolaos V Karadimas, Georgios Kouzas, Ioannis Anagnostopoulos, Vassili Loumos (2012) http://ducati.doc.ntu.ac.uk/uksim/journal/Vol-6/No.12-13/paper5.pdf

6. Kwee Kim Lim, Yew-Soon Ong, Meng Hiot Lim, Xianshun Chen, Amit Agarwal (2012) http://www.springerlink.com/content/5774082768421663/