Multi-Agent Systems for Education and Interactive Entertainment:

Design, Use and Experience

Martin Beer
Sheffield Hallam University, UK

Maria Fasli
University of Essex, UK

Debbie Richards
Macquarie University, Australia

Information Science REFERENCE

INFORMATION SCIENCE REFERENCE

Hershey · New York

Director of Editorial Content: Kristin Klinger
Director of Book Publications: Julia Mosemann
Acquisitions Editor: Lindsay Johnston
Development Editor: Michael Killian
Typesetter: Michael Brehm
Production Editor: Jamie Snavely
Cover Design: Lisa Tosheff

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Multi-agent systems for education and interactive entertainment : design, use and experience / Martin Beer, Maria Fasli, and Debbie Richards, editors.
p. cm.
Includes bibliographical references and index.
Summary: "This book presents readers with a rich collection of ideas from researchers who are exploring the complex tradeoffs that must be made in designing agent systems for education and interactive entertainment"--Provided by publisher.
ISBN 978-1-60960-080-8 (hardcover) -- ISBN 978-1-60960-082-2 (ebook) 1. Multiagent systems--Congresses. 2. Instructional systems--Congresses. 3. Computer science--Study and teaching--Congresses. I. Beer, M. D. (Martin D.) II. Fasli, Maria. III. Richards, Debbie.
QA76.76.I58.M863 2011
006.3--dc22
2010049832

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 1
MAgICS:
Toward a Multi-Agent Introduction to Computer Science

Forrest Stonedahl
Northwestern University, USA

Michelle Wilkerson-Jerde
Northwestern University, USA

Uri Wilensky
Northwestern University, USA

ABSTRACT

The authors present a preliminary version of the MAgICS (Multi-Agent Introduction to Computer Science) framework, which is a new approach for revitalizing introductory undergraduate or high school computer science curricula through the deep integration of agent-based modeling (ABM) and multi-agent systems (MAS) perspectives. The authors discuss the merits of using multi-agent systems as a lens for conceptual understanding across disciplines, compare multi-agent approaches to traditional serial ones, and explore how this approach can bring together disparate topics in computer science through the common focus on emergent systems to promote a broader view of the field as a whole. To exemplify this approach, they have developed a suite of curricular models for topics spanning from searching and sorting to machine learning and networks and security. By introducing these topics with a focus on parallel, distributed, and stochastic methods, they can make traditionally upper-level topics both motivating and accessible to introductory-level students. The authors review findings from a short implementation of several elements of MAgICS in an introductory computer science classroom with regard to student motivation and evidence of learning of distributed design strategies.

INTRODUCTION AND MOTIVATION

Two years ago, Rick Rashid, a senior vice president for research at Microsoft, asked the rhetorical question of whether computer science is a dying profession (Rashid, 2008). Indeed, shrinking undergraduate computer science enrollment and concern about the underrepresentation of both women and minorities in computer science has been the subject of much debate, particularly in

DOI: 10.4018/978-1-60960-080-8.ch001

North America (Denning & McGettrick, 2005; Goode, 2007; Katz, Allbritton, Aronis, Wilson, & Soffa, 2006). Diversifying the introductory curriculum is one method for reaching a broader audience (see, e.g., Cushing, Weiss, & Moritani, 2007; Denning & McGettrick, 2005; Downey & Stein, 2006), which has met with some success. In this chapter, we present the MAgICS (Multi-Agent Introduction to Computer Science) framework as a new and powerful approach to diversifying the introductory computer science curriculum. Through the MAgICS framework, we demonstrate the potential to address many conventional topics of computer science (such as searching, sorting, optimization, graphics, machine learning, networks/security) through an agent-based modeling (ABM) and multi-agent systems (MAS) perspective.

Agent-based modeling (Epstein & Axtell, 1996; Wilensky & Resnick, 1999) is a form of computational modeling whereby a population of individual ("micro-level") computational *agents* are given simple rules to govern their behavior: for example, traffic flow can be modeled by programming a number of "car" agents to speed up and slow down under different local conditions. The models are then run so that the aggregate ("macro-level") results of those agent behaviors can be investigated (e.g. traffic patterns; (Wilensky, 1997b)). The ABM/MAS paradigm has become increasingly popular within computer science (Davidsson, 2002; Panait & Luke, 2005; Wilensky & Rand, in press), and has proven to be a powerful computational modeling tool for the natural and social sciences (NRC, 2003). Because many systems in the world can be productively conceptualized as a collection of agents contributing to some macro-level phenomenon (atoms and molecules comprise matter, individual consumers comprise markets), it is conducive to interdisciplinary integration and applications. For instance, one particularly powerful interdisciplinary idea that can be explored using the agent-based paradigm is that of *emergence* – the notion that interactions between simple individual agents can result in surprising and complex aggregate-level phenomena that appears to be "more than the sum of its parts" (Johnson, 2001; Wilensky, 2001). For instance, an *emergent* outcome of a traffic system is that traffic jams move backward, even though the individual cars that comprise the jam each move forward (Wilensky & Resnick, 1999).

The benefits of an agent-based approach for understanding complex systems, emergence, and notions of parallelism and decentralization – topics that are typically very difficult – are well-established. Wilensky and Resnick (1999) have found that a number of difficulties that students have in understanding complex systems stem from a *deterministic/centralized* (or *DC*) mindset – for instance, they are likely to attribute the emergent behavior of a system of entities (such as the formation of a flock of birds; or the evolution of a species) to some single cause or intention, rather than as the result of a collection of behaviors and interactions in a distributed system. Agent-based modeling enables students to explore how the behaviors of individual agents can lead to unintended outcomes, and better understand why those outcomes occur in a multitude of disciplines (including chemistry, Levy, Novak, & Wilensky, 2006; materials science, Blikstein & Wilensky, 2006; physics, Sengupta & Wilensky, 2008; Wilensky, 2003; and biology Wilensky & Reisman, 2006).

We believe it is equally important for an ABM/MAS perspective to be an ingredient in computer science education – especially because understanding system-level behavior, emergence, and unintended outcomes is not only important for *understanding*, but also for *designing* systems of the future (Guckenheimer & Ottino, 2008). The MAgICS framework uses agent-based modeling and notions of emergence to focus on a variety of upper-level topics while encouraging connections to a wide array of interdisciplinary endeavors and real-world applications. As such, we refer to ABM/MAS as a "lens for conceptual

understanding" in computer science. Often, subjects such as neural networks, particle systems, genetic algorithms, or sorting and searching are organized topically within the computer science curriculum, and are thus taught separately, without making conceptual connections between them. In contrast, an ABM paradigm uses concepts such as agents and micro/macro level phenomena to highlight the similarities and differences between the mechanisms at work. For example, later in this paper we discuss how algorithms for solving very different problems (such as sorting colored objects, or ranking the relevancy of web pages) can be similarly explained in terms of agents interacting with each other or with their environment according to simple rules.

This work is motivated by two central goals. First, MAgICS seeks to enrich early ("low-level") computer science courses by engaging students with motivating and conceptually rich "high-level" topics. Second, it seeks to emphasize distributed, decentralized systemic thinking – a skill that is becoming increasingly relevant both within and beyond the domain of computer science. The MAgICS framework aims to address the first goal (motivation and engagement) by building a curriculum around a series of dynamic agent-based models and modeling activities, coupled with compelling and interactive visualizations. This model suite offers a survey of several conventionally upper-level topics, and gives introductory students a broader intellectual taste for what computer science has to offer. Concepts of elementary programming can be covered through experimenting with the provided source code, extending the models, and writing new agent-based models from scratch. The suite of curricular models is central to our approach, and we will elaborate on the constituent models of the suite in a later part of this chapter.

The second goal (distributed, decentralized thinking) is the result of the evolving technical content of computer science education and the future of computing. In recent years, computing has undergone an important shift toward parallelism. This includes the current prevalence of multiple and multi-core processors, the ubiquity of high performance computing clusters in academia and industry alike, cloud computing, massive peer-to-peer networks, social networking and Web 2.0 applications, increased deployment of massively parallel supercomputers for research, and parallel languages and language features that accompany these developments. We are not advocating that CS101 students should be forced to learn the intricacies of mutual exclusion semaphores for accessing shared memory. The point is a broader one: that it is time to re-examine whether the prevalent focus in contemporary introductory computer science courses on centralized, deterministic, serial algorithms is best at preparing our students for a world of computation that is ubiquitously distributed, potentially stochastic, and increasingly parallel. This view is supported by previous work by Stein (1999), who challenged the current centralized computational metaphor with an alternative view of computation as a community of interacting entities. Our approach is further motivated by the goal of providing universal instruction in "computational thinking," as described by Wing (2006), and Papert and diSessa's call for a widespread literacy and fluency in computational methods and models (diSessa, 2001; Papert, 1980).

The remainder of this chapter is structured as follows. We first discuss related research in computer science education, and argue for the merits of using agent-based modeling (ABM) as a "lens for conceptual understanding" when exploring topics that computer science educators might not traditionally consider to be in the domain of ABM or MAS (or even in the domain of introductory computer science!). Next, we explore three example curricular models in some detail, and discuss how they may be used to promote understanding of multi-agent systems while learning about computer science topics. We also offer a brief overview of each of the other models in the suite. The remainder of the chapter is dedi-

cated to discussion of a preliminary pilot study that we implemented to test out some of the ideas and models included in the MAgICS framework, and offer empirical support for the feasibility of our approach. We conclude with some remarks about potential implementation considerations and possibilities for future work.

RELATED WORK

There have been many suggested approaches for revitalizing computer science education. In this review we discuss only a few, for comparison to our own approach (for further coverage, we recommend readers to a recent survey paper on introductory computer science education; Pears et al., 2007). Specifically, we argue that integrating agent-based modeling into introductory computer science curriculum addresses many calls in the computer science education literature to engage students in motivating consequential tasks and to highlight the interdisciplinary nature of computer science and its applications.

Efforts to improve learning, motivation, relevance, and student retention in the introductory computer science sequence may be loosely classified into two categories (although much work falls at least partially into both categories): reforms based on pedagogy (i.e., *how* CS should be taught), and reforms based on content (i.e. *what* should be taught). Pedagogy-based reforms include, for example, the integration of design-first programming (Moritz, Wei, Parvez, & Blank, 2005), pair programming (Carver, Henderson, He, Hodges, & Reese, 2007), and non-computer-based activities focusing on computer science concepts (Bell, Witten, & Fellows, 1999). Content-based reforms include introducing computer science with a focus on robotics (Becker, 2001; Blank, 2006; Fagin & Merkle, 2003; Flowers & Gossett, 2002), game design (Overmars, 2004), multimedia such as sounds, images, and movies (Guzdial, 2003), or specific languages or language paradigms (Flowers & Gossett, 2002; Howland, 1997; Radenski, 2006). Our MAgICS framework also falls into this latter category, as it is largely a content-based reform yet it also has pedagogical entailments such as promoting the use of simple code fragments as behaviors, the importance of multiple solution paths and visual feedback. Another approach that is particularly relevant to our present work is that of Stein (1999), who designed a CS101 course centered around the paradigm of "interactions between entities", including a significant focus on issues of concurrency. Our approach differs from the broader work of Stein in that we specifically focus on the use of agent-based models as a cohesive thematic element to draw together a variety of interesting and challenging computer science topics, and make them accessible at an introductory level.

Rather than focusing on changes to the content within the introductory course, Cushing et al. (2007) suggested broadening introductory CS by offering interdisciplinary courses with math and science (such as ecology) as a means to improve retention and increase interest in the field. Although not described here, an interdisciplinary approach can also be easily employed using agent-based modeling, which is something we have productively done in other work (Levy & Wilensky, 2009; Sengupta & Wilensky, 2009; Wilensky & Reisman, 2006). Meanwhile, Buckley et al. (2008) advocate for the integration of socially relevant projects into early computer science education, and Denning and McGettrick (2005) call for a "recentering" of computer science, expanding the public's view of computer science as *programming*, and suggest an introductory CS sequence with a theme of technological innovation.

While there are a considerable variety of approaches to and opinions about how the current state of computer science education might be improved, we argue that agent-based modeling offers a possibility for addressing issues that many of these reforms are concerned with. As we hope to show, creating agent-based models provides a

natural opening to innovation and design of complex systems and simulations, as well as fertile ground for socially relevant projects. Furthermore, a curriculum designed around the ideas of ABM/MAS can provide an effective coupling of advances in computer science education methods (e.g., visualization technology) with the more general goals of active engagement and intellectual inquiry on the part of students. Learning the art of computer programming remains a central piece of our educational framework (students will simply be programming multi-agent simulations, rather than, for example, writing programs for counting prime numbers). However, we do agree that early computer science courses often provide too narrow a view of what it means to be a computer scientist, and suggest that our approach will offer broader exposure to "upper level" topics.

Finally, while most introductory sequences in computer science are offered at the college or university level and we present this framework primarily in that context, we also acknowledge the important role of computer science education at the pre-collegiate level (Patterson, 2005). Just as in college, enrollment in high school computer science courses is low, and there have been calls for a more diverse, integrated computer science curriculum (Goode, 2007). We suggest that an ABM/MAS paradigm, and the MAgICS framework specifically, is also a powerful way to introduce computer science to a younger audience. In support of this assertion, we note that the NetLogo modeling environment (Wilensky, 1999), and even several of the agent-based models discussed specifically in this chapter, have been successfully used with a wide range of students including in educational interventions as early as primary school.

Centralized vs. Decentralized Approaches

While there are many real-world problems which call for decentralized thinking and are amenable to agent-based approaches, there are also many for which traditional serial deterministic algorithms are better suited. Therefore, at least a brief discussion is in order regarding the relative merits of agent-based approaches versus more traditional centralized approaches.

For us, one vital learning goal is that students will able to use decentralized thinking to approach problems *when* there may be a benefit to doing so, and to understand the trade-offs between decentralized and centralized approaches. Another is that students are able to consider issues of parallelism and distribution in not only *programming*, but in the conceptual *design* of systems. However, in order to make such decisions, students must be able to "think in both styles" – something that is notably difficult (Wilensky & Resnick, 1999). In other words, we do not argue that ABM/MAS is always the best approach for every problem. However, in a field where distributed thinking and parallelism are becoming increasingly important, an awareness of the variety of approaches one could take toward designing a project or solving a problem is a fundamental issue. Futhermore, the applications for which MAS/ABM are particularly well-suited – such as computational simulation and modeling – also emphasize the relevance of computer science to other fields of study, as well as for real-world applications and problem solving. We hope that the following section, which introduces some core MAgICS models, illustrates how incorporating MAS/ABM into introductory computer science can bring both of these issues to the forefront.

Fortunately, the distinction between centralized and decentralized programming approaches need not be couched in absolute terms, and in the MAgICS framework we do not wish to enforce a false dichotomy. In our view, distributed programming almost always incorporates elements of serial programming as well. Specifically, when programming multi-agent systems, the code that controls the behavior of an individual agent is often written as a serial program, sequential

flow and logic, using standard procedural or functional components. Thus, as students learn to work with distributed multi-agent systems, they will concurrently be learning traditional programming techniques as well (such as loops, conditionals, variable assignment, functions, etc). This is especially important, as it also helps to address the potential concern about how students will handle the transition from MAgICS back to more traditional languages and curricula, as they progress in the CS sequence. Because the concepts of sequential programming will be included in the design of agent-based simulations, this transition is not as dramatic as it might appear.

While students will subsequently need to learn different languages after NetLogo, such a language change is not uncommon in many current computer science sequences (e.g. Scheme, then C++). Furthermore, we consider a computer science education that contains only a single programming language to be incomplete; students should be exposed to multiple languages and paradigms. As NetLogo contains both procedural and functional language features, this should facilitate the transition to other languages in later CS courses.

THE MAgICS FRAMEWORK

The MAgICS framework is designed to address several goals, including the enrichment of early CS courses with a broader range of content, improving students' understanding of parallel, non-deterministic, and distributed systems, and offering a more motivational and applications-oriented introduction to the field.

MAgICS Curricular Model Suite

We have developed a wide collection of agent-based models (available for download from the NetLogo Models Library (http://ccl.northwestern.edu/netlogo/models/), and for the purposes of this chapter we include a cross-section of those models that relate to important or motivating topics in computer science (Kornhauser & Wilensky, 2007; Rand & Wilensky, 2006; Stonedahl & Wilensky, 2008a, 2008b, 2008c, 2009; Wilensky, 1997a, 1998, 2003). The suite of models we describe consists of nine models spanning seven topics, as shown in Table 1. Some of these models have been used with great success in short workshops and introductory courses on multi-agent modeling. We present here a cohesive framework for introductory computer science that we hope will be refined through trial, as well as feedback from others working in this area.

These models are all implemented using the NetLogo agent-based language and integrated modeling environment, which permits interactive modification of a model's parameters as well as of the code itself. The NetLogo language, following the Logo tradition (Papert, 1980), has also been designed to be easy to read and easy to learn, and the integrated modeling environment contributes to a low barrier for entry (Tisue & Wilensky, 2004). Equally important, NetLogo is not a "toy language"; it is a full programming language currently being used by researchers across the globe, offering a wide range of control structures and data types, and it is extensible via the Java programming language if access to additional

Table 1. MAgICS suite models and related computer science topics, listed by order of appearance in this chapter

Model Name	Topic
PageRank	Searching
Painted Desert Challenge	Sorting
Virus on a Network	Network Security
Simple Genetic Algorithm	Optimization
Particle Swarm Optimization	Optimization
Artificial Neural Net	Machine Learning
Particle Systems Flame	Computer Graphics
Flocking 3D	Computer Graphics
Dining Philosophers	Operating Systems

libraries is required. We also wish to emphasize the "glass box" nature of the suite of models: besides the visual interfaces (shown in figures below), each model comes complete with educational documentation and full source code that students can easily edit and run within the NetLogo modeling environment.

This is certainly not intended to be a comprehensive list of topics in computer science that could benefit from re-examination from an agent-based perspective. Instead, we seek to highlight several examples where parallel, distributed, stochastic, and emergent methods can be fruitfully incorporated into early computer science curricula. Furthermore, this list contains only fully implemented and documented models that are presently ready for educational use. More models could be added to this list, highlighting other important ideas. Some of these topics (such as searching and sorting) are similar to those traditionally covered in an introductory CS sequence while others (such as particle swarm optimization) are more typically found in upper-level undergraduate or even graduate-level courses. We will discuss only the first three example models in detail, and then briefly explain the scope and purpose of the six remaining models. For all models, the provided source code is clear and concise (less than 100 lines), and the accompanying visualizations serve to enhance the accessibility of the content.

Searching: PageRank Model

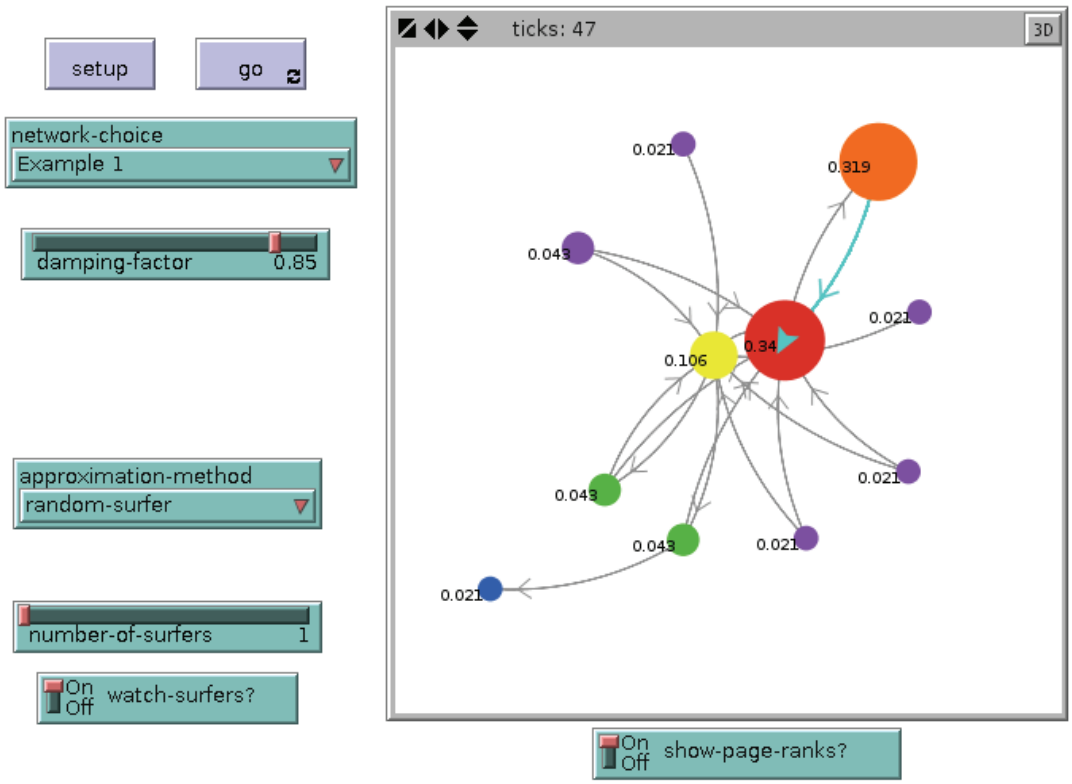
Traditional computer science curricula invariably include discussions of searching, often starting with students learning to do a sequential search in an array of numbers of strings. Later on, they are taught how to perform a binary search of sorted data, and to search other data structures such as trees or graphs, perhaps using depth-first search, breadth-first search (or perhaps Dijkstra's algorithm). While we have no desire to debate the merit of these venerable and classic algorithms, we note that they are all designed to run deterministically on a single processor accessing an unchanging data set.

Another approach to searching is to use a decentralized algorithm, which is especially useful for searching massive quantities of constantly changing data, i.e. the World Wide Web. Furthermore, we suspect students may be more motivated to learn about how Google "magically" returns relevant search results about their favorite curling team, as opposed to discovering how to find the position of "milk" in an alphabetized grocery list. The PageRank model (Stonedahl & Wilensky, 2009; see Figure 1), is based on the now famous PageRank algorithm developed by the founders of the Google search engine in the late 1990s (Brin & Page, 1998). PageRank is not technically a search algorithm, but rather a ranking algorithm, which provides a basis for ranking the information on one page as being more useful/important/relevant than the information on another page. The algorithm assigns a PageRank score to each web page, based on its relationship to other pages determined by the hyperlink structure of the web. Our PageRank model actually demonstrates two distinct agent-based methods for calculating the PageRank of a directed network (such as the web), though the two methods result in the same limiting behavior, and ultimately would assign the same PageRank scores to each page.

Method 1: Random Web Surfers

In this case, we assume there are "page" agents which are connected to each other in a directed network of hyperlinks, and there are also "web surfer" agents, which operate using these simple rules: Web surfer agents start at a random web page, and begin wandering the web. To wander, they either navigate a link from the current page to a new page, or they may jump directly to a random page somewhere on the web. If they run into a dead end page, they also jump to a random page. The probability with which they follow an existing link versus jump to a random page is

Figure 1. A screenshot from the PageRank model: Larger nodes represent higher PageRanks



controlled by a parameter called "damping factor" (typically set at 85% chance of link-following). As these agents, move, the model records the number of times a web surfer has visited each page. One definition for the PageRank metric is given by the probability of a single random web surfer being at that page at a given instant. Using the random web surfers model, this can be easily calculated by dividing the number of visits for each page by the total number of visits. In more formal mathematical terminology, this can be viewed as finding the stationary distribution for a certain Markov Chain, where each page is a state, and there are transitional probabilities specified between each pair of states. However, introductory CS students do not need to have acquired this level of mathematical formalism to appreciate the emergent behavior of the agent-based model.

Method 2: Diffusion of PageRank Scores

In this case, the primary agents in the model are the web pages themselves. Each page starts off with an equal amount of PageRank score. At each time step, pages divide equally and transfer their own PageRank value to each web page that they link to. (Pages with no out-bound hyperlinks are treated as if they linked to every single other page in the web.) Each page then sums the PageRank value received from each of the pages that link to it. Also, each page receives a certain amount of PageRank, just for existing (determined by the "damping-factor" parameter). This redistribution of PageRank via diffusion is carried out repeatedly, and over time the PageRanks converge toward the correct PageRank value. Mathematically, this method is related to the "power method" for

finding the dominant eigenvector of a modified adjacency matrix for the directed graph formed by the hyperlinks.

Beyond the clear benefits of exploring and understanding this classic algorithm that is so instrumental in making information accessible on the web, our PageRank model also provides an excellent launching point for students to experiment by creating their own distributed link analysis and/or ranking algorithms. For example, students could endow the "random surfer" agents with more sophisticated behavior (use of the "back" button, bookmarks) and see how the rankings would be affected. A broader discussion about emergent search techniques could also encompass ant foraging mechanisms, or the search of fitness landscapes performed by genetic algorithms (making a connection to the Simple Genetic Algorithm model also included in our suite).

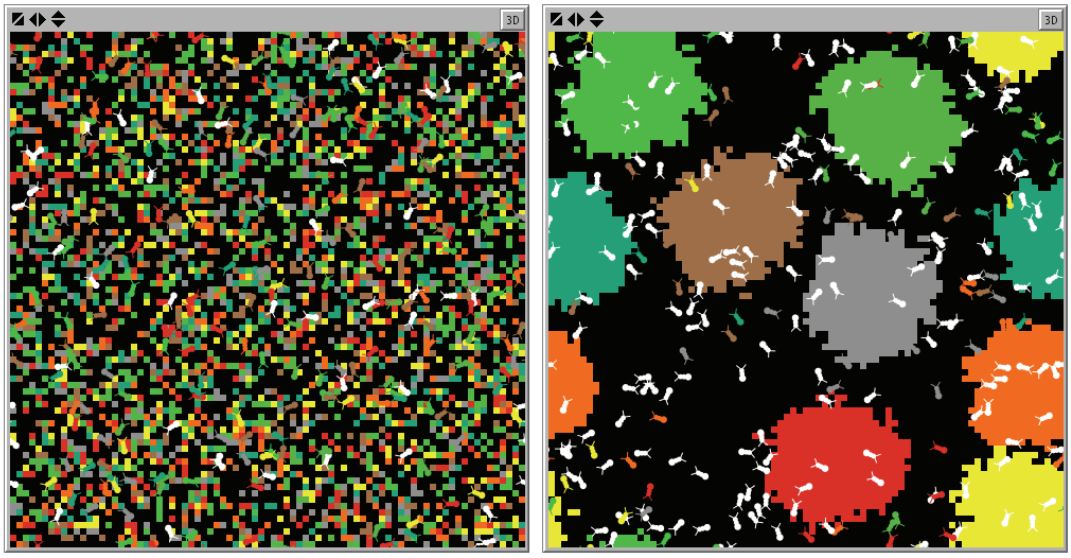
Sorting: Painted Desert Challenge Model

Sorting algorithms are another staple of early computer science education, inevitably including at least several of the following collection: bubble sort, selection sort, insertion sort, merge sort, quick sort, heap sort, bucket sort, shell sort, and radix sort. Again, a common theme is the deterministic single-threaded and serial aspects of sorting (although many of these algorithms can be at least partially parallelized). As a counterpoint, we wish to present a messier, distributed, and stochastic view of sorting, in the Painted Desert Challenge model (Resnick & Wilensky, 1992; Wilensky, 1997a; see Figure 2). While it may strike some as an inefficient approach to sorting, one should note that it is intrinsically parallel, reasonably robust, and could be applied in situations where the data is shifting during the sorting process, as a result of noise. However, it is important to keep in mind that we are not interested here in arguing for the merits of this particular sorting algorithm, but instead we are arguing for the merits of the ideas that students will be exposed to by exploring this model. The Painted Desert Challenge model offers insight into emergent systems, and in particular ant colony and other problem solving techniques inspired by nature.

The inspiration for this model goes back to a problem posed both at the Artificial Life III conference and to participants in a study on decentralized thinking (Resnick & Wilensky, 1998), where it was prefaced by a short whimsical vignette about insects that live in a painted desert and want to sort out each of the colors of sand after a windstorm mixed all the sand together. In this model, each termite follows the same set of very simple rules. It wanders in a 2D grid, wherein each grain of sand occupies one grid square. If it runs into a grain of sand, and it isn't already holding one, it picks it up. It continues to wander. If it runs into a grain of sand that is the same color as the one it is carrying, it drops its grain in an adjacent location. The emergent result of this random wandering and picking up and dropping is shown in Figure 2. This is, however, just one possible set of rules.

We do not expect students to learn computer science only by passive observation, any more than we expect people to learn to bicycle by watching the Tour de France on television. It is imperative that they get their hands dirty in the code, take the model apart and put it together again. For instance, a simple extension would be to have the sand shifting while the termites are working, and measure the rate of entropy-reduction the termites are capable of. A more complicated extension would be to give the termites greater vision and more intelligence, and test if more complicated rules yield more efficient sorting. On the more theoretical side, we might ask students to try to prove that the algorithm will eventually yield a complete separation of each of the different colors. It is worth noting that there are other emergent sorting algorithms, such as Brueckner's sorting networks (Brueckner, 2000), that could also be

Figure 2. "Before" and "after" from the Painted Desert Challenge model, demonstrating the reduction in entropy caused by the agents' behavior



discussed in class and/or implemented as student projects.

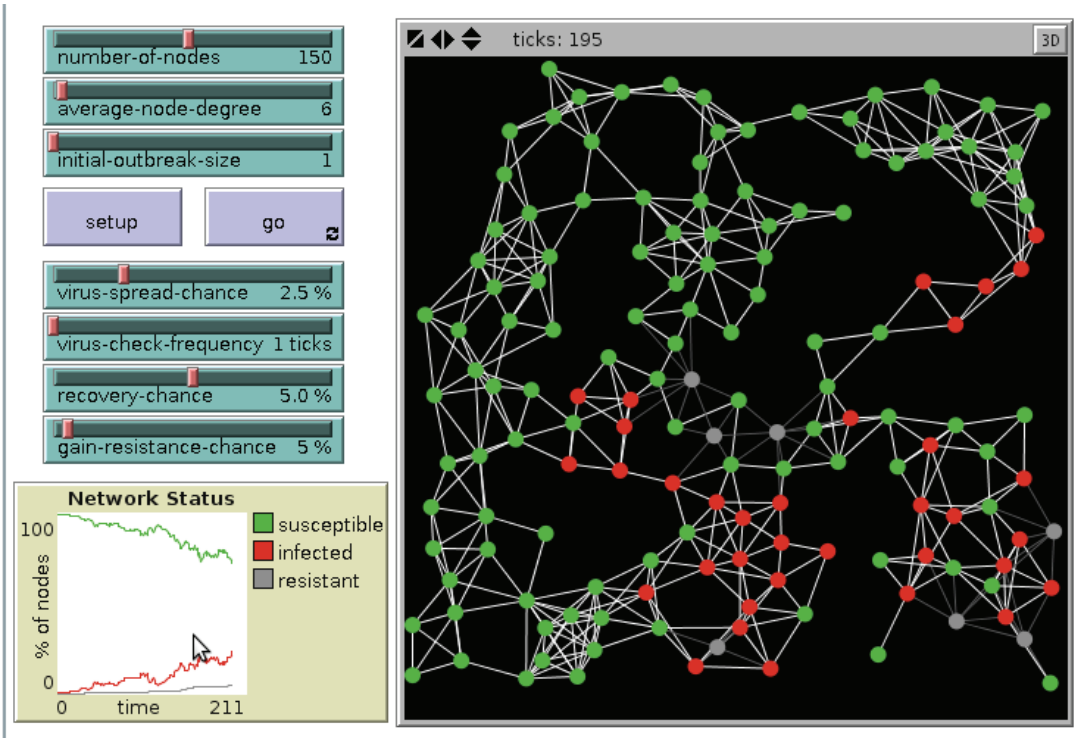
Security: Virus on a Network Model

Discussions about computer networks and security are not particularly common in introductory CS classes, which often focus more on programming and data structures. However, many computer science graduates go on to pursue careers in information technology where security is of paramount concern, which provides motivation for bringing this type of material into earlier coursework.

Rather than focusing on lower-level details of security, such as open ports or overrun buffer exploits, the Virus on a Network model (Stonedahl & Wilensky, 2008c) is concerned with security on a grander scale. In particular, worms and viruses that self-propagate from computer to computer through the Internet form a grave risk for today's society due in part to the creation of large "botnets" capable of acting in unison to carry out destructive distributed denial-of-service attacks, or other illicit activities. Virus on a Network is an abstract model, based on the SIR (Susceptible, Infected, Removed) models found in epidemiology (e.g., Hethcote, 1989). The setup consists of nodes (i.e. computers) on a network, and links between them, which could represent a variety of different connections depending on the attack vector of the virus (e.g., email contacts, shared network drives, shared USB keys, external hard drives, or floppy disks, etc). Nodes start as susceptible, except for some specified number that are infected with the virus. With some probability (which is controlled by an adjustable model parameter), a node that is infected by the virus can spread that virus to each of its neighboring nodes. Infected nodes also have a chance of recovering (e.g., an antivirus program removed the virus but didn't close up the vulnerability), and they have a chance of recovering and becoming resistant to future attacks (e.g., an antivirus program inoculated the computer against this virus). (see Figure 3)

Through exploration of the model, students can learn about how parameters such as number of nodes or average number of connections affect how quickly the virus moves through the network,

Figure 3. A screenshot from the virus on a network model

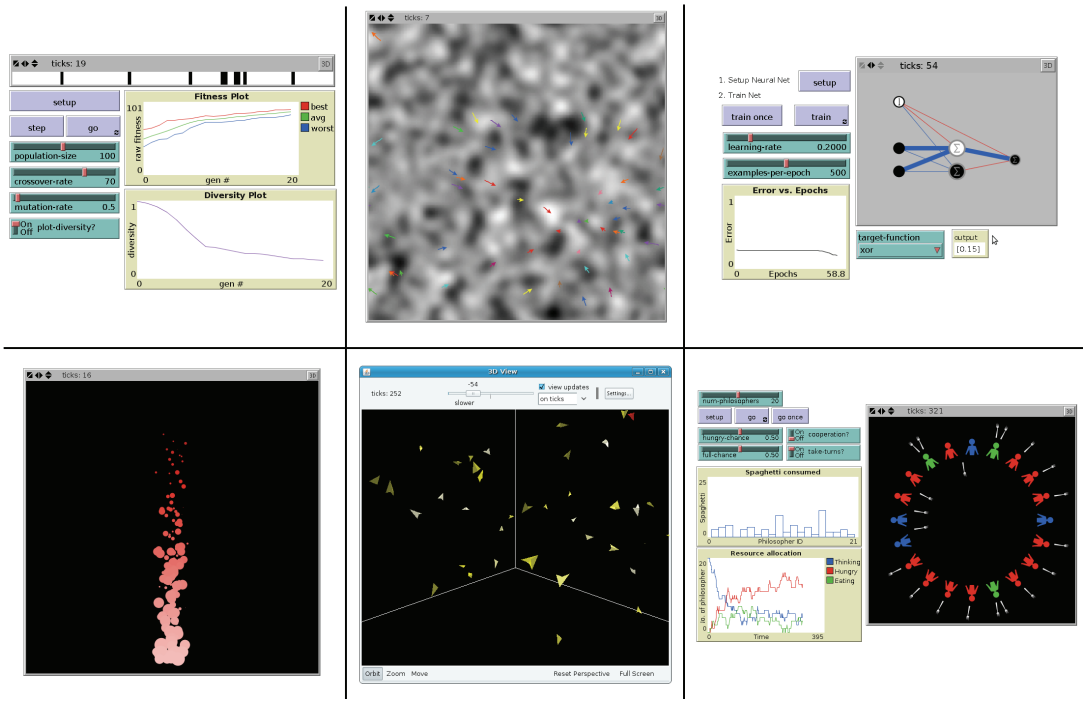


as well as the lifetime of the virus, and the extent to which vaccination of a few nodes can or cannot prevent a widespread epidemic. The Virus on a Network model also has potential connections to other disciplines (such as medicine, marketing, or sociology), and promotes high-level discussions about computer security practices, the structure of social and computer networks, and the Internet. This also leads naturally to student projects and extensions of the model. For instance, the default network structure found in this model is based on spatial proximity of the nodes, with nodes that are closer together in the 2D plane having a high probability of being linked, whereas there are no long-distance links. Students can discuss whether such a configuration is plausible for virus contagion[1] and write code to generate other types of network. A few other possible extensions include allowing the virus to mutate and evolve, and thus be able to re-infect computers which had become immune to a previous version of the virus, or to allow for coordinated (botnet) attacks by groups of infected nodes, or two have multiple different viruses present in the network. There are always opportunities for ambitious students to take this type of work further, and spin it off into summer research projects.

Remaining Model Suite Overview

The remaining six models in our model suite (see Figure 4) cover topics from an additional four areas of computer science. The *Simple Genetic Algorithm* and *Particle Swarm Optimization* models (Stonedahl & Wilensky, 2008a, 2008b) both offer an introduction to stochastic optimization algorithms by illustrating how agents, acting with limited intelligence and information, can move toward a goal. In the Simple Genetic Algorithm model illustrates an evolutionary search process in which fitness and diversity levels of a population change over time. In the Particle Swarm

Figure 4. Model screenshots. Top row: Simple genetic algorithm, particle swarm optimization, artificial neural betwork; Bottom row: Particle system flame, flocking 3D, dining philosophers



Optimization, the progress towards a goal can be viewed as agents traverse a 2D fitness landscape, searching for a global optimum. While these topics are not usually covered until much later in a traditional computer science curriculum (probably an upper-level elective course), we believe they are thoroughly accessible to introductory-level students from an ABM/MAS perspective. Additionally, they allow interdisciplinary connections to evolutionary biology and particle physics.

Artificial neural networks can also be productively understood from an agent-based perspective, as we hope to demonstrate to students through exploration of the *Artificial Neural Net* model (Rand & Wilensky, 2006). Each perceptron (simplified virtual neuron) can be conceived as an agent, which follows certain rules during the training phase, and then another set of rules when it is being tested. This is another fairly advanced topic, which admittedly may take some effort for students to understand and appreciate. However, it is not necessary for students to understand every detail of the back-propagation training algorithm or what the nice mathematical properties of a sigmoid function are — this can wait. The important thing is for students to gain an intuitive understanding of how the agents are activating each other, and that by automatically modifying the weights of connections between agents, it is possible for the system as a whole to "learn" pattern recognition skills. A classroom discussion comparing and contrasting this agent-based model with biological neural networks should also prove provocative and educational.

Agent-based modeling is useful in computer graphics as well, and is being increasingly explored as a means of automatically creating realistic procedural animations of systems with many interacting creatures or objects. Through the *Particle System* models[2] (Kornhauser & Wilensky, 2007) students can get a taste of the classic "particle systems" approach sometimes used in cinematic

animation to create the illusion of water, fire, or smoke. While each particle is fairly passive, being pushed or pulled by an externally imposed force field, it is still useful to think of each particle as one agent of a distributed multi-agent system, and it is not difficult to modify the code to make the agents take a more active and/or intelligent role in their movement patterns. For instance, more sophisticated agent behavior is exhibited in the "Boids" algorithm (Reynolds, 1987) for creating realistic-looking flocks of animated creatures, which is the inspiration for our *Flocking* model (Wilensky, 1998). As the Flocking screenshot in Figure 4 shows, the NetLogo modeling environment also provides facilities for the development and visualization of three-dimensional models, which opens more possibilities for students to extend, modify, or create their own multi-agent computer animations.

The *Dining Philosophers* model (Wilensky, 2003) introduces a classic case study in the synchronization of concurrent processes, posed as a puzzle about philosophers sitting around a table eating spaghetti that requires two forks to eat with, but having to share forks between them. Through this metaphor, concepts such as deadlock and resource starvation are explained. We mentioned above that part of the motivation for the MAgICS framework was the increasingly parallel nature of computing, such as the shift to multi-core and multi-processor machines. In reaction, multi-threaded and multi-process programming will become more pervasive, and it seems quite appropriate to include in the curriculum an agent-based model that addresses issues of resource sharing.

IMPLEMENTING MAgICS: A PILOT STUDY

Our hypothesis is that using an ABM/MAS perspective in introductory-level computer science courses, as the MAgICS framework does, offers some potential advantages over traditional computer science curricula: specifically, that it provides students with an early, but applicable background in concepts such as distributed computing and multi agent systems, and that the subject matter is more engaging than that found in many typical introductory computer science. In preparation for a complete implementation of the MAgICS framework in the future, we conducted a preliminary pilot study, which we describe below. It is important to note that while student modification and creation of agent-based models is an important component of the MAgICS framework, it was not a component of the pilot implementation. As such, this study addresses broad-scale questions regarding the *feasibility* of integrating ABM/MAS into classrooms sessions, students' *enjoyment* and the *perceived relevance* of the approach, and students' very general take-up and use of *distributed and decentralized techniques* when thinking about problems. Our primary intentions are to provide support for the ongoing development of the MAgICS framework, additional pedagogical support, and curricular materials (assignments, lecture notes, etc.).

Study Design

The pilot study was a 50-minute classroom-based implementation of several elements of the MAgICS framework, roughly two-thirds of the way through an introductory level computer science course at a private research university in the United States. The course is the first in the standard sequence for CS majors at this university, and draws students from both the school of engineering and the college of arts and sciences. The students were informed several days in advance that their usual professor would be out of town, and that a guest lecturer would be teaching the class instead. The total enrollment for the class was 39 students; 25 students attended class that day and all of those took part in the study. The class was learning (primarily functional) computer programming

using a variant of the Scheme language. Although the course does incorporate several contemporary pedagogical elements, such as the use of images and animation, in addition to standard numeric and string manipulation routines, it does not have any focus on distributed or decentralized topics or approaches.

The format of the study consisted of four parts: a pre-quiz (7-8 minutes), a period of lecture (35 minutes), a post-quiz (7-8 minutes), and a follow-up survey soliciting student feedback. The lecture component consisted of the following sequence: (1) a few introductory remarks contrasting centralized and decentralized approaches to designing computational systems, and recent technological shift toward more distributed/parallel systems in the real-world; (2) a demonstration of how multiple agents could be programmed to move simultaneously, using the NetLogo language; (3) a discussion/demo of the Painted Desert Challenge model discussed above (preceded by a simpler "Termites" model, which only involves one color); (4) a discussion/demo of the PageRank model, and Google's PageRank™ algorithm to rank the relevancy of web search results. In a semester-length implementation of the MAgICS curricula, the topics discussed in this compact lecture would be spread across at least three sessions, which would allow for more elaboration, class discussion, extension and production of models, and in-depth treatment. However, for this pilot study, we chose to give students a broader taste of the MAgICS curriculum by exposing them to at least two distinct multi-agent systems topics.

Research Questions, Data and Results

We are not arguing that the brief quiz responses obtained during a single class session exploring components of MAgICS are enough to illuminate whether or how students learned specific concepts or approaches in distributed computing. Instead, we view this study as a preliminary look into the feasibility and potential for an ABM/MAS perspective to present decentralized thinking as a relevant approach for addressing problems in computing. In this sense, our research questions related to this pilot implementation are:

- **R1:** Prior to the class session, did students have exposure to topics in parallel computing, such as distributed/decentralized design and multi-agent systems? Did they consider issues of distribution and decentralization when solving computational problems?
- **R2:** Can a brief presentation of topics from a multi-agent perspective using agent-based models influence how students think about, and design solutions for, computational problems?
- **R3:** Do students find the MAgICS approach relevant and engaging?

Pre/Post Quizzes

Before and after the class session, quizzes were distributed to students. Both groups were given the same first question on the pre-quiz and the post-quiz, but two versions of question 2 were included from the pre- to post quiz: so that roughly half of the students answered question 2A on the pre-quiz and 2B on the post-quiz, and the other half question 2B, then 2A. By having students answer a different question on the pre-quiz than the post-quiz, we hoped to give them a new context to apply the ideas explored in class, By making sure we had some pre- and post- responses for both questions via different groups, we hoped to disentangle whether differences in students responses were a result of the question itself (2A versus 2B), or the class session. Three students who attended the class did not complete both the pre-quiz and post-quiz, and were thus omitted from our analysis, leaving a sample size of 22 students that answered all 4 questions (2 before the lecture and 2 after).

Due to the time constraints of the implementation and quizzes, our analysis focuses on very coarse-level outcomes: namely, the presence of decentralized thinking and knowledge of the PageRank algorithm in question 1, and presence or absence of design elements that reflect different dimensions of decentralized computing in question 2 (we focus on the specific coding schematics in each respective section). Because students were not provided much time to complete the quizzes, and quiz performance is not considered for students' grades, it is likely that student responses were shorter and less complete than they might have otherwise been. This suggests that our results are, if anything, an underestimate of their adoption of decentralized design strategies after being presented with MAgICS activities.

Each question, our motivation for including it, and the coding schemes used to determine results are described in detail below. For our analysis, question responses were first anonymized and organized such that their status as pre- or post-quiz responses was not apparent to coders. The authors then coded these responses, and a subportion (~32%, split evenly among each question) were also coded by an independent interrater with expertise in agent-based modeling and multi agent systems. Interrater agreement on question 1 was 100%, and on question 2 was 97%.

Question 1

Pre Question: *If you perform a Google™ search for the word "turtle", there are over 5 million results. Describe how the Google search engine might choose to order the search results for you to view, so that the most useful/relevant web pages appear at the top.*

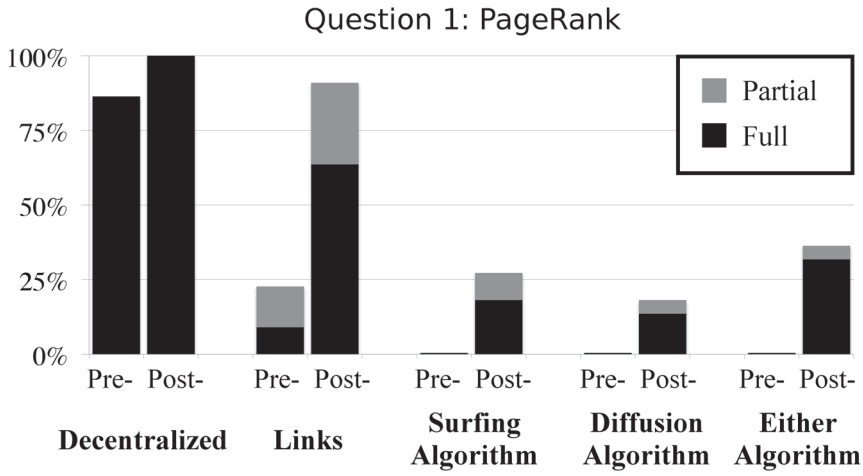
Post Question: *Briefly describe how Google's PageRank™ algorithm assigns scores to rank sites according to their general importance in the world wide web.*

Students' brief responses were coded for references to distributed/decentralized approaches to web page searching, across the following categories. At times, "partial credit" was given to responses, these are identified separately as the grey components of the histogram bars:

- **Decentralized.** Do student answers include discussion of any decentralized/ multi-agent factors? Specifically, did they mention ranking a web page by using features of the web page that are not contained within the page itself, but result from the interaction with other web pages or human agents.
- **Links.** Do the students mention using inbound links (that point to the page being ranked) as a factor (partial credit), and do they also take into consideration the importance of the page from which the inbound links originate (full credit)?
- **Surfing Algorithm.** Do students mention the use of automated surfer bots (partial credit), and general algorithms by which those bots can be used to calculate the PageRank scores (full credit)?
- **Diffusion Algorithm.** Do students mention the use of diffusion or value sharing (partial credit) that is repeated until an equilibrium is reached (full credit) to calculate the PageRank scores?
- **Either Algorithm.** This category is derived from the codes for the Surfing and Diffusion Algorithms, and features the number of students that mentioned either search algorithm (if they mentioned both, the highest level of credit is included).

As shown in Figure 5, more student responses fell into each of the coding categories across the board after the PageRank demonstration in class. These results are not particularly surprising, since one would hope that students should have a better understanding of the topic after

Figure 5. Student pre- and post-quiz results for question 1, separated by coding category



it has been discussed in the lecture. However, the data do offer support for several arguments. First, when confronted with a problem that is situated in a distributed context (such as the world wide web), even without prior exposure/ discussion about multi-agent systems, students often consider distributed ideas or approaches (this is in contrast with the results for question 2B, discussed below). Second, given only a very brief lecture (10-15 minutes) on this topic, many students were able to understand the basic ideas of PageRank (91% explicitly mentioned some form of hyperlink analysis), and a substantial fraction (32%) were able to give a reasonably complete ("full") description of at least one of the two algorithmic processes by which PageRank can be calculated. These preliminary results suggest that the subject matter is within reach of students, and offer support for the feasibility of our approach for teaching "upper-level" topics from a multi-agent systems perspective.

In our pilot implementation, the final lecture segment (discussing the PageRank algorithm) ended up being somewhat rushed, especially with regard to the rules of the two featured algorithms ("Random Surfer" and "Diffusion") for computing PageRank. Given ample class time to discuss, explain, and elaborate on the PageRank model, we would expect to see considerably greater improvement in the algorithmic understanding categories.

Question 2

2A. Planting: *(Adapted from Kolikant 2001) Suppose you are hired by Automagical Landscaping to design a robotic system for the very specific task of planting 1000 trees in an open field as* **efficiently** *as possible. (The planting sites will be designated by visual markers that are easy for robots to detect.) Planting one tree consists of three sub-tasks: digging a hole, dropping a seed in the hole, and filling in a hole. In order to keep costs down, robot physical capabilities and "intelligence" should be kept as simple as possible. Describe your approach.*

2B. Sorting: *(Adapted from Wilensky, 1997a) Suppose you are hired by the GWYPF (Get-What-You-Pay-For) Agricultural Cooperative to design a robotic system for sorting grain based on its quality. Some grain is harvested too early, too late, or didn't get as many nutrients as other grain, and GWYPF wants to differentiate between grain*

that's grade A, grade C, grade M, or grade Z! Your laboratory has already developed a hyperspectral imaging sensor that can measure the quality of a single kernel from close-range. However, now you are faced with a vast warehouse, with kernels of corn spread thinly (and randomly) across the floor. Your robotic system should (roughly) sort the grain from worst to best from East to West. Describe a possible design for this system.

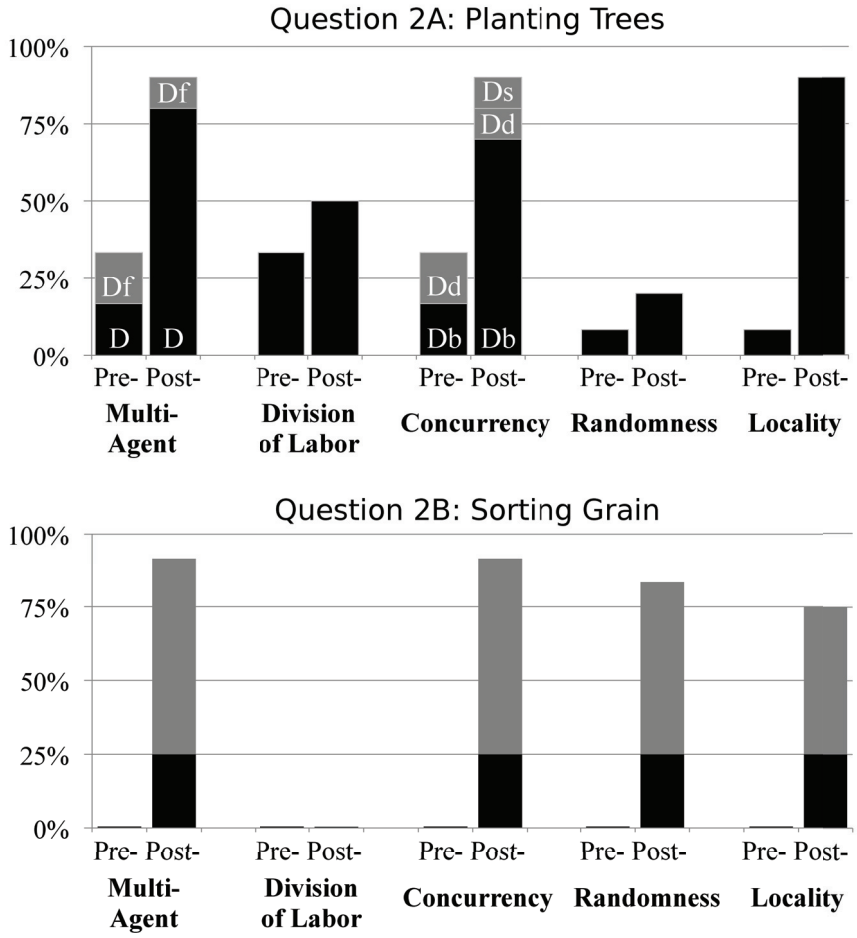
For question 2, responses were coded for the presence or absence of evidence of decentralized/distributed approaches along a number of dimensions. If the proposed solution involved any of the following elements, we coded that dimension as "**D**": in some cases, specific variants of these elements were identified separately as gray components of the histogram bars.

- **Multi-Agent**: Does the designed system involve *multiple agents*? (**D** signifies yes, while **Df** signifies only insofar as agents exist for different functions – for example, in the case of the gardening problem, a single "digger", "planter", and "burier".)
- **Division of Labor**: Does the designed system exhibit a *division of labor* whereby different types of agents perform specific tasks?
- **Concurrency**: Can the designed system be executed *concurrently* such that the same or different tasks can be executed at the same time? (**Ds** signifies that only the same task can be executed concurrently: for instance, if multiple robots can dig simultaneously, but planting cannot also occur at the same time; **Dd** signifies that only different tasks can be executed at the same time; **Db** signifies that both the same or different tasks can occur at the same time.)
- **Randomness**: Does the designed system involve elements of *randomness* in its behavior?
- **Locality**: Does the designed system involved updating and acting upon *local* information, rather than a prespecified knowledge of the entire system state?

Additionally, since the "sorting" question (2B) shares a lot of surface resemblance to the Termites/Painted Desert model that was presented in class, many students suggested a solution that involved that very same algorithm, without considering the additional constraints imposed by the quiz problem (namely the need to sort grain East to West by quality). We identify these solutions in our plotted results by grey bars, and solutions that explicitly adapted the Termites example to address the East/West constraint, or introduced a different solution that could be coded for the presence of those components, by the black components of the bars. The results all coding categories for each of the questions are shown in Figure 6.

We would first like to make a point about the effect of task selection on the differences in results between 2A and 2B. It is clear (and expected) that question type and reasonableness play an important role in students' application of decentralized approaches. Along these lines, it is not clear that specializing function in the case of sorting grain (question 2B), or randomizing the motion of robots rather than having them engage in more directed search and detection methods for finding markers in a garden (question 2A), are reasonably adding efficiency or other benefits a student's designed system. This is reflected in student responses: they did not apply every decentralized technique they were presented during the lecture "across the board", but rather suggested techniques that were appropriate to the problem statement. Furthermore, while it is established that students often adopt a deterministic/centralized mindset even toward phenomena that may not be well understood using this mindset and that such ways of thinking are difficult to change, we saw that (a) some students *did* use such reasoning even before the class period for the gardening tasks,

Figure 6. Student pre- and post-quiz results for questions 2A and 2B, presented separately by coding category



and (b) introducing agent-based models as computational problem solving techniques did appear to prompt these students to consider decentralized solutions. In other words, task selection might be one way to provide students with a "primer" to distributed thinking.

Our two main arguments given this very cursory data is that students did apply techniques that are characteristic of decentralized solutions to problems more often after the class session than they did before, and that in least in the case of question 2A, they sometimes did even before an explicit introduction to those techniques. We consider these findings to be quite promising, with regard to whether students possess an intuitive ability to think about issues of distribution as well whether students can learn the general concepts of distributed/decentralized computing at an early stage in their computer science education. It appears that students are considering issues of parallelism, and these considerations are brought to the forefront for consideration in issues of design by even a brief introduction to multi-agent systems in applied contexts. Although this study cannot speak to students' ability to productively integrate these concepts into instantiated designs, it certainly supports our claim that agent-based modeling and issues of distributed and decentral-

ized thinking are not only motivating and relevant, but accessible for students at an introductory level in a way that addresses problems of algorithm design and problem solving.

Student Survey

The student survey consisted of eight questions, with responses based on a 1-5 Likert scale, with 1 representing the most negative response and 5 representing the most positive, plus one open-response question which requested "any other comments about the lecture, topics discussed, or materials used?" Seventeen students responded to this voluntary anonymous survey, and the results are tabulated in Table 2. The eight category names correspond to the actual survey questions as follows.

- **Familiarity.** *Prior to this guest lecture, how familiar were you with distributed and multi-agent systems?*
- **Learning.** *How much do you feel you learned from this class?*
- **Quiz.** *How appropriate were the quiz questions for the topics presented and your level of knowledge?*
- **Applicability.** *How applicable do you think the discussed topics are to real problems in engineering and computing?*
- **Interest.** *How interesting and engaging was the class session?*
- **Termites.** *How much did you enjoy the "multi agent sorting with termites" topic?*
- **PageRank.** *How much did you enjoy the "Google PageRank" topic?*
- **Future Interest.** *How interested would you be in learning more about multi-agent systems?*

All the results should be interpreted with the standard limitations that apply to any self-reported survey data. That said, they mostly served to support/confirm our prior intuitions and hypotheses.

Table 2. Anonymous online student feedback survey results

		Mean	Median
1	Familiarity	2.1	2
	Learning	3.8	4
	Quiz	3.7	4
2	Applicability	4.4	5
3	Interest	4.2	4
	Termites	4.4	4
	PageRank	4.1	4
	Future Interest	4.3	4

However, several points are worth highlighting. First, on average students reported only a small amount of prior *familiarity* with distributed and multi-agent systems; more than half of the students reported a score of 2 or lower. Second, the results are generally quite positive, with over half of the class reporting either 4 or 5 (positive or very positive) for every question relating to how much they *learned* from the class, *enjoyed* particular subject matter, viewed the material as appropriate or *applicable*, etc. Third, more than half of the students responded that the topics discussed in class were "very applicable" (5) to real-world problems in engineering and computing. While it is the authors' view that the world is becoming increasingly distributed, and that decentralized thinking and the design of multi-agent systems are becoming a key skill for the next-generation of computational problem solvers, it was not evident to us (before administering the survey) whether or not students would share this view. When judging student motivation and engagement, it is difficult to disentangle characteristics of the curriculum/subject matter from style in which it is taught; a good lecturer can make even the driest material engaging. Accordingly, we must limit our findings to conclude only that the study provides evidence that the topics/ideas **can** be presented in a manner that students consider enjoyable and relevant.

Seven students responded to the free-form question soliciting any additional comments: several students commented on the quality of the lecture, or that the topics discussed were "very interesting". For instance, one student wrote "*This lecture was interesting and although not exactly relevant to the course, I think I learned a lot that I can apply to my life. Well done!*", and another wrote "*It was a very educational and interesting experience. Being a sort of beginning programmer I didn't understand a lot of what was going on, but I had a great appreciation for the program and interaction and everything. Very very interesting stuff. Thank you for the experience!*" These comments bring up a number of important points, both about the strengths and limitations of this pilot excursion toward implementing the MAgICS framework. The first response didn't judge the topic was "relevant to the course", and the second expressed difficulty understanding the computer programming aspects. These both speak to the fact that within the time constraints of a single class period, we were unable to delve deeply into learning the programming language, or have students write their own code to achieve a deeper understanding of *programming* multi-agent systems, instead of merely observing them and learning about the rules/algorithms in English language. We believe that this is an important component that needs to receive further consideration when implementing the MAgICS framework. In any case, these responses do show a clear indication of interest and enthusiasm for these topics and/or an appreciation for applying the ideas learned during this brief class period to real life issues.

Limitations

We would like to stress that this pilot implementation of the MAgICS framework was not intended as a rigorous test of its efficacy – indeed, many aspects and activities that we would consider essential components of the framework, such as students designing and constructing agent-based models, were absent from our pilot. Instead, we view the pilot as providing support for the feasibility of introducing these topics at an introductory level in the context of introductory computer science, as well as for our claims regarding the motivational potential relevance to real world topics. In other words, we believe this supports our claims that MAgICS to be a successful component of introductory computer science because it provides evidence that students' strategies, motivations, and ways of thinking are consistent with and responsive to the MAgICS approach.

On the other hand, this study does not illuminate very much about *how* or *how much* students learn. The short student quiz responses did not provide nearly enough information to draw any conclusions along these lines; a more intensive study would be required. Furthermore, the fact that attendance of the class session and completion of the online student survey were optional may have led to a sampling bias favoring individuals who were more academically motivated than the average student, which could have impacted our study results.

DISCUSSION AND FUTURE RESEARCH DIRECTIONS

Our intention in this work is to offer a window into an alternative introductory computer science curriculum. In practice, we would not expect this approach to be used to the complete exclusion of other curricula or approaches. We emphasize that in many cases it would be most beneficial to compare and contrast centralized and decentralized approaches to the same topic. Furthermore, the introductory course should still have a strong emphasis on learning to write computer programs. However, starting with existing programs (in this case, agent-based models) provides an opportunity for students to explore, modify, and learn to read the language while they are learning to write it.

The MAgICS framework permits the integration of a number of other techniques shown to be beneficial for computer science education. There is no reason, for example, that the pair programming approach (Carver, et al., 2007) or the integration of robotics (Blank, 2006) cannot be implemented successfully within the context of MAgICS. Indeed, the NetLogo programming environment includes interfaces to various physical devices and a variety of "bifocal modeling" (Blikstein & Wilensky, 2007) activities, which allow users in a variety of contexts to compare computational agent-based models with real-world data collected using robotic sensors and actuators. Also, the VBot materials (Berland & Wilensky, 2005), designed primarily for middle school students, engage users in programming independent robot agents, which can then interact with one another in a shared context (such as a robot soccer arena). Some inclusion of physical robotics would be very natural addition to an introductory course themed around ABM/MAS.

Our present work on the MAgICS framework serves as a starting place for future investigations about reinventing introductory computer science education with a focus on multi-agent systems. We plan to engage in more substantial research that includes more aspects of the MAgICS framework, such as having students design and program their own agent-based models tasks and performing more in-depth analyses (through interviews and detailed analyses of student work) of understanding of both the specific computer science topics explored, as well as overarching themes such as emergence, distributed/decentralized approaches to design and problem solving, etc. Such studies will help us to determine which topics and student activities are most appropriate for an introductory curriculum, and will contribute toward our continued refinement and development of the framework.

CONCLUSION

Through the MAgICS framework we are offering a first attempt at producing a coherent introductory CS curriculum centered on a series of agent-based models spanning a variety of computer science topics. We believe that this framework addresses recent calls by computer science educators to introduce widely applicable, engaging curricula early in the computer science sequence with a focus on the notion of "computational thinking", rather than specific algorithms and techniques. There unquestionably remains considerable room for improvement in this framework, and we hope that this work leads to expanded conversation and academic discourse about the fusion of multi-agent systems approaches with computer science education.

ACKNOWLEDGMENT

We wish to thank Josh Unterman for helping to refine our ideas and his contributions supporting our work on the pilot study, and Ian Horswill, for providing a context for the study. We also owe a debt of gratitude to the participants of the EduMAS 2009 workshop, all of our colleagues in the CCL research group, and several anonymous reviewers for their constructive feedback. This work was supported in part by NSF grant IIS-0713619.

REFERENCES

Becker, B. (2001). Teaching CS1 with karel the robot in Java. *ACM SIGCSE Bulletin*, *33*(1), 50–54. doi:10.1145/366413.364536

Bell, T., Witten, I. H., & Fellows, M. (1999). *Computer science unplugged: Off-line activities and games for all ages*.

Berland, M., & Wilensky, U. (2005). *Complex play systems: Results from a classroom implementation of Vbot*. Paper presented at the annual meeting of the American Educational Research Association, Montreal, Canada.

Blank, D. (2006). Robots make computer science personal. *Communications of the ACM*, *49*(12), 25–27. doi:10.1145/1183236.1183254

Blikstein, P., & Wilensky, U. (2006). *An atom is known by the company it keeps: A constructionist learning environment for materials science using multi-agent simulation*. Paper presented at the Annual Meeting of the American Educational Research Association, San Francisco, CA.

Blikstein, P., & Wilensky, U. (2007). *Bifocal modeling: a framework for combining computer modeling, robotics and real-world sensing*. Paper presented at the annual meeting of the American Educational Research Association, Chicago, IL.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems, 30*(1-7), 107 - 117.

Brueckner, S. (2000). *Emergent Sorting: Software Demonstration at the Fourth International Conference on Autonomous Agents (Agents 2000),* Barcelona, Spain.

Buckley, M., Nordlinger, J., & Subramanian, D. (2008, March 12-15). *Socially Relevant Computing.* Paper presented at the Proceedings of the 2008 Annual Meeting of the Special Interest Group in Computer Science Education, Portland, Oregon, USA.

Carver, J. C., Henderson, L., He, L., Hodges, J., & Reese, D. (2007). Increased Retention of Early Computer Science and Software Engineering Students using Pair Programming, *Proceedings of the 20th Conference on Software Engineering Education \& Training* (pp. 115-122).

Cushing, J. B., Weiss, R., & Moritani, Y. (2007). CS0++ broadening computer science at the entry level: interdisciplinary science and computer science. *J. Comput. Small Coll.*, *23*(2), 51–57.

Davidsson, P. (2002). Agent based social simulation: A computer science view. *Journal of Artificial Societies and Social Simulation*, *5*(1), 7.

Denning, P. J., & McGettrick, A. (2005). Recentering computer science. *Communications of the ACM*, *48*(11), 15–19. doi:10.1145/1096000.1096018

diSessa, A. (2001). *Changing minds: Computers, learning, and literacy*. Cambridge, MA: The MIT Press.

Downey, A. B., & Stein, L. A. (2006). Designing a small-footprint curriculum in computer science, *Proceedings of the 36th ASEE/IEEE Frontiers in Education Conference*.

Epstein, J., & Axtell, R. (1996). *Growing artificial societies: Social science from the bottom up*. Washington, DC: Brookings Institution Press.

Fagin, B., & Merkle, L. (2003). *Measuring the effectiveness of robots in teaching computer science*.

Flowers, T., & Gossett, K. (2002). Teaching problem solving, computing, and information technology with robots. *Journal of Computing Sciences in Colleges*, *17*(6), 45–55.

Goode, J. (2007). If You Build Teachers, Will Students Come? The Role of Teachers in Broadening Computer Science Learning for Urban Youth. *Journal of Educational Computing Research*, *36*(1), 65–88. doi:10.2190/2102-5G77-QL77-5506

Guckenheimer, J., & Ottino, J. M. (2008). *Foundations for Complex Systems Research in the Physical Sciences and Engineering*. National Science Foundation.

Guzdial, M. (2003). A media computation course for non-majors. *SIGCSE Bulletin*, *35*(3), 104–108. doi:10.1145/961290.961542

Hethcote, H. (1989). Three basic epidemiological models. *Applied Mathematical Ecology*, 119–144.

Howland, J. (1997). It's all in the language (yet another look at the choice of programming language for teaching computer science). *Journal of Computing in Small Colleges*, *12*(4), 58–74.

Johnson, S. (2001). *Emergence: The Connected Lives of Ants, Brains, Cities and Software*. Allen Lane.

Katz, S., Allbritton, D., Aronis, J., Wilson, C., & Soffa, M. L. (2006). Gender, achievement, and persistence in an undergraduate computer science program. *SIGMIS Database*, *37*(4), 42–57. doi:10.1145/1185335.1185344

Kolikant, Y. B.-D. (2001). Gardeners and cinema tickets: High school students' preconceptions of concurrency. *Computer Science Education*, *11*(3), 221–245. doi:10.1076/csed.11.3.221.3831

Levy, S. T., Novak, M., & Wilensky, U. (2006). *Students' foraging through the complexities of the particulate world: Scaffolding for independent inquiry in the connected chemistry (MAC) curriculum*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco, CA.

Levy, S. T., & Wilensky, U. (2009). Crossing levels and representations: The Connected Chemistry (CC1) curriculum. *Journal of Science Education and Technology*, *18*(3), 224–242. doi:10.1007/s10956-009-9152-8

Moritz, S. H., Wei, F., Parvez, S. M., & Blank, G. D. (2005, June 27-29). *From objects-first to design-first with multimedia and intelligent tutoring*. Paper presented at the Proceedings of the 10th annual conference on Innovation and techology in computer science education, Caparica, Portugal.

NRC. (2003). *BIO2020: Transforming undergraduate education for future research biologists*. Washington, DC.

Overmars, M. (2004). Teaching computer science through game design. *Computer*, 81–83. doi:10.1109/MC.2004.1297314

Panait, L., & Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, *11*(3), 387–434. doi:10.1007/s10458-005-2631-2

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. New York: Basic Books, Inc.

Patterson, D. A. (2005). Restoring the Popularity of Computer Science. *Communications of the ACM*, *48*(9), 25–28. doi:10.1145/1081992.1082011

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., et al. (2007). *A survey of literature on the teaching of introductory programming*. Paper presented at the Working group reports on ITiCSE on Innovation and technology in computer science education.

Radenski, A. (2006). "Python first": a lab-based digital introduction to computer science. *SIGCSE Bulletin*, *38*(3), 197–201. doi:10.1145/1140123.1140177

Rashid, R. (2008). Image Crisis: Inspiring a new generation of computer scientists. *Communications of the ACM*, *51*(7), 33–34. doi:10.1145/1364782.1364793

Resnick, M., & Wilensky, U. (1992). *StarLogo workshop*. Paper presented at the Artificial Life III.

Resnick, M., & Wilensky, U. (1998). Diving Into Complexity: Developing Probabilistic Decentralized Thinking Through Role-Playing Activities. *Journal of the Learning Sciences*, *7*(2), 153–172. doi:10.1207/s15327809jls0702_1

Reynolds, C. W. (1987). *Flocks, herds and schools: A distributed behavioral model.* Paper presented at the SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques, New York, NY, USA.

Sengupta, P., & Wilensky, U. (2008). *On Learning Electricity with Multi-agent Based Computational Models (NIELS).* Paper presented at the Proceedings of the International Conference of the Learning Sciences (ICLS), Utrecht, The Netherlands.

Stein, L. (1999). Challenging the computational metaphor: Implications for how we think. *Cybernetics and Systems*, *30*(6), 473–507. doi:10.1080/019697299125073

Tisue, S., & Wilensky, U. (2004). NetLogo: A Simple Environment for Modeling Complexity, *Proceedings of the International Conference on Complex Systems*.

Wilensky, U., & Rand, W. (in press). *An introduction to agent-based modeling: Modeling natural, social and engineered complex systems with NetLogo*. Cambridge, MA: MIT Press.

Wilensky, U., & Reisman, K. (2006). Thinking Like a Wolf, a Sheep, or a Firefly: Learning Biology Through Constructing and Testing Computational Theories-An Embodied Modeling Approach. *Cognition and Instruction*, *24*(2), 171–209. doi:10.1207/s1532690xci2402_1

Wilensky, U., & Resnick, M. (1999). Thinking in Levels: A Dynamic Systems Approach to Making Sense of the World. *Journal of Science Education and Technology*, *8*(1), 3–19. doi:10.1023/A:1009421303064

Wing, J., M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. doi:10.1145/1118178.1118215

ADDITIONAL READING

Kornhauser, D., & Wilensky, U. (2007). *NetLogo Particle System Flame model: Center for Connected Learning and Computer-Based Modeling*. Evanston, IL: Northwestern University.

Rand, W., & Wilensky, U. (2006). *NetLogo Artificial Neural Net model: Center for Connected Learning and Computer-Based Modeling*. Evanston, IL: Northwestern University.

Stonedahl, F., & Wilensky, U. (2008a). *NetLogo Particle Swarm Optimization model: Center for Connected Learning and Computer-Based Modeling*. Evanston, IL: Northwestern University.

Stonedahl, F., & Wilensky, U. (2008b). *NetLogo Simple Genetic Algorithm model: Center for Connected Learning and Computer-Based Modeling*. Evanston, IL: Northwestern University.

Stonedahl, F., & Wilensky, U. (2008c). *NetLogo Virus on a Network model: Center for Connected Learning and Computer-Based Modeling*. Evanston, IL: Northwestern University.

Stonedahl, F., & Wilensky, U. (2009). *NetLogo PageRank model: Center for Connected Learning and Computer-Based Modeling*. Evanston, IL: Northwestern University.

Wilensky, U. (1997a). *NetLogo Painted Desert Challenge model: Center for Connected Learning and Computer-Based Modeling*. Evanston, IL: Northwestern University.

Wilensky, U. (1997b). *NetLogo Traffic Basic model: Center for Connected Learning and Computer-Based Modeling*. Evanston, IL: Northwestern University.

Wilensky, U. (1998). *NetLogo Flocking 3D model: Center for Connected Learning and Computer-Based Modeling*. Evanston, IL: Northwestern University.

Wilensky, U. (1999). *NetLogo: Center for Connected Learning and Computer-Based Modeling*. Evanston, IL: Northwestern University.

Wilensky, U. (2003). *NetLogo Dining Philosophers model: Center for Connected Learning and Computer-Based Modeling*. Evanston, IL: Northwestern University.

ENDNOTES

1 Generally speaking, it is not. Real-world networks usually display a power-law degree distribution and "small-world" structure which includes the presence of long-distance links. However, we consciously chose this spatially-restricted network structure to support clear visualization of the contagion process.

2 The Particle Systems model is technically divided into four distinct NetLogo model files — Basic, Flame, Fountain, and Waterfall — but they are all grouped together because they express the same fundamental idea.